TRABAJO DE FIN DE GRADO

“Realización de sistemas SCADA basados en tecnologías WEB".

GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

AUTOR

Raúl Alberto Domínguez Santa Cruz

TUTOR ACADÉMICO

Alberto Hamilton Castro

AGRADECIMIENTOS

A mi madre, que desde que tengo uso de razón siempre me ha aconsejado bien y apoyado.

A mi familia y amigos, sin los que no sería la persona que soy.

Al Servicio Electrónica de la Universidad de la Laguna, por su predisposición siempre que he necesitado su ayuda para el diseño y montaje de circuitos electrónicos, en especial a Julio Brito, que me ha tutorizado en este ámbito.

A mi tutor, pues este proyecto ha sido propuesta suya, con el que he podido sentirme realizado y aplicar todos los conocimientos que he adquirido durante estos años.

Resumen

El resultado de este proyecto es la implementación de un sistema SCADA en un entorno web, las tecnologías web (HTML, CSS y JavaScritpt) son las utilizadas en las páginas web modernas.

Como podemos ver todos los días, estas están dotadas de dinamismo, gráficas interactivas y adaptación a los distintos dispositivos (ordenadores, tablets, móviles, etc.).

Estas tecnologías son muy adecuadas para la realización de aplicaciones de supervisión de procesos industriales (SCADA). Estas consisten en la representación gráfica de los distintos elementos del proceso con información en tiempo real y evolución histórica de los valores de distintas variables de interés.

En este proyecto han sido utilizados miniordenadores BeagleBone Black (BBB) para el control de algunos procesos sencillos (temperatura, señales PWM etc.). Los BBB proporcionan una librería de JavaScript que permite el uso de las tecnologías web, BoneScript.

Este sistema ha sido implementado en una planta encargada al Servicio Electrónica de la Universidad de la Laguna, en la cual se quería realizar la acción de control. Para ello ha sido necesario el diseño y la realización de varios circuitos electrónicos que conecten el sistema a la placa.

Una vez ejecutado el control, se obtiene la respuesta del sistema, con lo cual se analiza su dinámica.

Abstract

The result of this project is the implementation of a SCADA system in a web environment, web technologies (HTML, CSS and JavaScritpt) are used in modern web pages.

As we see it, these kind of technologies are dynamism, with interactive graphics and adaptation to different devices (computers, tablets, mobile phones, etc.).

These technologies are appropiate for the realization of industrial process supervision applications (SCADA). That consist in the graphic representation of the different elements of the process with information in real time and historical evolution of the values of different variables of interest.

In this project, BeagleBone Black (BBB) minicomputers has been used to control some simple processes (temperature, PWM signals, etc.). The BBBs provides a JavaScript library who allows the use of web technologies, BoneScript.

This system has been implemented in a plant demanded to the Electronic Service of the University of La Laguna, in which the control action was required. For that, has been necessary to design and make electronic circuits to connect the system to the board.

When the control is executed, the system response is obtained, with that is been analyzed its dynamics.

ÍNDICE

1. Introducción

1.1 Objetivos

El cometido principal de este trabajo es el estudio y la implementación de la programación web en el ámbito industrial, analizar sus potencialidades y sus carencias, realizar pruebas a través de un microcontrolador para su final ejecución en un caso real.

Los recursos que se disponen y lo realista que puede llegar a ser el control en comparativa con un autómata vendrán delimitados por los recursos que se disponen en el controlador y en la planta, en dicho sentido se intentará optimizar los recursos para obtener un resultado lo más eficaz posible.

Por este motivo se realizarán pruebas para ver la respuesta del controlador a diferentes estímulos y en las que quedarán estipuladas de manera progresiva la implementación de las tecnologías Web.

Otro objetivo que se ha desarrollado a lo largo del proyecto ha sido el acondicionamiento electrónico necesario para la adquisición de señales de sensores y también sobre los actuadores del sistema a controlar, en este sentido se han elaborado varios circuitos, desde su diseño hasta su montaje.

Por último, se desea implementar un sistema de control tanto en lazo abierto como en lazo cerrado que sea interactivo y funcional desde la web.

1.2 Estructura del proyecto

Esta memoria recoge el proceso de implementación de un entorno SCADA en un navegador web, por ese motivo se ha estructurado de la siguiente forma:

- Marco teórico: donde se definen conceptos básicos para el contenido de esta memoria.
- Pruebas: proceso de aprendizaje progresivo para comprobar el correcto funcionamiento de la placa en un entorno web
- Diseño y montaje de la planta: explicación del diseño de la planta en la que se pretende implementar el sistema SCADA, adaptación electrónica necesaria para la adquisición de señales y el control de los actuadores.

- Control del sistema: Análisis de la respuesta del sistema mediante el control en lazo abierto, planteamiento y análisis del lazo cerrado con la implementación de un PI digital.

- Estructura de la página Web: registro de la página para el acceso al SCADA desde un entorno web, explicación de las funcionalidades que esta ofrece.

- Conclusión: valoración final sobre la realización del proyecto, cuestiones que se podrían mejorar y líneas de investigación que se abren a través del proyecto.

2. Marco teórico

2.1 Sistemas SCADA

SCADA (Supervisión, Control y Adquisición de Datos) es un sistema ampliamente empleado en el ámbito industrial, consiste en un interfaz hombre-máquina desde una plataforma software, con monitorización en tiempo real, del valor y estado de sensores y actuadores, además, posibilitando su control.

Figura 1, Sistema SCADA [1]

Este sistema se implementa para la supervisión de procesos industriales, el programa informático debe ser intuitivo para el operario, con datos fácilmente interpretables.

Otra característica es que el usuario tenga la posibilidad de ejecutar acciones de control, manipulando los actuadores directamente (lazo abierto) o asignando valores de consigna al sistema (lazo cerrado).

Para la adquisición de datos debemos acceder a una correcta lectura del sensor, la adecuación electrónica para dicha adquisición es campo de la instrumentación electrónica.

2.2 Control de sistemas

Un sistema dinámico se puede considerar como aquel que tiene una respuesta de salida en función de unas entradas determinadas. El control de sistemas trata de poder influir en el funcionamiento de un sistema dinámico con la finalidad de conseguir un dominio en las variables de salida para que estas alcancen un valor previamente prefijado(consigna).

Un sistema de control ideal debe de cumplir los siguientes requisitos:

- Ser robustos frente a las perturbaciones, pudiendo llegar al valor de consigna en el estacionario.
- Ser los más eficiente posibles, evitando comportamientos bruscos e irreales en la entrada que podría derivar en un uso irresponsable de los actuadores.

Los elementos que participan en un sistema de control son:

- Sensor: registra el valor de salida del sistema.
- Controlador: compara el valor de consigna con la salida y calcula la acción que deben tomar las variables de control para realizar la estrategia de control.
- Actuadores: las variables a controlar del sistema.

Existen dos estrategias principales para la implementación del control:

- Sistema en lazo abierto: Es cuando se actúa sobre las variables de entrada independientemente del valor de la salida.
- Sistema en lazo cerrado: Las variables de control se ajustan a través del controlador para llegar al nivel de consigna en la salida.

2.3 Tecnologías Web

Existen varios lenguajes dedicados al desarrollo web. En primer lugar, se debe hacer una diferenciación entre tecnologías del lado del cliente (front-end) y tecnologías que operan del lado del servidor (back-end).

Por las características de este proyecto se ha utilizado únicamente programación front-end (HTML, CSS y JavaScript). Actualmente es la tecnología más utilizada según StackOverflow.

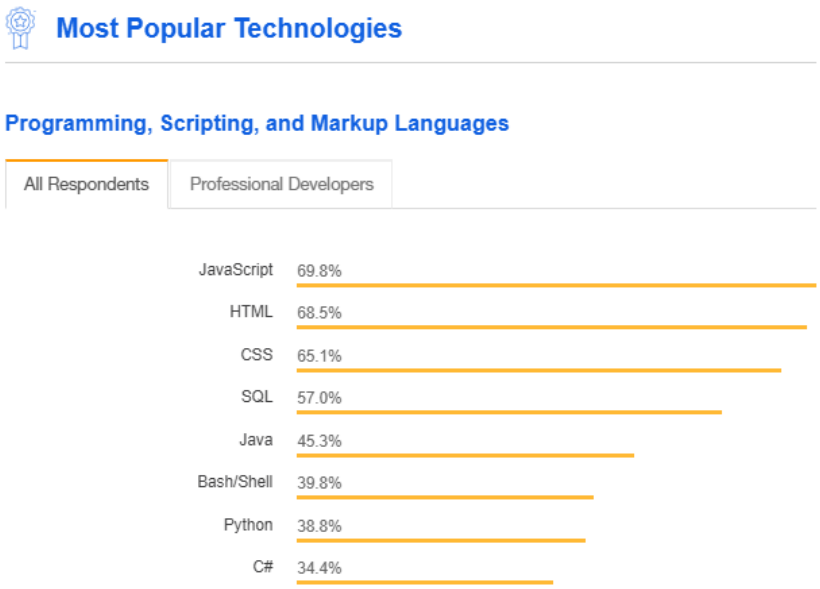


Figura 2, Tecnologías más utilizadas [2]

2.3.1 Lenguaje HTML

Se entiende como lenguaje HTML (HyperText Markup Language) a un conjunto de marcas o etiquetas que hacen referencias a información, es decir, HTML se define como lenguaje de marcado. Dichas etiquetas dotan de propiedades a la información, por ejemplo, la marca <img> sirve para idefinir una imagen, <button> un botón o <p> un párrafo de texto.

Figura 3, Ejecución de archivos en la Web [3]

Cuando accedemos a un navegador web e ingresamos una dirección ip o un dominio de nombre (DNS), estamos solicitando la información que se aloja en el documento HTML que tiene dicha dirección. El navegador es capaz de interpretar las marcas una a una y mostrarlas al usuario por pantalla.

2.3.2 Lenguaje CSS

Mientras con HTML sirve para definir el contenido de una página web, CSS (Cascading Styles Sheets) es utilizado para dar estilos visuales a dichos elementos, ejemplificando, desde un archivo CSS podemos instanciar a un elemento que ha sido definido en un documento HTML <img> a través de selectores de CSS, luego podremos definir una altura con el comando height y una anchura con width para redimensionarla.

Utilizando este lenguaje de estilos se pueden modificar colores, formas, márgenes, etc... Una de las funcionalidades principales de CSS es la de posicionar los elementos en la pantalla del usuario, incluso pudiendo hacerlo en función del tamaño que tenga dicha pantalla. Cuando hablamos de esta adaptación de tamaño es a lo que se denomina como Responsive Design (Figura 4) y en la programación Web es de vital importancia.

Figura 4, Responsive Design [4]

2.3.3 Lenguaje JavaScript

Habiendo definido anteriormente HTML como un lenguaje de marcas que dan contenido a una página web, CSS como un lenguaje de estilos que dota de cualidades visuales o estéticas a dicho contenido, JavaScript es un lenguaje de programación encargado de dar dinamismo a la página.

Es un lenguaje débilmente tipado, muy flexible, orientado a objetos y de lado del cliente (front-end). A diferencia de otros lenguajes no necesita ser compilado si no que se ejecuta directamente en el navegador, esto hace que sea una opción interesante si se quiere trabajar de una manera rápida e interactiva.

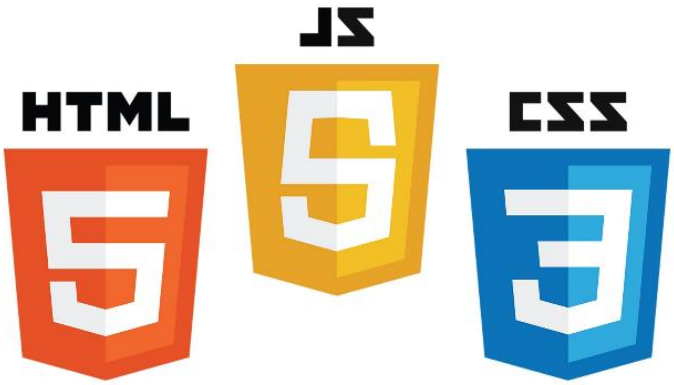


Figura 5, Tecnologías Front-end [5]

2.4 Microcontrolador BeagleBone Black (BBB)

BeagleBone Black es un MiniPc/microcontrolador de bajo coste y de código libre producido por Texas Instrument. Funciona con Debian, distribución de GNU/Linux. En la actualidad, cuenta con una gran comunidad desarrolladora, compitiendo con otros microcontroladores/miniPCs como son Arduino o Raspberry Pi.

Desde la propia página de BeagleBone definen a la BBB de esta manera. "Ha sido equipado con un conjunto mínimo de características para permitir al usuario experimentar la potencia del procesador y no se considera una plataforma de desarrollo completa, ya que muchos de las funciones e interfaces proporcionadas por el procesador no son accesibles desde BeagleBone Black mediante soporte onboard. No es un producto diseñado para hacer una función en particular. Es una base para la experimentación y el aprendizaje mediante la creación de tu propio software y hardware" [6].

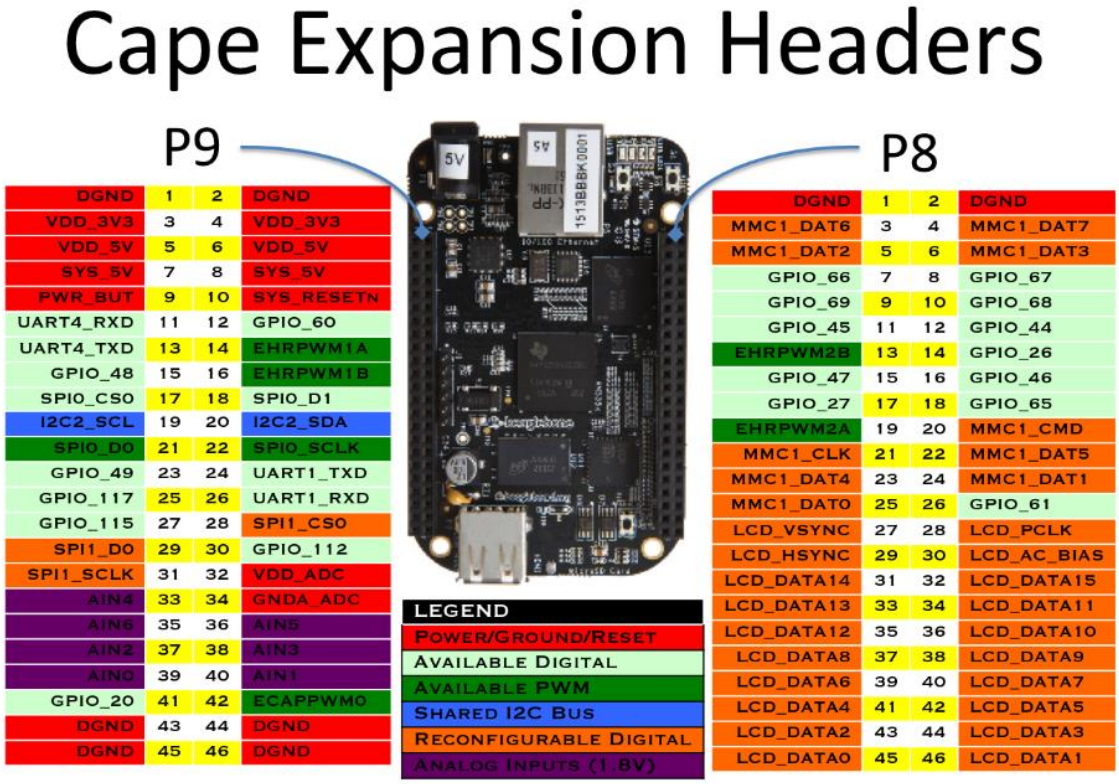


Figura 6, Configuración de pines en Beaglebone Black [7]

El punto fuerte respecto a su competencia es su funcionalidad, mientras que Raspberry Pi 3 puede ofrecer mejores especificaciones hardware que BBB, esta última otorga una gran cantidad de puertos, 92 para ser exactos, entradas analógicas, salidas PWM, comunicación I2C y SPI, etc.

También existen otras placas de la familia BeagleBone, como la BeagleBone Blue diseñada para sistemas robotizados, la PocketBeagle con un tamaño menor y más barata que la Black aunque más corta en características y la BeagleBone Black WiFi que reemplaza la conexión Ethernet de la Black original por un módulo con capacidad de comunicación Bluetooth y WiFi.

2.5 BoneScript

BoneScript es una librería optimizada para la familia BeagleBone de NodeJS (framework de JavaScript para back-end) que implementa muchas funciones parecidas a las de Arduino al navegador para que este pueda interpretarlas. Por tanto, esta librería posibilita la interacción entre el hardware y la web.

Algunas de las funciones de BoneScript que se han utilizado en este proyecto son:

Nombre	Parámetros	Descripción
pinmode()	(pin, [callback])	Configura un pin digital como entrada o como salida y opcionalmente se llama a una función (calback) cuando se complete la operación.
digitalWrite()	(pin, value, [callback])	Escribe HIGH o LOW en una salida o entrada digital y opcionalmente se llama a una función (calback) cuando se complete la operación.
digitalRead()	(pin, [callback])	Lee el valor de una entrada o salida digital devolviendo HIGH si es alta o LOW si es baja y opcionalmente se llama a una función (calback) cuando se complete la operación.
analogWrite()	(pin, value, [freq], [callback])	Escribe una señal PWM, value es el valor del ciclo de trabajo y freq el valor de la frecuencia (por defecto 2 kHz), opcionalmente se llama a una función (calback) cuando se complete la operación
analogRead()	(pin, [callback]	Lee el voltaje de una entrada analógica y opcionalmente se llama a una función (calback) cuando se complete la operación.

Tabla 1, Funciones de la librería BoneScript

2.6 Node-Red

Consiste en un entorno de programación basado en flujos para el IoT (Internet of Things), su versatilidad hace que sea muy útil a la hora de conectar diferentes dispositivos e interactuar entre ellos. Utiliza una interfaz gráfica con módulos entre los que se encuentran Arduino, BeagleBone, Raspberry Pi, etc.

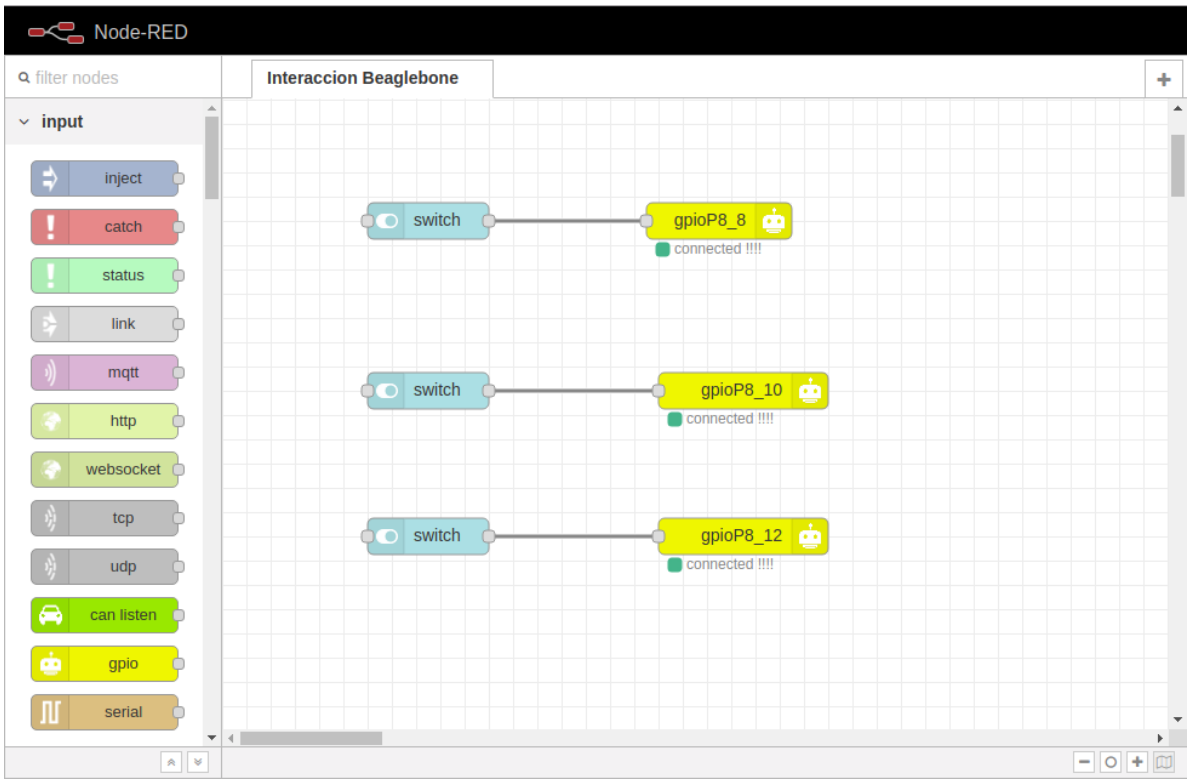


Figura 7, Entorno de programación Node-Red

2.7 Highcharts

Highcharts es una librería de JavaScript que utiliza imágenes vectoriales para la construcción de gráficos dinámicos. Ofrece una gran cantidad de funcionalidades como descargar los datos del gráfico a formato .CSV o .XLS, o exportar el gráfico a pdf o a una imagen. Sus gráficas son ampliamente personalizables tanto en el aspecto visual como en el manejo de datos.

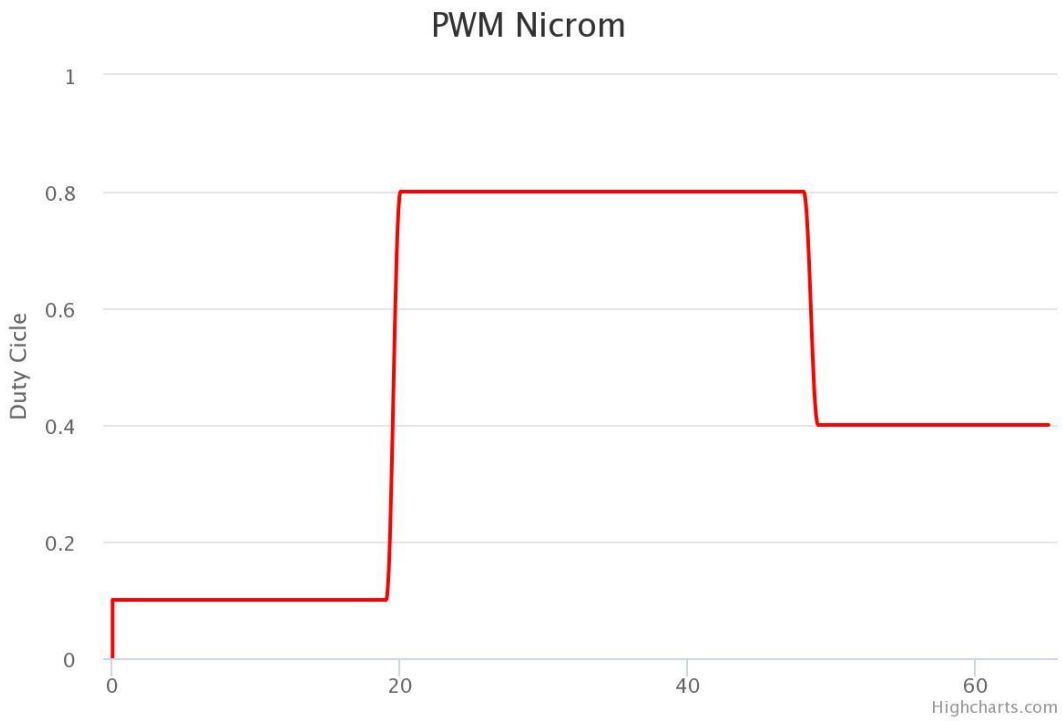


Figura 8, Gráfica en Highcharts

3. Pruebas

3.1 Introducción

El contenido de este capítulo recoge las pruebas que se han realizado para la Beaglebone Black, en ellas se han ido implementando de una manera progresiva el uso de las tecnologías web. Las pruebas son las siguientes:

- Prueba Semáforo: se probará la interacción con la placa, encendiendo unos diodos led desde botones que se han añadido al navegador.
- Prueba Gráfica: se probará la representación gráfica de una entrada analógica, en este caso un potenciómetro.
- Prueba Gráfica en tiempo real: se realizará la representación de una entrada analógica en función de la hora actual que será adquirida desde el propio navegador.
- Prueba Web: se insertará el gráfico de la "Prueba Gráfica" haciendo un diseño de una página web usando CSS.

3.2 Puesta en marcha de la BBB

En primer lugar, se debe comprobar que la BeagleBone Black tiene la última imagen de Debian. Utiliza una versión para IoT que podemos encontrar en la página oficial de Beaglebone.

Una vez iniciado el sistema operativo, si tenemos conectada la BBB vía USB se conectará a la dirección IP 198.168.7.2, si ingresamos la dirección nos redireccionará a la página principal de Beaglebone. Desde la propia página existen ejemplos desde los que se pueden comprobar la comunicación.

3.3 Prueba Semáforo

La finalidad de esta prueba es comprobar la comunicación con la placa, lo primero que se debe tener en cuenta es que desde la propia librería BoneScript nos asocia una dirección IP con los ficheros que internamente posee la BBB.

Figura 9, Prueba Semáforo

Para comprobar el correcto funcionamiento de la librería se ha montado un circuito de prueba, se trata de tres diodos LED emulando un semáforo, se comprueba la función digitalWrite() desde un entorno Web.

```
function botonVerde(){
  b.digitalWrite(ledPin1, b.HIGH);
  b.digitalWrite(ledPin2, b.LOW);
  b.digitalWrite(ledPin3, b.LOW);}
```

Código 1, Función que enciende led verde

El código y el circuito de la prueba están disponibles en la sección de anexos.

3.4 Prueba Gráfica

En un sistema SCADA se debe conseguir una correcta representación de los datos, para ello se necesita un entorno visual que muestre los datos de las señales monitorizadas. En esta prueba se pretende recoger la señal de una entrada analógica (en este caso un potenciómetro) para su posterior ingreso en un gráfico de la librería Highcharts de JavaScript.

Figura 10, Prueba Gráfica

Con esto, conseguimos un gráfico de carácter dinámico que se actualizará cada 200 ms, además si se inspecciona el navegador y se mira la consola, retornará el valor concreto de la entrada analógica en el mismo intervalo de tiempo.

```
function potRead(){
  b.analogRead(potPin, show);
  setTimeout(potRead, 200);
}
```

Código 2, Función que hace la lectura

El código y el circuito de la prueba están disponibles en la sección de anexos.

3.5 Prueba Gráfica en Tiempo Real

En la anterior prueba se logró un gráfico dinámico, sin embargo, en un sistema SCADA interesaría poder registrar esos cambios en función de otras variables, por ejemplo, el tiempo.

Una de las ventajas de la programación Web es poder recoger información desde el propio navegador, es posible registrar la hora actual a través del método .getTime(), en esta prueba se recoge la lectura del sensor y se almacena en un vector, dicho vector se pasa como parámetro al gráfico de Highcharts para que lo asigne a la hora actual registrada por el ordenador, de esta forma se consigue

observar la evolución del sensor, en este caso se trata de un LDR con salida analógica.

```
setInterval(function () {
        var x = (new Date()).getTime(), //Aquí registramos la hora actual
          y = resultados[49];// Pasamos la última posición del vector (la más
actualizada)
        series.addPoint([x, y], true, true);
      }, 1000);
```

Código 3, Función que añade los datos

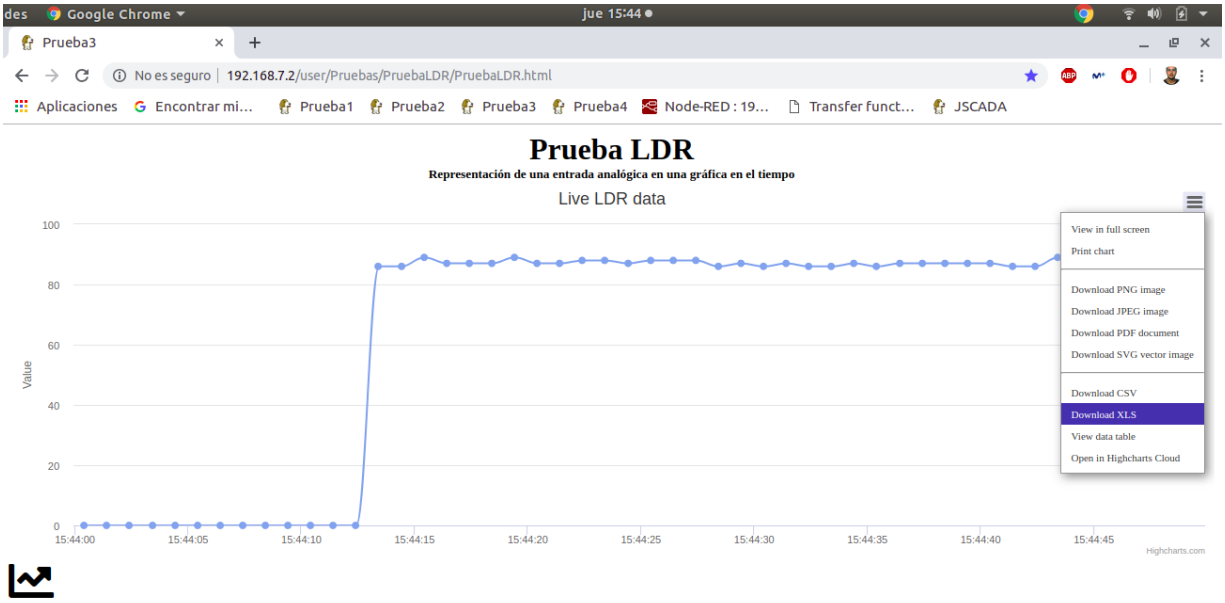


Figura 10, Prueba gráfica en tiempo real

La librería Highcharts ofrece varias funcionalidades, una de ellas es 'exporting' y en esta práctica ha sido habilitada, al activarse, muestra una pestaña en sus gráficos para poder exportar los datos a formato .XLS o .CSV, de esta forma se consigue almacenar los datos de manera permanente para luego ser manipulados en otras plataformas.

```
exporting: {
  enabled: true
},
```

Código 4, Exportar los datos del gráfico

El código y el circuito de la prueba están disponibles en la sección anexos.

3.6 Prueba Web

Esta prueba trata de integrar la gráfica de la Prueba 2 en un entorno Web con otros elementos, como imágenes o texto, para la correcta disposición de los elementos ha sido empleado Flexbox, un modelo de distribución del lenguaje CSS que consiste en definir un elemento 'padre' como contenedor y unos elementos 'hijos' como ítems que heredan las propiedades que se definen en el padre, si se define una altura en el padre, sus hijos se ajustarán a esa altura, si fijamos un ancho para uno de los hijos, el resto se redimensionan para ocupar el espacio del ancho del padre contenedor.

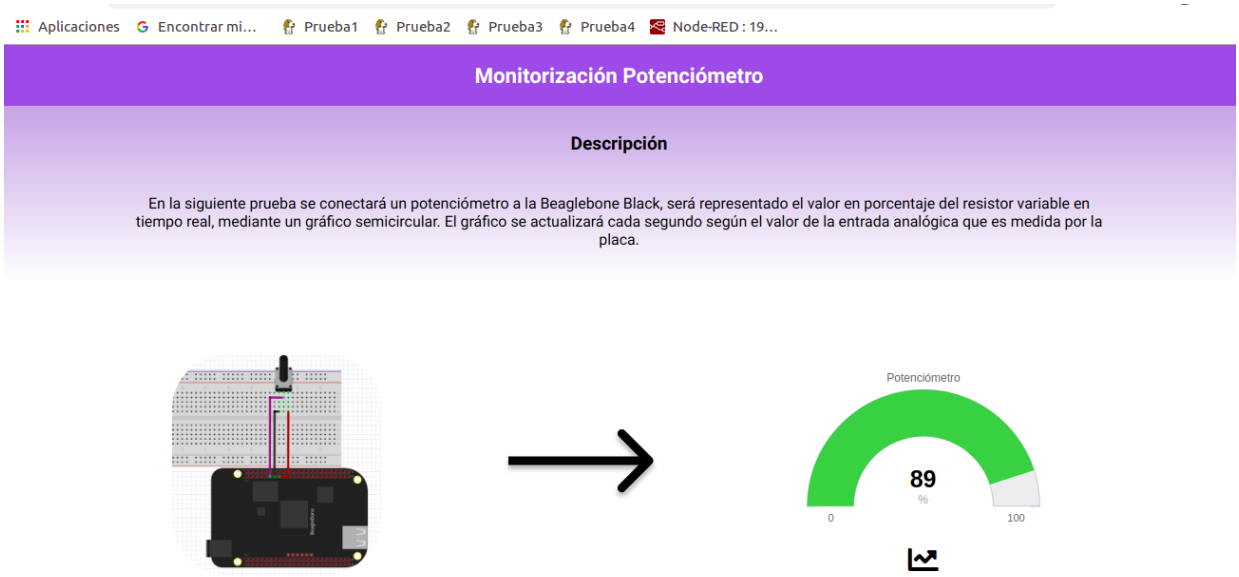


Figura 11, Prueba Web

De esta manera y utilizando unidades de ancho y alto de pantalla se puede lograr un diseño Responsive que se ajuste a cualquier dispositivo.

```
.elementos{
  display: flex;
  flex-direction: row;
  height: 60.5vh;
  justify-content: center;
}
```

Código 5, Utilizando Flexbox

El código y el circuito de la prueba están disponibles en la sección anexos.

4. Diseño y montaje de la planta

4.1 Descripción de la planta

Este sistema es un diseño que ha sido replicado de una planta orientada al ámbito de control que se encuentra en el laboratorio del Departamento de Ingeniería Informática y de Sistemas situado en el edificio de las Secciones de Física y Matemáticas de la Facultad de Ciencias. El funcionamiento consiste en propulsar y calentar aire a través de un tubo que tiene un sensor de temperatura, esta planta recoge los tiempos de retardo producidos en la lectura del sensor al final del tubo ya que está a cierta distancia del actuador. Las modificaciones en el actuador tardan tiempo en llegar al sensor.

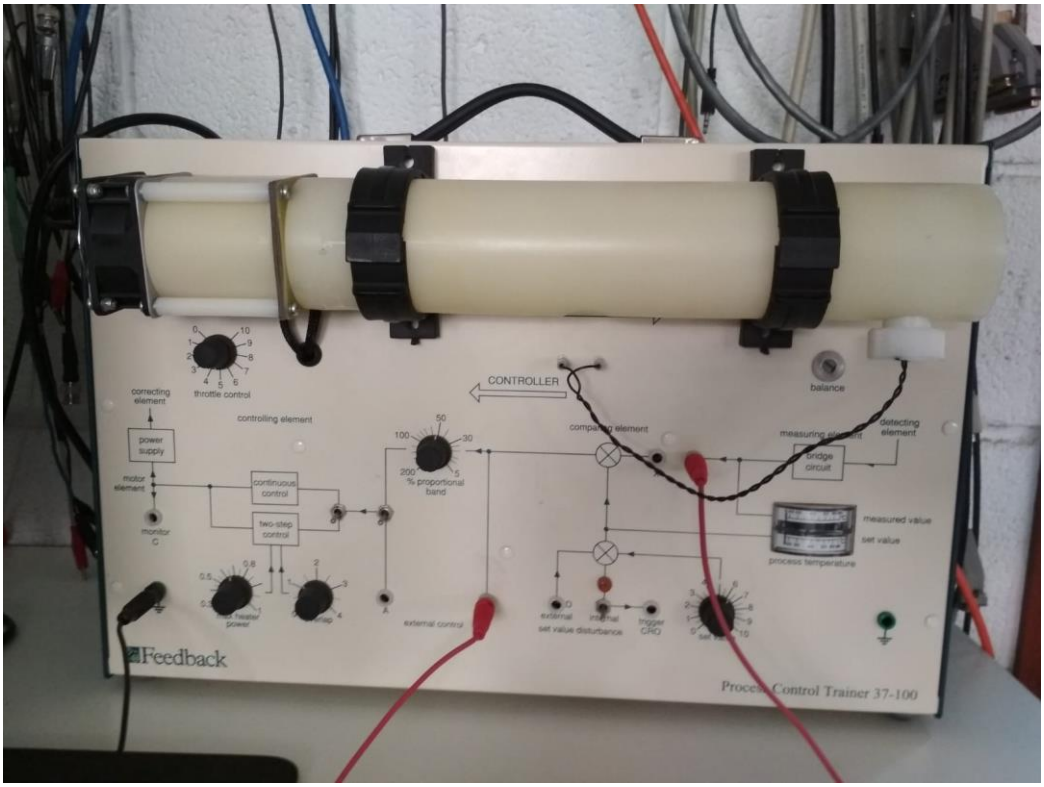


Figura 12, Estación original de control

Como réplica, el Servicio de Electrónica del SEGAI ha diseñado un sistema similar usando el microcontrolador Arduino en una PCB. La planta replicada consta de dos actuadores, un ventilador que impulsa el aire y una resistencia basada en nicrom que calienta el mismo. Se ha utilizado una sonda Pt100 para la lectura de la temperatura, todo esto integrado en un tubo de plástico que conduce el aire.

4.2 Funcionamiento general de la planta replicada

La lectura del sensor y el funcionamiento de los actuadores es coordinada a través de un microcontrolador Arduino NANO.

Para el sensado, el Servicio de Electrónica ha acondicionado la señal a través de un puente de Wheastone acoplando a su salida un amplificador de instrumentación tipo INA120, a la salida del INA un filtro pasa-bajas para eliminar ruido y finalmente se conecta a un pin analógico del Arduino que recogerá la medida.

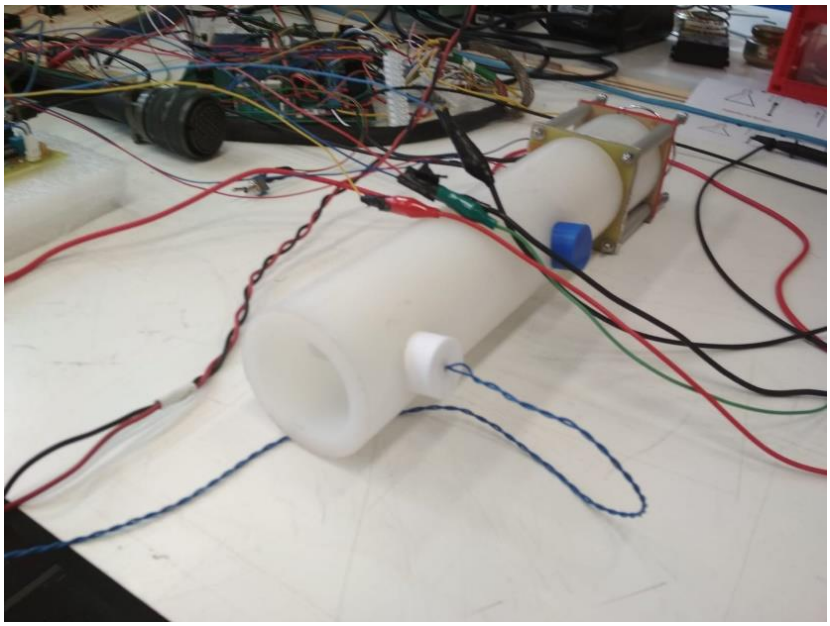


Figura 13, Planta replicada por el Servicio Electrónica

El control de los actuadores se ejecuta con señales PWM, modulando el ciclo de trabajo (duty cicle) de los pulsos, sin embargo, esta señal no es apta para la implementación directa tanto en el ventilador como en el nicrom, pues trabajan en tensión continua. Por este motivo, se diseñaron dos conversores frecuencia-voltaje para la señal de alimentación al nicrom y al ventilador, el resultado es que se tiene un control del voltaje de alimentación con la variación del duty cicle del pulso PWM.

Por cuestiones de diseño, en la planta se ha limitado el valor del voltaje con el que se alimenta al nicrom, al 80% no sigue aumentando, así mismo, el ventilador no podrá bajar de un 40%. Así se evitan sobrecalentamientos en el sistema. Esas especificaciones ya están contempladas en el propio Arduino.

El Arduino esta implementado de manera que pueda ser alimentado por voltaje o a través de la conexión serial con un puerto USB.

4.3 Adaptaciones necesarias a la BBB.

El diseño inicial de esta planta ha sido implementado desde el microcontrolador Arduino, que difiere en algunos aspectos en la Beaglebone Black.

El voltaje máximo permitido en las entradas analógicas es de 1,8 V, mientras que en Arduino son 5 V.

El voltaje máximo de salida de los pulsos PWM es de 3,3V, mientras que en Arduino son de 5 V.

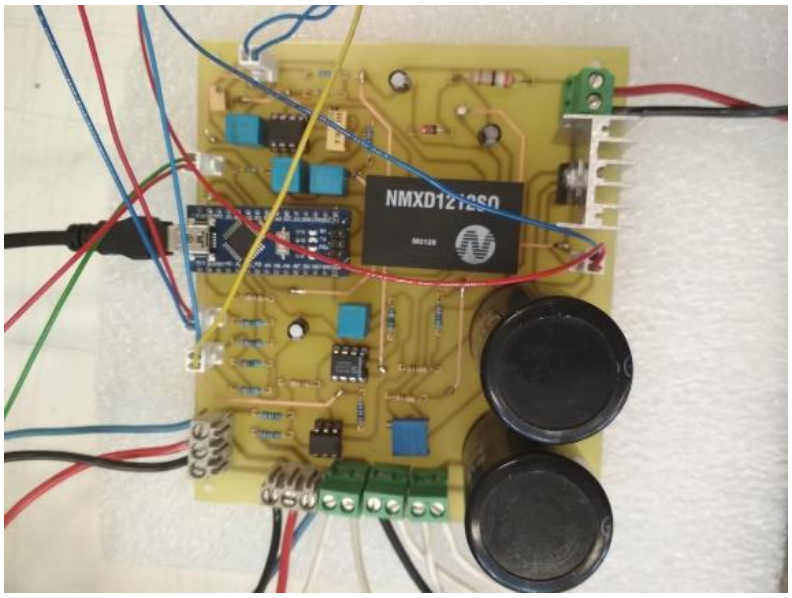


Figura 14, PCB con Arduino NANO

Por este motivo se deberán recalcular con estos parámetros y replicar los circuitos con los que esta implementado el Arduino, con el fin de hacer un control por voltaje desde la BBB al microcontrolador.

Otros aspectos a tener en cuenta si se quisiera integrar el sistema a la BBB es que el fabricante recomienda que una vez que se conecta la placa, se debe esperar a que se cargue el sistema operativo antes de conectar ninguna entrada analógica, de lo contrario el sistema podría dañarse. También sería recomendable que se pudieran acceder fácilmente a los dos botones que trae integrada la BBB, uno de reset y otro de apagado/encendido.

4.4 Adquisición de la señal de temperatura

4.4.1 Instrumentación electrónica para la RTD

El sensor encargado de la adquisición de señal es una Pt100, un transductor que varía su resistencia en función de la temperatura (RTD). Sin embargo, sería más útil si esta variación en la resistividad produjera una diferencia de potencial con una resistencia referencia que tuviera un valor fijo. Este es el principio de funcionamiento de un puente de Wheastone, como el que se muestra en la figura 15.

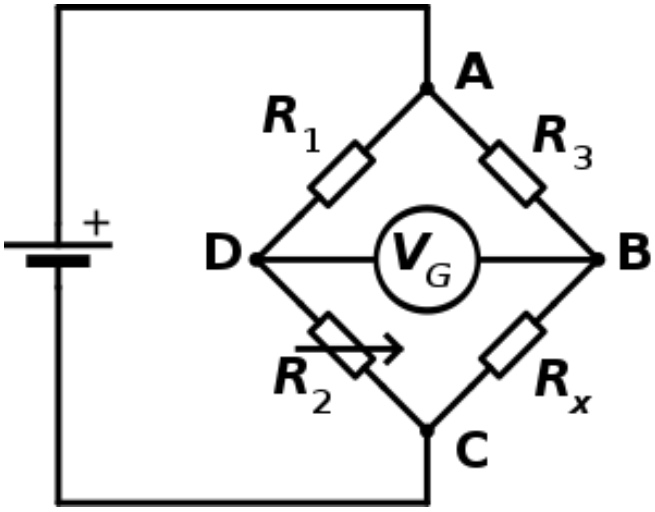


Figura 15, Puente de Wheastone [8]

De esta manera, al estar ambas sometidas a la misma tensión, la resistencia Rx, en la rama opuesta al sensor resistivo fijaría la resistencia de referencia y cualquier mínima variación de la Pt100 generaría una diferencia de potencial entre el punto D y B. Además, para que la proporción resistencia-voltaje sea lineal las resistencias del puente de Wheastone deben coincidir de la siguiente forma:

$$\frac{R_2}{R_1} = \frac{R_x}{R_3}$$

La sensibilidad a la salida del puente se tiene que es de 0,45 mV/ºC, mientras que el margen de entrada de la BeagleBone Black es de 0 a 1,8 V, además se considera que la temperatura máxima a la que trabajara la planta es de 75ºC. Teniendo estas especificaciones en cuenta se efectúan los siguientes cálculos:

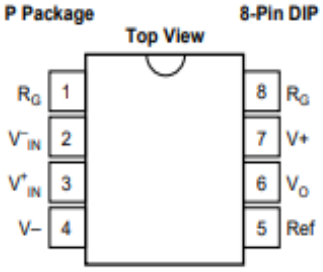
$$75℃ \rightarrow 1{,}8V$$

$$1℃ \rightarrow x$$

Entonces, según los cálculos, la sensibilidad se deberá aumentar a 24 mv/ºC para aprovechar el margen de entrada de la placa, por tanto, la ganancia G que deberá tener el amplificador de instrumentación INA114AP es la siguiente:

$$G = \frac{S}{Vs} = \frac{24\ mV}{0{,}45\ mV} = 53,\hat{3}$$

En un amplificador de instrumentación, la ganancia se ajusta por la resistencia Rg, en el caso del INA114AP esta resistencia se coloca entre los pines 1 y 8 del integrado, la relación entre ganancia y Rg viene dada por:



$$G = 1 + \frac{50\ k\Omega}{R_G}$$

Figura 16, Pineado y ganancia del INA114AP [9]

Con los datos anteriores y despejando la fórmula se tiene que Rg = 955,41 Ω.

Además, se desea eliminar el ruido que puede ser producido por varios motivos como, por ejemplo, un mal contacto del cableado. Un filtro pasa-bajas permite el paso de frecuencias bajas e impide las frecuencias altas, el ruido suele tener frecuencias muy altas por lo que es ideal para estos casos, si obtenemos mucho ruido en el pin analógico de la BeagleBone influirá en la medida y complicará mucho la implementación del control, la frecuencia de corte del filtro viene dada por la siguiente fórmula:

$$f_c = \frac{1}{2\pi RC}$$

El cálculo de este filtro se ha hecho de una manera experimental, asignando un valor fijo al condensador de 100 nF y variando la resistencia. Se comprobó la salida en un osciloscopio para ver la señal filtrada y también que la caída de voltaje que se produce por el filtro no fuera significativa.

El esquema eléctrico completo de la adquisición de temperatura:

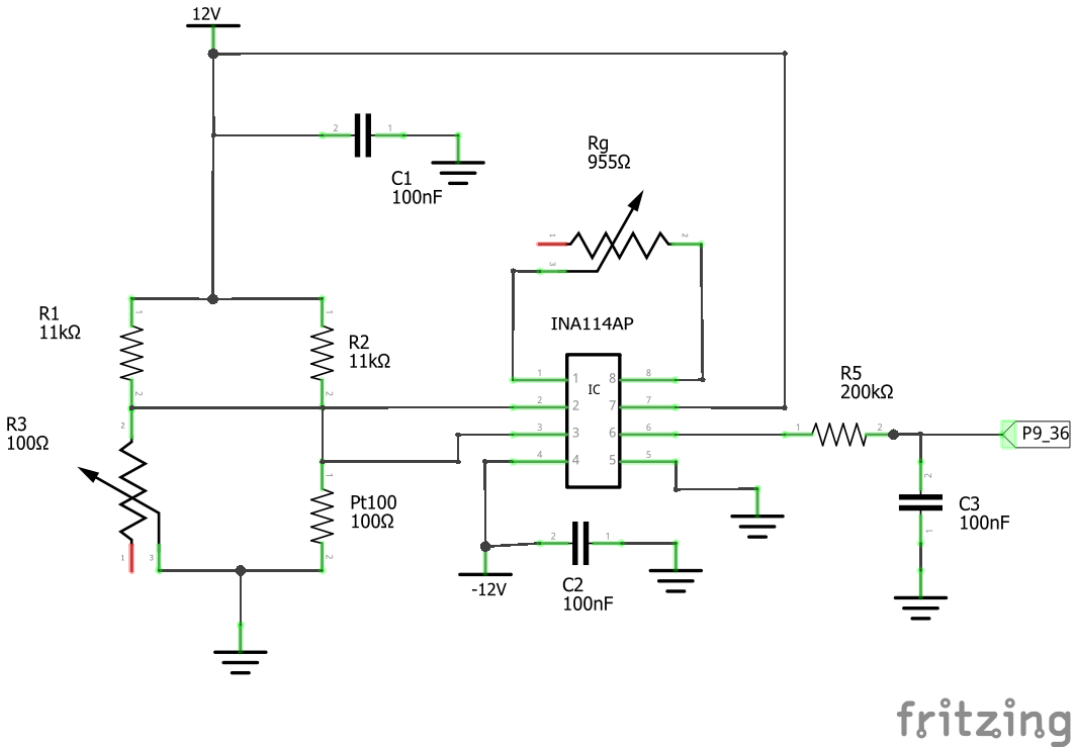


Figura 17, Circuito de adquisición de señal

Leyenda	Categoría	Valor	Función
R1	Resistencia	11KΩ	Puente de Wheastone
R2	Resistencia	11KΩ	Puente de Wheastone
R3	Potenciómetro	100 Ω	Puente de Wheastone
Pt100	RTD	100 + aT Ω	Puente de Wheastone
INA114AP	Integrado	“”	Amplificador de instrumentación
Rg	Potenciómetro	955 Ω	Amplificador de instrumentación
C1	Condensador	100 nF	Desacoplo
C2	Condensador	100 nF	Desacoplo
R5	Resistencia	200 kΩ	Filtro Pasa-bajas
C3	Condensador	100 nF	Filtro Pasa-bajas
P9_36	BeagleBone	“”	Entrada analógica

Tabla 2, Circuito de adquisición de señal

4.4.2 Calibración de los componentes

A continuación, se muestran imágenes de las metodologías empleadas para la calibración:

Para ajustar la ganancia del amplificador de instrumentación INA114AP hemos empleado un calibrador Yokogawa CA100 que emula la salida del puente de Wheastone, se establece el potenciómetro para que el voltaje a la salida del INA sea el calculado. Como se observa en la Figura 18, tanto el osciloscopio como el multímetro de banco muestran la salida del integrado:

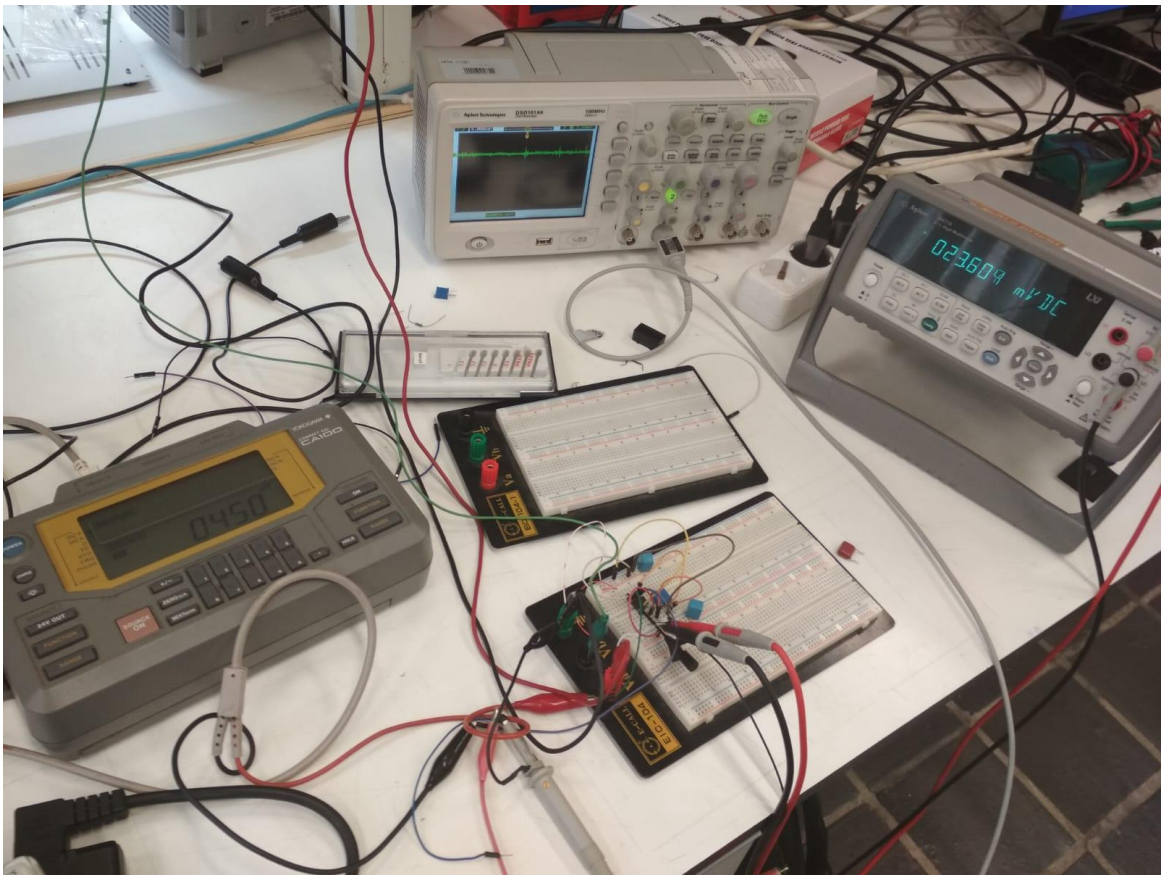


Figura 18, Calibración de la ganancia del INA

Llegados a este punto se observó que variando el voltaje del calibrador la ganancia del INA114AP no se mantenía constante, se hicieron varias comprobaciones de la correcta conexión del circuito, aun así, el integrado tenía el mismo comportamiento, incluso cambiando el amplificador. Así que se registró la variación de la ganancia con los voltajes que se deberían obtener a diferentes rangos de temperatura y se obtuvo esta función resultante (Figura 19):

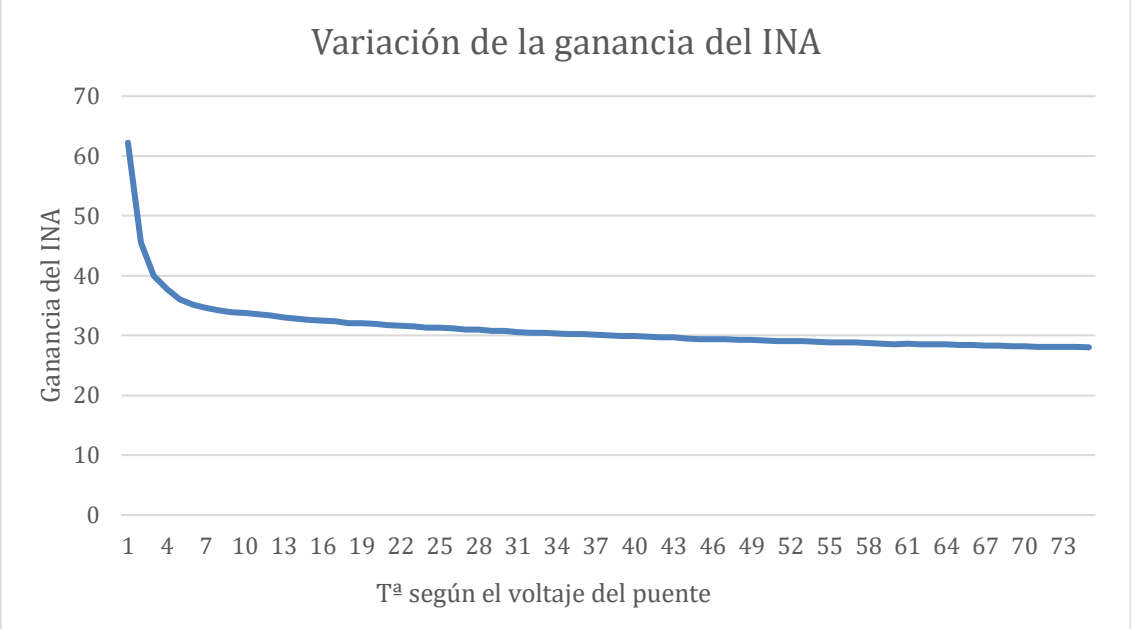


Figura 19, Variación de la ganancia del INA

En este punto se decidió emplear una media de la ganancia del INA ya que en los rangos de temperatura en los que se iba a utilizar la pt100, el valor de la ganancia del INA se mantenía prácticamente constante, finalmente se estableció que la ganancia fuera de 29.03.

El siguiente ajuste se realiza acoplando el puente de Wheastone al amplificador. Con la ayuda de un termopar conectado a un multímetro podemos comprobar si la lectura de datos es la correcta (Figura 20).

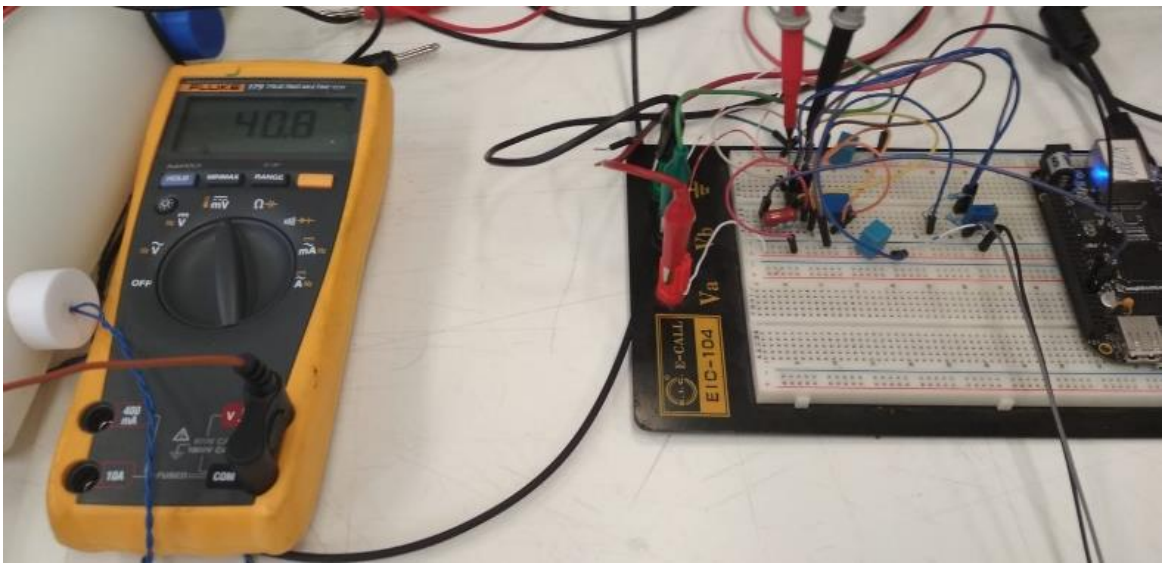


Figura 20, Calibración con multímetro y termopar

Observamos el valor recogido por la BBB en la consola del navegador, coincidiendo con el valor del multímetro (Figura 21).

4.4.3 Código para la adquisición de señal

Una vez acabada la instrumentación electrónica necesaria, debemos implementar un código software capaz de interpretar los datos.

Con la ayuda de la librería BoneScript, utilizamos la función b.analogRead() que accederá a la lectura del voltaje de salida del INA(INAPin), el segundo parámetro es una función que se ejecuta cada vez que se realiza la lectura (50 ms).

```
function INARead(){ //Realiza la lectura del sensor cada 50 ms
  b.analogRead(INAPin, Vo);
  setTimeout(INARead, 50)
}
```

Código 6, Lectura a la salida del INA

La siguiente función Vo(), almacena en la variable sumatorio las medidas de INARead() hasta llegar a 20, donde se le pasa por parámetro a la función show() y la ejecuta. Por tanto, la función show se ejecuta cada 1 s.

```
function Vo(x){ //Almacena valores del sensor en sumatorio
  if (count == 20){ //Al llegar a 20 significa que ha pasado 1 seg y pasa las 20
muestras sumadas
    count = 0;
    show(sumatorio);
    sumatorio = 0;
  }else{
    count++;
    sumatorio += x.value*1.8; // 1.8 por ser el voltaje de la BBB
  }
```

Código 7, Recogiendo datos de la salida del INA

La función show() muestra los valores de Vo (salida del INA), Vs (salida del puente de Wheastone) y T (temperatura) cada segundo por la consola del navegador.

- Vo: Tenemos que hacer la media a sumatorio que tiene almacenada 20 medidas.

- Vs: Se obtiene dividiendo Vo entre la ganancia.

- T: Utilizamos la fórmula de la temperatura que viene dada por la Pt100

$$T = \frac{Vs(k+1)^2}{Vin * k * \alpha}$$

```
function show(sumatorio){  //Muestra los valores de salida de Vo (Salida del INA)
                           //Vs(Salida del puente de Wheastone)
                           //T Temperatura

    let Vo = (sumatorio / 20); //Media de las medidas recogidas por el sensor (20
por seg)
    let Vs;

       Vs = Vo/29.0296296296296; //Media de la ganancia del INA (Amplificador de
Instrumentación)

      T = ((Vs*(100 + 1)*(100 + 1))/(12*100*0.00385));
      //T = (VS*(k+1)*(k+1))/(Vin*k*alpha);


      console.log("Vo = " + Vo.toPrecision(4) + " V, Vs = " +
(1000*Vs).toPrecision(4) +
      " mV, T = " + T.toPrecision(3) + " ºC");

  }
```

Código 8, Impresión de las variables de interés por consola

Por último se utiliza console.log() para visualizar los datos en la consola del navegador, aquí está el resultado:

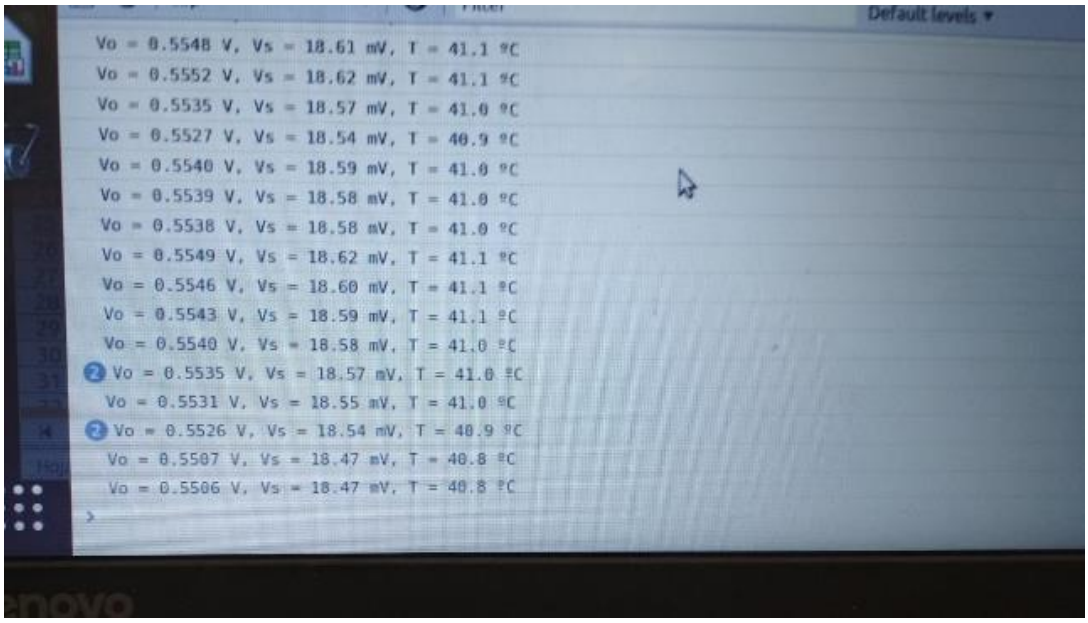


Figura 21, Consola del navegador

4.5 Señal de control por voltaje para el Nicrom y el Ventilador

4.5.1 Instrumentación electrónica para los actuadores

La señal PWM (pulse-width-modulation) se trata de una señal digital, periódica en el tiempo, una de sus características más relevantes es su ciclo de trabajo (duty cicle) que es el factor que indica cuanto está el pulso en su valor alto. Desde la BeagleBone Black podemos regular este factor, esto hace de esta señal ideal para la implementación del control.

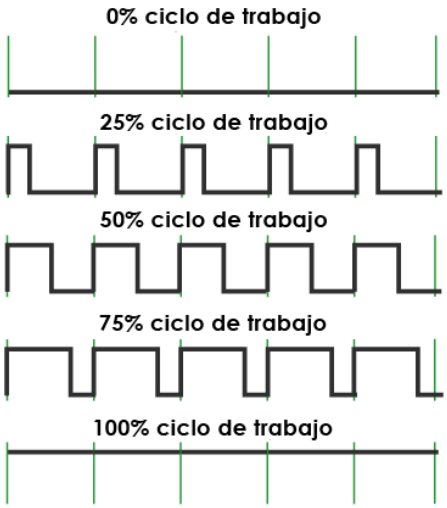


Figura 22, Duty Cicle de un PWM [10]

Sin embargo, esta señal no puede ser enviada directamente al Arduino pues sería interpretado al ventilador como constante arranque-parada, una solución eficiente sería la implementación de un circuito conversor frecuencia-voltaje.

Para ello se hace uso del integrado LM358N [11], un amplificador operacional de dos canales con un compensador de frecuencia interno. Utilizando sólo uno de los canales para conseguir la configuración deseada.

Antes de enviar la señal cuadrada al operacional se pasa por un filtro pasa-bajas para limitar el paso del ruido, que como vemos en la Figura 23 es importante:

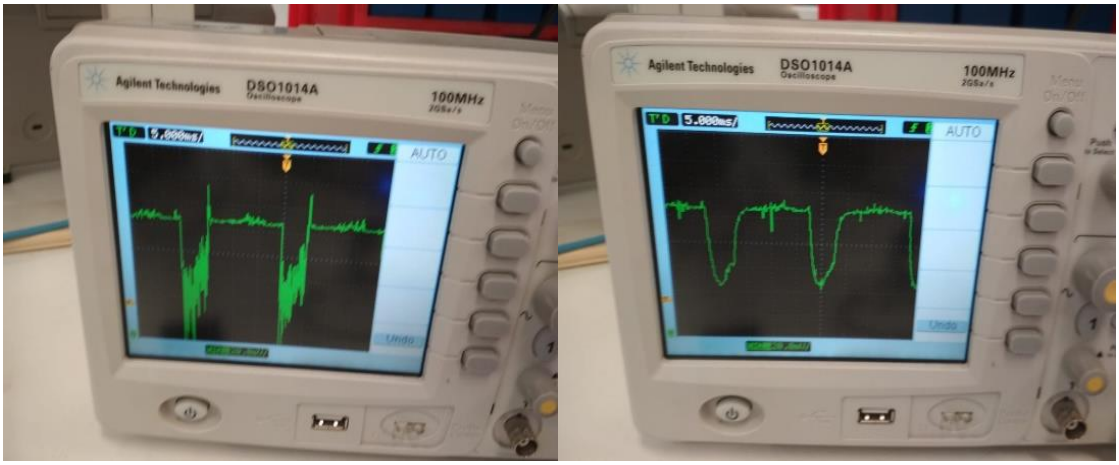


Figura 23, Ruido SIN/CON filtro pasabajas

Por lo que el circuito para el control por voltaje para el Nicrom al Arduino es el siguiente (Figura 24):

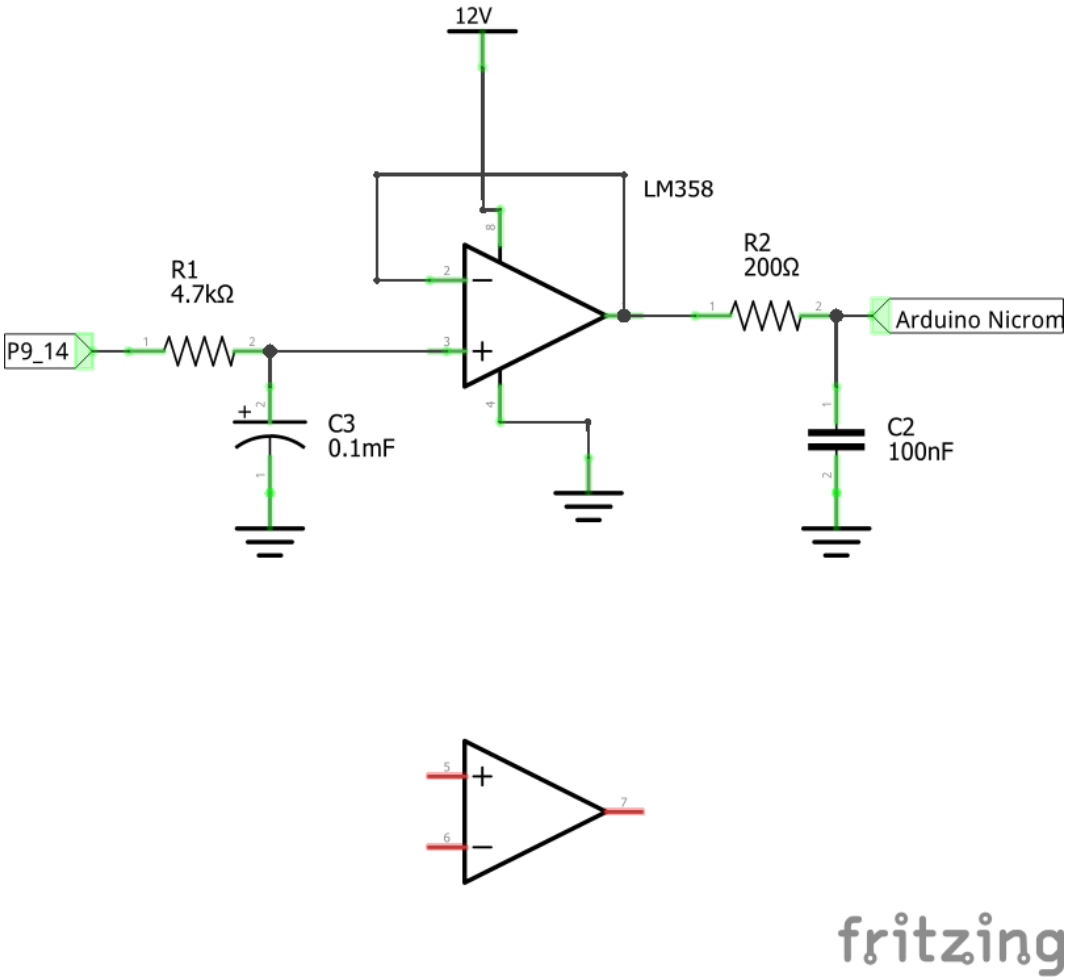


Figura 24, Circuito para el control por voltaje del Nicrom

Leyenda	Categoría	Valor	Función
R1	Resistencia	4,7K	Filtro Pasa-bajas Entrada
C3	Condensador	100µ	Filtro Pasa-bajas Entrada
LM358	Integrado	""	Conversión Frecuencia-voltaje
R2	Resistencia	200	Filtro Pasa-bajas Salida
C3	Condensador	100n	Filtro Pasa-bajas Salida
P9_14	BeagleBone	""	Señal PWM Nicrom

Tabla 3, Circuito para el control por voltaje del Nicrom

Mientras que el circuito para el control por voltaje para el ventilador al Arduino (Figura 25):

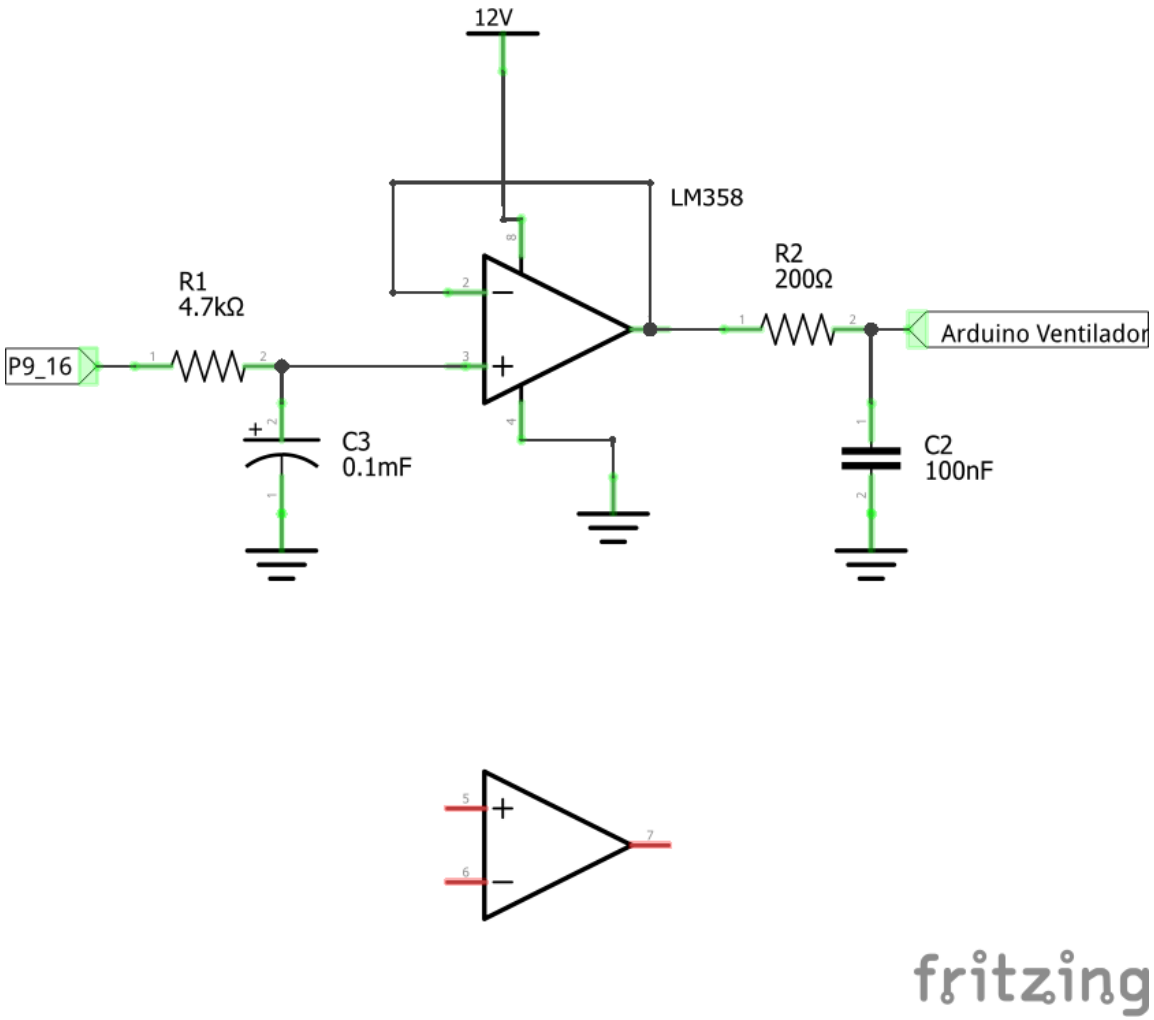


Figura 25, Circuito para el control por voltaje del Ventilador

Leyenda	Categoría	Valor	Función
R1	Resistencia	4,7K	Filtro Pasa-bajas Entrada
C3	Condensador	100µ	Filtro Pasa-bajas Entrada
LM358	Integrado	""	Conversión Frecuencia-voltaje
R2	Resistencia	200	Filtro Pasa-bajas Salida
C3	Condensador	100n	Filtro Pasa-bajas Salida
P9_16	BeagleBone	""	Señal PWM Ventilador

Tabla 4, Circuito para el control por voltaje del Ventilador

4.5.2 Código para el control por voltaje

Teniendo ya configurado el hardware para el control sobre los actuadores, necesitamos un programa capaz de poder enviar las señales correctamente.

Usando la librería BoneScript, se dispone de la función analogWrite() responsable del envío de la señal PWM, esta requiere de tres parámetros, el pin de donde se enviará la señal, el ciclo de trabajo y la frecuencia de la señal cuadrada (Que tiene el valor de 490 Hz por el diseño del Arduino, que emplea el mismo valor).

```
function PWMWrite(){ //Envía las señales PWM al Ventilador y al Nicrom
  b.analogWrite(PWMPin, FanVoltage, 490);
  b.analogWrite(PWM2Pin, NicromVoltage, 490);
  if(inicio == 0){
      INARead();
      inicio = 1;
      }
      return(inicio);
}
```

Código 9, Enviando señales PWM

Esta función además es la que inicia el programa con la lectura del sensor mediante INARead(), ya que está asociada a un icono del html. De modo que se ejecute al hacer click en el icono. La variable inicio se utiliza para que si la función se ejecuta varias veces al hacer click o desde otra función no se ejecuten varias lecturas del sensor de una manera asíncrona.

```
<i class="fas fa-chart-line" onclick = "PWMWrite()"></i>
```

Código 10, Icono en html (importado de FontAwesome)

Lo ideal sería que el usuario pudiera introducir el parámetro de duty cicle para modular el voltaje, usando la etiqueta <input> de HTML e instanciándola en JavaScript se puede conseguir.

El duty cicle según la librería BoneScript varía entre 0 y 1, por tanto, se limita el parámetro a este rango. Entonces, si ejecutamos la función PWMWrite() se envía la señal actualizada a los actuadores.

```
function actualiza(){//Actualiza los valores del ventilador y nicrom
  let a = document.querySelector(".Fan .faninput").value;
  if((a <= 1)&&(a >= 0)){
    FanVoltage = a;
  }
  let k =  document.querySelector(".Nicrom .nicrominput").value;
  if((k <= 1)&&(k >= 0)){
    NicromVoltage = k;
  }
  PWMWrite();
}
```

Código 11, Recogiendo datos introducidos por usuario

Llegado a este punto ya es posible la implementación del lazo abierto en el sistema. Por tanto, se puede conseguir la respuesta del sistema para una entrada escalón.

5. Control del sistema

5.1 Obtención de datos experimentales

Acabado el apartado anterior ya tenemos el hardware necesario para la implementación del control, sin embargo, necesitamos de un programa software que interprete los resultados y los almacene de tal manera que se pueda conseguir el control digital en lazo cerrado.

Como las tecnologías que se utilizan son de front-end, nuestra web está limitada al uso de bases de datos que corresponden al back-end, la solución para el registro de los datos ha sido almacenarlos en un vector de 500 posiciones que pasaremos a la librería Highcharts para la construcción del gráfico, de una forma similar a la Práctica 3: Prueba Gráfica en tiempo real aprovechando que esta librería permite la descarga de sus datos.

```
setInterval(function () {
            var x = tiempoV[499],// Pasamos la última posición del vector (la más
actualizada)
              y = parseFloat(resultadosTemperatura[499]);
            series.addPoint([x, y], true, true);
          }, 1000);
```

Código 12, Introduciendo datos en el gráfico

El vector resultadosTemperatura[] es el que registra los resultados de la temperatura que es la salida del sistema. Se repite el mismo proceso para el voltaje del nicrom y el voltaje del ventilador (compartiendo la misma línea temporal) que serían las dos entradas del sistema.

Cargando el programa de la sección 6.2.1 en el navegador podremos visualizar e interactuar con las entradas y salidas, con la posibilidad de descargar los datos en sus respectivos gráficos.

	A	B	C
1	Tiempo	Temperatura	PWMFAN
2	991	48,61370927	0.4
3	992	48,74787774	0.4
4	993	48,72634453	0.4
5	994	48,69818572	0.4
6	995	48,8588566	0.4
7	996	48,76112895	0.4
8	997	48,75615975	0.4
9	998	48,6600885	0.4
10	999	48,91186143	0.4
11	1000	48,65346289	0.4
12	1001	48,79425697	0.4

Figura 26, Datos obtenidos desde el gráfico

5.2 Cálculo de la función de transferencia del sistema

Con los datos obtenidos en la sección anterior del control en lazo abierto del sistema podemos analizar la respuesta del sistema ante una entrada de tipo escalón. Teniendo en cuenta que el sistema consta de dos entradas (nicrom y ventilador) se pueden plantear funciones de transferencia en función a ambos actuadores.

La metodología que se ha empleado ha sido la siguiente: para un valor fijo en uno de los actuadores, variar el otro para analizar la respuesta. A través de un análisis numérico de los datos experimentales se pretende calcular la función de transferencia del sistema, para ello utilizamos la herramienta Excel.

En primer lugar, se eligen los datos en el instante en el que se produce la entrada escalón (Tiempo exp, T exp) Figura 27.

	A	B	C	D	E	
1	Tiempo	Temperatura (C°)	Nicrom (%)	Tiempo exp	T exp	T
2	1	40,58121159	30	1	41,1228293	
3	2	40,64139134	30	2	41,1646208	
4	3	40,71828769	30	3	41,2632488	
5	4	40,71995935	30	4	41,4956095	
6	5	40,70992939	30	5	41,8800912	
7	6	40,78348242	30	6	42,3966341	
8	7	40,87040873	30	7	43,0235065	
9	8	40,85369213	30	8	43,6537222	
10	9	40,90551358	30	9	44,2505047	
11	10	40,95900669	30	10	44,9258552	
12	11	40,98909657	30	11	45,5226378	
13	12	40,99243989	30	12	46,1561968	
14	13	40,99578321	30	13	46,8031291	
15	14	41,02420142	30	14	47,4266582	

Figura 27, Datos del experimento

A continuación, si restamos el último valor de temperatura del experimento suponiendo que llega al estacionario frente al primero y dividimos por el cambio en el escalón obtenemos la ganancia K del sistema.

Para un sistema de orden 1 tenemos que la salida viene dada por la siguiente expresión:

$$y(t) = k(\Delta u)[1 - \exp\left(\frac{-t}{\tau}\right)]$$

Implementamos esta función en Excel dando un valor aproximado a la constante de tiempo, aunque también podemos calcularlo teniendo en cuenta su definición, el tiempo que tarda el sistema en alcanzar el 63,21% del valor del estacionario.

Si seleccionamos las columnas de la temperatura experimental y la temperatura estimada por la función de transferencia en función del tiempo y hacemos una gráfica tipo línea obtenemos lo siguiente (Figura 28):

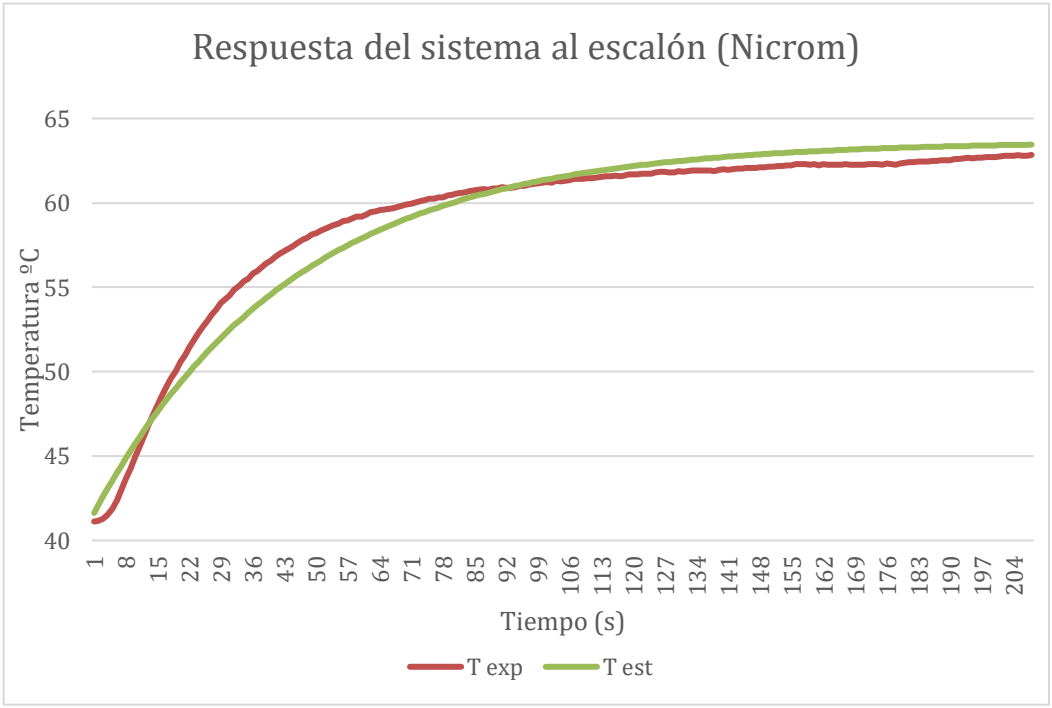


Figura 28, Respuesta del Nicrom

Repetimos el proceso con el nicrom a un valor fijo y variamos el ventilador (Figura 29):

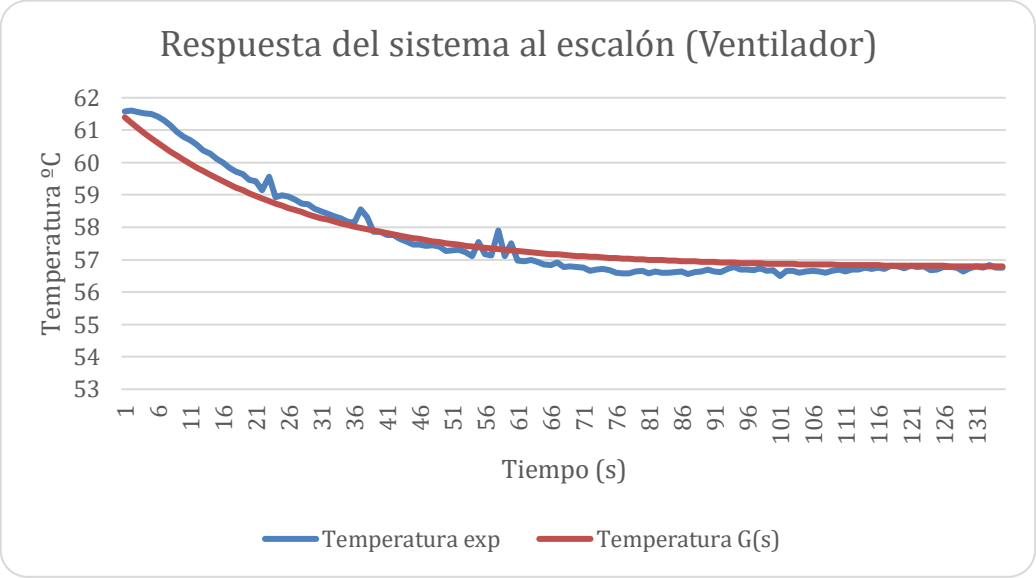


Figura 29, Respuesta del Ventilador

	Constante de tiempo (τ)	Ganancia(k)
Nicrom	44	0,45
Ventilador	27	-0,24

Tabla 5, Parámetros de la función de transferencia obtenidos

5.3 Análisis de la respuesta del sistema

Los resultados obtenidos en el apartado anterior describen como se comporta el sistema, de ellos se sacan las siguientes conclusiones:

Se trata de un sistema con una respuesta muy lenta, con una constante de tiempo τ de 44 s para cambios en el nicrom y de 27 s para cambios en el ventilador, esto se puede deber a que el calentamiento del nicrom aumentando la intensidad eléctrica que pasa por el mismo de por sí es un proceso lento, también que, al tratarse de un sistema abierto el proceso no es adiabático y se tarda en alcanzar el estacionario.

Ante un sistema tan lento, el tiempo de retardo que se produciría desde que la señal PWM llega a los actuadores hasta que la temperatura cambia significativamente en el sensor no es relevante.

También se observa que la ganancia del sistema para el ventilador es negativa mientras que la del nicrom es positiva, es decir, que el aumento de la señal en el nicrom produce un incremento en la temperatura del sistema mientras que cuando es el ventilador el que aumenta su velocidad de rotación, la temperatura tiende a decrecer.

5.4 Diseño del control digital (lazo cerrado)

Al tratarse de un sistema con varias entradas pueden plantearse varias estrategias para cerrar el lazo de control. Se podría llevar el control de ambas entradas para llegar al valor de consigna o solo de una de ellas considerando la otra como una perturbación.

Se acordó con la persona que encargó la planta llevar el control del nicrom y considerar el ventilador como una perturbación, por lo que se tendría el siguiente lazo de control (Figura 30):

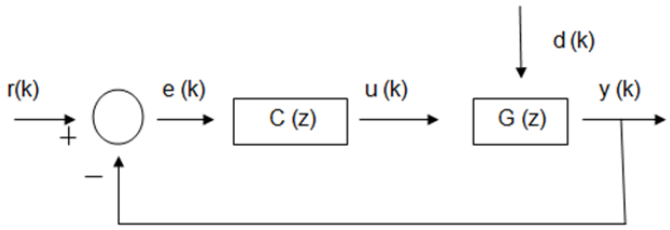


Figura 30, Diagrama del lazo de control

r(k)	Consigna
e(k)	Error (consigna-salida)
u(k)	Entrada (PWM Nicrom)
d(k)	Perturbación (PWM Ventilador)
y(k)	Salida (Lectura del sensor)
C(z)	Controlador
G(z)	Sistema

Tabla 6, Variables del lazo de control

La variable sobre la que se llevará a cabo el control es u(k), para un controlador PI tenemos que:

$$u(k) = p1 * e(k-1) + p2 * e(k-2) + u(k-1)$$

Donde p1 y p2 son parámetros que se establecen con las ganancias del controlador PI (Kp,Ki) y están relacionadas de la siguiente forma:

$$p1 = Kp$$

$$p2 = -Kp + KiT$$

Para implementar el lazo de control necesitamos datos, de los muestreos anteriores e(k-1), e(k-2) y datos de la entrada en el instante anterior u(k-1) por ese motivo declaramos los siguientes vectores:

```
//Vector que almacena los últimos 500 valores de tiempo
let tiempoV = [];
//Vector que almacena los últimos 500 valores de T
let resultadosTemperatura = [];
//Vector que almacena los últimos 500 valores de NicromVoltage
let resultadosNicromVoltage = [];
//Vector que almacena los últimos 500 valores de FanVoltage
let resultadosFanVoltage = [];
```

Código 13, Vectores para la construcción de los gráficos

Rellenamos el contenido de los vectores de la siguiente forma. Con el método shift() eliminamos el primer elemento del array, mientras que el método push() añade un elemento a la última posición del vector:

```
    //Actualización del vector de Temperatura (Para el gráfico)
    resultadosTemperatura.shift();
    resultadosTemperatura.push(T);

    //Actualización del vector del tiempo (Para el gráfico)
    tiempoV.shift();
    tiempoV.push(tiempo);

//Actualización del vector del Voltaje del Ventilador (Para el gráfico)
    resultadosFanVoltage.shift();
    resultadosFanVoltage.push(FanVoltage);

//Actualización del vector del Voltaje del Ventilador (Para el gráfico)
    resultadosNicromVoltage.shift();
    resultadosNicromVoltage.push(NicromVoltage);
```

Código 14, Actualizando los datos de los vectores

Los valores de Kp, Ki, Consigna y Voltaje del ventilador serán solicitados al usuario a través de la etiqueta <input> de HTML, estas están declaradas como globales y se actualizarán al ejecutar la función:

```
let Kp = 0.0005;    //Ganancia proporcional
let Ki = 0.00005;    //Ganancia integral
let Consigna = 30;   //Consigna
```

```
function actualiza(){ //Actualiza los valores del ventilador, Ki,Kp y Consigna
  let a = parseFloat(document.querySelector(".Fan .faninput").value);
  if((a <= 1)&&(a >= 0)){
    FanVoltage = a;
  }
  let t =  parseFloat(document.querySelector(".Nicrom .nicrominput").value);
  if((t <= 75)&&(t >= 25)){ //Rango de temperaturas (25-75 ºC)
    Consigna = t;
  }
  Kp =  parseFloat(document.querySelector(".Kp .Kpinput").value);
  Ki =  parseFloat(document.querySelector(".Nicrom .Kiinput").value);
  PWMWrite();
}
```

Código 15, Recogiendo datos introducidos por usuario

Aseguramos que la ganancia Kp y Ki estén entre (0 y 100) y realizamos los cálculos para que estén integradas en p1 y p2:

A partir del tercer muestreo (cuando la variable tiempo sea igual a 3) se implementa el control, esto se debe hacer porque no se tienen los datos suficientes para resolver la ecuación del controlador para la variable controlada hasta ese instante. Así que antes de ese instante se establece que el valor del duty cicle del PWM del nicrom sea de un 10% (0,1).

```
    if(((Kp >= 0) && (Kp <= 100))&&((Ki >= 0)&&(Ki <= 100))){
      //Kp, Ki entre 0 y 100
      p1 = Kp;
      p2 = -Kp + Ki*T;
    }

    if (tiempo >= 3){// Si se han completado 3 muestreos
      u1 = parseFloat(resultadosTemperatura[499]);
      u2 = parseFloat(resultadosTemperatura[498]);
      uAnt = parseFloat(resultadosNicromVoltage[499])

    // u(k) = p1*e(k-1) + p2*e(k-2) + u(k-1)
      NicromVoltage = p1*(Consigna - u1)
      + p2*(Consigna - u2) + uAnt;
    }else {
      NicromVoltage = 0.1;
    }
    if (NicromVoltage < 0) NicromVoltage = 0;
    if (NicromVoltage > 1) NicromVoltage = 1;
    PWMWrite();
```

Código 16, Calculando la acción de control

A continuación, se asignan los valores necesarios para calcular la ecuación del controlador asegurando que se están obteniendo números ya que en JavaScript no es un lenguaje fuertemente tipado. Una vez calculada la ecuación del controlador, ejecutamos la función PWMWrite() para enviar el valor al Nicrom.

Por último, si el valor calculado fuera menor a 0 o mayor a 1, se establecería en 0 o 1 respectivamente. Esto se hace porque la librería BoneScript define que el valor del duty cicle debe estar entre 0 y 1, antes de pasar los datos para que sean almacenados por el vector se comprueba que se cumpla este criterio, de lo contrario, se podría inestabilizar el lazo de control.

Con todo esto, ya es posible la implementación del lazo de control cerrado. A continuación, analizaremos el control del PI para el sistema.

5.5 Análisis de la respuesta del sistema al lazo cerrado

Teniendo el software necesario para el lazo cerrado se puede analizar el sistema para diferentes casos, pudiéndose modificar los siguientes parámetros:

- Consigna Temperatura: establece el valor que debe tomar la salida del sistema (por defecto 30 ºC).
- PWM Ventilador: asigna el valor de la perturbación (por defecto 10%).
- Ganancia proporcional (Kp): ajusta la velocidad a la que se llega al estado estacionario (por defecto 0,0005).
- Ganancia Integral (Ki): indica la velocidad con la que se repite la acción proporcional (por defecto 0,00005).

- **Experimento nº1**:

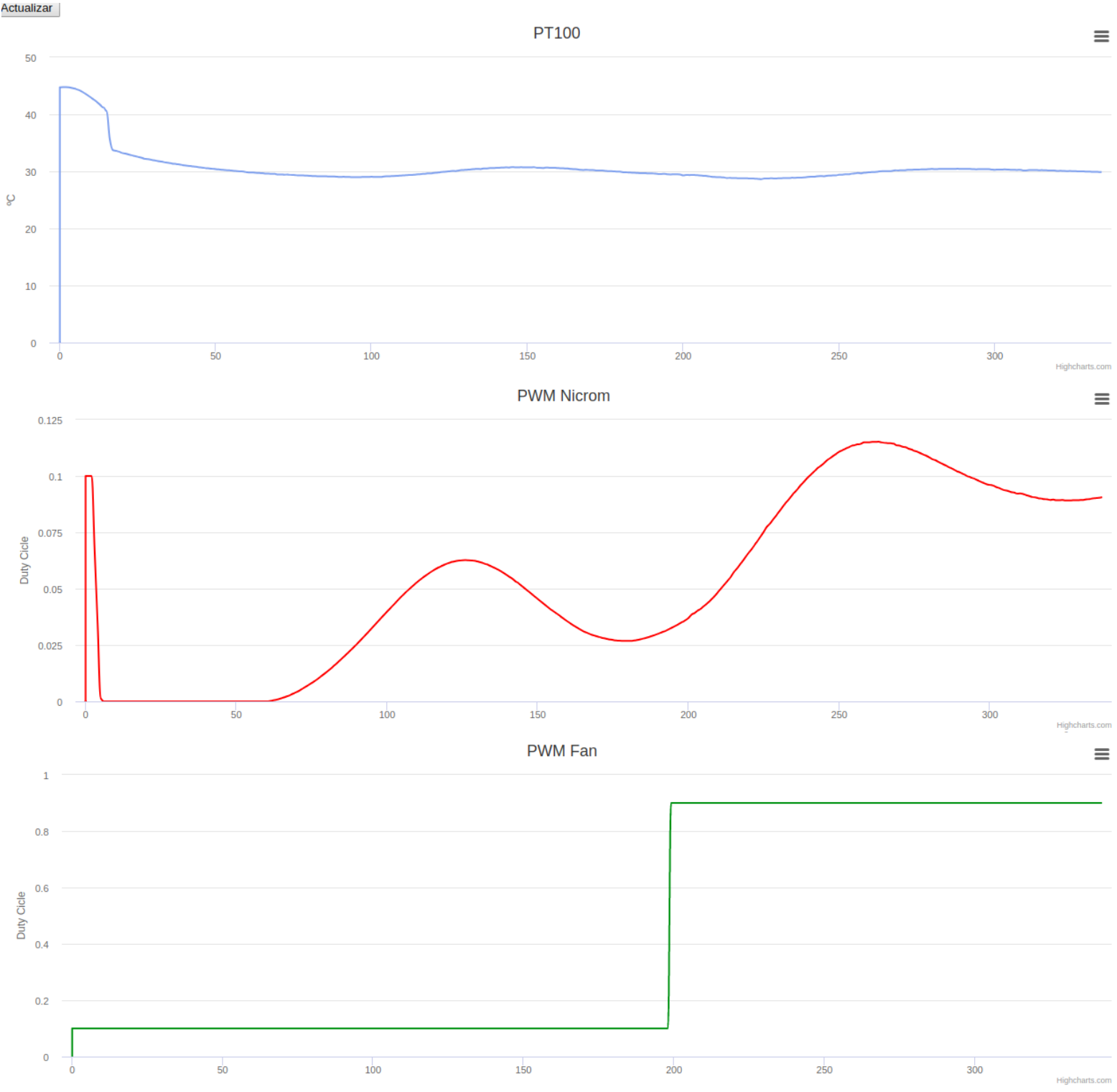


Figura 31, Experimento nº 1

Estos son los resultados al ejecutar el software de control en lazo cerrado (apartado 6.2.2), en ellos podemos ver (en tiempo real) el valor de la temperatura (gráfica Pt100), el valor del duty cicle del PWM del nicrom (gráfica PWM Nicrom) y el valor del duty cicle del PWM del ventilador (gráfica PWM Fan). O lo que es lo mismo, el valor de la salida, la entrada y la perturbación del sistema en función del tiempo (en segundos).

Tiempo 0-3 s:

Se inicia el programa con los valores por defecto, en este intervalo de tiempo la variable de control no cambia pues aún faltan muestras para implementar la acción del controlador.

Tiempo 3-200 s:

La variable de control se modula para llegar al valor de consigna. En el inicio, el sistema detecta un valor muy alto en la salida y rápidamente disminuye el valor del nicrom al mínimo para bajar la temperatura. Una vez alcanzado el valor de consigna, trata de estabilizar el valor de la salida, oscilando el valor de la variable de control según dictamine el PI.

Tiempo > 200 s:

En este punto se aumenta significativamente la perturbación (del 10 al 90%) manteniendo los mismos valores de consigna y de ganancias, se puede observar como la variable de control aumenta su valor para compensar la perturbación.

- **Experimento nº2:**

Tenemos que la ganancia por defecto para la parte proporcional es Kp = 0,0005 y la integral Ki = 0,00005. ¿Por qué se han elegido esos valores tan pequeños?, muy bien en el experimento nº2 analizaremos que ocurre al aumentar estos parámetros.

Figura 32, Experimento nº 2

Tiempo 300-400 s (aprox):

En este rango de valores se establecieron los siguientes parámetros: Consigna = 40ºC Kp = 0.05, Ki = 0.005, es decir las ganancias son 100 veces más que las que se designan por defecto, se puede apreciar como el sistema no es capaz de alcanzar el valor de consigna porque el control se vuelve totalmente inestable, oscilando entre el valor máximo (1) y el valor mínimo (0) de la variable controlada. Como el nicrom no tiene la capacidad de enfriar el sistema, inevitablemente tiende a aumentar la temperatura.

Que las ganancias tengan que ser tan pequeñas está justificado por la lentitud del sistema, que no es capaz de seguir la dinámica que marca el controlador.

Tiempo > 400s:

Las ganancias Kp y Ki se reducen a 0,005 y 0,0005 respectivamente (10 veces la impuesta por defecto) mientras que la consigna se mantiene en 40ºC. Ahora el sistema sí que es capaz de ejecutar la acción de control y se llega a 40ºC con cierta sobreoscilación.

6. Estructura de la página Web

6.1 Inicio

Este proyecto trata de implementar un sistema SCADA en un navegador Web, es por ello que ha sido integrado en la página web JSCADA de fácil acceso para el usuario, ingresando la dirección http://192.168.7.2/user/TFG/Inicio.html tenemos acceso a la web (Figura 33).

Figura 33, JSCADA Inicio

Como se puede observar en la Figura 33 desde la barra superior de la página de Inicio se podrá acceder al resto de partes del proyecto.

- Inicio: página de inicio en la que nos encontramos actualmente.
- Pruebas: sección donde se recogen las diferentes pruebas del proyecto.
- Planta: Aquí encontraremos el sistema SCADA con la planta en lazo abierto o lazo cerrado
- Info: Información sobre el proyecto realizado, con datos que se incluyen en esta memoria.

En el cuerpo de la página encontramos una breve descripción del proyecto y una imagen de la planta sobre la que se ha llevado el control.

Además, se han añadido unos recursos a la derecha que nos redireccionaran a páginas externas, que están contenidas en la sección 7.3 Enlaces de interés de esta memoria.

6.2 Planta

6.2.1 Control en lazo abierto

En la pestaña planta de la página Web JSCADA, podemos seleccionar la opción de "Control en lazo abierto" que accederá a la siguiente interfaz (Figura 34):

Figura 34, JSCADA Control en lazo abierto Salida

En primer lugar, se debe seleccionar el icono con forma de gráfica que está ubicado encima de los parámetros de entrada. Al iniciar el programa se asignarán a las entradas del Nicrom y Ventilador el valor de 0,1 (10%) que son los valores por defecto.

También comenzará la acción de muestreo del sensor que pasará los datos al gráfico para que se actualice de una forma dinámica.

La gráfica tiene la capacidad de cambiar dinámicamente sus ejes, hará un autoajuste en todo momento según su contenido. El máximo de datos que se

mostrarán por pantalla al usuario es de 500, como está programado para que se haga una muestra por segundo, se mostrarán 500 segundos. Una vez pasado este tiempo se autoajustará mostrando las últimas 500 medidas.

Al igual que para la salida (Temperatura), han sido implementados dos gráficos más para las entradas como se muestra en la Figura 35.

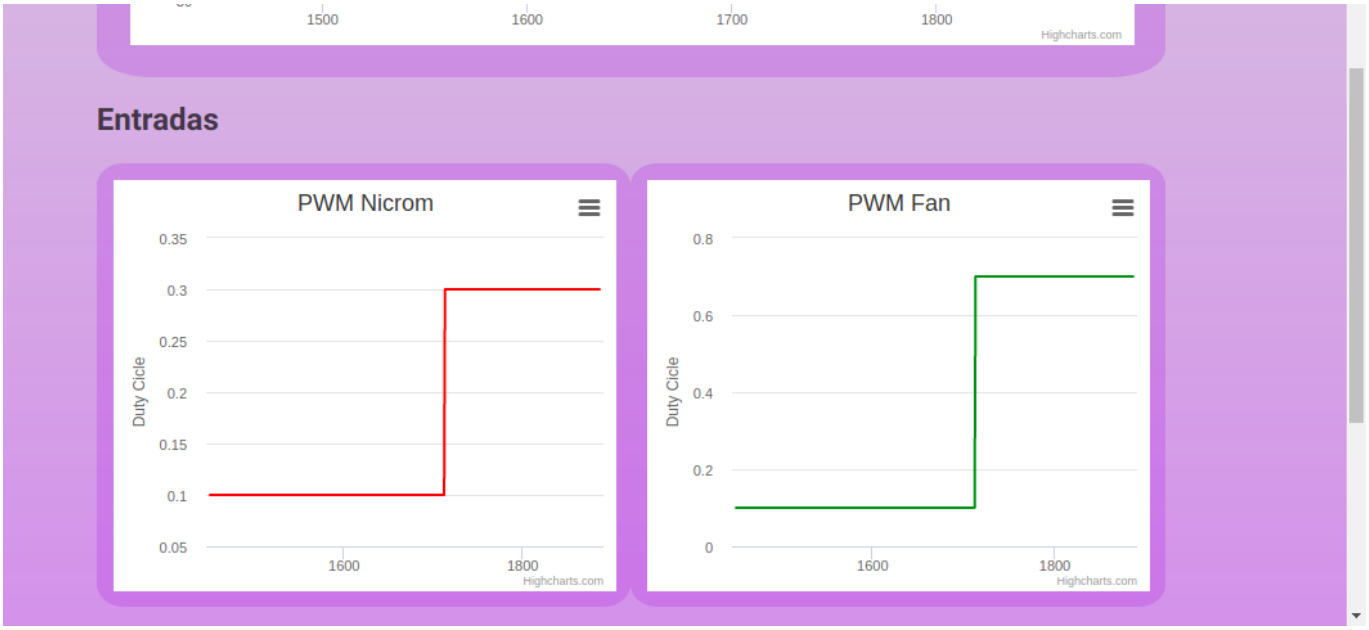


Figura 35, JSCADA Control en lazo abierto Entradas

A la izquierda del gráfico de salida, bajo el icono que ejecuta el programa (con forma de gráfica) existen dos entradas disponibles para que el usuario ingrese los valores de Nicrom y Ventilador. El valor está limitado por software entre 0 y 1.

Todas las gráficas tienen en su esquina superior derecha una pestaña con diversas funcionalidades:

- Ver en pantalla completa
- Descargar Gráfica a formato .PNG, .JPEG, .PDF, .SVG
- Descargar Datos a .XLS o .CSV
- Ver tabla de valores
- Abrir Gráfico en la nube de Highcharts

6.2.2 Control en lazo cerrado

En la pestaña planta de la página Web JSCADA, podemos seleccionar la opción de "Control en lazo cerrado" que accederá a la siguiente interfaz (Figura 36):

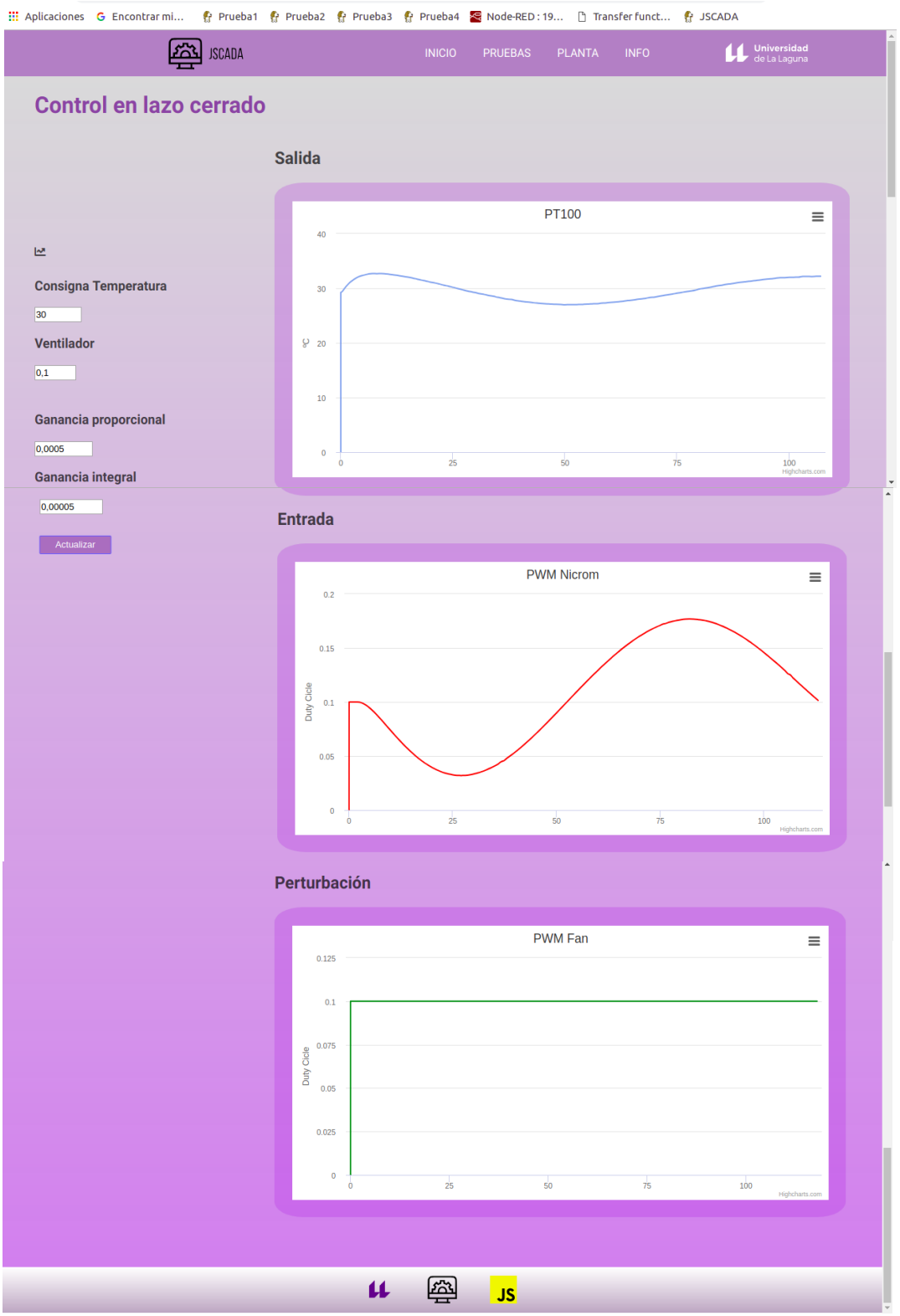


Figura 36, JSCADA Control en lazo cerrado

Al igual que en la sección anterior, se debe seleccionar el icono con forma de gráfica que está ubicado encima de los parámetros de entrada. Al iniciar el programa, para los tres primeros muestreos, se asignarán a las entradas del Nicrom y Ventilador el valor de 0,1 (10%) que son los valores por defecto. Después de esos tres primeros muestreos comenzará la acción de control, como valor por defecto la consigna de temperatura será de 30 °C, el valor de la ganancia proporcional de 0,0005 y el valor de la ganancia integral de 0,00005.

Tanto la salida (Temperatura), la entrada (PWMNicrom) y la perturbación (PWMVentilador) del sistema disponen de sus propias gráficas, compartiendo los mismos valores de tiempo, todas conservan las propiedades que se han comentado en la sección anterior.

A la izquierda del gráfico de salida, bajo el icono que ejecuta el programa (con forma de gráfica) existen cuatro entradas disponibles para que el usuario ingrese los valores de:

- Consigna de temperatura. Limitado de 25 a 75 C°.
- PWM Ventilador. Limitado de 0 a 1.
- Ganancia proporcional. Limitado de 0 a 1.
- Ganancia integral. Limitado de 0 a 1.

7. Conclusión

7.1 Inconveniencias durante el proyecto

Durante el proyecto, surgieron varios inconvenientes:

En la fase de construcción del código, se trató de utilizar el entorno de programación Node-red en la BeagleBone Black, ya que viene integrado en la propia placa. Para aplicaciones básicas como encender un led no había problema alguno, sin embargo, cuando se trataba de registrar el valor de una entrada analógica en una gráfica, se consumían demasiados recursos en la placa. En mi opinión, es una herramienta interesante a futuro, aunque a día de hoy está mal optimizada para la BeagleBone Black.

En la fase de adquisición de la señal de temperatura se intentó conectar la Pt100 a la BBB implementando el protocolo I2C, utilizando el conversor analógico-digital ADS1115 ya que tenía una gran resolución (16 Bits). Sin embargo, no se logró encontrar drivers de I2C en JavaScript para este dispositivo.

Una vez que fueron realizadas las pruebas en la BBB, se tuvo que esperar a la finalización de la planta por el Servicio de Electrónica. En concreto, la construcción de la planta estuvo parada por el encargo de una resistencia de potencia que hacía falta para la disipación de calor.

Al ser un sistema con una dinámica tan lenta, no se ha podido comprobar los tiempos de retardo que se calculaban en la planta original, aunque ese inconveniente está en el diseño de la propia planta que no es competencia de este proyecto.

7.2 Conclusión Final

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

Durante el transcurso del TFG, a fin de adquirir conocimientos de las tecnologías WEB (HTML, CSS y JavaScript) he realizado dos cursos de formación desde la Fundación General de la Universidad de la Laguna. Un curso de CSS y otro de JavaScript.

Para la familiarización con la placa BeagleBone Black y la librería BoneScript, se han realizado unas pruebas que comunican el hardware con el navegador y que implementan las tecnologías web de una forma progresiva. En estas pruebas podemos encontrar desde encender un led haciendo click en un botón del navegador hasta gráficas dinámicas que registran el valor de luminosidad que da un LDR en función de la hora actual recogida desde el propio navegador.

Una vez realizadas las pruebas, con la finalidad de hacer un SCADA de un sistema real, se ha trabajado con una planta que había elaborado el Servicio de Electrónica de la Universidad de la Laguna

Para ello, como la planta estaba siendo coordinada a través de un microcontrolador Arduino en una PCB, hemos tenido que buscar una forma de comunicarnos con el Arduino aprovechando que era implementable el control por voltaje. Se han replicado los circuitos del Arduino rediseñando con los valores correspondientes para la BeagleBone. Se han calibrado los circuitos electrónicos para la correcta adquisición de la temperatura y el control de los actuadores con señales PWM conectadas al control por voltaje del Arduino.

Llegados a este punto se realizó por software un entorno para el control en lazo abierto, con gráficas en tiempo real que permitieran ella descarga de datos. Con los datos de la respuesta de la planta al estimularla con una entrada tipo escalón se han calculado gráficas de la respuesta del sistema. También se ha hecho una valoración de los datos obtenidos.

A continuación, se procedió a cerrar el lazo de control mediante la implementación de un controlador PI, fue realizado el software necesario para el control digital, dejando libertad al usuario para que introduzca varios parámetros del sistema. Se añadió, igual que al lazo abierto, gráficas en tiempo real y permitiendo la descarga de datos. Los datos del sistema en lazo cerrado han sido interpretados sintonizando los parámetros del controlador PI.

Por último, se ha desarrollado una web que permite el acceso tanto al software de control en lazo abierto como en lazo cerrado, las pruebas realizadas e información que se recoge en esta memoria.

7.3 Final conclusion

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

During the TFG, to a better understanding of WEB technologies (HTML, CSS and JavaScript), I have done two courses on the General Foundation of the University of La Laguna. A CSS course and a JavaScript one.

To work correctly with the BeagleBone Black and the BoneScript library, has been done some tests to communicate the hardware with the browser and the implementation of web technologies in a progressive way. In these tests we can find how to turn on with the click on a browser button or the construction of a dynamic charts wich take the value of the brightness on a LDR in the function of the current time taken by the browser.

Once the tests were done, in order to make a SCADA in a real system, we worked with a plant that been developed by the Electronic Service of the University of La Laguna.

Firstly, the plant was coordinated by Arduino microcontroller on a PCB, we had to find a way to communicate with Arduino, that can be posible if we use the voltage control. The circuits of the Arduino have been replicated, redesigning the values corresponding to the BeagleBone. The electronic circuits have been calibrated for the correct acquisition of temperature and the control of the actuators with PWM signals connected to the voltage control of the Arduino.

At this point, software for control in the open loop was developed, with the graphs in real time wich allow data download. Graphs of the response of the system have been used with the response data of the plant with a step-type input. A studio of the corresponding data has also been made.

Next, we will see how the control loop works with the implementation of a PI controller, for that reason was developed a software for digital control, with freedom for the input user data. Added, as in the open loop, charts in real time and data download is available. The system data in the closed loop has been interpreted by tuning of parameters of the PI controller.

Finally, a website has been developed wich allows the access to the open loop and closed loop control software, the tests and the information found in this document.

7.4 Líneas Futuras

Haciendo uso de las tecnologías Web se abre un gran abanico de posibilidades. Pues solo necesitamos del uso de un navegador para la ejecución del código, pudiéndose implementar en todo tipo de dispositivos a través del protocolo TCP/IP. Esta idea esta enlazada con tecnologías al alza como es el IoT (Internet de las Cosas). Por tanto, se abre una vía muy interesante en este ámbito.

Relacionado con el apartado anterior, se podría intentar hacer una página Web totalmente Responsive para el acceso desde otros dispositivos como móviles o tablets usando frameworks de JavaScript para páginas dinámicas como son React, Angular o Vue.

A través de la web que se ha desarrollado, se podría implementar un SCADA con más funcionalidades, este TFG se ha centrado en la implementación del control de la planta, pero se podría añadir seguridad al acceso, solicitando un user y un password, botones de parada de emergencia, displays de las medidas, etc… La idea es seguir con el desarrollo de web pudiendo extrapolarse fácilmente a otros sistemas. En ese sentido el código que se ha desarrollado no es muy cerrado y es totalmente realizable.

En la comunicación con el Arduino, es posible que se pudiera haber realizado mediante I2C directamente sin la necesidad del montaje para el control por voltaje, a través de una comunicación maestro-esclavo. Esta idea surgió al final del proyecto y no se llegó a implementar.

Al analizar la planta en lazo abierto, se comprobó que en el estacionario existía sobreoscilación, se podría implementar por software un filtro sobre la acción de control en lazo cerrado. Por ejemplo, un filtro Butterworth al que el usuario le pudiera pasar también los parámetros.

8. Bibliografía y enlaces de interés

8.1 Bibliografía consultada

- Charles A.Hamilton, "**BeagleBone Black Cookbook**", Packt Publishing, 2015
- Brian McLaughlin, "**The BeagleBone Black Primer**", Que, 2015
- Jason Kridner; Mark A. Yoder, "**BeagleBone Cookbook**", O'Reilly Media, 2015

8.2 Referencias

1. https://www.plcacademy.com/scada-system/
2. https://insights.stackoverflow.com/survey/2018/#most-popular-technologies
3. https;//www.emezeta.com
4. https://es.wikipedia.org/wiki/Dise%C3%B1o_web_adaptable
5. https://medium.com/level-up-web/amazingly-useful-html-css-and-javascript-tools-and-libraries-d73b10fbae29
6. https://github.com/beagleboard/beaglebone-black/wiki/System-Reference-Manual
7. https://beagleboard.org/static/images/cape-headers.png
8. https://es.wikipedia.org/wiki/Puente_de_Wheatstone
9. http://www.ti.com/lit/ds/sbos014/sbos014.pdf
10. https://www.arduino.cc/en/tutorial/PWM
11. http://www.ti.com/lit/ds/symlink/lm158-n.pdf

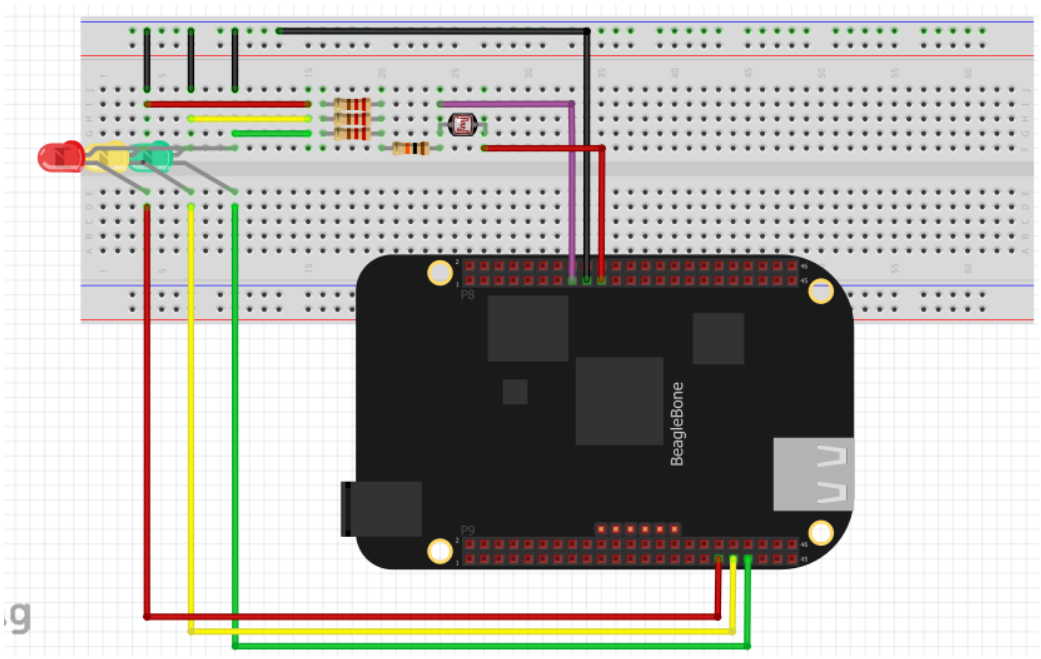
8.3 Enlaces de interés

- https://nodered.org/docs/
- https://beagleboard.org/
- https://lenguajehtml.com/p/html/
- https://lenguajecss.com/p/css/
- https://lenguajejs.com/p/javascript/
- https://nodejs.org/es/docs/guides/
- https://www.highcharts.com/demos/
- https://www.chartjs.org/docs/latest/

9. Anexos

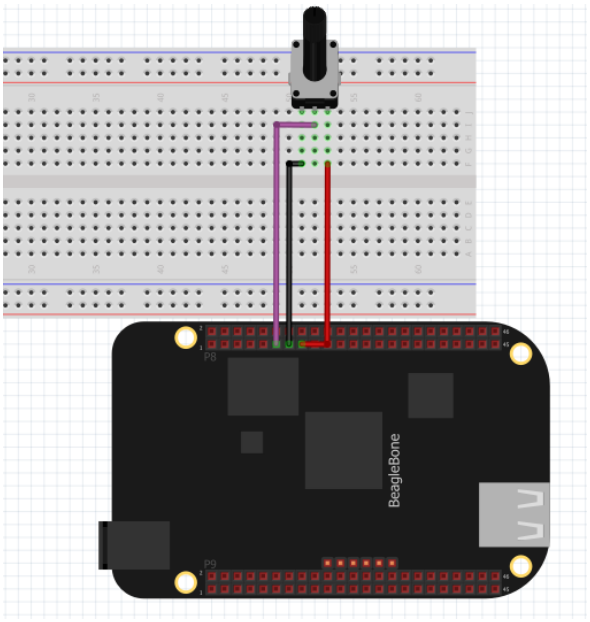
9.1 Circuitos de las pruebas

9.1.1 Prueba 1 y 3



Pin BBB	Conexión	Función
P8_8	Led verde	Salida digital
P8_10	Led amarillo	Salida digital
P8_12	Led rojo	Salida digital
P9_36	LDR	Entrada analógica
P9_32	VDD	Alimentación
P_34	GND	Tierra

9.1.2 Prueba 2 y 4



Pin BBB	Conexión	Función
P9_36	Potenciómetro	Entrada analógica
P9_32	VDD	Alimentación
P_34	GND	Tierra

9.2 Código de las pruebas

Prueba 1

```
//http://192.168.7.2/user/Pruebas/PruebaSemaforoManual/PruebaSemaforoManual.html

setTargetAddress(location.host, {
  initialized: run
  }); //Apuntamos a la direccion y ejecutamos en funcion run


let b = undefined;

let ledPin1 = "P8_8"; //led verde
let ledPin2 = "P8_10";//led amarillo
let ledPin3 = "P8_12"; //led rojo


function run(){

  b = myrequire('bonescript'); //libreria Bonescript
  b.pinMode(ledPin1, b.OUTPUT);
  b.pinMode(ledPin2, b.OUTPUT);
  b.pinMode(ledPin3, b.OUTPUT);
}

function botonVerde(){

  b.digitalWrite(ledPin1, b.HIGH);
  b.digitalWrite(ledPin2, b.LOW);
  b.digitalWrite(ledPin3, b.LOW);
}

function botonAmarillo(){

  b.digitalWrite(ledPin1, b.LOW);
  b.digitalWrite(ledPin2, b.HIGH);
  b.digitalWrite(ledPin3, b.LOW);
}

function botonRojo(){

  b.digitalWrite(ledPin1, b.LOW);
  b.digitalWrite(ledPin2, b.LOW);
  b.digitalWrite(ledPin3, b.HIGH);
}
```

Prueba 2

```
//http://192.168.7.2/user/Pruebas/PruebaPotenciometro/PruebaPotenciometro.html

setTargetAddress(location.host, {
    initialized: run
    }); //Apuntamos a la dirección y ejecutamos en función run

  let b = undefined;

  let potPin = "P9_36"; //Potenciómetro
  let arrayY = [0]; //Vector necesario para construir la grafica

  function run(){
    b = myrequire('bonescript'); //librería Bonescript
  }

  function ldrRead(){

    b.analogRead(potPin, ldrWrite);
    setTimeout(ldrRead, 200);
  }

  function ldrWrite(x) {
    arrayY.shift();
    arrayY.push(Math.round(x.value * 100)); //Aquí actualizamos el vector
  }
var chartSpeed = Highcharts.chart('container', Highcharts.merge(gaugeOptions,
{ … }));

let gaugeOptions = { … };

  // Bring life to the dials
  setInterval(function () {
    // Speed
    var point,
      newVal,
      point = chartSpeed.series[0].points[0];
      if (arrayY != 100){
      newVal = arrayY;
      }

      point.update(newVal);

  }, 1000);
```

Prueba 3

```
//http://192.168.7.2/user/Pruebas/PruebaLDR/PruebaLDR.html

setTargetAddress(location.host, {
  initialized: run
  }); //Apuntamos a la direccion y ejecutamos en funcion run



let b = undefined;

let ldrPin = "P9_36";
let ledPin1 = "P8_8";
let ledPin2 = "P8_10";
let ledPin3 = "P8_12";
let resultados =
  [ 0,0,0,0,0,0,0,0,0,0,
    0,0,0,0,0,0,0,0,0,0,
    0,0,0,0,0,0,0,0,0,0,
    0,0,0,0,0,0,0,0,0,0,
    0,0,0,0,0,0,0,0,0,0,]; //Vector necesario para construir la grafica

function run(){
  b = myrequire('bonescript'); //libreria Bonescript
  b.pinMode(ledPin1, b.OUTPUT);
  b.pinMode(ledPin2, b.OUTPUT);
  b.pinMode(ledPin3, b.OUTPUT);
}

function ldrRead(){
  b.analogRead(ldrPin, ldrWrite);
  setTimeout(ldrRead, 1000);
}


function ldrWrite(x) {
  b.digitalWrite(ledPin1, b.LOW);
  b.digitalWrite(ledPin2, b.LOW);
  b.digitalWrite(ledPin3, b.LOW);
  resultados.shift();                    //Aqui actualizamos el vector
  resultados.push(Math.round(x.value * 100));
  if(x.value <= (33.33/100)){
      b.digitalWrite(ledPin3, b.HIGH);
  }
  if(x.value > (33.33/100) && x.value <= (66.66/100)){
      b.digitalWrite(ledPin2, b.HIGH);
```

```
  }
  if(x.value > (66.66/100)){
      b.digitalWrite(ledPin1, b.HIGH);
  }
}

Highcharts.chart('container', {
chart: {
  type: 'spline',
  animation: Highcharts.svg, // don't animate in old IE (NO INTERNET EXPLORER!!)
  marginRight: 10,
  events: {
    load: function () {

      // Funcion que actualiza el grafico a cada segundo
      var series = this.series[0];
      setInterval(function () {
        var x = (new Date()).getTime(), // Aqui registramos la hora actual
          y = resultados[49];               // Pasamos la ultima posicion del vector
(la mas actualizada)
        series.addPoint([x, y], true, true);
      }, 1000);
    }
  }
},

time: {
  useUTC: false
},

title: {
  text: 'Live LDR data'
},
xAxis: {
  type: 'datetime',
  tickPixelInterval: 150
},
yAxis: {
  title: {
    text: 'Value'
  },
  plotLines: [{
    value: 0,
    width: 1,
    color: '#808080'
  }]
},
```

```
tooltip: {
  headerFormat: '<b>{series.name}</b><br/>',
  pointFormat: '{point.x:%Y-%m-%d %H:%M:%S}<br/>{point.y:.2f}'
},
legend: {
  enabled: false
},
exporting: {
  enabled: true
},
series: [{
  name: 'LDR value',
  data: (function () {
    // Generamos un vector que vaya guardando los datos anteriores
    var data = [],
      time = (new Date()).getTime(),
      i;

    for (i = -49; i <= 0; i += 1) {
      data.push({
        x: time + i * 1000,
        y: resultados[i + 49]
      });
    }
    return data;
  }())
}]
});
```

Prueba 4 (CSS)

```
@import
url('https://fonts.googleapis.com/css?family=Roboto:300,400,500,700&display=swap');

.outer {
    margin: 0 auto;
  }
  .outer .chart-container {
    width: 300px;
    margin: 0 auto;
    height: 200px;
  }
  .highcharts-yaxis-grid .highcharts-grid-line {
    display: none;
  }

  @media (max-width: 600px) {
    .outer {
      width: 100%;
      height: 400px;
    }
    .outer .chart-container {
      width: 300px;
      float: none;
      margin: 0 auto;
    }

  }
body{
  margin: 0;
}
main{
  display: flex;
  flex-direction: column;
}
.encabezado{
  display: flex;
  align-items: center;
  height: 10vh;
  background: rgba(94, 62, 222, 0.71);
  color: white;
  font-family: Roboto;
  justify-content: center;
}
.encabezado h1{
  font-size: 3.5vh;
```

```
}
.entradilla{
  height: 25vh;
  font-family: Roboto;
  padding: 2vh;
  text-align: center;
  background: linear-gradient(180deg, rgba(152, 137, 212, 0.6) 0%, rgba(152, 137,
212, 0) 100%);
}
.entradilla p{
  font-size: 2.5vh;
  padding: 4vh 9vw;
  text-align: center;
}
.entradilla h2{
  font-size: 3vh;
}
.elementos{
  display: flex;
  flex-direction: row;
  height: 60.5vh;
  justify-content: center;
}
.elementos .imagen{
  display: flex;
  justify-content: center;
  align-items: center;
  height: 100%;
}
.elementos .flecha{
  display: flex;
  justify-content: center;
  padding: 0 10vw;
}
.imagen img{
  border-radius: 58px;
  height: 60%;
}
.grafica{
  display: flex;
  flex-direction: column;
  justify-content: center;
}
.elementos i{
  text-align: center;
  font-size: 5vh;
```