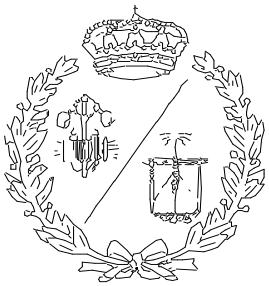
ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACIÓN

UNIVERSIDAD DE CANTABRIA



Proyecto Fin de Grado

SISTEMA DE CONTROL PID DE TEMPERATURA PARA LA DOCENCIA DE ASIGNATURAS DE REGULACIÓN AUTOMÁTICA

(PID temperature control system for the teaching of automatic control subjects)

Para acceder al Título de

GRADUADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

Autor: Canduela Ilundain, Jose Ignacio

febrero - 2022

AGRADECIMIENTOS

Quiero agradecer a todos los profesores que he tenido, tanto dentro como fuera de la universidad, por haber fomentado el desarrollo de mi curiosidad. También quiero agradecer a aquellos que me han formado como profesional y como persona, realizando un trabajo que nunca se podrá valorar lo suficiente.

También quería agradecerle a Julia que me ha llevado de la mano en todo este recorrido, y sin cuya sonrisa no habría podido llegar hasta aquí. Y a mis padres, que me han dedicado todo su tiempo, todo su esfuerzo y todos sus recursos, con tal de educarme y formarme lo mejor posible para afrontar la vida.

SISTEMA DE CONTROL PID DE TEMPERATURA PARA LA DOCENCIA DE ASIGNATURAS DE REGULACIÓN AUTOMÁTICA

RESUMEN

Se ha decidido crear un sistema para el control de temperatura de un espacio cerrado.

Se desarrollará tanto de forma teórica como real, y se trata de diseñar un controlador Proporcional-Integral-Derivativo capaz de medir la temperatura del sistema mediante un sensor de temperatura y una fuente de calor. Además constará de un ventilador en una de las paredes que funcionará como perturbación, además de fomentar la salida del calor/variación de temperatura.

Se podrá introducir la temperatura deseada y la velocidad del ventilador a partir de unos potenciómetros, así como seleccionar los parámetros del regulador, ya que el propósito del proyecto es didáctico. Tanto la temperatura deseada como la del sistema serán muestreadas en un display LED.

El sistema estará implementado a partir del controlador Arduino y tendrá un manual de usuario para que los alumnos puedan realizar unos ejercicios y aprender el funcionamiento de la retroalimentación en el control de una planta de forma muy visual y directa.

PID temperature control system for the teaching of automatic control subjects

ABSTRACT

It has been decided to create a system for the temperature control of an enclosed space.

It will be developed both theoretically and real, and it is to design a PID controller capable of measuring the temperature of the system by means of a temperature sensor and a heat source. It will also consist of a fan on one of the walls that will function as a disturbance, in addition to promoting the heat output/temperature variation.

It will be possible to enter the desired temperature and fan speed from potentiometers, as well as to select the regulator parameters, since the purpose of the project is didactic. Both the desired temperature and the system temperature will be displayed on a LED display.

The system will be implemented from the Arduino controller and will have a user manual so that students can perform some exercises and learn the operation of the feedback in the control of a plant in a very visual and direct way.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1 INTRODUCCIÓN

El control Proporcional-Integral-Derivativo (PID) [1] es un mecanismo de control que, a través de un lazo de retroalimentación, permite regular la respuesta de los procesos físicos entre otras variables de un proceso general. Este se encarga de calcular la señal de control en función de la diferencia entre el valor deseado y el valor actual de la magnitud que se desea controlar.

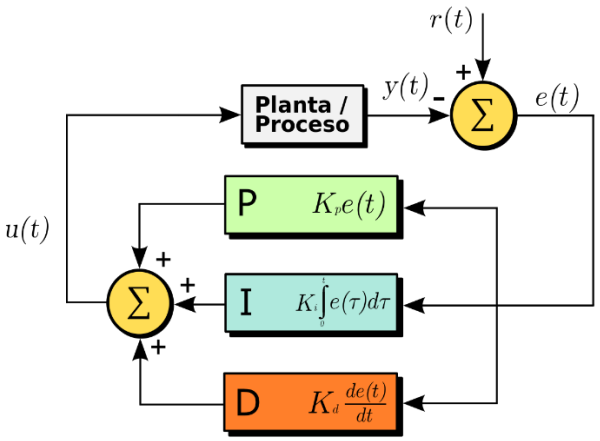


Figura 1.1 Diagrama de bloques de un controlador PID con realimentación

Uno de los ejemplos más antiguos fue un controlador PID [2] diseñado por Elmer Sperry en 1911, mientras que el primer análisis teórico de un controlador PID empezó con el diseño de los limitadores de velocidad y posteriormente fue usado para el gobierno automático de buques gracias al ingeniero ruso americano Nicolás Minorsky en 1922, el cual, observando al timonel y notando así que el timonel no solo controlaba la nave por el error actual, sino también por errores pasados y en la tasa actual de cambio, logrando así desarrollar un modelo matemático para esto.

Mientras que el control proporcional brinda estabilidad ante pequeñas perturbaciones, se requería un término integral para tratar perturbaciones constantes. Finalmente se agregó el término derivativo también para mejorar el control.

Se realizaron pruebas con el controlador en el USS New Mexico (BB-40), donde se encargaba de controlar la velocidad angular del timón. El control PI se mantuvo virando con ±2º de error, mientras que al añadir el derivativo, el error disminuyó hasta el ±1/6º, mucho mejor que lo que un timonel podría lograr.

1.1 OBJETIVOS

La finalidad de este proyecto es controlar la temperatura de la caja, de forma que se consiga de una forma rápida y visualmente directa la temperatura previamente indicada por el alumno, ya que es un proyecto didáctico.

Se podrán controlar las variables proporcional, derivativa e integral mediante unos potenciómetros, para observar el control del sistema según las vamos variando. Así mismo se podrá controlar la velocidad de un ventilador ubicado en una de las pareces de la caja, de forma que tenga ventilación.

Esto se logrará mediante el control PID consiguiendo un sistema estable y con el mínimo error.

Tanto la temperatura deseada como la real, será muestreada en un display de siete segmentos, de forma que se pueda apreciar facilmente la variación temperatura.

Se tratará de corregir el error del sistema en función de la temperatura objetivo de forma que pueda servirle al alumno para entender como funciona el control de una planta, así como el funcionamiento de un PID.

1.2 FINALIDAD

El proyecto es una aplicación didáctica para los alumnos de la Escuela Técnica Superior de Ingenieros Industriales y de Telecomunicación de la Universidad de Cantabria. Sirviendo de apoyo para el departamento TEISA(Tecnología Electrónica Ingeniería de Sistemas y Automática) y sus laboratorios, para enseñar el funcionamiento del control de una planta, la función de transferencia y como influyen las tres componentes de un controlador PID(proporcional, derivativa e integrativa) visualmente, de forma que resulte más sencillo comprender los conceptos y el funcionamiento.

1.3 METODOLOGÍA

El proceso de estudio de este proyecto se realizará en dos bloques, el teórico y el práctico.

En la primera parte, tenemos:

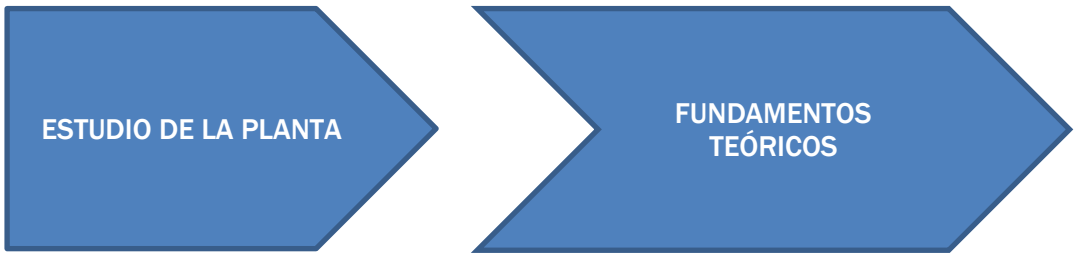


Figura 1.2 Esquema 1ª parte

Además tenemos la parte del diseño, construcción, y la parte de software:

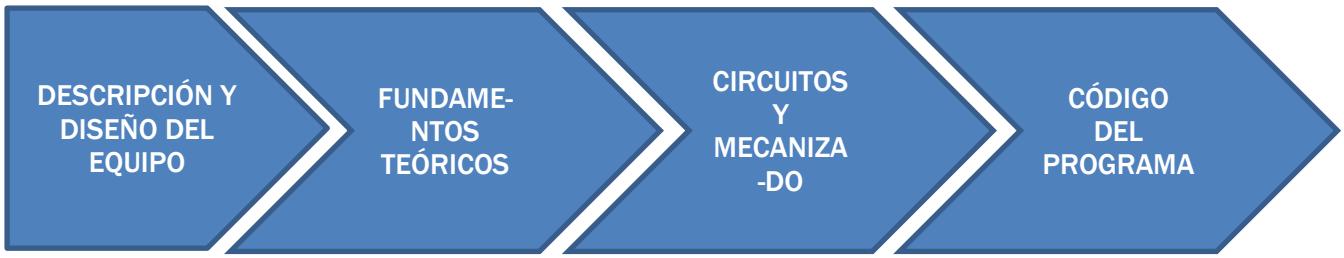


Figura 1.3 Esquema 2ª parte

2 ESTUDIO DE LA PLANTA

Lo primero es hacer un estudio de modelo de planta, para saber a que sistema nos estamos enfrentando.

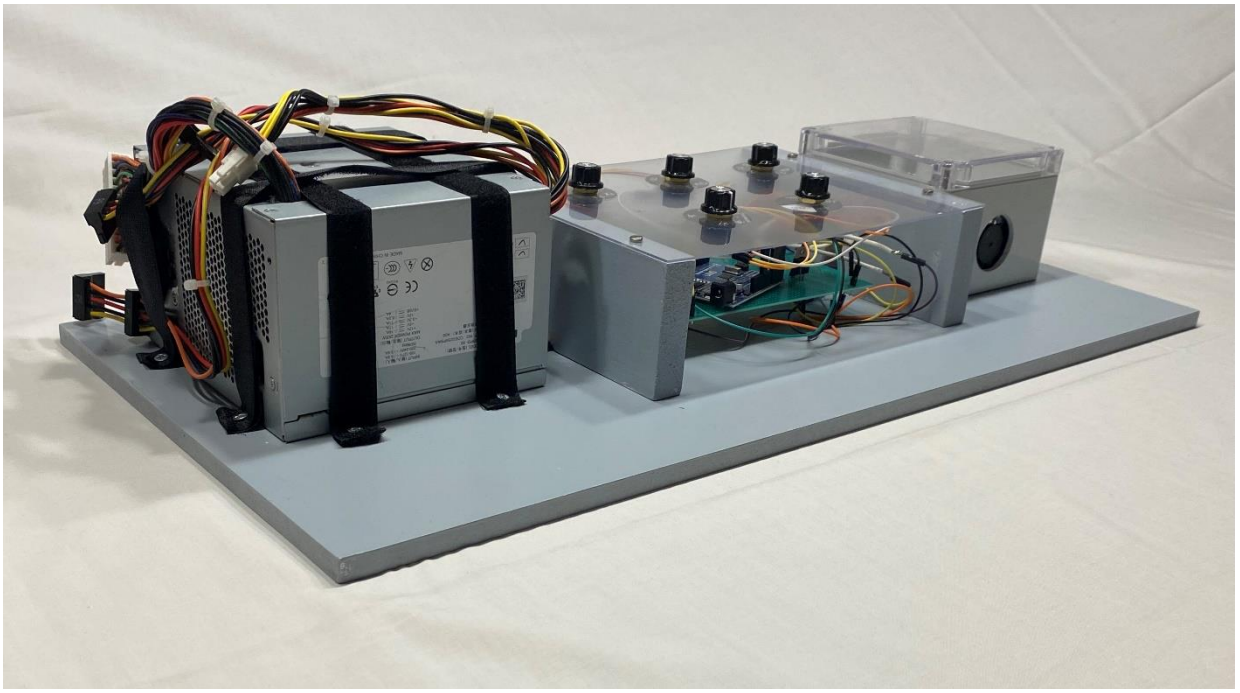


Figura 2.1 Planta al completo

El sistema se podría dividir en tres partes, la fuente de alimentación, la placa Arduino con sus potenciómetros y la caja contenedora donde se llevará a cabo el control de temperatura.

La fuente de alimentación, proviene de un ordenador que ya no tenía uso, pudiendo aprovechar la salida de 12V para controlar la bombilla (la cual es nuestro foco de calor).

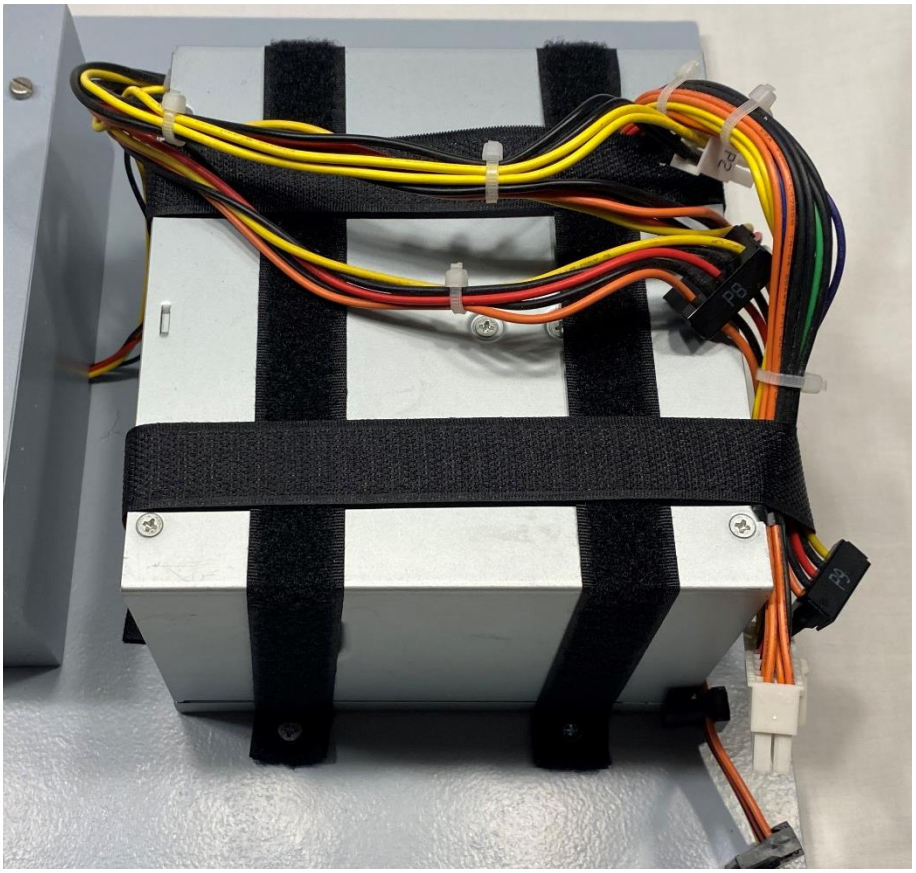


Figura 2.2. Fuente de alimentación

Una placa Arduino One, la cual tiene como entradas la lectura del sensor de temperatura, cinco potenciómetros para controlar las tres variables del controlador PID(k_p ,k_d y k_i), la temperatura deseada y el control de la velocidad de giro del ventilador. Como salidas tiene un display de 7 segmentos, el cual muestrea intermitentemente la temperatura deseada frente a la temperatura en el interior de la caja y la señal de control al transistor MOSFET para que la fuente de alimentación pueda alimentar la bombilla.

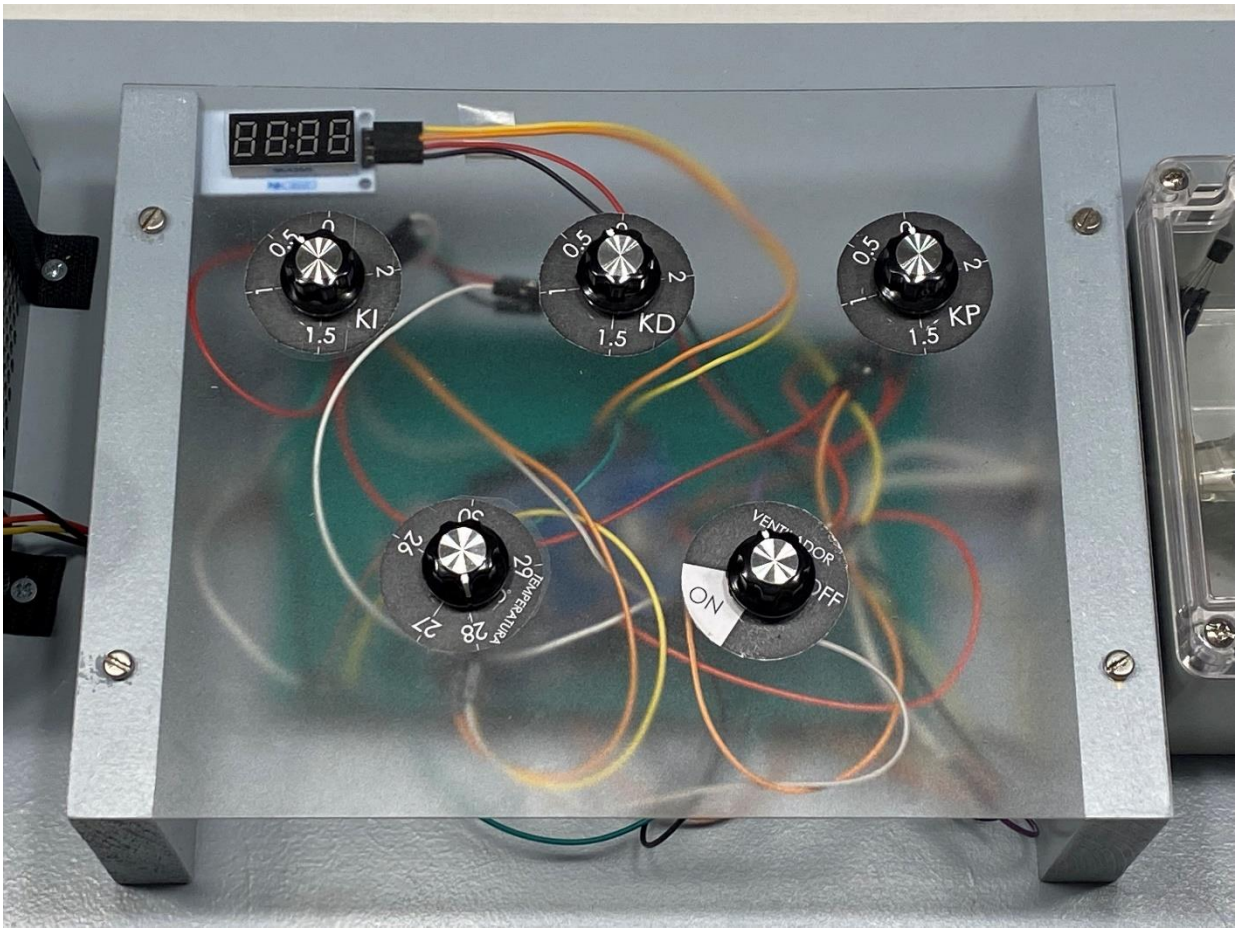


Figura 2.3.Potenciometros, display y debajo la placa Arduino con sus conexiones soldadas

El contenedor, se trata de una caja contenedora Bopla de policarbonato gris de 122 x 120 x 85mm con IP65, con la tapa superior de cristal y junta de neopreno para proteger su interior, en la cual hemos introducido un sensor de temperatura LM35, una bombilla halógena y un ventilador de 12V.

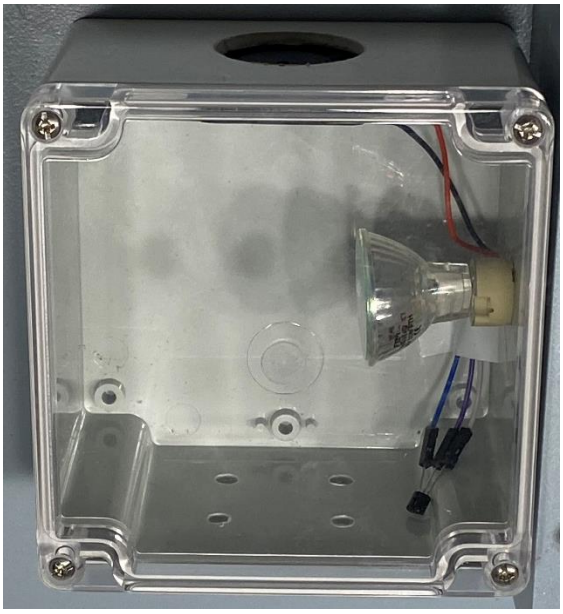


Figura 2.4 Caja con sus componentes

Figura 2.5 Caja con sus componentes

Al seleccionar una temperatura y los parámetros del controlador con los potenciómetros, vemos como varía la iluminación dentro de la caja y como va acercándose la temperatura del interior del contenedor a la deseada, de forma amortiguada.

3 FUNDAMENTOS TEÓRICOS

Para realizar el presente proyecto es necesario introducir unos conceptos teóricos que nos ayudarán a comprender mejor el sistema.

3.1 PID

Un controlador PID [3] proporciona una variación continua de la salida dentro de un mecanismo de retroalimentación de bucle de control para controlar con precisión el proceso, eliminando la oscilación y aumentando la eficiencia.

Estudios empíricos demuestran que un PID es la estructura que por lo general tiene la suficiente flexibilidad como para alcanzar excelentes resultados en muchas aplicaciones, y por ello se emplea en más del 90% de los lazos de control

El algoritmo del PID consta de tres parámetros: el proporcional, el derivativo y el integral. El conjunto de los tres sirve para ajustar un sistema a partir de un elemento de control.

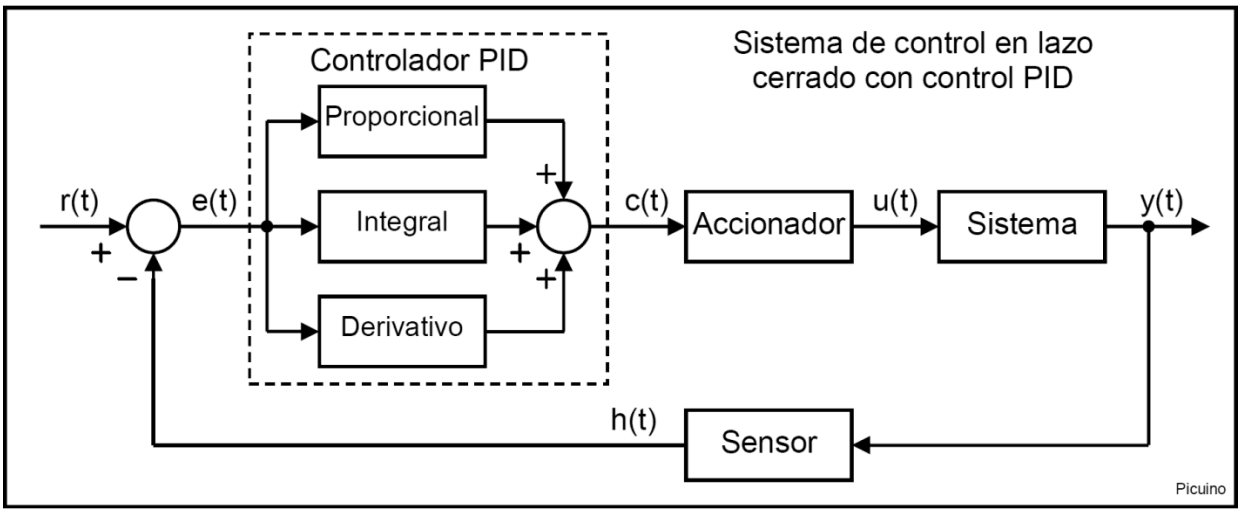


Figura 3.1 Diagrama de bloques de un controlador PID en lazo cerrado

$$u(t) = \underset{P}{K_p e(t)} + \underset{I}{K_i \int_0^t e(\tau)d\tau} + \underset{D}{K_d \frac{de}{dt}} \quad (3.1)$$

3.1.1 P: ACCIÓN DE CONTROL PROPORCIONAL

Da una salida del controlador que es proporcional al error, es decir:

$$u(t) = K_p . e(t) \quad (3.2)$$

donde K_p es una ganancia proporcional ajustable. Un controlador proporcional puede controlar cualquier planta estable, pero posee un desempeño limitado y error en régimen permanente (off-set). La parte proporcional consiste en el producto entre la señal de error y la constante proporcional para lograr que el error en estado estacionario se aproxime a cero, pero en la mayoría de los casos, estos valores solo serán óptimos en una determinada porción del rango total de control, siendo distintos los valores óptimos para cada porción del rango. Sin embargo, existe también un valor límite en la constante proporcional a partir del cual, en algunos casos, el sistema alcanza valores superiores a los deseados. Este fenómeno se llama sobreoscilación y, por razones de seguridad, no debe sobrepasar el 30%, aunque es conveniente que la parte proporcional ni siquiera produzca sobreoscilación.

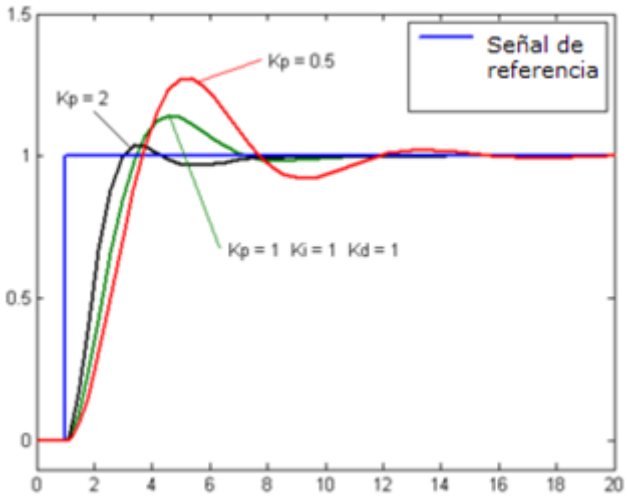


Figura 3.2 Respuesta del sistema con control proporcional

3.1.2 I: ACCIÓN DE CONTROL INTEGRAL

El modo de control Integral tiene como propósito disminuir y eliminar el error en estado estacionario, provocado por perturbaciones exteriores y los cuales no pueden ser corregidos por el control proporcional. El control integral actúa cuando hay una desviación entre la variable y el punto de consigna, integrando esta desviación en el tiempo y sumándola a la acción proporcional. El error es integrado, lo cual tiene la función de promediarlo o sumarlo por un período determinado; Luego es multiplicado por una constante Ki. Posteriormente, la respuesta integral es adicionada al modo Proporcional para formar el control P + I con el propósito de obtener una respuesta estable del sistema sin error estacionario.

La señal de control $u(t)$ tiene un valor diferente de cero cuando la señal de error $e(t)$ es cero. Por lo que se concluye que dada una referencia constante, o perturbaciones, el error en régimen permanente es cero.

$$u(t) = K_i \int_0^t e(\tau)d\tau \qquad (3.3)$$

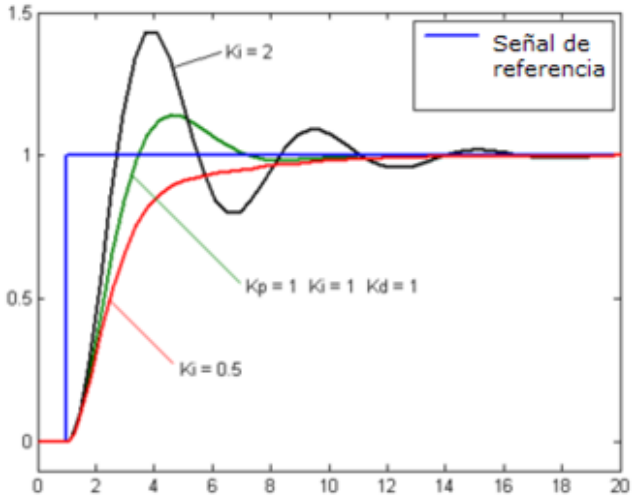


Figura 3.3 Respuesta del sistema con control integral

3.1.3 D: ACCIÓN DERIVATIVA

La acción derivativa se manifiesta cuando hay un cambio en el valor absoluto del error(si el error es constante, solamente actúan los modos proporcional e integral).

El error es la desviación existente entre el punto de medida y el valor consigna, o "Set Point".

La función de la acción derivativa es mantener el error al mínimo corrigiéndolo proporcionalmente con la misma velocidad que se produce; de esta manera evita que el error se incremente.

Cuando el tiempo de acción derivada es grande, hay inestabilidad en el proceso. Cuando el tiempo de acción derivada es pequeño la variable oscila demasiado con relación al punto de consigna. Suele ser poco utilizada debido a la sensibilidad al ruido que manifiesta y a las complicaciones que ello conlleva.

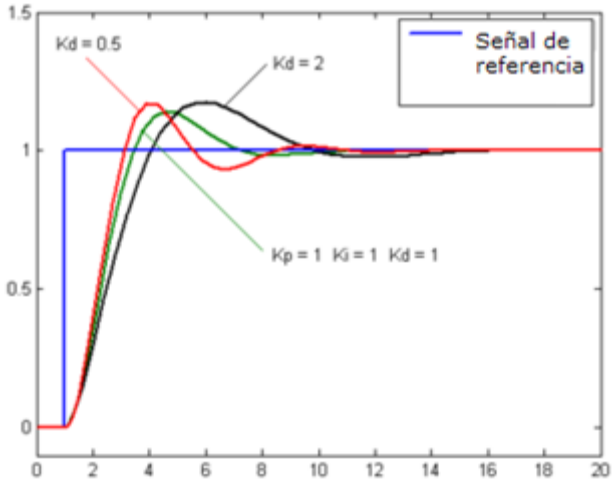


Figura 3.4 Respuesta del sistema con control derivativo

3.1.4 PI: ACCIÓN DE CONTROL PROPORCIONAL INTEGRAL

La función de transferencia resulta:

$$CPI(s) = K_p + \frac{1}{T_i \cdot s} \tag{3.4}$$

Donde Ti se denomina tiempo integral y es quien ajusta la acción integral.

Con un control proporcional, es necesario que exista error para tener una acción de control distinta de cero. Con acción integral, un error pequeño positivo siempre nos dará una acción de control creciente, y si fuera negativo la señal de control será decreciente. Este razonamiento sencillo nos muestra que el error en régimen permanente será siempre cero.

Muchos controladores industriales tienen solo acción PI. Se puede demostrar que un control PI es adecuado para todos los procesos donde la dinámica es esencialmente de primer orden.

3.1.5 PD: ACCIÓN DE CONTROL PROPORCIONAL DERIVATIVA

La función de transferencia resulta:

$$CPD(s) = K_p(1 + T_d \cdot s) \tag{3.5}$$

Esta acción tiene carácter de previsión, lo que hace más rápida la acción de control, aunque tiene la desventaja importante que amplifica las señales de ruido y puede provocar saturación en el actuador. La acción de control derivativa nunca se utiliza por sí sola, debido a que solo es eficaz durante periodos transitorios.

Cuando una acción de control derivativa se agrega a un controlador proporcional, permite obtener un controlador de alta sensibilidad, es decir que responde a la velocidad del cambio del error y produce una corrección significativa antes de que la magnitud del error se vuelva demasiado grande. Aunque el control derivativo no afecta en forma directa al error de estado estacionario, añade amortiguamiento al sistema y, por tanto, permite un valor más grande que la ganancia K, lo cual provoca una mejora en la precisión en estado estable.

3.1.6 PID: ACCIÓN DE CONTROL PROPORCIONAL INTEGRAL DERIVATIVA

Esta acción combinada reúne las ventajas de cada una de las tres acciones de control individuales. La ecuación de un controlador con esta acción combinada se obtiene mediante:

$$u(t) = K_p \cdot e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{de}{dt} \tag{3.6}$$

y su función transferencia resulta:

$$CPID(s) = K_p + \frac{1}{T_i \cdot s} + T_d \cdot s \tag{3.7}$$

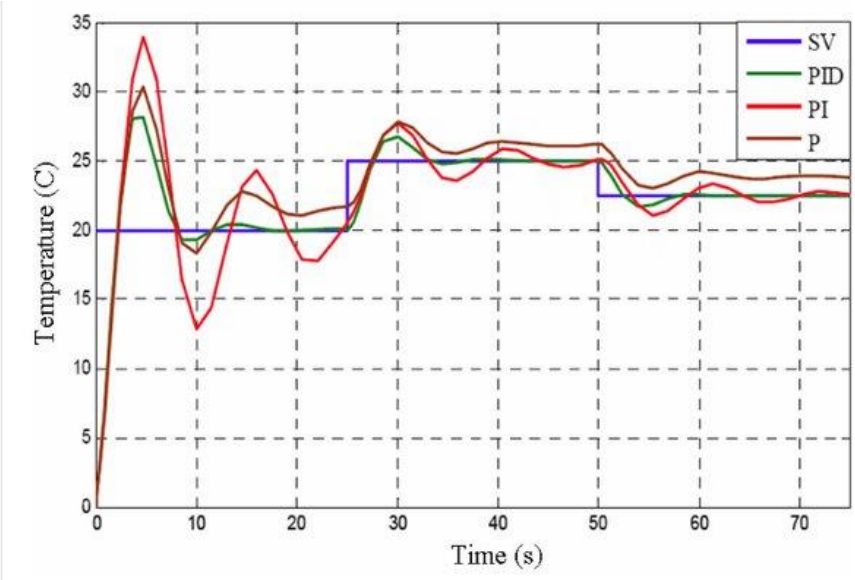


Figura 3.5 Respuesta del sistema con control PID

3.2 AJUSTE DE PARÁMETROS DE UN PID

El objetivo de los ajustes de los parámetros PID es lograr que el bucle de control corrija eficazmente y en el mínimo tiempo los efectos de las perturbaciones; se tiene que lograr la mínima integral de error. Si los parámetros del controlador PID (la ganancia del proporcional, integral y derivativo) se eligen incorrectamente, el proceso a controlar puede ser inestable, por ejemplo, que la salida de este varíe, con o sin oscilación, y está limitada solo por saturación o rotura mecánica. Ajustar un lazo de control [4] significa ajustar los parámetros

del sistema de control a los valores óptimos para la respuesta del sistema de control deseada. El comportamiento óptimo ante un cambio del proceso o cambio del "setpoint" varía dependiendo de la aplicación. Generalmente, se requiere estabilidad ante la respuesta dada por el controlador, y este no debe oscilar ante ninguna combinación de las condiciones del proceso y cambio de "setpoints". Algunos procesos tienen un grado de no linealidad y algunos parámetros que funcionan bien en condiciones de carga máxima no funcionan cuando el proceso está en estado de "sin carga". Hay varios métodos para ajustar un lazo de PID. El método más efectivo generalmente requiere del desarrollo de alguna forma del modelo del proceso, luego elegir P, I y D basándose en los parámetros del modelo dinámico. Los métodos de ajuste manual pueden ser muy ineficientes. La elección de un método dependerá de si el lazo puede ser "desconectado" para ajustarlo, y del tiempo de respuesta del sistema. Si el sistema puede desconectarse, el mejor método de ajuste a menudo es el de ajustar la entrada, midiendo la salida en función del tiempo, y usando esta respuesta para determinar los parámetros de control.

3.2.1 AJUSTE MANUAL

- Este método consiste en establecer primero los valores de I y D a cero.

- A continuación, incrementamos la constante proporcional, hasta que la salida del lazo oscile.

- Luego establecemos la K_p a aproximadamente la mitad del valor configurado previamente.

- Después incrementamos la variable integral, K_i, hasta que el proceso se ajuste en el tiempo requerido (aunque subir mucho I puede causar inestabilidad).

- Finalmente, incrementamos K_d, si se necesita, hasta que el lazo sea lo

suficientemente rápido para alcanzar su referencia tras una variación brusca de la carga.

3.2.2 MÉTODO DE ZIEGLER-NICHOLS

El método de Ziegler-Nichols permite ajustar o "sintonizar" un controlador PID de forma empírica, sin necesidad de conocer las ecuaciones de la planta o del sistema controlado. Estas reglas de ajuste propuestas por Ziegler y Nichols fueron publicadas en 1942 y desde entonces es uno de los métodos de sintonización más ampliamente difundido y utilizado.

Los valores propuestos por este método intentan conseguir en el sistema realimentado una respuesta al escalón con un sobreimpulso máximo del 25%, que es un valor robusto con buenas características de rapidez y estabilidad para la mayoría de los sistemas.

El método de sintonización de reguladores PID de Ziegler-Nichols permite definir las ganancias proporcional, integral y derivativa a partir de la respuesta del sistema en lazo abierto o a partir de la respuesta del sistema en lazo cerrado. Cada uno de los dos ensayos se ajusta mejor a un tipo de sistema.

- **Método Ziegler-Nichols en lazo abierto:**

En este método la planta recibirá en su entrada una entrada escalón unitario y a partir de la respuesta en la salida se obtendrán los parámetros del controlador PID. Para poder usar este método la respuesta tiene que ser una curva de forma S o sigmoidal, es decir, que la respuesta no puede tener impulsos en lazo abierto.

La curva S depende del retardo en el tiempo L y por la constante de tiempo τ, cuya **función de transferencia es:**

$$G_p(s) = \frac{Ke^{-L \cdot s}}{\tau \cdot s + 1} \tag{3.8}$$

Por lo tanto, podemos sacar los parámetros de retardo y la constante de tiempo dibujando la tangente en el punto de inflexión de la curva sigmoidal y determinando las intersecciones de la línea tangente con el eje del tiempo y el eje donde $c(t) = K$ como se muestra:

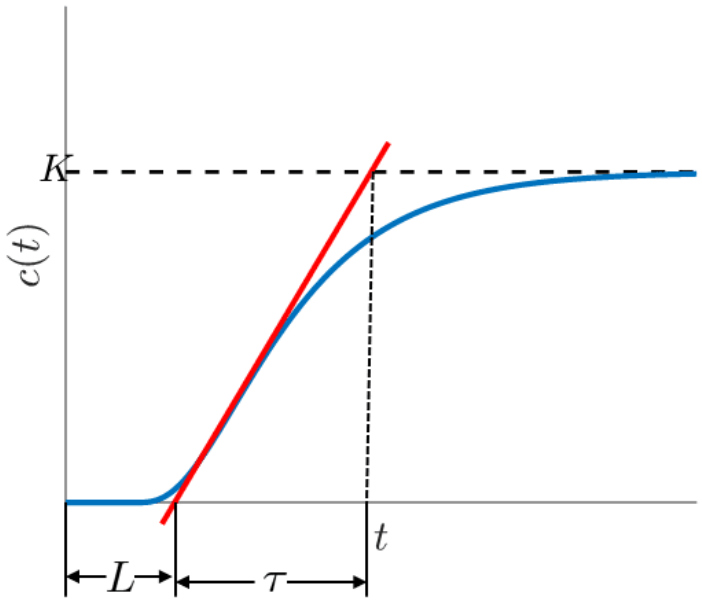


Figura 3.6 Parámetros en la curva sigmoidal

De ahí, Ziegler y Nichols sugirieron un ajuste del PID de acuerdo con la tabla 3.1:

Tabla3.1 Tabla ziegler-Nichols lazo abierto

	K_p	T_i	T_d
P	$\frac{\tau}{K \cdot L}$	∞	0
PI	$0.9\frac{\tau}{K \cdot L}$	$\frac{L}{0.3}$	0
PID	$1.2\frac{\tau}{K \cdot L}$	$2 \cdot L$	$0.5 \cdot d$

Finalmente, sustituimos valores en la ecuación del PID:

$$Gc(s) = K_p(1 + T_i s + T_d s) \tag{3.9}$$

-<u>Método Ziegler-Nichols en lazo cerrado:</u>

Comenzamos poniendo el parámetro integral y el derivativo a cero, y vamos aumentando el proporcional lentamente, hasta conseguir una respuesta oscilatoria de amplitud constante.

Esa ganancia que tenemos en el momento que la respuesta oscila con amplitud constante, es la ganancia crítica K_u y a partir del gráfico sacamos el periodo crítico P_u, o bien, con la frecuencia crítica ω_u:

$$P_u = 2\pi/\omega_u \tag{3.10}$$

Con estos parámetros, encontraremos los valores de las constantes de nuestro PID con la siguiente tabla:

Tabla3.2 Tabla ziegler-Nichols lazo cerrado

	K_p	T_i	T_d
P	$0.5K_u$	∞	0
PI	$0.45K_u$	$\frac{P_u}{1.2}$	0
PID	$0.6K_u$	$0.5P_u$	$0.125P_u$

Finalmente reemplazamos los valores en la ecuación del PID.

4 DESCRIPCIÓN Y ANÁLISIS DEL EQUIPO

4.1 ARDUINO

Se trata de uno de los tipos de placas más usados en proyectos de electrónica, el cual no tiene un solo modelo, si no que ofrece un hardware abierto para que otros fabricantes puedan crearlas.

Arduino es una plataforma de creación de electrónica de código abierto, basada en el hardware y software libre, la cual cuenta con todos los elementos necesarios para conectar periféricos a las entradas y salidas del microcontrolador.

Arduino es un circuito integrado en los que se puede escribir un código de programación en el entorno IDE(lenguaje C++) para grabar unas instrucciones que hacen interaccionar los circuitos de la placa. Se utilizan las entradas para mandar información desde los periféricos a la placa, el cual se encargará de utilizar esta información para mandar mediante las salidas, información a otros periféricos.

Los proyectos Arduino pueden ejecutarse sin estar conectados a un ordenador, ya que una vez cargado el código con sus librerías en la placa solo será necesario conectarlo a la placa.

En este caso, se ha programado un código desde el ordenador y se ha subido a Arduino para que este trabaje independientemente, alimentando dispositivos y tomando decisiones de acuerdo con el programa y con la ayuda de sensores y actuadores.

Debido al número de entradas y salidas que tenemos en nuestro sistema, hemos escogido una placa Arduino One la cual cuenta con 14 pines digitales y 6 pines analógicos, programables a través de un USB tipo B.

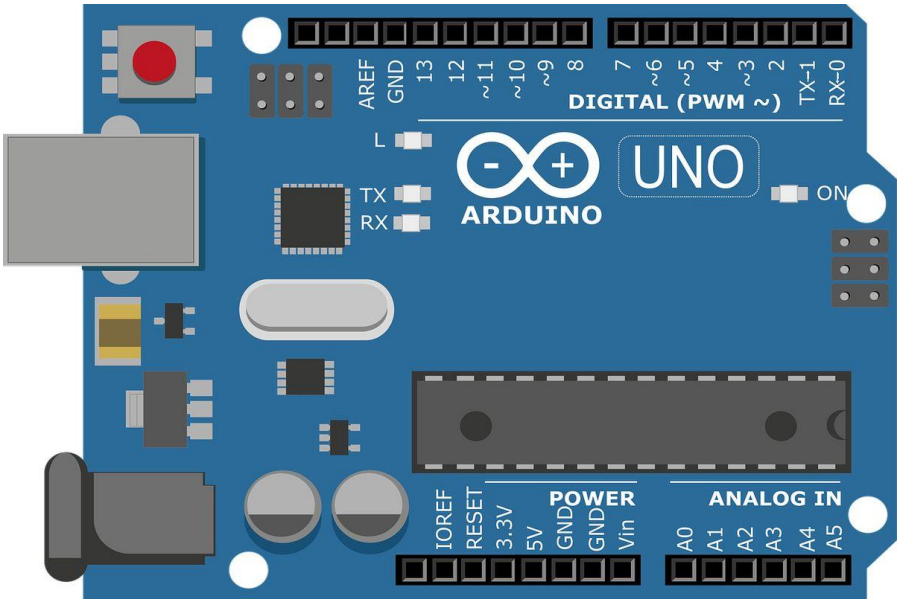


Figura 4.1 Placa Arduino One

Esta placa tiene una memoria Flash de 32k bytes, una memoria SRAM de 2k bytes, la EEPROM de 1k byte y un oscilador de cristal de 16 MHz.

Las señales de salida del sistema serán PWM(pulse-width modulation) de una señal es una técnica en la que se modifica el ciclo de trabajo de una señal periódica, ya sea para transmitir información a través de un canal de comunicación o para controlar la cantidad de energía que se envía a una carga. El ciclo de trabajo es la relación del tiempo que permanece la señal en estado activo y el periodo completo de la señal.

En este tipo de señales es posible variar el tiempo que la señal se mantiene en estado alto, pero siempre manteniendo el periodo constante.

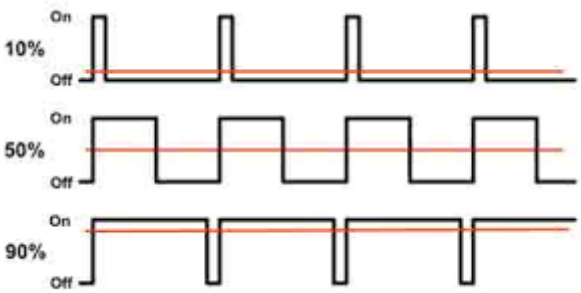


Figura 4.2 Señales PWM

Gracias a esto es muy sencillo controlar el voltaje que sale de un pin, por ejemplo, para controlar la temperatura de la bombilla, que es la que nosotros necesitamos. Esta aplicación es imprescindible ya que estamos alimentando la señal con una tensión fija de 0 o 5 V y acabamos controlando una variable con valores intermedios entre 0-255.

4.2 SENSOR

Para medir la temperatura que hay en el interior de la caja, necesitamos un sensor [5], el cual es un dispositivo que capta magnitudes físicas del exterior u otras alteraciones del entorno, y las transforma en una señal eléctrica que llega hasta un sistema electrónico.

Los diferentes tipos de sensores de temperatura son:

-Termopares:

El termopar es el sensor mas utilizado en los sistemas de medición de temperatura, ya que son precisos, baratos y fáciles de instalar.

Figura 4.3 Sensor termopar

El funcionamiento de los termopares, se basa en dos hilos metálicos de diferentes materiales unidos por el mismo extremo, llamado junta caliente o junta de medición. También cuenta con otro extremo, llamado junta fría.

La diferencia de temperatura entre ambas, produce una diferencia de potencial, que será la señal enviada de nuestro sensor al dispositivo electrónico.

Los más comunes serían:

-Tipo T: Una alambre de cobre y constatan. Tiene un rango de temperatura de entre los -250ºC y los 350ºC
-Tipo J: Una combinación de hierro y constatan. Tiene un rango de temperatura de entre los 0ºC y los 750ºC
-Tipo K: Una junta de chromega y otra de alomega. Tiene un rango de temperatura de entre los -200ºC y los 1100ºC
-Tipo E: Una combinación de chromega y constatan. Tiene un rango de temperatura de entre los -200ºC y los 900ºC

-Sensores RTD:

Las PT100 y PT1000 son sensores los cuales se basan en la resistencia a la temperatura del material del que está compuesto. Se suelen componer de un alambre enrollado con un núcleo de vidrio o cerámica a su alrededor.

Figura 4.4 Sensor RTD

Este tipo de sensores suele utilizarse para mediciones en ámbitos industriales, gracias a su protección ante el ruido eléctrico. El sensor PT100 tiene rango de temperaturas de -200ºC a 850ºC.

Suele estar compuesta para leer sondas de tres hilos, de esta manera se evita en gran parte el error de medida debido a las resistencias de los cables.

Su nombre hace referencia a los 100 ohms de resistencia que tiene la PT100 a 0ºC, su resistencia irá aumentando a medida que aumente la temperatura.

-Termistores NTC y PTC:

Se trata de un sensor compuesto de materiales semiconductores cuya resistencia a la temperatura varía con los grados de esta misma, debido a la

variación de la concentración de portadores.

Figura 4.5 Sensores NTC y PTC

Para los termistores NTC, al aumentar la temperatura, aumentará también la concentración de portadores, por lo que la resistencia será menor, de ahí que el coeficiente sea negativo. Para los termistores PTC, en el caso de un semiconductor con un dopado muy intenso, este adquirirá propiedades metálicas, tomando un coeficiente positivo en un margen de temperatura limitado.

-Sensores infrarrojos sin contacto

Este tipo de se sensores, son usados en sistemas de medición a los cuales es inaccesible llegar, están en movimiento o en aplicaciones donde el sensor debe ser aislado de la superficie de medición. El diseño más básico de un sensor infrarrojo consiste en una lente para enfocar los rayos infrarrojos de energía a un pirómetro, que convierte la energía en una señal eléctrica.

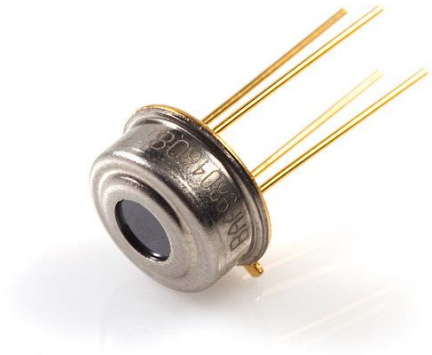


Figura 4.6 Sensor infrarrojo

Tiene un rango de temperaturas de -20ºC a 2000ºC.

-LM35:

En este caso nosotros usaremos un sensor LM35, el cual, es un sensor de temperatura con una precisión calibrada de 1ºC. Lo más relevante es que es muy económico y que ya está calibrado directamente en Celsius, por lo que la tensión es proporcional a la temperatura. Además, tiene una precisión garantizada de 0.5ºC a 30ºC y baja impedancia de salida, lo que hace posible que este integrado se instale en circuitos de control.

Figura 4.7 Sensor LM35

Debido a su baja corriente de alimentación se produce un efecto de auto calentamiento muy reducido. Su rango de medición es de -55ºC hasta 150ºC.

4.2.1 FILTRO DEL SENSOR

Un sensor analógico no suele ser del todo preciso, y es muy sensible a ruidos. Lo que se puede hacer es aplicarle un filtro que minimice el error de medida.

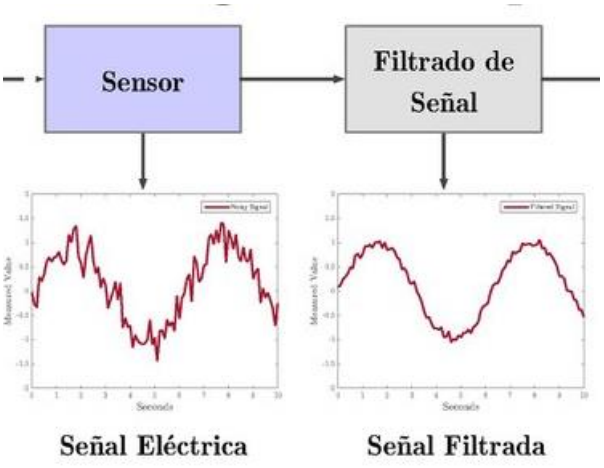


Figura 4.8 Señal filtrada

La señal es filtrada por medio de software, creando una función que, con los valores obtenidos, consiga un conjunto de medidas que sean más precisas y que tengan menos ruido para poder controlar el sistema de forma correcta.

```
for (int i = 0; i < 140; i++) {
  SENSOR = analogRead(A5);
  TEMPERATURA = ((SENSOR * 5000.0) / 1023) / 10;
  SUMA = TEMPERATURA + SUMA;
  delay(5);
  //
}
```

Con esta función la señal de medida recibida por la entrada analógica A5 será convertida a grados Celsius y sumada en la variable 'SUMA' cada 5 nanosegundos, hasta 140 veces.

Hacemos la media dividiendo ese valor entre las 140 iteraciones. Y el valor obtenido es nuestra medida filtrada mediante software por el método de la media.

El número de iteraciones y el tiempo entre ellas se han obtenido experimentalmente, viendo cuáles eran los que daban mejor resultados en el sistema y en las mediciones.

4.2.2.- CALIBRACIÓN DEL SENSOR

Este sensor ya está directamente calibrado a Celsius, y solamente hace falta esta ecuación para tener la temperatura en grados Celsius en el dispositivo:

$$TEMPERATURA = ((SENSOR * 5000.0) / 1023) / 10; \quad (4.1)$$

4.3 DISPLAY 7 SEGMENTOS

El display usado en este proyecto es el TM1637, el cual es el idóneo, pues el resto de displays suele ser muy tedioso el conexionado, pues tenemos que multiplexar y realizar gran cantidad de conexiones.

Figura 4.9 Display TM1637

Este módulo está compuesto por varios componentes con el fin de reducir al mínimo las conexiones con la protoboard, y ya incluye el multiplexor y los circuitos latchs.

El TM1637 solo necesita 4 conexiones, dos de ellas sirven de alimentación y las otras dos son la señal de reloj y la de datos. Los datos se ingresan al módulo por medio de comunicación serial, de ahí que sólo un pin es necesario para datos, mientras tanto el pin de reloj define el tiempo en que se envían tales datos.

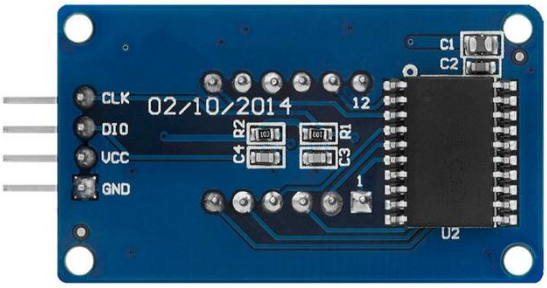


Figura 4.10 Parte trasera Display TM1637

Las conexiones del módulo son:

Tabla 4.1 Conexiones del módulo

TM1637 4-Digit Display	Arduino
VCC	5 V
GND	GND
CLK	Digital pin 2
DIO	Digital pin 3

El fabricante nos da una librería("TM1637.h") para instalar en Arduino IDE con el fin de poder mandar datos a nuestro display de la forma más sencilla posible.

4.4 CAJA BOPLA

Bopla es uno de los mayores fabricantes de cajas de plástico del mundo

para productos industriales y electrónicos. Hemos escogido esta caja por su acabado y por tener la tapa superior de cristal, para poder visualizar tanto los componentes del interior como la luz que arroja la bombilla al calentar con más o menos intensidad.

Figura 4.11 Caja contenedora

La parte inferior de la caja es de policarbonato,IP65.

A esta caja le hemos hecho un agujero de 3.7 cm de diámetro en una de sus caras para el ventilador. En la cara contigua le hemos hecho dos agujeros de 1cm y 0.7 cm para la salida de los cables de la bombilla halógena, del sensor de temperatura LM35 y del ventilador BLS12/40.

En la cara opuesta al ventilador, hemos hecho cuatro agujeros de 0.8cm cada uno, para que el sistema tenga una salida del aire que entra por el ventilador, creando un flujo de aire pequeño pero constante con la temperatura ambiente del exterior, consiguiendo enfriar más rápido.

4.5 ACTUADOR

Un actuador es un componente que emplea la energía hidráulica, neumática o eléctrica recibida para activar el funcionamiento de un proceso automatizado. Este recibe la orden de un regulador o controlador y en función a ella genera la orden para activar un elemento final de control, como por ejemplo una bombilla.

En nuestro caso el actuador será una bombilla, de forma que si quiere calentar se encenderá y si no se apagará.

Por precio y sencillez hemos elegido como foco de calor una bombilla halógena, la cual gasta casi toda su potencia en dar calor. Además se encienden al instante y tienen gran capacidad de iluminación, con lo que será muy sencillo e intuitivo comprobar visualmente como el sistema maneja el calor que da la bombilla de forma que controle la temperatura final del contenedor.

Figura 4.12 Bombilla halógena

Hemos optado por una bombilla de 15 W, ya que experimentalmente hemos visto que era la optima por el tiempo que tardaba en calentarse.

El único inconveniente es que para funcionar necesita estar conectado a 12V o 24V, así que necesitamos una fuente de alimentación y un sistema de potencia para que se ilumine y poder controlarla.

El sistema de potencia se encarga de controlar el voltaje que le llega a la bombilla de 12V mediante una placa Arduino One de 5V, esto se consigue con la ayuda de un transistor MOSFET.

Figura 4.13 Transistor MOSFET

Un transistor MOSFET (metal-oxide-semiconductor field-effect transistor) es un transistor utilizado para amplificar o conmutar señales electrónicas. Un transistor MOSFET tiene 3 patillas, este conduce corriente eléctrica entre dos de ellas cuando aplicamos tensión en la otra patilla. En este caso lo estamos usando como un interruptor que se activa por tensión.

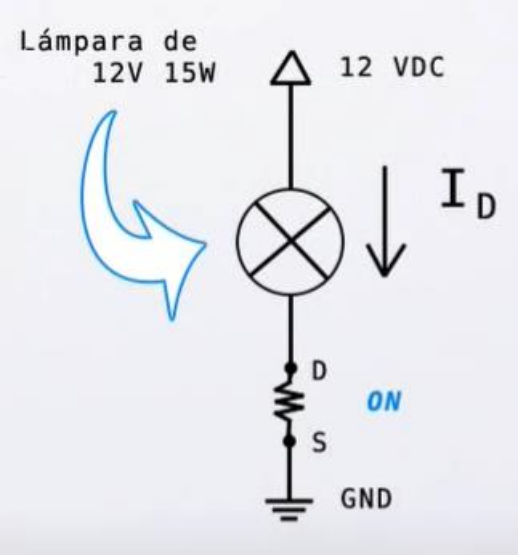


Figura 4.14 Circuito de bombilla con equivalente MOSFET

Este sería nuestro circuito donde tendríamos nuestra bombilla de 15W 12V conectada a su tensión y pasando por el MOSFET, el cual estando encendido equivale a 0.2 Ω, luego haciendo cálculos:

$$V_{gs} = 5V;$$

$$R_{ds} = 0.2\Omega;$$

$$I = \frac{P}{R} = \frac{15W}{12V}; \tag{4.2}$$

$$I_d = 1.25A;$$

$$V = I_d * R_{ds}(on) = 1.25A * 0.2\Omega;$$

$$\mathrm{V} = 0.25\mathrm{V} \quad => \quad \text{Caida de tensión en MOSFET}$$

$$V_{bombilla} = 12V - 0.25V = 11.75V$$

4.6 FUENTE DE ALIMENTACIÓN

Figura 4.15 Fuente alimentación

Es muy importante tener todos los componentes bien alimentados a la tensión necesaria, al usar una bombilla halógena de 12V, necesitaremos una fuente de alimentación adicional puesto que la placa Arduino solo alimenta a 3.3V y 5V.

Para ello hemos cogido la fuente de alimentación de un ordenador antiguo la cual alimenta a la tensión deseada.

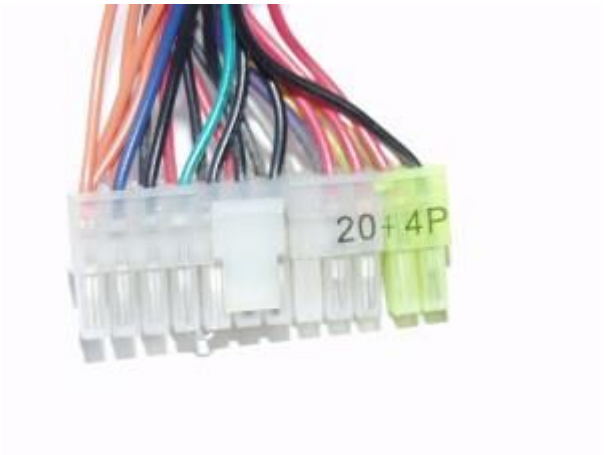


Figura 4.16 Cableado fuente alimentación

En la fuente de alimentación de un ordenador antiguo podemos encontrar los siguientes cabes principales:

Negro: Negativo GND

Naranja: +3.3 V

Rojo: +5 V

Amarillo: +12 V

Verde: Encendido

Para poder encenderla necesitamos cortocircuitar el cable verde y tierra, simulando que hemos pulsado el botón de encendido del ordenador.

4.7 VENTILADOR

El sistema solo es capaz de calentar pero no de enfriar, por lo que se ha colocado un ventilador en uno de los lados de la caja, mejorando el paso de aire e intercambio de temperatura entre interior y exterior.

Figura 4.17 Ventilador 12V

A su vez sirve de perturbación para la bombilla, la cual en función de si el ventilador esta al 100% de velocidad o no, tendrá que calentar más o menos, para conseguir la temperatura deseada.

Este ventilador es de 12V 100mA, por lo que está alimentado por la fuente de alimentación. Su velocidad se controla mediante un potenciómetro, el cual tiene una resistencia de 10kΩ.

4.8 POTENCIÓMETRO

Un potenciómetro [6] es un elemento que permite variar su resistencia al paso de la corriente eléctrica, en función de la posición del terminal móvil o perilla.

En electrónica, esta resistencia posee tres terminales a los cuales se conectan el voltaje a medir. Al hacer variar la resistencia varía la diferencia de

potencial entre los terminales.

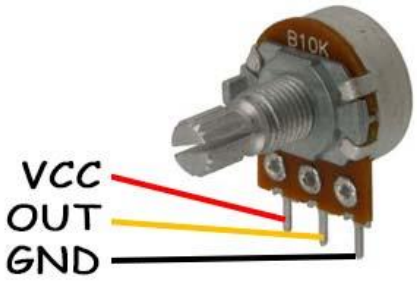


Figura 4.18 Potenciómetro de eje ranurado

El valor de un potenciómetro viene dado en ohmios y representa el valor máximo de resistencia que puede llegar a tener.

Dos patillas se usan para alimentación, mientras que la central es la salida del valor con la resistencia, en este caso, resistencia variable de 10kΩ.

En este proyecto, hemos utilizado 4 potenciómetros para introducir señales al Arduino, como la temperatura, la K_p, la K_d o la K_i. A su vez, hemos utilizado uno para controlar el voltaje que le llega al ventilador, y así poder cambiar su tensión de entrada.

La placa Arduino UNO tiene 6 pines analógicos, los cuales se suelen usar para lectura de datos, como en nuestro caso. Estas entradas tienen una resolución de 10 bits, lo que implica que podemos leer 1024 valores diferentes.

Leeremos en un rango de entre 0V y 5V detectando cambios de voltaje de 0.004V.

$$\frac{5V}{1024} = 0.004V \qquad (4.3)$$

5 CONEXIONADO Y MECANIZADO

Para facilitar el entendimiento del sistema con cada uno de los elementos, se va a proceder a dibujar de una manera esquemática los circuitos de los elementos más importantes del proyecto. A su vez también voy a mostrar unos planos de la caja contenedora, en la cual se produce el control de temperatura.

Circuito de la bombilla de 12 V con el MOSFET

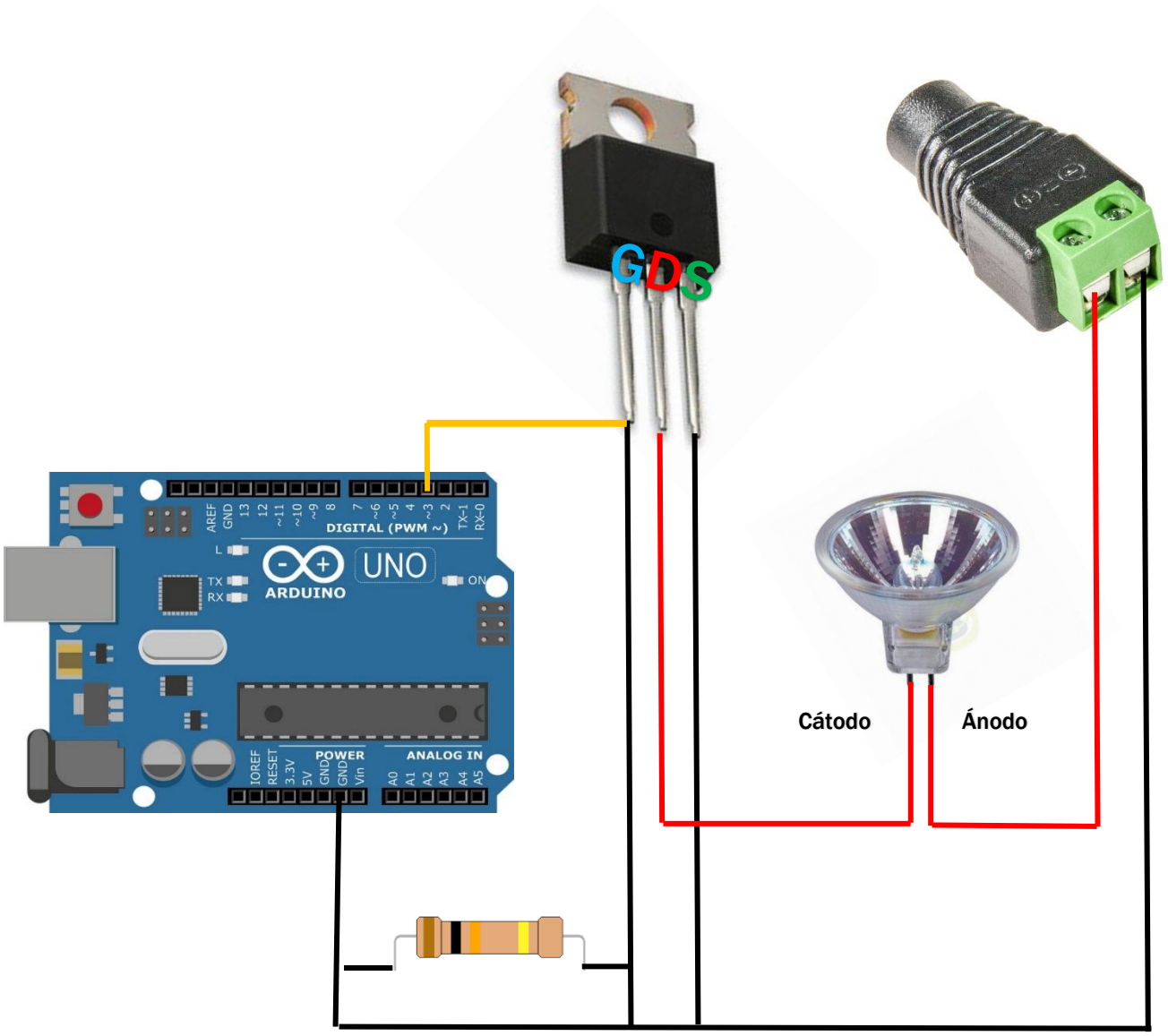


Figura 5.1 Circuito de potencia para la bombilla

En este circuito nos encargamos de controlar la tensión que le llega a la bombilla desde los 0V a los 12V, mediante la placa Arduino, que tiene 5V.

Apoyándonos en un transistor MOSFET, crearemos un circuito en el cual tiene la función de interruptor o conmutador, ya que en función de la tensión que le llegue al MOSFET pasará más corriente o menos por la bombilla, es decir, en función de la tensión, que llegue a la patilla G del MOSFET, conducirá más corriente habrá entre sus dos patillas.

5.1 CIRCUITO POTENCIÓMETRO:

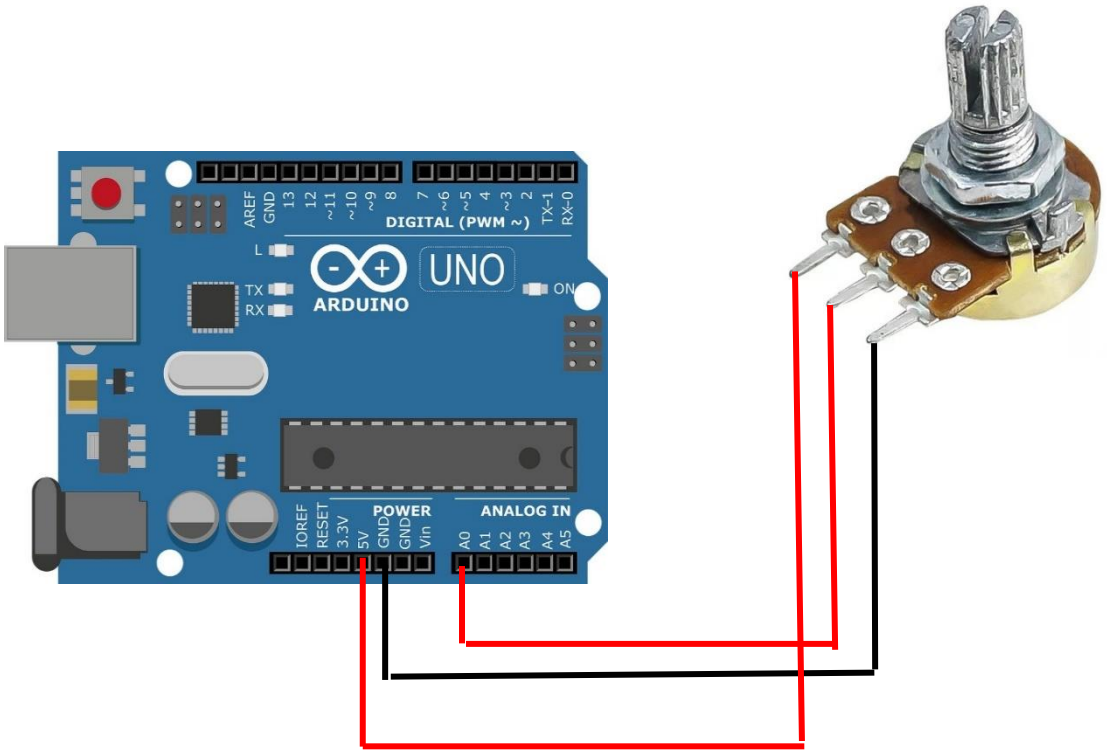


Figura 5.2 Circuito para los potenciómetros

En este caso tenemos un potenciómetro conectado a la alimentación y con la patilla de salida enviamos la señal a nuestra placa.

Dependiendo de cuanto hayamos girado la parte móvil del potenciómetro mandaremos una señal u otra.

Esto lo usaremos para enviarle a la placa la temperatura deseada, la K_p, la K_d o la K_i, para que el sistema pueda ser 'controlado' por el usuario o alumno.

5.2 CIRCUITO LM35:

En este montaje tenemos el sensor LM35(sensor de temperatura), el cual tiene tres patillas, la 1 y 3 van conectadas a Vcc y Gnd respectivamente, mientras la patilla 2 es la salida del sensor, que manda la información al arduino mediante una entrada analógica

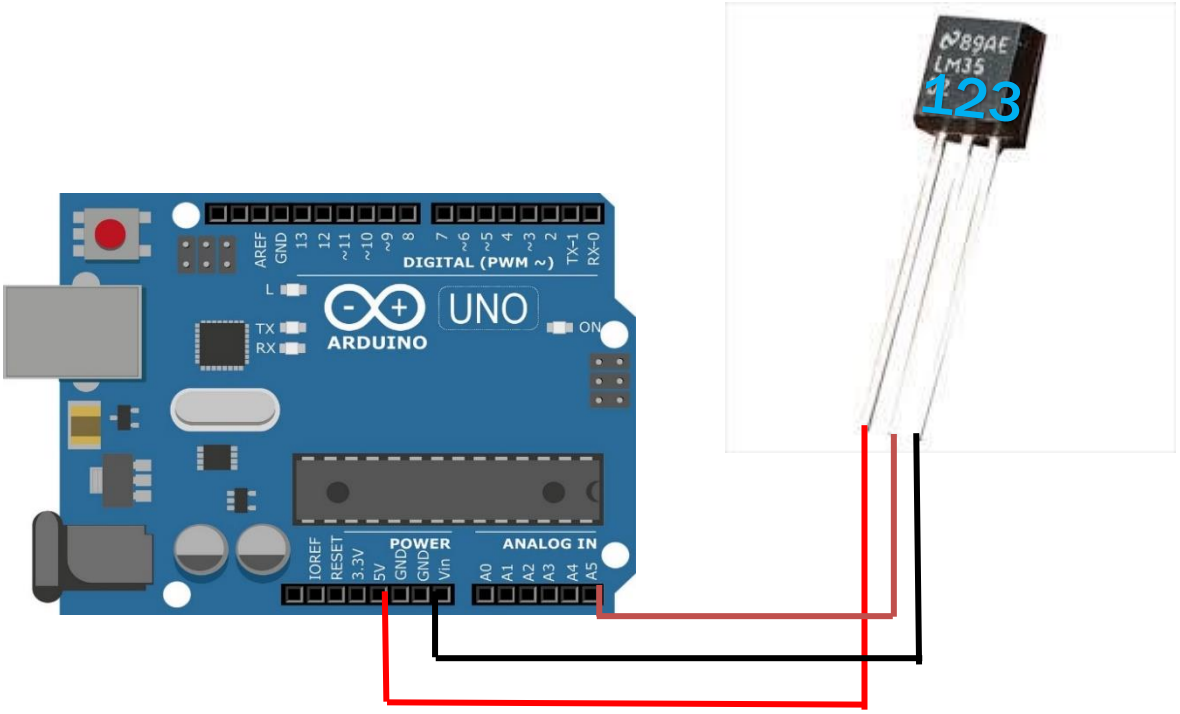


Figura 5.3 Circuito LM35

5.3 CIRCUITO PANTALLA 7 SEGMENTOS

La primera patilla CLK va conectada al pin de salida PWM 2, y la segunda patilla DIO va conectada al pin de salida PWM 5, por esta se transmiten los datos de los dígitos que tiene que muestrear el display.

Los otros dos cables, van directos a la alimentación, es decir, a 5V y tierra.

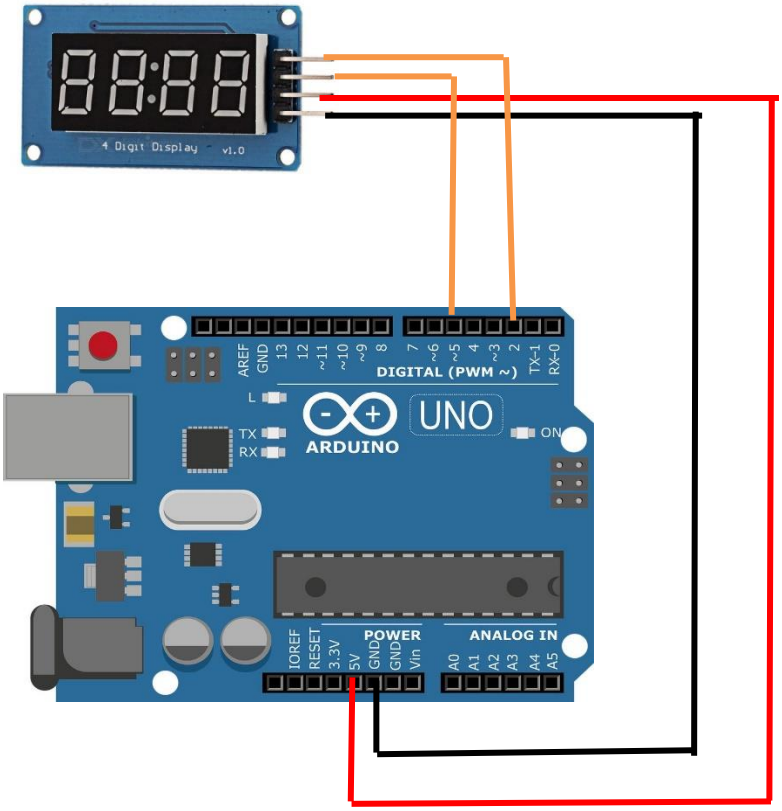


Figura 5.4 Circuito display 7 segmentos

5.4 MECANIZADO DE LA CAJA

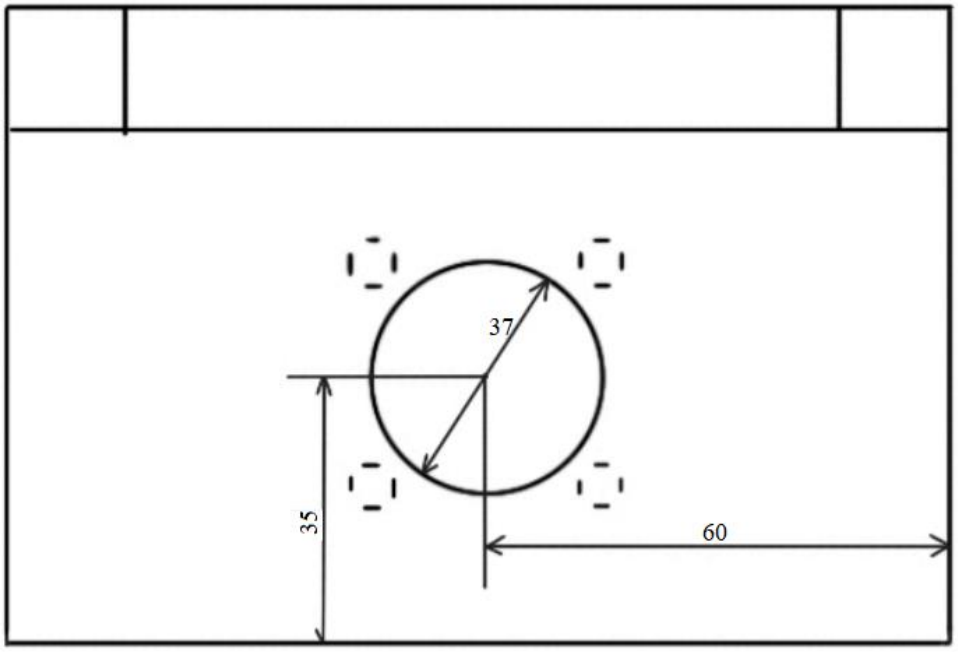


Figura 5.5 Alzado caja bopla, con agujero para el ventilador, en la parte trasera tiene 4 agujeros de 7mm

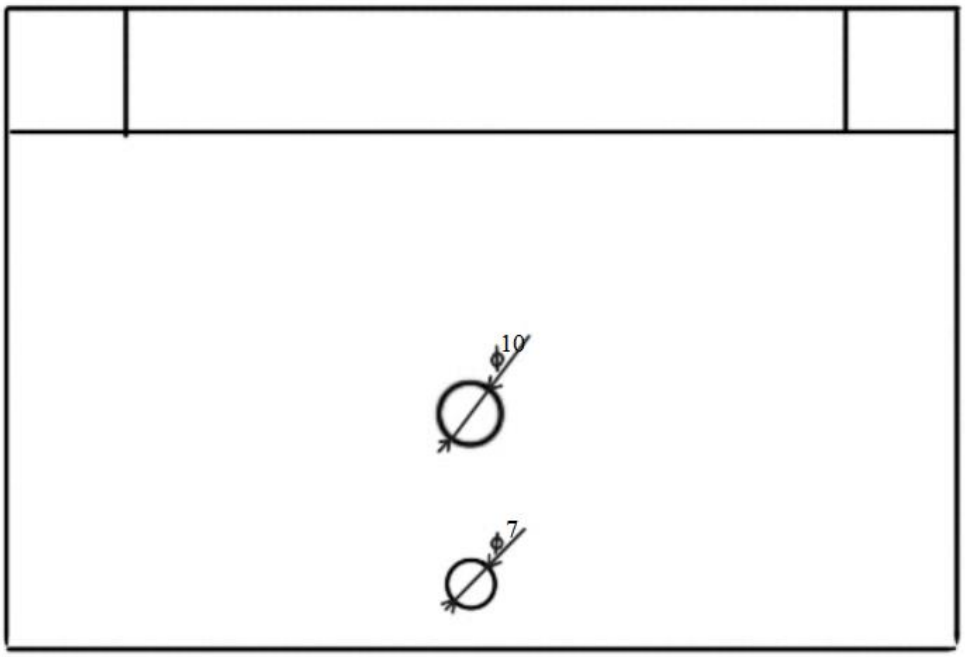


Figura 5.6 Perfil caja bopla, agujeros de salida para los cables

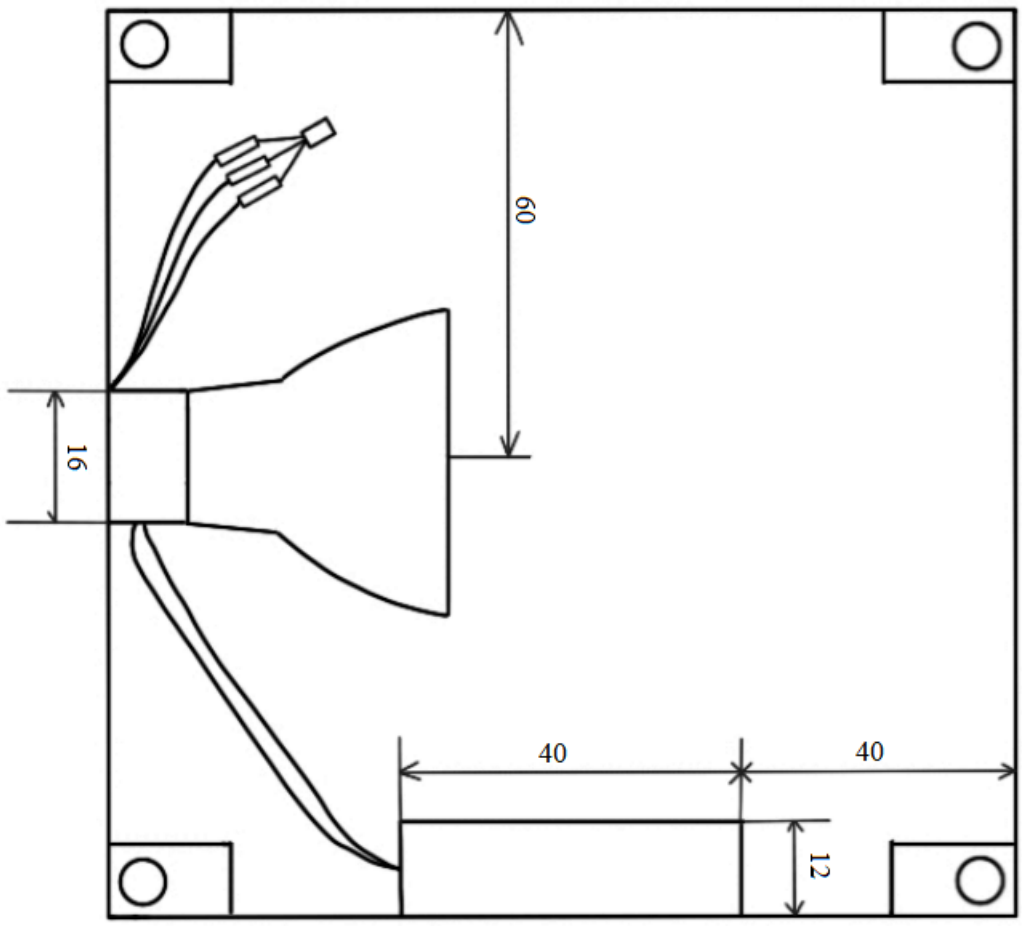


Figura 5.7 planta de caja bopla, con el sensor, ventilador y bombilla dentro

Todas las medidas en milímetros.

Este mecanizado se ha realizado mediante taladros de diferentes diámetros y unalija para mejorar el acabado.

6 CÓDIGO ARDUINO

El código Arduino es una serie de comandos de programación que dirigirán a nuestro microcontrolador como configurarse al iniciarse y qué acciones tiene que realizar mientras esté funcionando. Estos comandos utilizados en Arduino son sentencias muy fáciles e intuitivas.

El ambiente de programación Arduino es realmente similar a C++, asumiendo parámetros relativos al controlador, para facilitar su programación. El código siempre debe estar dividido en dos partes:

- La función setup():

En esta parte programaremos el código, para que el Arduino trabaje correctamente.

Las sentencias que declaremos se ejecutarán solamente al iniciar Arduino. Aquí estableceremos la entrada/salida de los pines, elegiremos órdenes de inicio del programa, inicializaremos variables…etc.

- La función loop():

Esta debe ir siempre detrás de la función setup(), en ella escribiremos los bucles, sentencias y llamadas a funciones para que el Arduino pueda ejecutar las acciones del programa.

A la hora de programar tenemos que empezar siempre instalando y declarando las librerías necesarias. En este caso, sería instalar la librería del display de 7 segmentos TM1637, la cual nos facilita el fabricante. Posteriormente, definimos las salidas, que en nuestro caso será, la salida 2 y 5 serán el pin del temporizador y de datos del display y la 3 será, el valor de tensión que queremos mandarle a la patilla G del MOSFET.

```
3 #define PWM_pin 3
4 #define CLK 2
5 #define DIO 5
```

Figura 6.1 Definimos las outputs

Dentro de la función setup(), declaramos todas las variables globales que vamos a usar en este programa, y las inicializamos como convenga.

```
8 //delaramos variables globales
9 float elapsedTime, Time, timePrev;
10 //parámetros recogidos por potenciometros
11 int kp = 0;   int ki = 0;   int kd = 0; int kp1;int kd1;int ki1;
12 //valores para el PID
13 int PID_P = 0;    int PID_I = 0;    int PID_D = 0;
14 float PID_value, PID_error = 0, previous_error = 0;
15 //variables que guardan la temperatura deseada y la real.
16 float SENSOR;
17 float TEMPERATURA, TEMPERATURA0, TEMPERATURAS;
18 //variables  para luego hacer operaciones
19 float SUMA;
20 int valor = 0;
```

Figura 6.2 Definimos las variables a usar

Obtenemos los valores de K_p, K_d y K_i introducidos por el alumno mediante potenciómetros, a las entradas analógicas A0 , A1 y A2, mediante la función 'analogread()'. Los cuales, debido a la resolución de 10 bits de Arduino sus valores serán entre 0 y 1024, como nosotros queremos que las constantes vayan de 0 a 2, dividiremos el valor obtenido por el Arduino entre 512, y así obtener el rango que queremos.

Posteriormente leeremos el valor de temperatura deseada que se introducirá por medio de un potenciómetro, a la entrada analógica A4 con la

misma función que antes. Al igual que en el anterior caso, tenemos que obtener una fórmula para que el valor que leamos de la temperatura deseada oscile entre 26 y 30 grados.

$$Temperatura_{desesada} = 26 + \frac{señal\ obtenida\ potenciómetro}{256} \quad (6.1)$$

Para obtener la medida del sensor, leemos el valor de tensión que nos llega a la entrada analógica A5, lo pasamos a grados y luego la almacenamos y sumamos en la variable suma, en la cual se sumará la medida del sensor cada 5 milisegundos durante 140 iteraciones, antes de volver a repetir el mismo proceso.

$$Temperatura_{Sensor}\ (ºC) = \frac{señal\ obtenida\ por\ el\ sensor * 500}{256} \quad (6.2)$$

Para reproducir por el display de 7 segmentos la temperatura deseada y la temperatura real, debemos muestrear ambas temperaturas de forma que sea fácil apreciar y entender ambas temperaturas. Esto lo haremos con la función 'display.showNumberDecEx(number, dots, leading_zeros, length, position)', incluida en la librería del display. El primer argumento es el número que quieres mostrar, el segundo argumento te permite iluminar o no los puntos entre los números, el tercer argumento puede utilizarse para activar o desactivar los ceros a la izquierda, el cuarto argumento especifica el número de dígitos a modificar (0-4), y el quinto argumento es la posición en la que se mostrará.

De forma que cuando se muestre la temperatura obtenida mostraremos la temperatura con una cifra decimal en el display y en la segunda cifra decimal haremos que salga una raya en la parte A del 7 segmentos, mientras que para la temperatura deseada se hará de la misma forma solamente que en la segunda cifra decimal, pondremos la raya debajo (en la parte D del 7 segmentos).

```
16 const uint8_t medida[] = {
17   SEG_A
18 };
19 const uint8_t solicitada[] = {
20   SEG_D
21 };
```

```
64  display.showNumberDecEx((SUMA / 140.0) * 10, 0b01000000, false, 3, 0);
65  display.setSegments(medida, 1, 3);
66  delay(1000);
67  display.showNumberDecEx((TEMPERATURAS) * 10, 0b01000000, false, 3, 0);
68  display.setSegments(solicitada, 1, 3);
```

Figura 6.3 Código muestreo por display

Calculamos el PID, para ello calculamos el error y las partes proporcional, derivativa e integral. Y calculamos de nuevo el error para la siguiente iteración.

```
70  PID_error = (TEMPERATURAS - (SUMA / 140));
71  //CALCULAMOS LA KP
72  PID_P = kp * PID_error;
73  PID_I = PID_I + (ki * PID_error);
74  timePrev = Time;
75  elapsedTime = (Time - timePrev) / 1000;
76  PID_D = 0.01 * kd * ((PID_error - previous_error) / elapsedTime);
77
78  PID_value = PID_P + PID_I + PID_D;
79  if (PID_value < 0)
80  {
81    PID_value = 0;
82  }
83  if (PID_value > 255)
84  {
85    PID_value = 255;
86  }
87  analogWrite(PWM_pin, PID_value);
88  previous_error = PID_error;
```

Figura 6.4 Cálculo PID

7 LISTA DE MATERIALES Y COSTE

7.1 RESUMEN DE COMPONENTES

Comenzamos haciendo una lista de los materiales y herramientas usados en la realización de este proyecto.

- Arduino Uno Rev 3
- Protoboard de 830 puntos de conexión, circuito de prueba de 2 Buses
- Ventilador 12V 100mA BLS12/40
- Bombilla halógena de 35W, 12V
- Resistencia 10KΩ
- Cables dupont hembra-macho
- Cables dupont macho-macho
- Sensor de temperatura LM35
- Soldador de estaño
- Filamento de estaño
- Madera para la base y apoyos
- 5 Potenciometros 10KΩ
- Transistor MOSFET
- Caja bopla 122 x 120 x 85 mm
- 5 Perillas para los potenciómetros
- Fuente de alumentación 12V
- Taladro con distintas brocas
- Tornillos y tuercas para la estructura

7.2 COSTE DE MATERIAL

Muchos de estos materiales ya los teníamos o los hemos reciclado de otro proyecto en laboratorio. Sacamos el presupuesto con los componentes que si hemos tenido que comprar.

Tabla7.1 Coste material

Material	**Precio**
Arduino Uno Rev 3	15.95€
Protoboard	1.24€
Ventilador BLS12/40	6.90€
Resistencia 10kΩ	0.35€
Cables dupont hembra-macho	1.69€
Cables dupont macho-macho	1.69€
Sensor de temperatura LM35	1.35€
5 Potenciometros 10KΩ	34.1€
Transistor MOSFET	1.98€
Caja bopla 122 x 120 x 85 mm	31.06€

5 Perillas para los potenciómetros	5.00€
TOTAL	101.31€

8 CONCLUSIONES

- En este proyecto hemos estudiado el control PID de una planta, y lo hemos llevado a cabo en su modelo real. Han sido tres meses de trabajo, en los cuales me he enfrentado a la realización de un proyecto físico y real con los problemas que ello plantea. Como resumen, he puesto en práctica y ampliado los conocimientos aprendidos durante el grado, en especial sobre automática y electrónica.

- He tenido muchos problemas para encontrar una bombilla de 12V que calentase lo suficiente y a la vez que no fuese demasiado, ya que al comienzo del proyecto estuve probando con bombillas incandescentes de coches y también lo intente con la de un frigorífico antiguo, pero calentaban demasiado. Para esta planta da mucho mejores resultados una halógena de 35W, la cual calienta mucho, pero con su recubrimiento resulta idóneo para esta aplicación.

- Hemos aprendido también como establecer el control por realimentación, sobre una variable controlada, cuando una perturbación intenta alterar el sistema y modifica la variable controlada, cuando se activa y desactiva el ventilador.

- También a reprogramar el bootloader [7] de una placa de Arduino, ya que esta dejó de funcionar sin razón alguna e informándome vi que tenía que ser el bootloader. Para reprogramarlo se necesita otra placa Arduino con el bootloader programado y así grabarlo en la que no lo tenía programado o se había desprogramado.

- Para subir códigos a la placa de Arduino, hay que desconectar todos los cables que tenemos conectados en la placa. En el datasheet indica que no podemos tener la patilla 1 ni la 2 de las outputs conectadas para subir archivos, pero al menos en la mía había que desconectar siempre todos, o si no salía este error:

```
avrdude: Version 6.3, compiled on Dec 16 2016 at 13:33:19
         Copyright (c) 2000-2005 Brian Dean, http://www.bdmicro.com/
         Copyright (c) 2007-2014 Joerg Wunsch

         System wide configuration file is "C:\Program Files (x86)\Arduino\hardware\tools\avr/etc/avrdude.conf"

         Using Port                    : COM3
         Using Programmer              : arduino
         Overriding Baud Rate          : 115200
avrdude: stk500_recv(): programmer is not responding
avrdude: stk500_getsync() attempt 1 of 10: not in sync: resp=0x08
avrdude: stk500_recv(): programmer is not responding
avrdude: stk500_getsync() attempt 2 of 10: not in sync: resp=0x08
avrdude: stk500_recv(): programmer is not responding
avrdude: stk500_getsync() attempt 3 of 10: not in sync: resp=0x08
avrdude: stk500_recv(): programmer is not responding
avrdude: stk500_getsync() attempt 4 of 10: not in sync: resp=0x08
avrdude: stk500_recv(): programmer is not responding
avrdude: stk500_getsync() attempt 5 of 10: not in sync: resp=0x08
avrdude: stk500_recv(): programmer is not responding
avrdude: stk500_getsync() attempt 6 of 10: not in sync: resp=0x08
avrdude: stk500_recv(): programmer is not responding
avrdude: stk500_getsync() attempt 7 of 10: not in sync: resp=0x08
avrdude: stk500_recv(): programmer is not responding
avrdude: stk500_getsync() attempt 8 of 10: not in sync: resp=0x08
avrdude: stk500_recv(): programmer is not responding
avrdude: stk500_getsync() attempt 9 of 10: not in sync: resp=0x08
avrdude: stk500_recv(): programmer is not responding
avrdude: stk500_getsync() attempt 10 of 10: not in sync: resp=0x08

avrdude done.  Thank you.

Problem uploading to board.  See http://www.arduino.cc/en/Guide/Troubleshooting#upload for suggestions.
```

Figura 8.1 Error en el IDE arduino

- Hemos conseguido que el sistema alcance la temperatura deseada en régimen estacionario en un tiempo y con un error tolerable. Lo cual será una práctica visualmente educativa los alumnos de la Universidad de Cantabria.

- Personalmente, me ha gustado mucho poder llevar a cabo el control de una planta en físico, teniendo que trabajar con el hardware y unirlo con el software, además de ir viendo día a día cada uno de los cambios que se iban planteando.

9 BIBLIOGRAFÍA

- [1] Fernando Morilla García, *Fundamentos de los controladores PID* http://www.dia.uned.es/~fmorilla/MaterialDidactico/El%20controlador%20PID.pdf
  Fecha de acceso: 20/10/2021

- [2] Arduino, «Arduino,» 2022.
  https://www.arduino.cc/
  Fecha de acceso: 18/10/2021

- [3]C. Kuo, Benjamín, *Sistemas de control automático,7º Edición, Editorial pHH, 1996 , ISBN: 9789688807231*

- [4] Sergio Andrés Castaño, *Todo sobre Ziegler Nichols – Sintonía de Control PID* https://controlautomaticoeducacion.com/control-realimentado/ziegler-nichols-sintonia-de-control-pid/
  Fecha de acceso: 05/12/2021

- [5] srcsl,*TIPOS DE SENSORES DE TEMPERATURA*
  https://srcsl.com/tipos-sensores-temperatura/
  Fecha de acceso: 27/10/2021

- [6]Areatecnologia, *Potenciómetro*
  https://www.areatecnologia.com/electronica/potenciometro.html
  Fecha de acceso: 27/10/2021

- [7]Luis llamas, *USAR ARDUINO PARA REPROGRAMAR EL BOOTLOADER DE OTRO ARDUINO*
  https://www.luisllamas.es/usar-arduino-para-reprogramar-el-bootloader/
  Fecha de acceso: 28/11/2021

Hojas de características

Las hojas de características o datasheets de los componentes utilizados en el desarrollo de este proyecto se encuentran alojados en la capeta adjunta llamada "TFG_datasheets".

ANEXO

Código

```
//Declaro la librería
#include <TM1637Display.h>
//Asignamos a las variables los pines que vamos a utilizar
#define PWM_pin 3
#define CLK 2
#define DIO 5

TM1637Display display(CLK, DIO);

//delaramos variables globales
float elapsedTime, Time, timePrev;

//parámetros recogidos por potenciometros
int kp = 0;   int ki = 0;   int kd = 0; int kp1;int kd1;int ki1;

//valores para el PID
int PID_P = 0;    int PID_I = 0;    int PID_D = 0;
float PID_value, PID_error = 0, previous_error = 0;

//variables que guardan la temperatura deseada y la real.
float SENSOR;
float TEMPERATURA, TEMPERATURA0, TEMPERATURAS;

//variables  para luego hacer operaciones
float SUMA;
int valor = 0;

//Guardamos el segmento superior e inferior en las
//variables medida y solicitada para luego usarlas en el display

const uint8_t medida[] = {
  SEG_A
};

const uint8_t solicitada[] = {
  SEG_D
};

//Inicializamos la placa, declaramos la variable de salida
//y la configuración del display

void setup() {
```

```
  // put your setup code here, to run once:

  Serial.begin(9600);
  pinMode(PWM_pin, OUTPUT);
  display.setBrightness(7);
  display.clear();
  Time = millis();
}

//Bucle

void loop() {
  //Obtenemos los valores de las constantes obtenidas desde
  //los potenciómetros. Y pasamos los valores a un rango de
  //valores de 0 a 2.
  kp1=analogRead(A2);
  kp=(kp1/512);
  kd1=analogRead(A1);
  kd=(kd1/512);
  ki1=analogRead(A0);
  ki=(ki1/512);
  //Podemos ver los valores a tiempo real en el Monitor Serie
  Serial.println(kp);
  Serial.println(kd);
  Serial.println(ki);
  //Obtenemos el valor de la temperatura deseada y pasamos el
  //valor a la variable TEMPERATURAS, el cual ira desde 26 hasta 30
  TEMPERATURA0 = analogRead(A4);
  TEMPERATURAS = (TEMPERATURA0 / 256) + 26;
  //Inicializamos la variable SUMA antes de cada iteración
  SUMA = 0;
  //Obtenemos los valores del sensor cada 5 milisegundos, 140 veces.
  //Almacenamos la suma en la variable SUMA
  for (int i = 0; i < 140; i++) {
    SENSOR = analogRead(A5);
    TEMPERATURA = ((SENSOR * 5000.0) / 1023) / 10;
    SUMA = TEMPERATURA + SUMA;
    delay(5);
    //
  }
  //Mostramos en el display la media de todas esas muestras obtenidas
  //Configuramos el display
  //Mostramos 3 cifras, ocupando los dígitos de la izquierda
  display.showNumberDecEx((SUMA / 140.0) * 10, 0b01000000, false, 3, 0);
  //En el dígito de la derecha ponemos el ségmento superior
  display.setSegments(medida, 1, 3);
  //Lo mostramos durante 1 segundo cada uno
  delay(1000);
  //Mostramos 3 cifras, ocupando los dígitos de la izquierda
  display.showNumberDecEx((TEMPERATURAS) * 10, 0b01000000, false, 3, 0);
  //En el dígito de la derecha ponemos el ségmento inferior
```

```
  display.setSegments(solicitada, 1, 3);
  //Calculamos la diferencia entre temperaturas en grados
  PID_error = (TEMPERATURAS - (SUMA / 140));
  //Calculamos la parte proporcional
  PID_P = kp * PID_error;
  //Calculamos la parte integral
  PID_I = PID_I + (ki * PID_error);
  //Calculamos la parte derivativa
  timePrev = Time;
  elapsedTime = (Time - timePrev) / 1000;
  PID_D = 0.01 * kd * ((PID_error - previous_error) / elapsedTime);
//Sacamos el valor del regulador PID
  PID_value = PID_P + PID_I + PID_D;
  if (PID_value < 0)
  {
    PID_value = 0;
  }
  if (PID_value > 255)
  {
    PID_value = 255;
  }
  //Pasamos el valor a la variable de salida
  analogWrite(PWM_pin, PID_value);
  previous_error = PID_error;
}
```