Trabajo de Fin de Grado

Grado en Ingeniería de las Tecnologías Industriales

Sistema electrónico para el control de temperatura con doble salida para rango de aplicaciones biomédicas

Autor: Alberto Gallego Risco

Tutor: Francisco Perdigones Sánchez

Dpto. Ingeniería Electrónica
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2024

Trabajo de Fin de Grado
Grado en Ingeniería de las Tecnologías Industriales

Sistema electrónico para el control de temperatura con doble salida para rango de aplicaciones biomédicas

Autor:
Alberto Gallego Risco

Tutor:
Francisco Perdigones Sánchez
Profesor titular de Universidad

Dpto. Ingeniería Electrónica, Microsistemas
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2024

Proyecto Fin de Carrera: Sistema electrónico para el control de temperatura con doble salida para rango de aplicaciones biomédicas

Autor: Alberto Gallego Risco

Tutor: Francisco Perdigones Sánchez

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2024

El Secretario del Tribunal

A mi familia

A mis maestros

Agradecimientos

Este Trabajo supone la culminación de años de esfuerzo para conseguir los objetivos que un joven yo se puso en la vida. Ese chico estaría muy feliz de ver como ha terminado todo.

Me gustaría agradecer primero a mi familia, que me han visto crecer y esforzarme, sobre todo a mi madre y a mi hermana, que siempre han estado a mi lado.

También a mis abuelos, que me quieren muchísimo y siempre me apoyan y me comprenden, aunque las cosas no salgan como espero.

Por supuesto, como no mencionar a mis amigos, tanto a los que ya estaban como a los que han llegado durante este viaje a mi vida, por todos los momentos que nos quedan y los que nos quedan por pasar.

En último lugar, muchísimas gracias a mi tutor Francisco por ser tan paciente y hacer que saquemos este proyecto adelante, sin él esto no sería posible.

Resumen

La reciente pandemia provocada por el COVID-19 ha traído consigo la necesidad, cada vez más apremiante, de detectar infecciones víricas. Mediante el desarrollo de PCB-MEMS, este trabajo se puede realizar de manera bastante eficiente.

En el departamento de Electrónica industrial se ha desarrollado un proyecto para poder hacer pruebas biomédicas que requieran de un control preciso de temperatura de manera sencilla gracias a Arduino. Este consiste en colocar la muestra extraída del paciente sobre un calentador y variar la temperatura dentro de un rango establecido. Sin embargo, al variar la temperatura del calentador dentro de un rango muy amplio en poco tiempo, la muestra se calentaba de más y la prueba era muchas veces desechada.

En este proyecto se pretende realizar un control PID de temperatura mediante una placa PCB conectada a Arduino. La finalidad de hacer esto es la posibilidad de calentar y enfriar cuanto se quiera la muestra sin miedo a que la temperatura supere el umbral determinado.

Abstract

The recent COVID-19 pandemic has brought with it an increasingly pressing need to detect viral infections. By developing PCB-MEMS, this work can be done quite efficiently.

In the Department of Industrial Electronics, a project has been developed to be able to carry out biomedical tests that require precise temperature control in a simple way thanks to Arduino. This consist of placing the sample taken from the patient on a heater and varying the temperature within a set range. However, by varying the heater temperature across a very wide range in a short time, the sample would overheat, often resulting in the test being discarded.

In this project, it is intended to carry out a PID temperature control using a PCB board connected to Arduino. The purpose of doing this is to be able to heat and cool the sample as desired without fear of exceeding the specified temperature threshold.

Índice

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

1 INTRODUCCIÓN

Este apartado incluye la descripción general del proyecto y un breve resumen de los conceptos más importantes del mismo.

1.1 Introducción general al proyecto

En la actualidad, la tecnología se ha convertido en una herramienta fundamental en diversos campos de aplicación, incluyendo la biología y la medicina. En particular, la dependencia de la temperatura en las reacciones biológicas y químicas es un común objeto de estudio y optimización.

Esta dependencia es un factor crítico que influye de manera significativa en las reacciones químicas y biológicas y es fundamental para entender y optimizar diversos procesos en el ámbito científico y tecnológico. La capacidad de controlar y ajustar la temperatura durante estas reacciones es crucial para lograr tiempos de reacción óptimos y resultados reproducibles.

Muchas reacciones en sistemas biológicos y químicos son altamente sensibles a la temperatura, ya que esta afecta a la velocidad de las reacciones enzimáticas, la estabilidad de las moléculas y la cinética de las interacciones moleculares. Por lo tanto, comprender y ajustar la temperatura es esencial para mejorar la eficiencia y la precisión de estos procesos.

Una aplicación destacada de esta dependencia de la temperatura es la optimización de las reacciones en tiempo real. Esto es importante en técnicas como la reacción en cadena de la polimerasa (PCR), la amplificación isotérmica mediada por bucle (LAMP) y en cultivos celulares y de tejidos.

Además, en el análisis clínico, el control de la temperatura es esencial para medir variables bioquímicas en humanos y animales, como glucosa, colesterol, bilirrubina o creatinina. La precisión y la fiabilidad de estas mediciones dependen en gran medida de mantener condiciones térmicas estables y controladas.

En resumen, la dependencia de la temperatura en las reacciones biológicas y químicas es un aspecto clave en la optimización de procesos científicos y tecnológicos.

En este contexto, el presente trabajo de fin de grado se centra en el diseño e implementación de un sistema de control de temperatura basado en Arduino utilizando un controlador Proporcional-Integral-Derivativo (PID).

1.2 Justificación

Este proyecto se realiza para poder exponer los conocimientos adquiridos de la electrónica y el diseño de circuitos. De esta forma se decide llevar a cabo un proyecto que incluya un poco de todos los aspectos más importantes de la electrónica: diseño de circuitos y programación.

Además, es una buena oportunidad para que cualquier persona que esté interesada en entrar al mundo de la electrónica pueda tener como referencia esta memoria, que explica de manera sencilla y clara un circuito que puede ser realizado incluso en casa, mientras se comprenden fácilmente conceptos básicos de electrónica y automática, como puede ser el funcionamiento de un controlador. También se usan herramientas sencillas y de software libre, como Arduino, de uso educativo y apto para todo el mundo, incluyendo novatos.

En definitiva, este proyecto se ha realizado para que el dispositivo elaborado, además de poder ser usado para realizar pruebas funcionales en el ámbito de la biomedicina, pueda ser fácilmente replicado por alumnos que tengan la intención de adentrarse en el mundo de la electrónica y el usa de controladores.

Como motivación personal para realizar este proyecto, es la de poder realizar el control de un sistema utilizando Arduino. Aprender a programar un sistema real utilizando esta plataforma es útil en muchos ámbitos. Además, la fabricación de microsistemas está presente en cualquier especialidad, y en este proyecto se puede observar perfectamente como crear placas de circuito impreso, presentes en cualquier aparato electrónico. Es decir, consiguiendo los objetivos de este proyecto, se abre un amplio abanico de posibilidades que se pueden realizar con los conocimientos adquiridos.

2 ESQUEMA GENERAL Y OBJETIVOS

A continuación, se establecen los objetivos relacionados con el proyecto, así como un esquema general para resumir los pasos a seguir y comprender el sistema que se va a diseñar en este proyecto. Se va a empezar por los objetivos.

2.1 Objetivos

El objetivo principal de este proyecto es el desarrollo de un sistema electrónico de control de temperatura para pruebas biomédicas.

Por otro lado, se van a establecer unos requisitos mínimos para el funcionamiento del dispositivo, los cuales se enumeran a continuación:

- **Rango entre 25ºC y 100ºC:** Este va a ser el rango de temperatura dentro del que se va a mover el control del dispositivo. Lo ideal es que la temperatura del calentador no se salga de ese rango.
- **Control de temperatura doble, con temperaturas diferentes:** Como se ha mencionado, se requiere que la placa tenga dos salidas para poder conectar dos calentadores simultáneamente, y que estos puedan estar operando al mismo tiempo con temperaturas diferentes, para poder realizar así pruebas diferentes a la vez.
- **Error menor a 0,5ºC para 37ºC:** Otro requisito es que cuando el control sea establecido en 37ºC, el error no sea mayor a 0,5ºC, esto se requiere para comprobar que el sistema es estable y fiable, pudiendo mantenerse en un rango pequeño durante un largo periodo de tiempo.
- **Control de la referencia con potenciómetros:** Se ha establecido que el control de la temperatura de referencia sea introducido en el programa mediante potenciómetros. Al variar manualmente su resistencia se podrá saber, mirando la pantalla, cuál es la temperatura que se está imponiendo en el dispositivo.
- **Uso de pantalla como interfaz con el usuario:** Se desea disponer de una pantalla como interfaz, que muestre los valores de la temperatura de la muestra y de la temperatura de referencia establecida. De esta manera, se podrá apreciar de una manera muy sencilla como actúa el dispositivo y ver si la prueba es válida, ya que sólo habrá que mirar la pantalla para ver la diferencia entre los valores en un momento determinado.
- **Uso sencillo del dispositivo:** Por último, es importante introducir la idea de que se va a realizar un dispositivo sencillo de controlar para realizar pruebas complejas.

2.2 Esquema general

A continuación, se puede observer un esquema general con los distintos dispositivos que conforman el Proyecto, y se procederá a una breve explicación a modo de introducción de cada uno de los mismos:

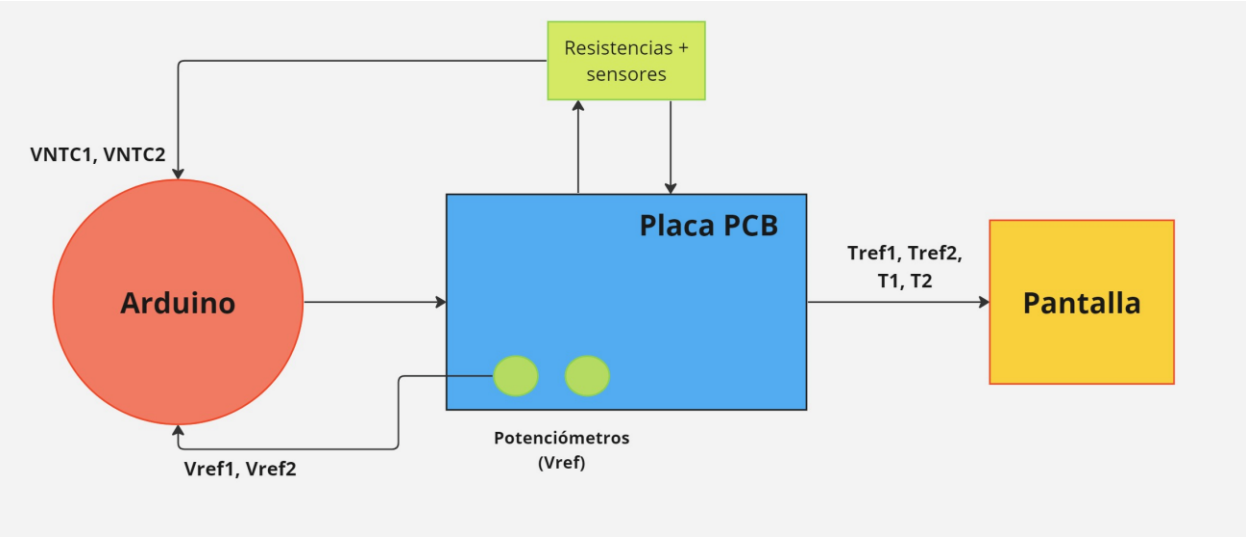

Figura 2-1. Esquema general del proyecto

- **Potenciómetros:** Estos serán los encargados de establecer la temperatura de referencia. Haciendo uso de un divisor de tensión, depende de la posición en la que se ponga el potenciómetro de manera manual, se enviará una tensión al programa de Arduino, que este convertirá en temperatura.

  Los potenciómetros irán colocados en un extremo de la placa PCB, para sean accesibles para su manipulación.

- **Arduino:** La placa de Arduino irá conectada por debajo de la placa PCB. En Arduino se ha realizado el programa principal, que consta de: Controlador PID, para comparar las temperaturas real y de referencia y actuar sobre el circuito en consecuencia, la pantalla, para mostrar gráficamente los resultados, y las transformaciones de voltaje a temperatura.

  La placa de Arduino será alimentada a 12 voltios para poder nutrir de más corriente a los calentadores para que lleguen antes a temperatura que se desea.

- **Pantalla:** En este componente, que se monta sobre la placa PCB, se van a ver los resultados de las temperaturas de los calentadores y las referencias. Estará programada en Arduino y tiene un uso meramente visual, no se podrá interactuar con ella.

- **Resistencias + Sensores:** Habrá dos calentadores conectados a la placa PCB, cada uno con su correspondiente sensor de temperatura. El calentador estará alimentado a 12 voltios mientras que el sensor irá conectado directamente a Arduino para pasarle la temperatura a la que se encuentra la muestra, para que se pueda comparar con la referencia.

  Estos componentes también irán conectados en un borde de la placa PCB, por si en algún momento se quiere cambiar el tipo de calentador para realizar alguna prueba específica.

- **Placa PCB:** Es donde se montan y se conectan todos los componentes electrónicos del sistema. Esta placa estará diseñada para que los componentes que se van a manipular, como los calentadores y los potenciómetros, queden en sitios fácilmente accesibles

3 ESTADO DEL ARTE

En el ámbito de las aplicaciones biomédicas, el control preciso de la temperatura es esencial para garantizar resultados confiables y reproducibles en diversos procesos, como la PCR, el cultivo celular, la termoterapia y la electrofisiología. A lo largo de los años, se han desarrollado y utilizado diversos métodos y técnicas para abordar este desafío, algunos de los cuales se describen a continuación:

3.1 Control mediante termostatos convencionales

Los termostatos convencionales son sido ampliamente utilizados en aplicaciones biomédicas para controlar la temperatura. Estos dispositivos emplean un interruptor eléctrico o mecánico que enciende o apaga el calentador en función de la lectura del sensor de temperatura. Aunque son ampliamente accesibles y asequibles, su precisión y estabilidad pueden ser limitadas, lo que puede afectar la reproducibilidad y la exactitud de los experimentos biomédicos. En la actualidad se usan los siguientes tipos de termostatos:

3.1.1 Bimetálicos

Este tipo de termostato se fabrica uniendo dos láminas de metales de diferentes coeficientes de dilatación. De esta manera, cuando la temperatura cambia, los metales se deforman dependiendo de esta, lo que hace que se abra o se cierre el circuito eléctrico. Pueden ser manueales o aotomáticos y solo tienen dos estados, abierto o cerrado. Como se puede deducir por la descripción del mismo, este termostato es analógico, por lo que no es todo lo preciso que se podría esperar para aplicaciones médicas.

Figura 3-1. Ejemplo de termostato bimetálico

3.1.2 De parafina

El funcionamiento de este tipo de termostatos es parecido al anteriormente mencionado. Estos contienen parafina encapsulada. La parafina es un compuesto constituido por una mezcla de hidrocarburos derivados del petróleo, tiene numerosas aplicaciones industriales, pero para este estudio nos centramos en una de sus muchas propiedades: al ser una cera, mientras la temperatura va en aumento, esta se empieza a derretir, haciendo que el volumen de material vaya en aumento y, por lo tanto, se expanda.

En conclusión, cuando este material se derrite, empuja un dusco que permite el paso del fluido y, cuando el fluido baje su temperatura, un resorte devuelve el disco a su posición inicial.

Este tipo de termostatos es utilizado en el sistema de enfriamiento de los motores de combustión interna [1].

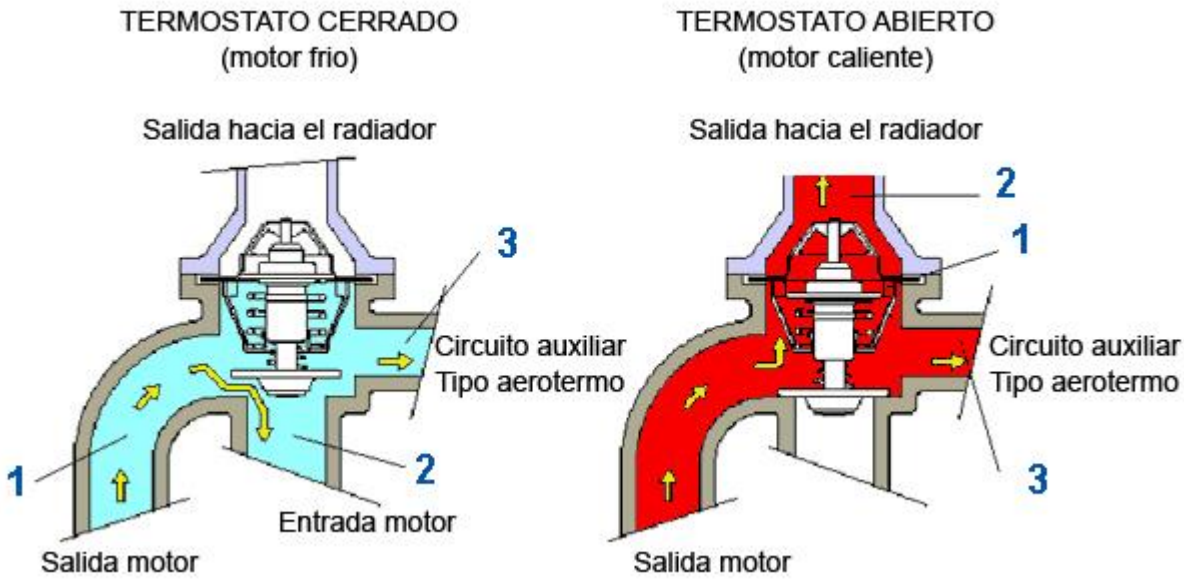

Figura 3-2. Ejemplo de funcionamiento del termostato de parafina en un motor de combustión interna

3.1.3 De gas encerrado

Consiste en un gas encerrado dentro de un tubo. Al subir la temperatura, el gas se expande y empuja una válvula. Para regular estos termostatos, lo que se hace es modificar el volumen del tubo. Con esto se consigue que varíe la presión y, por lo tanto, la temperatura. Es muy usado en electrodomésticos que requieran de un control de temperatura, como neveras u hornos.

Figura 3-3. Ejemplo de termostato de gas encerrado [2]

3.1.4 Termostatos electrónicos

Los tipos de termostatos anteriormente citados no son compatibles con aplicaciones biomédicas puesto que no son excesivamente precisos, sin embargo, existe un tipo de termostato que sí cumple con los requerimientos en cuanto a precisión se refiere, el electrónico.

El funcionamiento de estos está basado en sensores de temperatura. Cuando estos detectan que se ha superado la temperatura límite, mandan una señal para abrir el circuito eléctrico, y al revés, cuando se rebasa un límite inferior al establecido, se manda otra señal para cerrar el circuito y volver a subir la temperatura.

Estos termostatos son más precisos que los analógicos, pero presentan dos problemas:

1. Son bastante caros
2. El circuito se cierra cuando se llega al umbral de temperatura, por lo que para aplicaciones médicas en las que se necesita controlar la temperatura en todo momento esto es un problema, ya que la temperatura va a presentar grandes oscilaciones y no se va a estabilizar nunca.

3.2 Extensión del controlador PID

El control de variables preciso es esencial en el ámbito de la medicina, donde un ligero desvío de temperatura, presión u otras magnitudes, pueden traer consecuencias negativas significativas para ciertas necesidades.

Debido a esto, se han llevado a cabo numerosos estudios para determinar la variedad de dispositivos que se pueden usar para estas aplicaciones y analizar sus resultados.

En los últimos años, se ha observado un creciente interés en el desarrollo de dispositivos controlados por algoritmos PID que tengan la precisión suficiente para poder ser usados en entornos médicos y laboratorios, lo que es fundamental paea mejorar la calidad de la atención médica.

A finales de la década de los 2000, con la llegada de los Smartphones y el auge mundial de la electrónica, se empezó a estudiar métodos para controlar magnitudes tales como la temperatura, sobre todo en el ámbito de la domótica. Surgieron muchos estudios para regular variables con controladores PID, como, por ejemplo, el estudio realizado por Radu Bălan y sus compañeros en 2009 [3], en el que se realiza un control PID de temperatura para un edificio, implementado en Simulink.

Los resultados de este y otros estudios confirmaron la eficacia del algoritmo de control predictivo para la regulación de temperatura. Se observó una mejora sustancial en precisión y eficiencia en comparación con métodos de control tradicionales, como los vistos anteriormente.

3.3 Integración de Arduino en controles PID

A partir de estos experimentos, se empezaron a implementar controles PID con diferentes microcontroladores, mayoritariamente obteniendo buenos resultados, hasta la llegada de Arduino.

Los microcontroladores Arduino llegaron al mercado en 2005. En principio estaban pensados para uso escolar y formativo, pero pasado un tiempo se empezó a vislumbrar como una herramienta de un potencial increíble, con un entorno y programación muy fáciles de utilizar y con unos resultados muy buenos.

No fue hasta a partir del año 2018 cuando el uso de Arduino se extendió mundialmente y fue empezado a ser tomado en cuenta para estudios precisos. Se puede ver como ejemplo un artículo del mismo año, publicado por Bambang Supriyo, entre otros [4]. En este estudio, se describe un sistema de un calentador de aire controlado por Arduino a través de un controlador PID. Los resultados de la investigación revelan que el calentador de aire basado en PID logra una regulación estable y precisa de la temperatura del sistema.

De ahí en adelante, se han realizado numerosas investigaciones con esta combinación, entrando cada vez en ámbitos que requieren más precisión o que los equipos de medida existentes sean muy caros y con algo tan barato como un Arduino se pueda replicar cualquier dispositivo.

En el campo de la medicina, son varios los estudios que se han realizado recientemente, se ha elegido como

ejemplo un estudio realizado por Clarissa Grace Santoso en 2023 [5]. Este estudio trata de un control de la temperatura de la sangre de un paciente durante las transfusiones de sangre. El valor que debe tener la temperatura para no desestabilizar al paciente debe ser entre 36,5 y 37,5ºC. Con un simple sensor de temperatura, un microcontrolador Arduino y un calentador, ajustando las constantes del PID, se logra obtener un error muy pequeño y mantenerse dentro de ese rango. Existen muchos más artículos parecidos en los que se controlan variables en procesos médicos, como, por ejemplo, la liofilización de muestras o la presión.

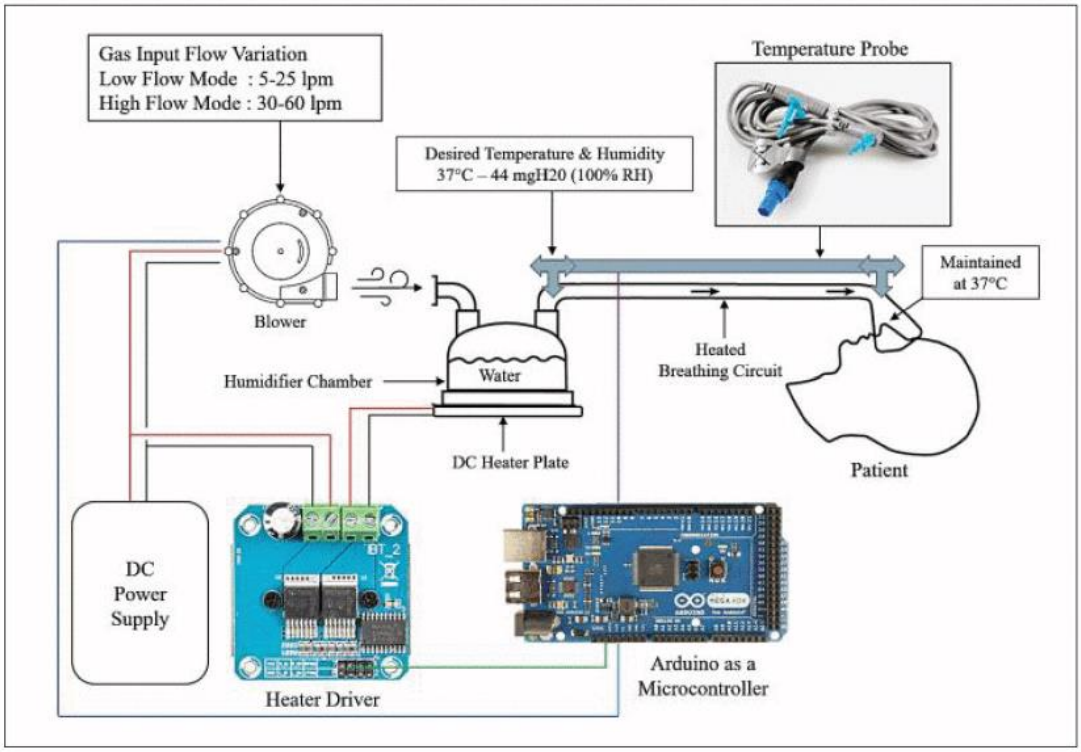

Figura 3-4. Control de temperatura de un sistema humidificador para la respiración [6]

En conclusión, la tendencia en el control preciso de variables médicas es realizar experimentos precisos con infraestructura barata, debido a la inmensa cantidad de pruebas a realizar, sobre todo después de la crisis del COVID, donde se ha requerido de la fabricación de numerosos dispositivos de control de temperatura, ya que las pruebas PCR se basan en este principio. Debido a la alta demanda y a los grandes avances en electrónica, se están desarrollando numerosos estudios con controladores PID combinados con controladores del tipo Arduino.

4 DISEÑO DEL CIRCUITO

En este capítulo se va a describir el proceso de diseño del circuito regulador. El objetivo de este circuito es establecer un límite de tensión de 0 a 5 voltios, siendo 0 voltios 100ºC y 5 voltios 25ºC. Por otro lado, se hace lo mismo con el potenciómetro y las dos señales llegan a un Amplificador operacional, donde se mide la diferencia que hay entre estas. Esta diferencia sería la salida del circuito y, por tanto, la entrada al programa donde se ejecuta el controlador PID.

A continuación, se va a hacer una descripción de los elementos electrónicos y programas utilizados:

4.1 Programas utilizados

El programa utilizado para este apartado es Micro-Cap 12.

Micro-Cap es un programa de simulación de circuitos analógicos y digitales con un editor de esquemas integrado que es compatible con cualquier versión de Windows.

Figura 4-1. Logo de Micro-Cap 12

Las funciones principales que han llevado a que este sea el software utilizado para realizar esta tarea son:

1. Es gratis. Esto no era así hasta 2019, cuando la empresa fundadora del programa cerró, desde ese entonces el software se puede descargar y ser utilizado libremente de manera gratuita.
2. Su interfaz es simple e intuitiva, lo cual hace que, si ya partes de una buena base, no va a ser trabajoso lograr tu objetivo inicial.
3. Con motivo del cierre de la empresa, la aplicación no va a ser actualizada en un futuro, pero, a pesar de esto, se pueden meter componentes si alguien realiza el modelo con los datos internos para poder ser integrado en el programa, por lo que le da mucha versatilidad.

Una vez descrito el software, se va a proceder a explicar los elementos que van a ir integrados en el circuito, tales como componentes electrónicos o bien conceptos electrónicos.

4.2 Elementos de la simulación

4.2.1 Divisor de tensión

Un divisor de tensión es un circuito que, a partir de un nivel de señal de entrada de alimentación, obtiene una determinada tensión de salida, cuyo valor será el cociente de la resta entre la tensión de alimentación y la tensión de referencia, dividido por las sumas de las resistencias que componen el divisor, multiplicado por el valor de la resistencia conectada a tierra. Su fórmula se expresa de la siguiente manera:

$$V0 = \frac{V}{R1 + R0} \cdot R0 \qquad \text{Ecuación 4-1}$$

A continuación, se representa un modelo de divisor de tensión implementado en Microcap. Los nombres dados a cada componente corresponden con la fórmula superior.

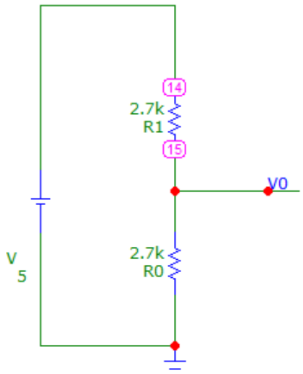

Figura 4-2. Esquema de un divisor de tensión con dos resistencias

Este sistema presenta la característica de que, si las resistencias que lo componen son del mismo valor, la diferencia de potencial será la misma en cada una de ellas. En el caso anterior, con dos resistencias, en cada una de ellas habrá una caída de tensión equivalente a la mitad de la tensión de alimentación.

4.2.2 Amplificador operacional diferencial

Un amplificador operacional es un amplificador diferencial. Desde el punto de vista de una señal, tiene tres terminales: dos de entrada y uno de salida. Este elemento es un dispositivo que necesita una potencia de continua para funcionar, y requiere de dos fuentes de continua, una negativa y otra positiva, siendo habitual que sean iguales en valor absoluto.

La ecuación que gobierna el comportamiento de este dispositivo en bucle abierto es la siguiente:

$$V_0 = A \cdot (V^+ - V^-) \qquad \text{Ecuación 4-2}$$

Siendo A la ganancia de tensión.

Es decir, la salida es la diferencia de ambas entradas multiplicada por A. Cabe destacar que esta ecuación es la de un amplificador real, la ecuación real difiere un poco, añadiendo al voltaje de salida efectos no ideales.

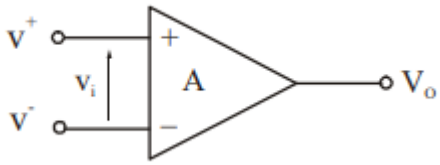

Figura 4-3. Esquema de un amplificador operacional

En este proyecto se va a hacer uso del amplificador operacional de dos maneras diferentes, dentro de las muchas maneras de utilizarlo que se aplican actualmente, aquí nos vamos a encontrar las siguientes:

- **Amplificador Restador de Tensión Diferencial**

Como el propio nombre del componente indica, este se usa para amplificar la diferencia de los valores de tensión que llegan a sus terminales, obteniendo una tensión varios valores de magnitud más alta que la que se obtendría normalmente. El valor de salida puede ser establecido como se desee gracias al uso de resistencias.

Para entender bien el funcionamiento de esta configuración, hay que introducir el concepto de **realimentación negativa,** que consiste en conectar la salida V_0 con la entrada V^-. Esto se hace principalmente para estabilizar el sistema.

Una particularidad de este montaje, es que nos permite establecer el voltaje de salida como un valor definido gracias a los componentes que conectemos en el circuito, y que no sea un valor interno del amplificador, esto es debido a que, al conectar componentes externos al amplificador, este trabaja siempre en modo lineal (no saturado), haciendo que la diferencia entre V+ y V- sea insignificante, por lo que la ganancia interna del amplificador operacional no tendrá nada que ver en la tensión de salida . A continuación, se representa el esquema simplificado de un amplificador de tensión diferencial:

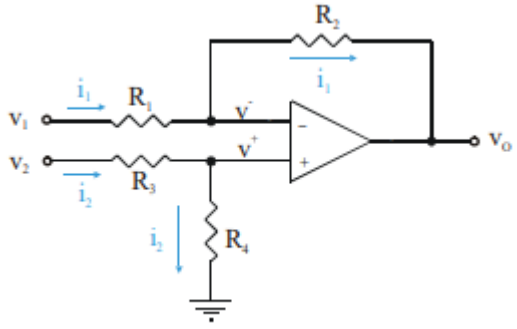

Figura 4-4. Esquema de un amplificador de tensión diferencial

Una vez visto como es la representación del circuito, se va a hacer una demostración de las fórmulas que gobiernan el comportamiento del mismo. Esto es bastante sencillo, ya que solo se van a utilizar resistencias como componentes.

En primer lugar, se van a establecer las bases, gracias a lo anteriormente descrito tenemos que, al haber realimentación negativa:

- $V^+ = V^-$

Esto hace que se puedan igualar los valores de tensión de los dos nudos, como vemos a continuación.

Nudo V+:

$$\left.\begin{array}{l} V^+ = R_4 \cdot i_2 \\ i_2 = \frac{v_2}{R_3 + R_4} \end{array}\right\} \longrightarrow V^+ = \frac{R_4}{R_4 + R_3} \cdot v_2 \qquad \text{Ecuación 4-3}$$

Nudo V-:

$$\left.\begin{array}{l} i_1 = \frac{v_1 - V^-}{R_1} \\ i_1 = \frac{V^- - v_0}{R_2} \end{array}\right\} \rightarrow \frac{v_1 - V^-}{R_1} = \frac{V^- - v_0}{R_2} \rightarrow V^- = \frac{R_2}{R_1 + R_2} \cdot v_1 + \frac{R_1}{R_1 + R_2} \cdot v_0 \qquad \text{Ecuación 4-4}$$

Ahora queda igualar ambas ecuaciones, con lo que quedaría:

$$\frac{1}{R_1 + R_2} \cdot (R_2 \cdot v_1 + R_1 \cdot v_0) = \frac{R_4}{R_4 + R_3} \cdot v_2 \qquad \text{Ecuación 4-5}$$

La incógnita es V_0, puesto que los valores de las resistencias se establecen por la persona que diseñe el circuito. En este caso, vamos a tomar resistencias del mismo valor para R1 y R3, pasando lo mismo para R2 y R4, con lo que la ecuación se va a ver notablemente simplificada:

$$\frac{1}{\cancel{R_1 + R_2}} \cdot (R_2 \cdot v_1 + R_1 \cdot v_0) = \frac{R_2}{\cancel{R_1 + R_2}} \cdot v_2 \longrightarrow v_0 = \frac{R_2}{R_1} \cdot (v_2 - v_1) \qquad \text{Ecuación 4-6}$$

Esta es, finalmente, la ecuación que va a gobernar el circuito. Lo que hay que hacer es establecer valores de las resistencias en relación a la ganancia que se quiera obtener.

- **Seguidor de tensión**

Como se ha visto anteriormente, la realimentación negativa estabiliza el sistema haciendo que la zona de funcionamiento lineal del amplificador operacional aumente, además de eliminar la ganancia del amplificador.

En un seguidor de tensión sólo se conecta la salida con la entrada negativa, sin ningún componente de por medio. Esto hace que se eliminen los efectos de carga que pueden aparecer en la tensión de salida por la configuración interna del amplificador.

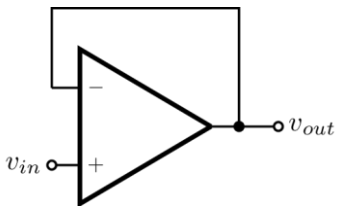

Figura 4-5. Esquema de un seguidor de tensión

4.3 Descripción del circuito

El objetivo de realizar este montaje es, teniendo un sensor de temperatura, hacer que la salida del circuito sea un valor entre 0 y 5 voltios aproximadamente, se dice aproximadamente por que los valores de las resistencias usadas en el amplificador operacional están normalizados, y es complicado obtener el 5 exacto.

Se desea obtener un valor entre 0 y 5, siendo 0 cuando la temperatura del sensor sea de 100ºC y 5 cuando sea 25ºC. Esto se ha establecido así debido a que es el rango que se requiere para los campos en los que se va a aplicar este proyecto.

4.3.1 Divisor de tensión con potenciómetro

En primer lugar, se va a establecer la tensión de entrada positiva del amplificador operacional. Esta entrada va a ser constante, pero se hará uso de un potenciómetro para regularla, por si se quiere ampliar o reducir el rango de temperatura controlable. Una vez definido, se pasa al montaje de esta parte:

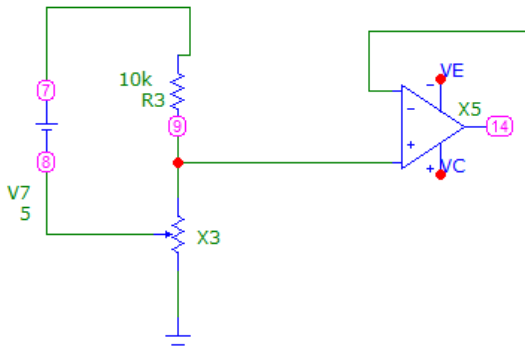

Figura 4-6. Esquema del nodo V+ del amplificador

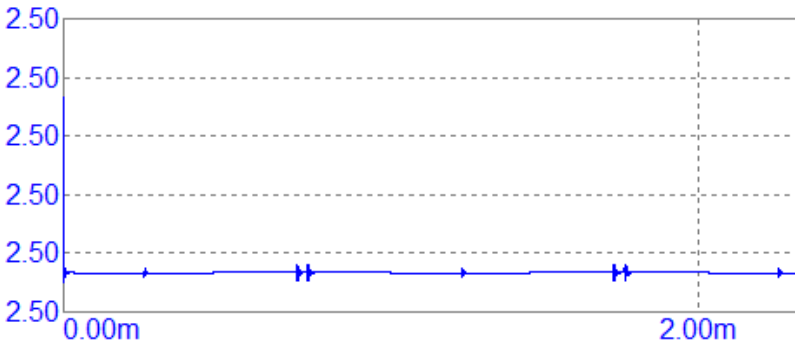

Figura 4-7. Tensión resultante del nodo V+

Como se puede observar, se ha hecho uso de un divisor de tensión regulado por un potenciómetro. Este se ha establecido con valor de 10kΩ para que la salida sea 2.5V, esto es así porque la resistencia de la otra parte del divisor también tiene el mismo valor, por lo que la salida es la mitad del valor de la tensión de entrada. Si se modifica el valor de la resistencia del potenciómetro, la tensión de salida irá disminuyendo, por lo que sabemos que el rango de salida es de 0 a 2.5V.

Para la tensión del nudo de la entrada negativa del amplificador operacional, se monta el mismo circuito, con la excepción de que, en lugar de dos resistencias, se pone una resistencia y un NTC de temperatura. Esto se hace para simular un flujo de temperatura cambiante.

4.3.2 Divisor de tensión con sensor NTC

La NTC se establece en un rango de 25 a 100 ºC, esto se puede modelar conectándole una fuente de tensión de 25 a 100 V, que actúa como simulador de la temperatura. En la parte de abajo se pone una resistencia de 1kΩ, de modo que el valor de salida de esta parte del circuito tiene un rango de 0.455 a 2.496 V.

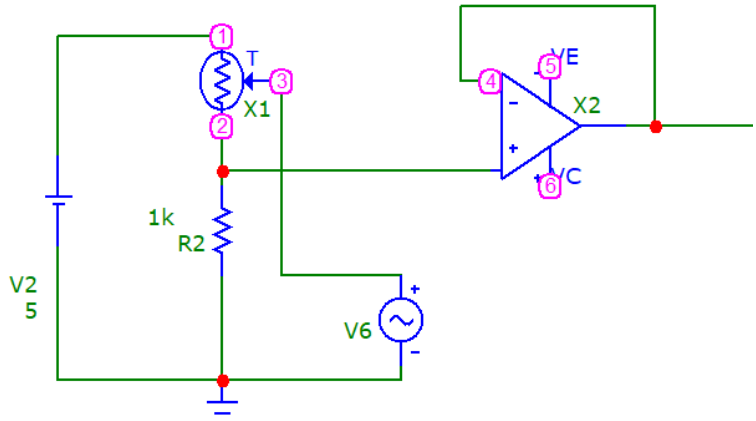

Figura 4-8. Esquema del nodo V- del amplificador

A continuación, se muestran los resultados de las simulaciones de estos dos nodos después de pasar por el seguidor de tensión. Estos serán los valores con los que se operará para sacar la tensión de salida del circuito:

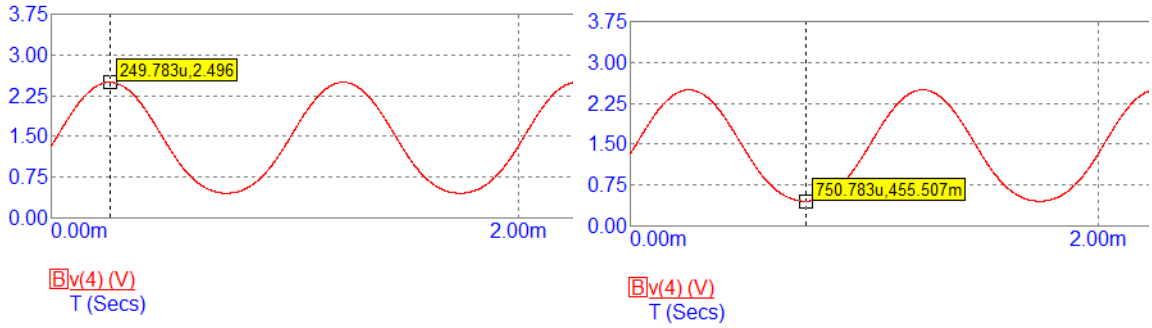

Figura 4-9. Tensión resultante del nodo V-

4.3.3 Amplificador Operacional

Una vez obtenidos estos valores, se pasa al diseño del amplificador operacional. Como recordatorio, el objetivo es obtener un valor de salida que vaya de 0 a 5 voltios, o, como mínimo, que se quede lo más cercano posible a este rango. Esto es difícil de conseguir debido a que los valores de las resistencias están normalizados y las combinaciones no son infinitas.

En el apartado 4.2.2 se ha deducido la fórmula de la tensión de salida de un amplificador operacional, que, como se puede ver, es la siguiente:

$$v_0 = \frac{R_2}{R_1} \cdot (v_2 - v_1)$$ Ecuación 4-7

Ahora bien, conocemos todas las incógnitas a excepción de R2 y R1:

- V0 = 5 V
- V2 = 2.5 V
- V1 = 0.455 V

Despejando, se obtiene que la relación entre las resistencias debe ser igual a 2.445, a partir de aquí, lo que queda es ir a la tabla de resistencias normalizadas y coger dos cuya división resulte en un valor aproximado a 2.445:

x 1	x 10	x 100	x 1.000 (K)	x 10.000 (10K)	x 100.000 (100K)	x 1.000.000 (M)
1 Ω	10 Ω	100 Ω	1 KΩ	10 KΩ	100 KΩ	1 M Ω
1,2 Ω	12 Ω	120 Ω	1K2 Ω	12 KΩ	120 KΩ	1M2 Ω
1,5 Ω	15 Ω	150 Ω	1K5 Ω	15 KΩ	150 KΩ	1M5 Ω
1,8 Ω	18 Ω	180 Ω	1K8 Ω	18 KΩ	180 KΩ	1M8 Ω
2,2 Ω	22 Ω	220 Ω	2K2 Ω	22 KΩ	220 KΩ	2M2 Ω
2,7 Ω	27 Ω	270 Ω	2K7 Ω	27 KΩ	270 KΩ	2M7 Ω
3,3 Ω	33 Ω	330 Ω	3K3 Ω	33 KΩ	330 KΩ	3M3 Ω
3,9 Ω	39 Ω	390 Ω	3K9 Ω	39 KΩ	390 KΩ	3M9 Ω
4,7 Ω	47 Ω	470 Ω	4K7 Ω	47 KΩ	470 KΩ	4M7 Ω
5,1 Ω	51 Ω	510 Ω	5K1 Ω	51 KΩ	510 KΩ	5M1 Ω
5,6 Ω	56 Ω	560 Ω	5K6 Ω	56 KΩ	560 KΩ	5M6 Ω
6,8 Ω	68 Ω	680 Ω	6K8 Ω	68 KΩ	680 KΩ	6M8 Ω
8,2 Ω	82 Ω	820 Ω	8K2 Ω	82 KΩ	820 KΩ	8M2 Ω
						10M Ω

Figura 4-10. Tabla de resistencias normalizadas []

Para este caso en particular se han seleccionado las resistencias R1 = 1.2 kΩ y r2 = 2.7 kΩ, cuya división da un valor de 2.25, lo que hace que el pico de tensión a la salida esté en 4.605 V, un valor aceptable.

A continuación, se pueden ver las configuraciones finales del amplificador operacional y del circuito completo:

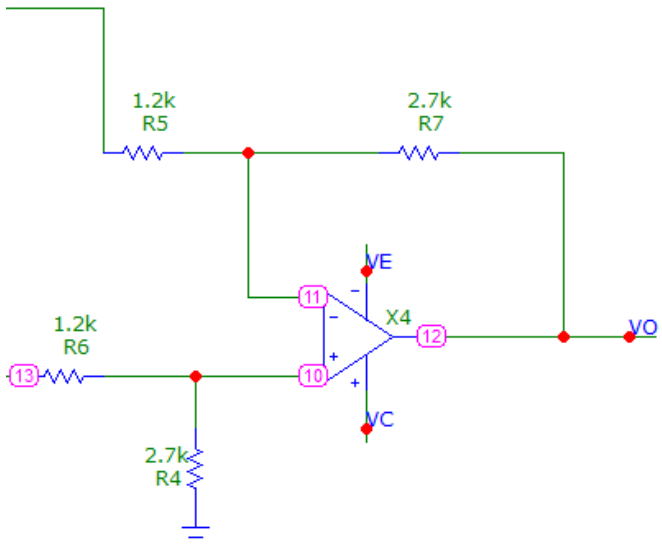

Figura 4-11. Esquema del amplificador operacional

4.3.4 Circuito de Referencia de Temperatura

Por otra parte, se va a implementar el circuito que servirá como referencia de temperatura, este constará de un divisor con un potenciómetro, que será el que se manejará acorde a la temperatura que se desea en el sistema a controlar. La salida de este circuito será una entrada al microcontrolador Arduino.

A continuación, se puede ver cómo sería el montaje de este circuito, cuyo valor de salida tendrá un valor de 0 a 5 voltios:

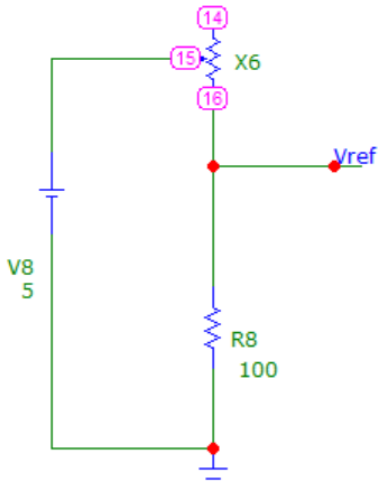

Figura 4-12. Divisor de tensión para captar la temperatura de referencia

La resistencia usada será de un valor bajo para garantizar que la salida está entre 5 voltios y un valor muy cercano a los 0 voltios.

De esta manera quedaría el diseño del circuito completo:

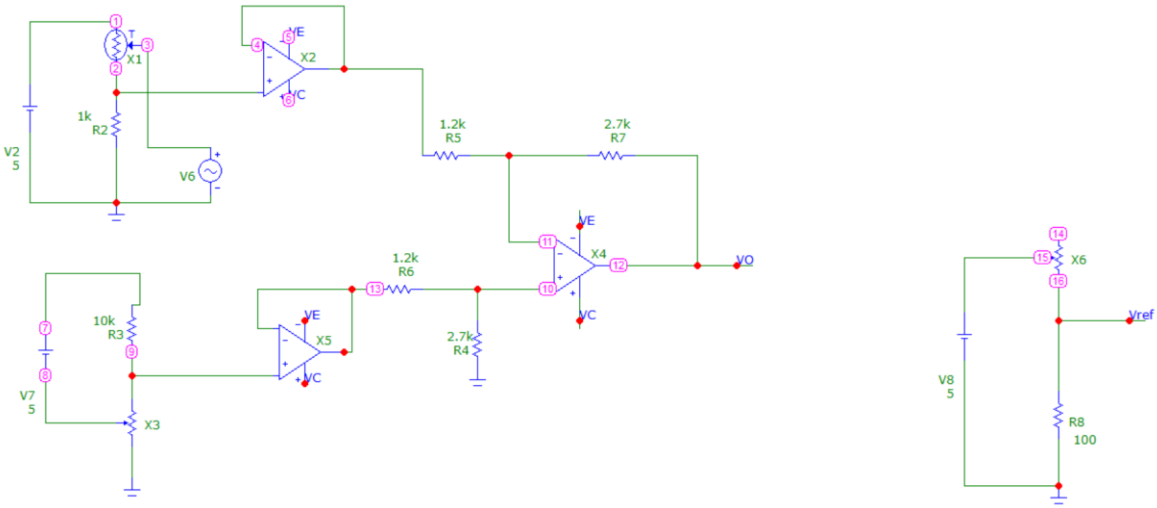

Figura 4-13. Esquema final del circuito

Finalmente, se muestran los resultados de la tensión de salida del circuito, la cuál será la entrada al programa en Arduino:

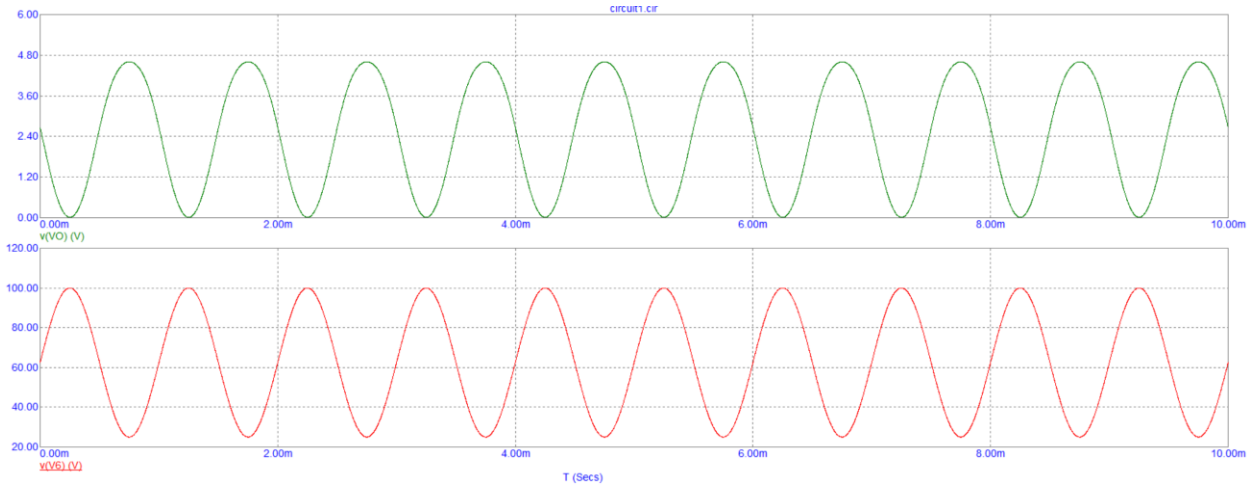

Figura 4-14. Tensión de salida del circuito (verde) en relación con la temperatura (rojo)

Como puede observarse, cuando el sensor mide 100ºC, la salida será de 4,605 voltios y, cuando mide 25ºC, la salida será de 0 voltios, tal y como se deseaba.

4.4 Actuación

La señal de salida del amplificador operacional será la entrada del programa implementado en Arduino, y, a la salida de este, la señal PWM será recogida por un circuito actuador.

Este circuito estará formado por un driver, un transistor MOSFET y una resistencia de cobre.

4.4.1 Driver

Un driver es un dispositivo electrónico diseñado para controlar o manejar otro dispositivo, como un transistor, relé, motor, etc. Los drivers son esenciales cuando se necesita operar dispositivos que requieran corrientes o voltajes específicos que no pueden ser suministrados directamente por el microcontrolador. Dependiendo del tipo de dispositivo a la salida, los drivers pueden variar en complejidad y diseño.

En este caso, el dispositivo a operar es un transistor MOSFET con una señal por ancho de pulso o PWM.

Para una señal PWM, el driver recibe esta señal y la utiliza para controlar la cantidad de tiempo durante el cual el MOSFET está encendido o apagado en cada ciclo de la señal. Cuando la señal PWM está en su nivel alto, el MOSFET está encendido, permitiendo que la corriente fluya a través de él y, por lo tanto, activando la carga conectada. Por otro lado, cuando la señal está en su nivel bajo, el MOSFET se cierra, interrumpiendo el flujo de corriente.

En resumen, el driver se encarga de interpretar la señal PWM para controlar el encendido y el apagado del transistor.

4.4.2 Transistor MOSFET

Un transistor MOSFET es un tipo de transistor utilizado en electrónica para controlar el flujo de corriente entre dos terminales, el drenador y la fuente, mediante la aplicación de una tensión en un tercer terminal llamado puerta. El funcionamiento básico de un transistor MOSFET se puede entender en tres modos de operación: corte, saturación y zona activa [8].

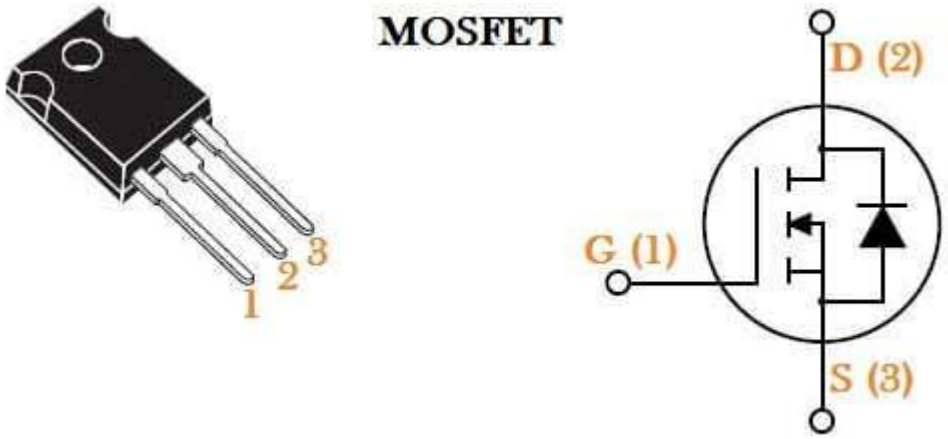

Figura 4-15. Esquema de un transistor MOSFET

- **Modo de corte:** En este modo, no hay corriente que fluya entre el drenador y la fuente. La unión entre el sustrato semiconductor y el canal en el MOSFET actúa como un diodo polarizado en inversa, lo que previene el flujo de corriente entre el drenador y la fuente.
- **Modo de saturación:** Cuando se aplica una tensión adecuada entre la compuerta y la fuente, se forma un campo eléctrico que modifica la conductividad del canal entre el drenador y la fuente. Esto permite que la corriente fluya. En este modo, el MOSFET se comporta como un interruptor cerrado, y hay una resistencia de conducción muy baja entre el drenador y la fuente. Esto significa que el MOSFET está completamente activado y permite el flujo máximo de corriente.
- **Zona lineal / óhmica:** En este modo, el MOSFET se encuentra en un estado intermedio entre el corte y la saturación. Hay una corriente entre el drenador y la fuente, pero no está en su máximo valor. La resistencia entre el drenador y la fuente es intermedia y controlada por la tensión aplicada a la compuerta.

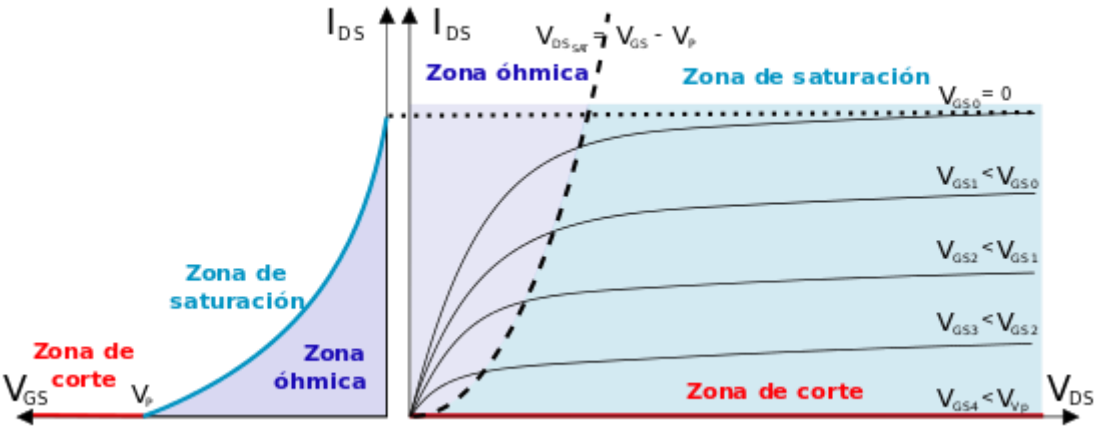

Figura 4-16. Curvas de funcionamiento características de un transistor MOSFET

El transistor MOSFET es especialmente útil en aplicaciones donde se requiere un control eficiente del flujo de corriente, como en circuitos de conmutación de alta velocidad, amplificadores de señal, reguladores de voltaje, entre otros.

Existen MOSFET de tipo canal N y canal P, que se diferencian en la polaridad de la tensión necesaria para usar el dispositivo. En este proyecto se usarán MOSFET tipo canal N.

4.4.3 Resistencia de cobre

A la salida del transistor MOSFET, se encuentra la resistencia de cobre, la cual se va a ir calentando o enfriando (a temperatura ambiente) en función de la corriente que pase por ella.

Finalmente, el esquema del circuito completo quedaría de la siguiente manera:

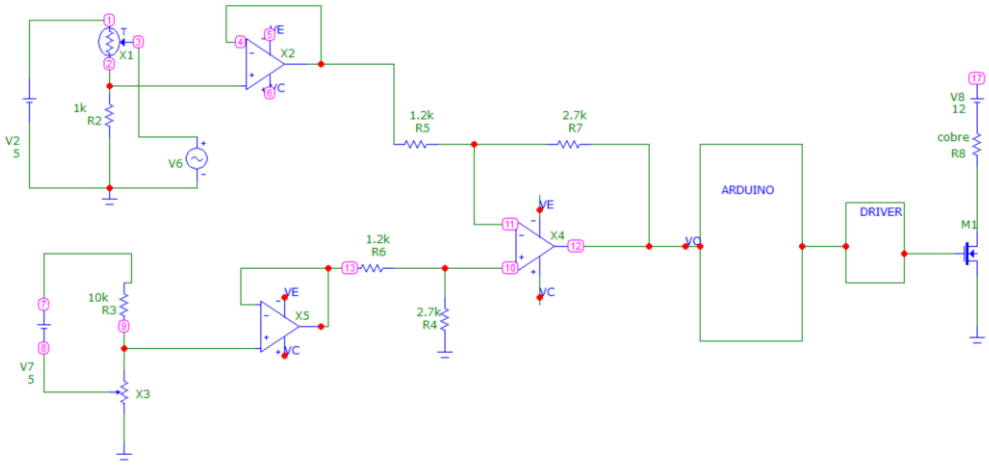

Figura 4-17. Esquema del circuito completo

5 PROGRAMACIÓN

En este capítulo se va a describir el programa en Arduino donde se implementa el PID, la base de este proyecto. Se recogerán los datos de la temperatura de referencia que se quiera obtener y la real, se compararán los valores y el PID actuará en consecuencia, regulando la temperatura de la resistencia de cobre para adaptar el valor que recoge el sensor de temperatura al valor deseado.

El programa consta de varios bloques:

- **Inicialización de pines, PID y pantalla**: Como veremos más adelante, la programación en Arduino se realiza en dos fases, el *setup* y el *loop*. En el setup es donde se inicializan los pines y variables que se van a usar globalmente.
- **Lectura de datos**: En el programa, habrá datos que tendrán que ser leídos constantemente, como las entradas analógico digitales.
- **Cálculos**: Una vez leídos los datos, hay que adaptarlos a la medida que se va a utilizar. En este caso, por ejemplo, hay que convertir voltaje en temperatura, como se verá en los siguientes apartados. También hay que calcular el error entra las temperaturas reales y de referencia, lo cuál va a ser el valor de entrada del PID.
- **Información visual**: Se va a añadir una componente visual al proyecto, haciendo que, cuando el error sea mayor del esperado, se encienda una señal luminosa.
- **Actuación de PID**: Una vez obtenidos los datos de entrada, se acciona el PID para ontener una salida PWM, que será la salida de el programa.
- **Actualización de la pantalla**: Ya obtenido el valor PWM de salida proveniente del PID, se procede a actualizar la pantalla con los valores reales del experimento. Esto se hará una vez cada 20 ciclos (20 ms).

A continuación, se puede observar el diagrama de bloques del programa completo:

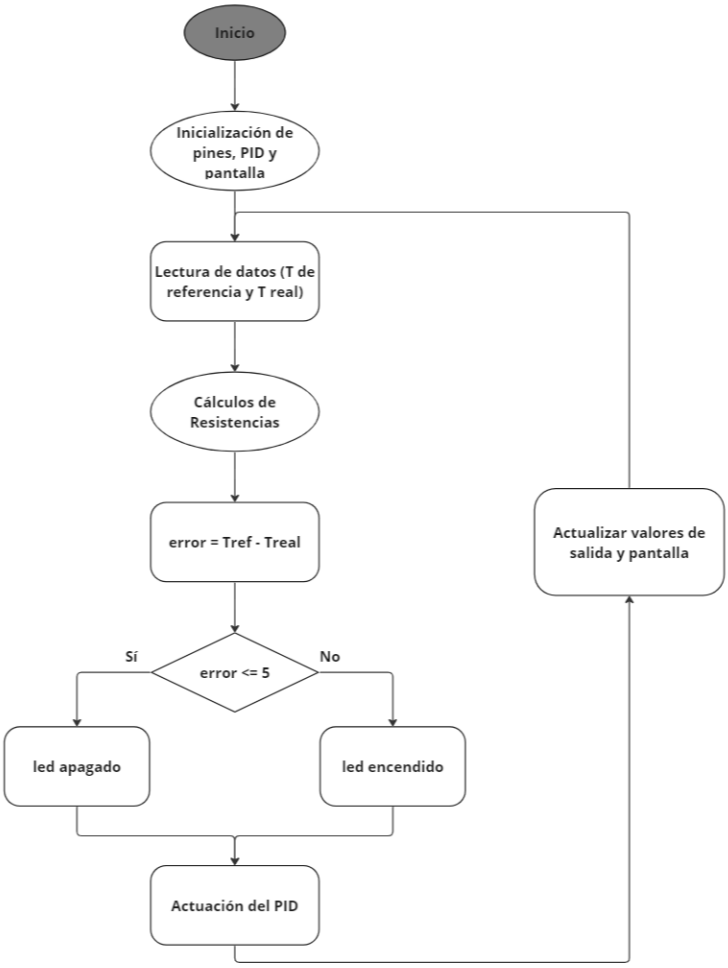

Figura 5-1. Diagrama de bloques del programa en Arduino

Una vez resumida la función del programa y los pasos a seguir, se va a desglosar el proceso seguido bloque a bloque, además de ver una breve introducción a Arduino.

5.1 Descripción del Software

Como ya se ha mencionado, para la programación de este proyecto, se va a hacer uso de Arduino [9].

Arduino es una plataforma de hardware y software de código abierto diseñada para facilitar la creación de proyectos electrónicos. Se compone de una placa de microcontrolador que puede ser programada para realizar diversas funciones mediante la conexión de sensores, actuadores y otros componentes electrónicos. Esta flexibilidad y facilidad de uso la hacen popular en la creación de proyectos de robótica, dispositivos portátiles, etc.

La versatilidad de Arduino radica en su capacidad para interactuar con el entorno físico a través de sensores y controlar dispositivos mediante actuadores. Esto permite a los usuarios crear sistemas interactivos que respondan a estímulos como la detección de luz o temperatura, entre otros. Además, el extenso conjunto de bibliotecas y recursos disponibles facilitan el desarrollo y la experimentación para los usuarios.

Una de las principales ventajas de Arduino es su accesibilidad. El hardware es relativamente económico y está ampliamente disponible, lo que lo hace asequible para estudiantes, aficionados y profesionales por igual. Además, el software de programación Arduino es gratuito y de código abierto, compatible con Windows, Mac

y Linux, lo que permite a los usuarios empezar a programar con facilidad. Asimismo, su amplia comunidad ofrece un rico ecosistema de soporte y recursos en línea, que van desde tutoriales hasta foros de discusión y proyectos compartidos.

El modelo de Arduino que se va a utilizar en este proyecto es Arduino UNO, una de las placas más populares y ampliamente utilizadas dentro de la familia Arduino. Se destaca por su facilidad de uso y versatilidad, con una serie de pines de entrada/salida que permiten la conexión de una amplia variedad de componentes electrónicos. Su diseño compacto y robusto lo hace ideal para proyectos tanto educativos como profesionales.

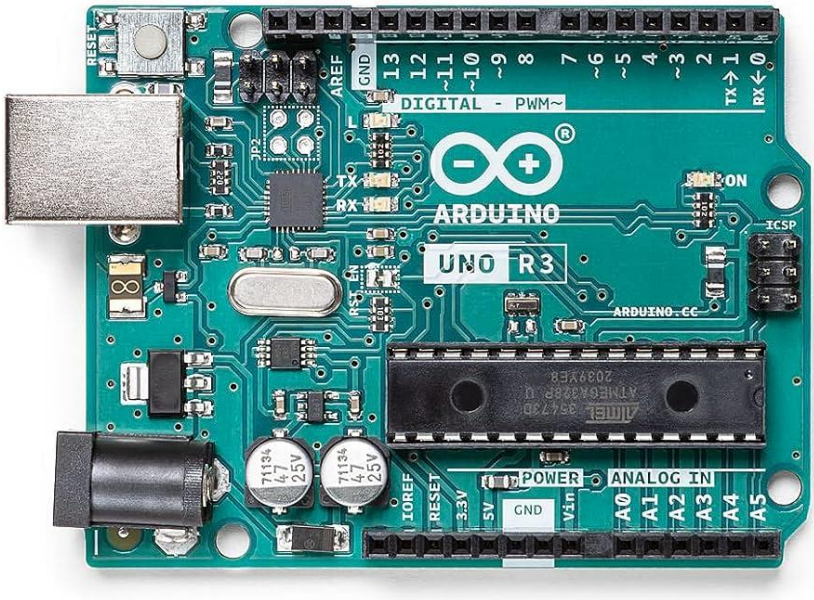

Figura 5-2. Placa Arduino UNO

En Arduino Uno, las funciones *setup* y *loop* son dos componentes fundamentales del programa que determinan el comportamiento del microcontrolador.

La función *setup* se ejecuta una vez al inicio del programa y se utiliza para realizar configuraciones iniciales, como la inicialización de pines de entrada/salida, la configuración de la velocidad de comunicación serial, la inicialización de variables y la preparación del entorno para la ejecución del programa principal. Es el lugar donde se definen las condiciones iniciales del sistema y se realizan las configuraciones necesarias para el funcionamiento correcto del dispositivo.

Por otro lado, la función *loop* se ejecuta continuamente después de que *setup* ha finalizado su ejecución. En esta función, se implementa el código principal del programa, que se ejecuta de forma repetida en un bucle infinito. Aquí es donde se definen las acciones y tareas que el microcontrolador debe realizar de manera recurrente, como la lectura de sensores, el procesamiento de datos, la toma de decisiones y el control de actuadores. La función *loop* es responsable de mantener en funcionamiento el sistema y de realizar las acciones necesarias en respuesta a las condiciones del entorno o a eventos específicos.

5.2 Variables utilizadas

En este apartado se va a hacer un resumen de las variables más importantes del programa y con que componente física del proyecto se corresponden. En la siguiente tabla se puede ver un resumen de las variables globales utilizadas y una breve descripción de las mismas:

Tabla 5–1 Variables del programa

Variable	Descripción
ADCref1	Esta variable se corresponde con la entrada analógica de la temperatura de referencia. Es un valor que va de 0 a 1023 y que será necesario convertir a temperatura para poder operar con ella.
Vref1	Paso intermedio entre ADCref1 y Tref1, se calcula haciendo una regla de tres entre 0 y 1023, y 0 y 5, siendo 5 V equivalente a 1023 bits del ADC
Tref1	Valor de la temperatura de referencia del sistema
ADCref2, Vref2 y Tref2	Equivalentes de lo anterior para el segundo circuito
ADCNTC1	Entrada analógica de la temperatura medida por el sensor NTC, igualmente se necesita pasar de valor de bits a temperatura. Esta es la entrada proveniente del circuito anteriormente simulado en Microcap
VNTC1_f	Valor de tensión de salida del circuito de medición de temperatura
V2_1	Paso intermedio para llegar al valor de temperatura medido por la NTC
R1	Valor de resistencia del sensor NTC
T1	Valor de la temperatura medida por el sensor NTC
ADCNTC2, VNTC2_f, V2_2, R2 Y T2	Equivalentes de lo anterior para el segundo circuito

Más adelante se verá detalladamente como se calcula en el programa cada uno de estos valores, a continuación, se procede a aclarar como se inicializan los pines, el PID y la pantalla.

5.3 Inicialización de pines, PID y pantalla *(setup)*

5.3.1 Inicialización de pines

El primer paso a la hora de realizar un Proyecto con Arduino, es identificar los pines que se van a usar como entrada o salida.

A continuación, se detalla una tabla resumen de los pines que se van a utilizar, aunque se verán más en detalle cuando se defina el componente al que pertenecen, como, por ejemplo, la pantalla.

Tabla 5–2 Pines usados en el programa

Nº del pin	Tipo	Componente
13	E/S Digital	Pantalla
11	E/S Digital	Pantalla
10	E/S Digital	Pantalla
9	E/S Digital	Pantalla
8	E/S Digital	Pantalla
2	Salida Digital	LED 1
4	Salida Digital	LED 2
6	Salida Digital	LED 3
7	Salida Digital	LED 4
3	Salida PWM	PID
5	Salida PWM	PID

Nº del pin	Tipo	Componente
A0	Entrada analógica	Convertidor A/D
A1	Entrada analógica	Convertidor A/D
A2	Entrada analógica	Convertidor A/D
A3	Entrada analógica	Convertidor A/D

Como se ha visto anteriormente, en la función *setup* se inicializan los pines que sirven como entrada o salida digital. En este caso, los únicos 4 pines que hay de este tipo son los correspondientes a los LEDs, los cuales, según la librería de funciones de Arduino [9] , se inicializan de la siguiente manera:

```
pinMode(pin, mode);
```

- Pin: número del pin del que se desea establecer el modo
- Mode: puede ser INPUT, OUTPUT o INPUT_PULLUP, dependiendo de si es de entrada, de salida o de entrada con una resistencia conectada, respectivamente

En la siguiente tabla se puede ver la inicialización de pines establecida en este apartado:

Tabla 5–3 Pines inicializados en la función *setup*

pin	mode	Componente
2	OUTPUT	LED 1
4	OUTPUT	LED 2
6	OUTPUT	LED 3
7	OUTPUT	LED 4

5.3.2 Inicialización de PID

Un controlador PID es un mecanismo de control por realimentación que calcula la desviación o error existente entre el valor medido y el que se quiere obtener pare realizar una corrección sobre el mismo.

El cálculo del control PID se da en tres parámetros distintos: el proporcional, el integral y el derivativo. El valor proporcional determina la reacción del error actual. El integral genera una corrección proporcional a la integral del error acumulado, esto asegura que el error en estado estacionario en régimen permanente se reduzca a cero. El derivativo considera la tendencia del error y permite una repercusión rápida de la variable después de presentarse una perturbación en el proceso. Ajustando estas tres variables en el algoritmo de control del PID, el controlador puede preveer un control diseñado para las especificaciones técnicas que requiera el proceso a realizar.

Dicho de otra manera, cuanta más acción proporcional se ponga, más mejora la rapidez del sistema, pero es menos estable. La acción integral elimina el error en estado estacionario, es decir, actúa muy bien en el régimen permanente, pero cuanta más acción integral se le ponga al sistema, más aumenta la precisión, pero a su vez disminuye la estabilidad. Cuanta más acción derivativa se ponga más mejora el rebose del sistema, es decir, aumenta el amortiguamiento del sistema, lo que aumenta la estabilidad del sistema, pero hace que el error se disminuya más lentamente.

Dicho esto, la base para obtener un control óptimo es jugar con las diferentes constantes de proporcionalidad (Kp, Kd y Ki), y ver que combinación de las tres proporciona un control óptimo.

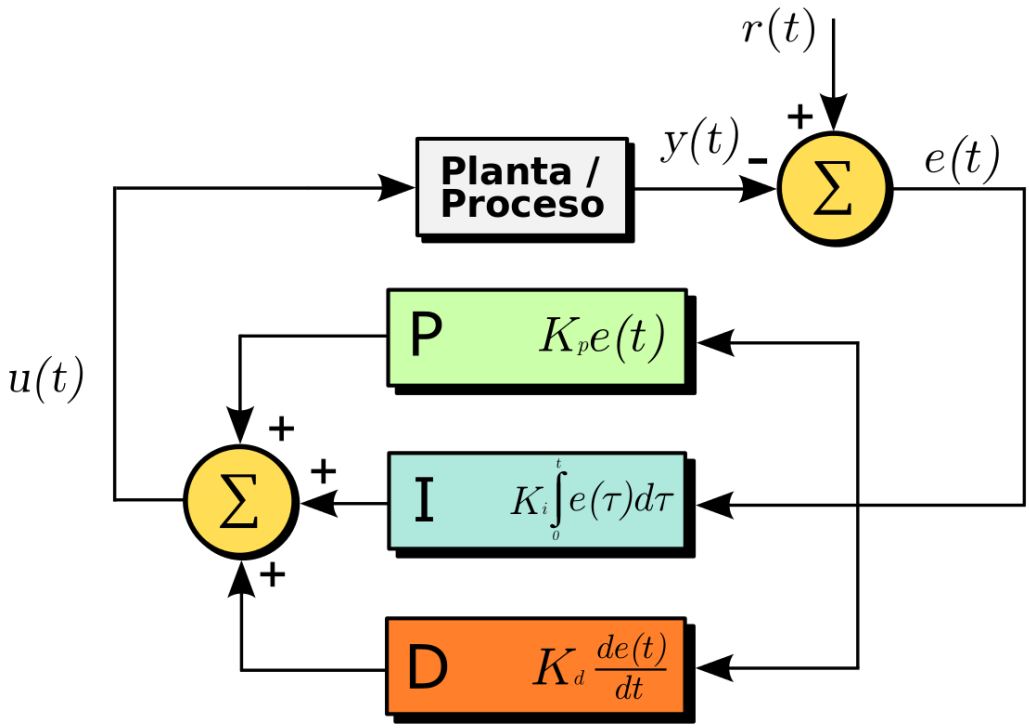

Figura 5-3. Esquema de funcionamiento de un controlador PID

Ahora se van a ver detalladamente los diferentes tipos de control y el efecto que tienen en el control, de esta manera será más fácil entender como trabajan en conjunto [10].

- **Control proporcional, P:**

  En este tipo de control la salida sube hasta el valor final, produciendo un rebose y un error estacionario. Este control es bastante impreciso, sobre todo con señales grandes, que producirán un error en estado estacionario muy grande.

  Lo bueno de este controlador es que tiene una respuesta muy rápida, pero genera mucho error, pudiendo llegar en algunos casos a desestabilizar el sistema.

- **Control proporcional derivativo, PD:**

  Como se ha explicado antes, este control se utiliza para eliminar el rebose y mejorar la estabilidad del sistema, para ello es necesario un mecanismo que prediga que va a producirse ese rebose. Esto se consigue añadiendo a la señal de control un término proporcional a la derivada del error.

  Esta acción no es capaz de eliminar el error en el estado estacionario, lo disminuye, pero no lo elimina completamente.

  Este sistema amplifica mucho las señales de alta frecuencia, como el ruido, por lo que no se suele implementar en cierto tipo de sistemas que contengan este tipo de señales.

  En general, valores elevados de Kd hacen al sistema más lento, pero más estable.

- **Control proporcional integral, PI:**

  Este tipo de control se usa para eliminar el error en estado estacionario, es decir, para realizar un control preciso. Consigue eliminar completamente este error debido a la acción integral. Lo malo es que el término integral disminuye la estabilidad del sistema.

  El control integral es adecuado para señales que presentan mucho ruido, ya que filtra el ruido de alta frecuencia.

- **Control proporcional integral derivativo, PID:**

  El control proporcional integral derivativo presenta una cmbinación de los tres anteriores, entre los que hay que buscar una armonía para que la respuesta sea lo mejor posible. Dependiendo del sistema y de las perturbaciones que se puedan encontrar en el mismo se podrá aumentar el efecto de un control o de otro.

  En general, en los instantes iniciales de la etapa de control, predomina el control proporcional y, a medida que va pasando el tiempo y persistiendo el error, la acción integral va aumentando.

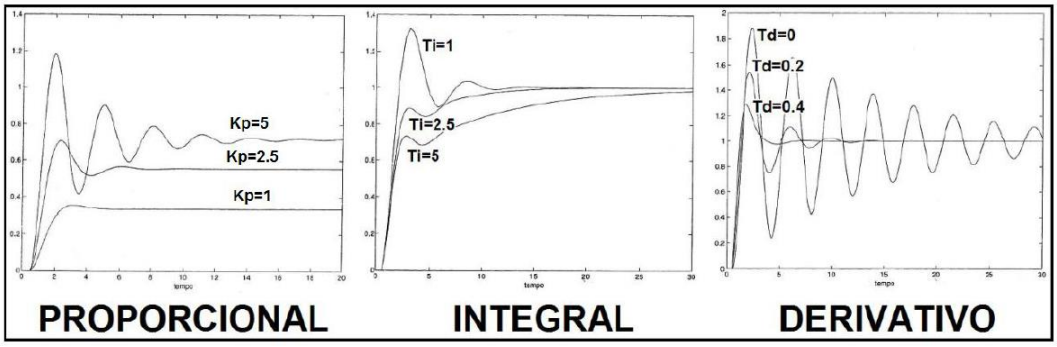

Figura 5-4. Ejemplo de efectos de los diferentes controladores sobre un sistema

Como se puede observar en la imagen, en el control proporcional, un valor muy elevado de Kp produce inestabilidad, valores altos de Ki generan una respuesta precisa, pero con sobreoscilaciones y valores altos de Kd producen una respuesta rápida pero cada vez más inestable.

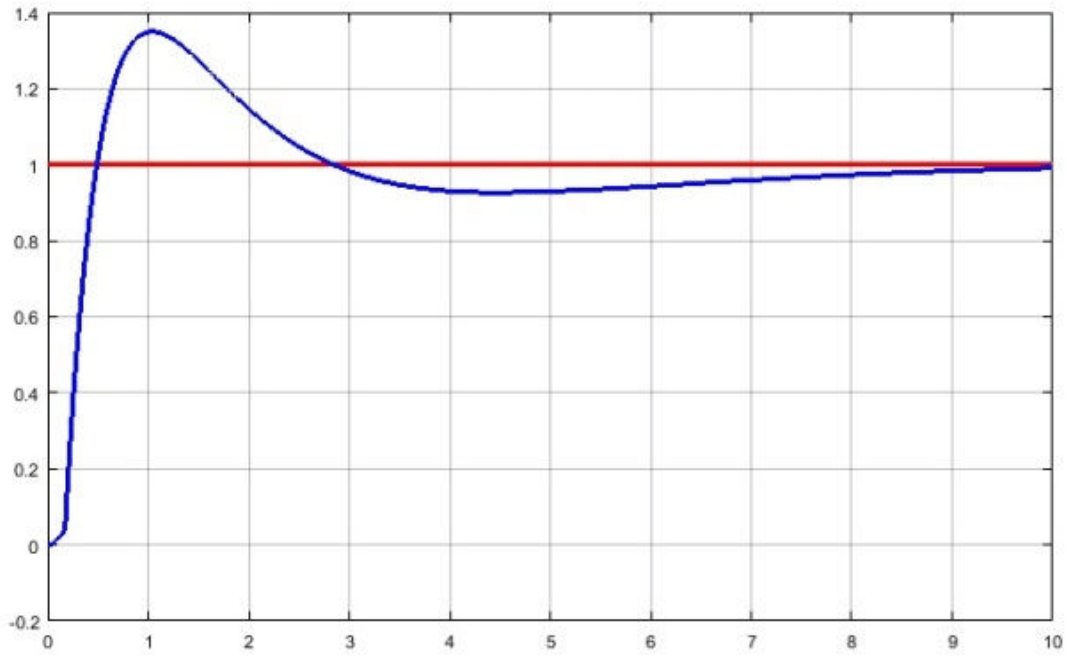

Figura 5-5. Ejemplo de respuesta dada por un controlador PID

Este es un ejemplo de salida de un controlador PID, siendo la línea roja la señal de referencia y la azul el valor corregido por el PID. Podemos ver que este sistema sobreoscila, pero su respuesta es rápida y su adaptación al error también.

Una vez vistos los tipos de controlador, y, habiendo entendido el funcionamiento del controlador PID, se va a proceder a su implementación en Arduino. Para ello, Arduino cuenta con programas de ejemplo en su librería para implementar controladores sin mayor dificultad en cualquier programa.

Para este programa se van a implementar dos controladores PID, uno por cada circuito, pero la programación para ambos es idéntica, ya que van a realizar la misma función, pero en circuitos distintos.

Como se ha visto antes, se va a utilizar el controlador PID para establecer el error entre la temperatura de referencia y la real del sistema, para, posteriormente, corregir ese error de manera precisa en el menor tiempo posible y sin sobreoscilaciones.

En términos de programación, el controlador PID se divide en cuatro variables: **Input, Output, Setpoint** y **error.**

El Setpoint es el valor estacionario que se desea alcanzar. En este caso, se corresponde con las temperaturas de referencia (Tref1 y Tref2).

El Input es la temperatura real del sistema. Esta señal debe ser previamente filtrada (se verá como más adelante) y se usa para ver el error que existe en ese momento en el sistema.

El error es la diferencia existente entre el valor real entre la temperatura real y la referencia, o lo que es lo mismo, entre el Input y el Setpoint. Con este valor el controlador PID, dependiendo del valor de las constantes, sabe como actuar.

El Output es el valor de salida del controlador, este valor es la señal PWM que entra al driver, como se vió en el *Apartado 4.4.1.*

La señal de salida es del tipo PWM por una razón: Arduino no es capaz de proporcionar auténticas salidas analógicas, por lo que para salvar esa limitación se usan las señales PWM, que consisten en activar una salida digital durante un tiempo y mantenerla apagada durante el resto del ciclo. El promedio de la tensión a la salida, a lo largo del tiempo, será igual al valor analógico deseado. La proporción de tiempo que está activada la señal es el ***duty cycle*** y, generalmente, se expresa en tanto por ciento.

Como se ha especificado antes, para usar un PID en Arduino, se puede implementar una librería que contiene todas las funciones y cálculos internos [11]. La librería es la siguiente:

```
#include <PID_v1.h>
```

De momento, en la función Setup, solo se va a inicializar el controlador y, más adelante, se verá la asignación de variables, el filtrado de la señal de medida y la puesta en funcionamiento.

Para inicializar el PID solamente hay que especificar el tiempo de muestreo que este tendrá y el modo de actuación, que en este caso va a ser automático. Estas son las funciones a utilizar:

```
myPID.SetSampleTime(10);
myPID.SetMode(AUTOMATIC);
```

5.3.3 Inicialización de Pantalla

La pantalla que se va a utilizar para este proyecto es un modelo 'Adafruit_ST7735', mostrada a continuación:

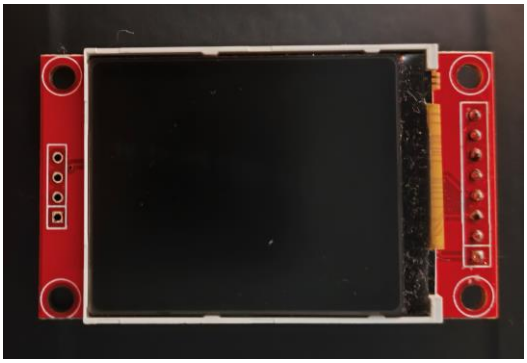

Figura 5-6. Pantalla Adafruit_ST7735

Esta es una pantalla a color de matriz activa TFT (Thin-Film Transistor) ampliamente utilizada en proyectos de electrónica, está disponible en varios tamaños, aunque la que se va a usar es de 1,8 pulgadas (128x160 mm). Tiene una resolución bastante nítida. Esta pantalla además puede interactuar a través de interfaces como SPI o GPIO, las cuales, aunque no se van a usar en este proyecto, pueden ser útiles para futuras ampliaciones, pudiendo hacer que el mecanismo se controle desde, por ejemplo, un dispositivo móvil a través del SPI.

El dispositivo cuenta con 8 pines, los cuales irán conectados tal y como se muestra en la siguiente tabla:

Tabla 5–4 Pines de la pantalla

Nombre del pin (Pantalla)	Nº pin Arduino
LED	3,3V
SCK	13
SDA	11
A0	9
RESET	8
CS	10
GND	GND
VCC	5V

A continuación, se va a hacer una breve descripción de la funcionalidad de cada uno de los ocho pines:

- **LED:** Este pin controla la iluminación LED de la pantalla. Al aplicar voltaje aquó, se enciende la pantalla.
- **SCK:** Es el pin del reloj. Su función es sincronizar la transferencia de datos entre la pantalla y el microcontrolador.
- **SDA:** Es el pin de datos de serie. Se usa para enviar y recibir datos desde la pantalla hacia el microcontrolador, y viceversa.
- **A0:** Es el pin de selección de datos. Se utiliza para indicar a la pantalla si se están enviando datos.
- **RESET**: Este pin se utiliza para restablecer la pantalla
- **CS (Chip Select):** Se utiliza para seleccionar la pantalla cuando se envían datos desde el microcontrolador
- **GND:** Es el pin de tierra.
- **Vcc**: Se conecta a la fuente de alimentación y proporciona energía a la pantalla.

El primer paso para inicializar la pantalla es incluir las librerías necesarias, sacadas de la web de Arduino, e incluir los pines:

```
#include <Adafruit_GFX.h>
#include <Adafruit_ST7735.h>
#define TFT_CS     10
#define TFT_RST    8
#define TFT_DC     9
#define TFT_SCLK 13
#define TFT_MOSI 11
```

El siguiente paso es definir la pantalla como un objeto con la función escrita a continuación:

```
Adafruit_ST7735 tft = Adafruit_ST7735(TFT_CS, TFT_DC, TFT_RST);
```

En este caso, 'Adafruit_ST7735' es la clase del objeto, 'tft' el nombre del mismo, y el resto es la conexión de pines que se han definido antes con la pantalla.

Seguidamente, una vez ya empezada la función Setup, hay que iniciar la comunicación con la pantalla. Para ello se usan las siguientes funciones:

- `tft.initR(INITR_BLACKTAB);`

  Este comando hace que se inicialice la pantalla con una configuración específica de varios parámetros internos de la pantalla, como la polaridad de los píxeles o la orientación de la pantalla.

- `tft.setRotation(3);`

  Este comando establece la rotación de la pantalla. Su entrada puede ser '0', '1', '2' o '3', dependiendo de la orientación en la que queramos leer los valores de la pantalla. Para este ejemplo concreto, se aplicaría una rotación de 270 grados con respecto a la configuración original.

- `tft.fillScreen(ST7735_BLACK);`

  Este comando es utilizado para rellenar la pantalla de un color específico. En este ejemplo, se llenaría la pantalla de color negro.

Para cerrar la función Setup, se va a rellenar la pantalla con los valores que deseamos representar. En este caso, se ha considerado que los valores a representar van a ser, en primer lugar, la temperatura de referencia del dispositivo de control, para que sea visible y pueda ser regulada con facilidad y, en segundo lugar, la temperatura real captada por el sensor de temperatura, para poder llevar un seguimiento preciso del comportamiento del sistema y se vea que el control es preciso.

En primer lugar, se va a rellenar la pantalla de blanco con la función vista anteriormente, y después se van a representar las letras y números en color negro.

Para representar las letras y números que se desea, se han definido dos funciones, las cuales se pueden encontrar en el *'Anexo I: Código Completo'*. Son las siguientes:

- `void testdrawtext(char *text, uint16_t color, int pos_x, int pos_y)`

  Esta es la función para dibujar un texto. Los valores de entrada son el texto que se desea escribir, el color en el que se quiere, y las posiciones x e y en las que se va a empezar a escribir.

- `void testdrawnum(float num, uint16_t color, int pos_x, int pos_y)`

  Esta es la función para dibujar un número. Los valores de entrada son el valor que se desea escribir, el color en el que se quiere, y las posiciones x e y en las que se va a empezar a escribir.

5.4 Lectura de datos (Convertidor Analógico-Digital)

En este punto comienza la función **loop()**, es decir, todo lo aplicado de aquí en adelante volverá a repetirse cada ciclo de reloj mientras el programa esté activo.

Como se ha dicho anteriormente, son cuatro las variables que se van a leer por el convertidor analógico digital, estas son las dos temperaturas de referencia y las dos temperaturas leídas por cada uno de los sensores. Esta temperatura viene en forma de voltaje y, una vez medida, se calculará la temperatura en función al valor del voltaje de entrada.

En primer lugar, se va a entender como funciona el convertidor analógico digital y como se mide la señal:

El ADC (Analog-Digital Converter) tiene una resolución de 10 bits. Esto significa que puede representar valores en una escala de 0 a 1023. El valor 0 es el mínimo voltaje, mientras que 1023 es el máximo.

Arduino utiliza una referencia de voltaje para comparar la señal analógica de entrada. Por defecto, esta referencia es la tensión de alimentación, es decir, 5 voltios. Sin embargo, también se puede establecer una referencia de voltaje mediante código, aunque este no va a ser el caso.

El dispositivo Arduino Uno cuenta con 6 entradas de ADC (A0-A5), que se conectan a un multiplexor analógico. Este multiplexor permite al microcontrolador seleccionar una entrada analógica en específico para convertirla en digital.

El proceso de conversión se realiza mediante dos pasos: muestreo y cuantificación.

- **Muestreo:** En esta etapa, el ADC toma muestras de la señal a intervalos regulares. La velocidad de muestreo es controlada por el micro.
- **Cuantificación:** Después, el ADC compara la referencia de voltaje y asigna un valor digital en función de su relación con la referencia.

Para leer un valor analógico en Arduino, se utiliza la función **'analogRead(pin)'**, donde 'pin' es el número de pin analógico que se desea leer. Esta función devuelve un valor de 0 a 1023, 0 cuando el valor es 0 voltios y 1023 cuando es de 5 voltios.

5.5 Cálculos de Resistencias y error

A continuación, se van a describir todos los cálculos realizados en el programa para obtener los valores de temperatura y la resistencia de los sensores de temperatura.

5.5.1 Cálculo de la temperatura de referencia

El valor leído por los pines A0 y A1, son los valores de referencia de la temperatura de cada uno de los circuitos. Como ya hemos visto, el convertidor digital lo que nos proporciona es un valor entre 0 y 1023, dependiendo del valor del voltaje a la entrada.

El primer paso está bastante claro, hay que volver hacia atrás, y, con el valor proporcionado por el ADC, sacar el valor del voltaje a la salida. Esto se consigue fácilmente mediante una simple regla de tres:

$$Vref1 = \frac{ADCRef1 * 5}{1023} \qquad \text{Ecuación 5-1}$$

De esta manera, cuando ADCRef1 sea 1023, el voltaje será 5 voltios y, cuando sea 0, el voltaje será de 0 voltios igualmente.

Una vez obtenida la temperatura, si repasamos lo anteriormente descrito, se establece el mínimo de temperatura en 25ºC y el máximo en 100ºC. Es decir, se van a controlar temperaturas siempre dentro de este rango, por lo que interesa que, para 0 voltios el valor de la temperatura de referencia sea de 25ºC, y lo mismo para 100ºC y 5 voltios.

De esta manera, con otra regla de tres, el cálculo final quedaría así:

$$Tref1 = \frac{Vref1 * 75}{5} + 25 \qquad \text{Ecuación 5-2}$$

Con esto se obtendría la temperatura de referencia del circuito 1, para el segundo, las operaciones serían exactamente iguales.

Seguidamente, se va a ver el cálculo de la temperatura real. Este cálculo no va a resultar tan sencillo, ya que no es un potenciómetro lo que se quiere medir, si no un sensor.

5.5.2 Cálculo de la temperatura de referencia

El primer paso es el mismo, hay que leer las entradas ADC conectadas a los pines A2 y A3, que corresponden con la salida del circuito visto en el *apartado 4.3.*

El primer paso es el mismo que en el punto anterior, hay que hacer una regla de tres para conocer el valor del voltaje de salida.

$$VNTC1\_f = \frac{ADCNTC1 * 5}{1023}$$ Ecuación 5-3

Una vez obtenido el voltaje, hay hacer el camino inverso por el circuito, hasta llegar al valor de la resistencia del sensor NTC. A modo de recordatorio, el circuito es el siguiente:

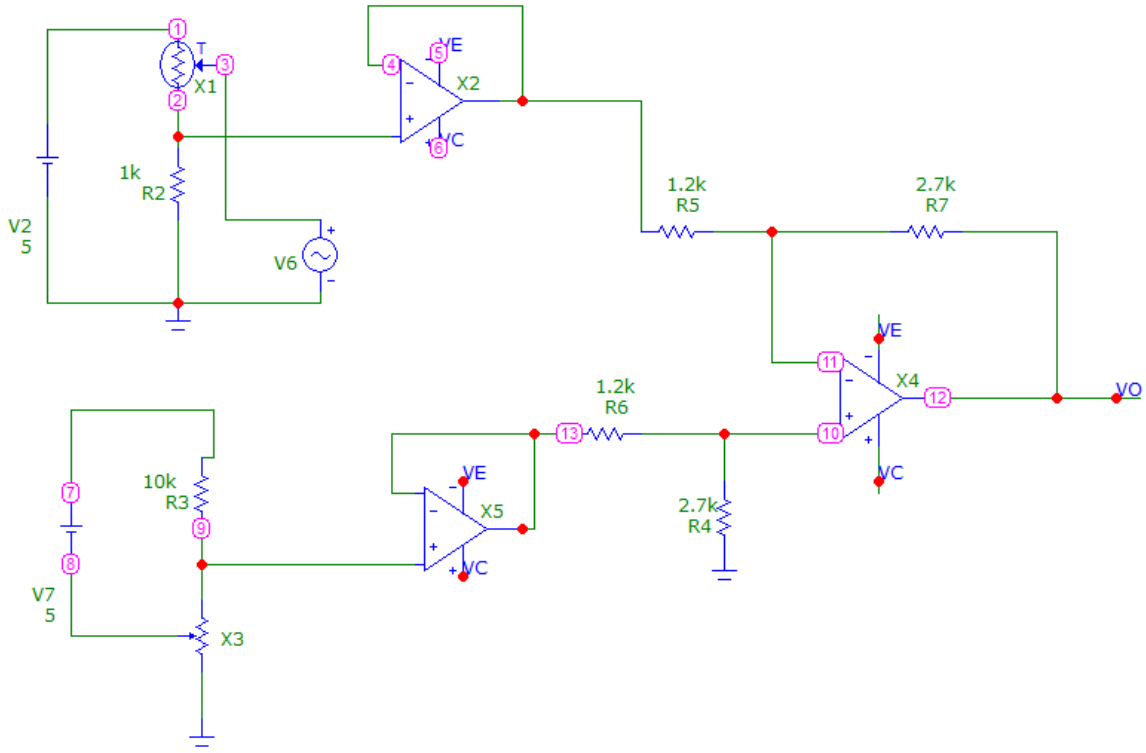

Figura 5-7. Esquema del circuito

En primer lugar, hay que calcular el voltaje antes de entrar en el amplificador operacional. Como datos tenemos que la relación de resistencias R2/R1 es igual a 2,25. También se sabe que el valor que entra por el terminal negativo del Amplificador es 2,5 voltios, por lo que tenemos la siguiente ecuación:

$$V0 = (2{,}5 - V2\_1) * 2{,}25 => V2\_1 = \frac{5{,}625 - VO}{2{,}25}$$ Ecuación 5-4

De esta manera se logra sacar V2_1, que se corresponde con la salida del divisor de tensión de la NTC y la resistencia R2 en la imagen. El siguiente paso es deshacer este divisor de tensión para sacar el valor de la resistencia del sensor.

Para conseguir este valor lo más exacto posible, se ha medido la resistencia de 1k con un voltímetro y ha dado un valor de 988Ω, por lo que se va a hacer la operación con este valor.

Se ha denominado R1 al valor de resistencia del sensor NTC.

$$R1 = 5 * \left(\frac{988}{V2\_1}\right) - 988$$ Ecuación 5-5

Una vez obtenido el valor de la resistencia del sensor, debido a su configuración interna, se puede definir el valor de temperatura debido a su configuración interna. Esto se puede encontrar en la hoja del fabricante del sensor NTC. En este caso, las ecuaciones para obtener la temperatura son las siguientes:

$$alfa1 = \left(\frac{1000}{T0}\right) + (\frac{1000}{beta} * \ln(\frac{R1}{R0}))$$ Ecuación 5-6

$$T1 = \frac{1000}{alfa1} - 273{,}15$$ Ecuación 5-7

Donde:

- **Beta:** es una constante de diseño del sensor NTC, cuyo valor es 3410.
- **T0:** valor de temperatura ambiente, 25ºC, aunque las unidades utilizadas son en kelvin, por lo que sería igual a 298,15 K.
- **R0:** es una constante de diseño del sensor NTC que mide el valor de la resistencia máxima que puede alcanzar el sensor. En este caso R0 = 10000.

5.5.3 Cálculo del error

El cálculo del error se realiza restando el valor de la temperatura real con la temperatura de referencia, para ver la diferencia entre ambas.

Se va a implementar un sistema muy simple para poder observar de una manera muy visual como está siendo el error cometido por el control. Esto consiste en instalar 4 LEDs, dos por circuito, y, dependiendo del LED que se encienda, se puede ver si en ese momento el error es mayor de un valor establecido.

En la pantalla también se puede ver el valor de las temperaturas en todo momento, pero para verlo de una manera más fácil se va a hacer uso de los LEDs.

Dependiendo de si el valor del error es mayor o menor de 5ºC, se va a encender un LED u otro, así se puede ver si el control es estable o si tarda demasiado en llegar al rango de valores deseados.

Los LEDs se activan o desactivan gracias a la siguiente orden en Arduino:

```
digitalWrite(pin, HIGH) //Encender LED
digitalWrite(pin, LOW) //Apagar LED
```

5.6 Actuación del control PID

Paso previo a la actuación del PID, hay que asignar las señales Input, Output y Setpoint.

Setpoint será el valor de la temperatura de referencia previamente calculado. Pero con el Input hay que realizar una acción antes de asignarlo. Input será el valor de la temperatura real, pero para calcularla, se ha pasado por alguna operación complicada, como, por ejemplo, un logaritmo. Por este motivo, es necesario filtrar la señal antes de su entrada en el PID.

Para ello, se establece un valor constante llamado 'alfa' de valor 0,05 y se aplica el siguiente filtro paso bajo:

$$medida1 = T1 \quad \text{Ecuación 5-8}$$

$$medidaf1 = medida1 * alfa + (1 - alfa) * medidaf1 \quad \text{Ecuación 5-9}$$

El valor de 'alfa' controla cuanto peso se le da a la nueva medida en comparación con la anterior, un valor más alto de 'alfa' hará que el filtro responda más rápidamente a los datos, pero será mas propenso a contener ruido, cosa que se desea evitar.

Una vez asignados el Input y el Setpoint, se activa el PID con el siguiente comando:

```
myPID.Compute();
```

Y, acto seguido, se manda el Output al pin de salida que corresponda. En el caso de el circuito 1 se mandará al pin 3 y, en caso del circuito 2, al pin número 5.

```
analogWrite(pin, Output);
```

El controlador PID actúa en cada ciclo de reloj para proporcionar un control preciso y actualizado de la temperatura.

5.7 Actualización de la Pantalla

Por último, una vez actualizados todos los valores, hay que mostrarlos por la pantalla. Al ser los ciclos demasiado rápidos, la pantalla se satura si tiene que actualizarse en cada uno de ellos. Para prevenir esto, se ha creado una variable llamada 'tempo', que va sumando uno cada ciclo que pasa.

Cuando esta variable llega a 20, es decir, cada 20 ciclos de reloj, se entra aun if para actualizar los valores de la pantalla, siendo este valor bastante adecuado para garantizar una actualización rápida de los valores y, a su vez no saturar la pantalla.

Dentro del if, el comando que se ordena es el mismo que se vió en el *Apartado 5.3.3,* para escribir un número en pantalla:

```
void testdrawnum(float num, uint16_t color, int pos_x, int pos_y)
```

Sin embargo, haciendo pruebas surgió un problema. El nuevo número se sobrescribía encima del anterior y después de 5 actualizaciones los números eran ilegibles.

En un primer momento se pensó en pintar toda la pantalla de blanco y volver a escribir todo encima, incluido el texto, pero, al final se optó por crear una función que pinte un rectángulo blanco únicamente encima de los sitios en los que haya números, evitando modidicar el texto.

Esta función se puede encontrar completa en el *'Anexo I: Código Completo'* y es la siguiente:

```
void testdrawrects(uint16_t color, int pos_y,int pos_final)
```

Sus valores de entrada son: el color del rectángulo, la posición y inicial y la posición y final. El ancho en el eje x está definido dentro de la propia función.

A continuación, se muestra como quedarían todos los valores en pantalla (los valores mostrados están establecidos manualmente, no es resultado de ninguna simulación del circuito):

Figura 5-8. Pantalla funcionando

Una vez actualizada la pantalla, se vuelve a ejecutar la función loop para seguir actualizando los valores de temperatura y seguir controlando el sistema hasta que este se desconecte.

6 MONTAJE EN PLACA DE PRUEBAS

En este capítulo se van a describir los pasos a seguir para montar el circuito diseñado en una placa de pruebas. Esto se hace para probar el montaje y el programa diseñados antes de mandar a fabricar la placa que se va a diseñar posteriormente para el diseño definitivo del dispositivo. A continuación, se va a proceder a listar los componentes físicos que se van a usar y el procedimiento de montaje.

El objetivo principal de este procedimiento es probar que lo diseñado anteriormente funciona correctamente e ir haciendo diversas pruebas, como el funcionamiento del PID mediante una resistencia LDR y un LED.

6.1 Elementos físicos

Primeramente, se van a describir los elementos que se van a usar para el montaje, tales como potenciómetros, resistencias, amplificadores operacionales, etc.

6.1.1 Placa de pruebas

Será una placa de pruebas base para uso educativo, con 830 puntos de conexión.

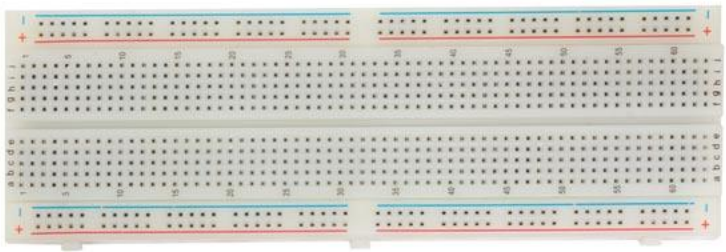

Figura 6-1. Placa de pruebas

6.1.2 Amplificador operacional

Se van a usar 3 amplificadores operacionales por circuito, es decir, un total de 6.

El modelo de amplificador que se va a utilizar será el LT1490, que consta de 8 pines. Este amplificador es ideal para esta aplicación ya que es un componente bastante versátil y de bajo costo. Además, este dispositivo costa con dos amplificadores operacionales en su interior, por lo que se pueden agrupar de 2 en 2, lo que implica que se puede simplificar de 6 componentes a 3.

A continuación, se puede ver el conexionado interno de este elemento.

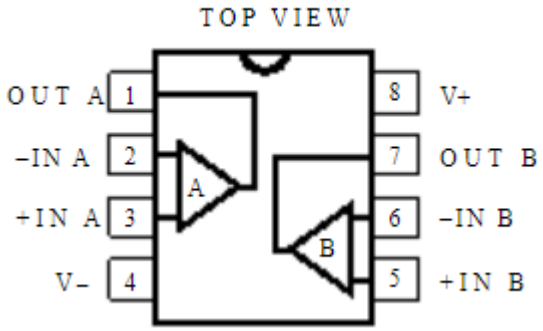

Figura 6-2. Conexionado interno del LT1490

6.1.3 Resistencias

Como se ha visto en el diseño del circuito, todas las resistencias usadas se han elegido de manera que estén dentro de los valores normalizados de fabricación. Estas resistencias tienen una tolerancia del 5%.

Se van a utilizar resistencias de orificio pasante normalizadas, cuyo valor se puede ver con un código de colores. En la siguiente tabla, se ve cuantas resistencias se van a usar para cada valor:

Tabla 6–1 Resistencias utilizadas

Valor (kΩ)	Cantidad
0,1	2
0,68	4
1	2
1,2	4
2,7	4
10	2

Figura 6-3. Resistencia de orificio pasante

6.1.4 Potenciómetro

Se van a usar 2 potenciómetros por circuito, es decir, un total de 4. En concreto, se usará uno para el circuito de referencia de temperatura y otro para fijar el valor de entrada a restar en el amplificador operacional.

El potenciómetro usado será el modelo BOURNS 3296 de 10 kΩ.

Figura 6-4. Potenciómetro BOURNS 3296

6.1.5 Drivers

El driver usado será el modelo TC4427. Su diseño robusto y su capacidad de operación en un amplio rango de voltajes lo convierten en una opción versátil para sistemas que requieren control preciso de la conmutación de potencia.

Se hará uso de un driver por circuito que, como se ha visto anteriormente, recogerá la salida PWM del controlador para determinar el ciclo de conmutación del transistor MOSFET y controlarlo eficientemente.

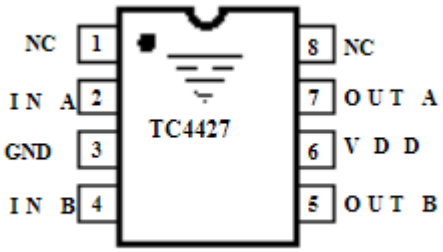

Figura 6-5. Esquema de conexionado del driver

Figura 6-6. Driver TC4427

6.1.6 Transistor MOSFET

Como ya se ha visto, un transistor MOSFET es un tipo de transistor utilizado en electrónica para controlar el flujo de corriente entre sus terminales.

El transistor elegido ha sido el modelo K30E06N1, un MOSFET tipo canal-n diseñado para aplicaciones de conmutación de baja tensión. Tiene una capacidad de drenaje de hasta 60 voltios y una corriente de drenaje continua de 30 amperios, lo que lo hace adecuado para una amplia gama de aplicaciones de electrónica de potencia.

Figura 6-7. MOSFET K30E06N1

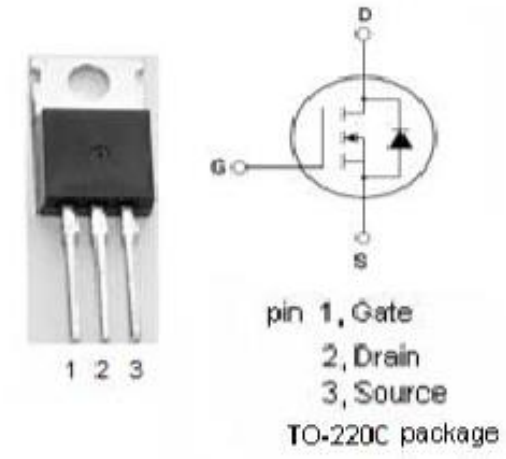

Figura 6-8. Esquema de conexionado del transistor MOSFET

6.1.7 Diodos LED

Se conectarán 4 diodos LED al dispositivo, dos verdes y dos rojos. Cada par de LEDs, uno de cada color, irá asociado a un circuito calentador, cuando el error entre la referencia y la temperatura de la muestra sea menor de 3 ºC, se mantendrá encendido el LED verde, en caso contrario, se encenderá el rojo.

6.2 Proceso de Montaje

A continuación, se va a montar el circuito siguiendo todos los pasos de diseño vistos en los apartados anteriores para realizar las primeras pruebas experimentales.

6.2.1 Circuito de temperatura de referencia

Esta parte del circuito es muy simple, solo tiene un potenciómetro y una resistencia, completando así un divisor de tensión. Se ha montado solo uno de los dos circuitos para la prueba, el otro será igual, pero la salida entrará en el pin A2 de Arduino, en vez del A0.

El circuito quedaría de la siguiente manera:

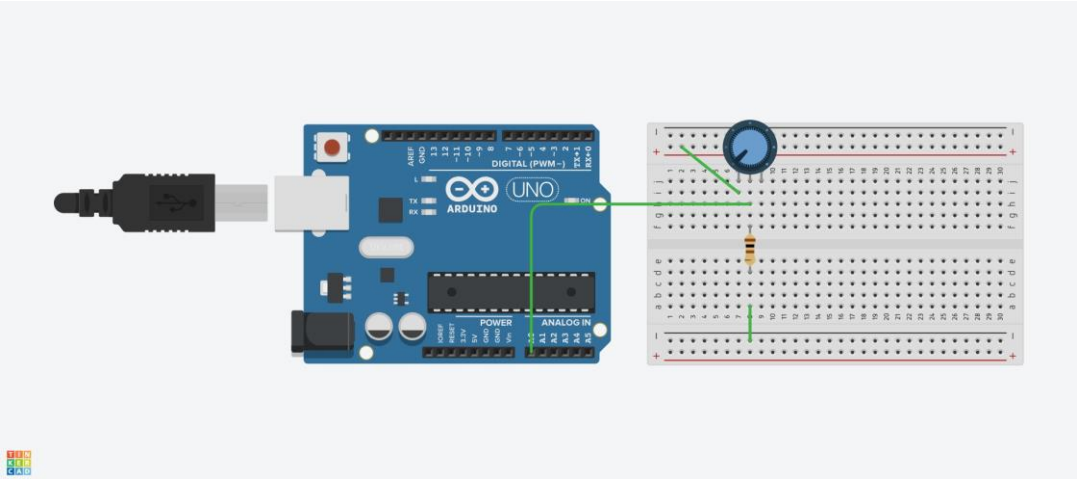

Figura 6-9. Montaje del circuito de temperatura de referencia

6.2.2 Circuito de actuación

En esta parte se va a montar en la placa el circuito actuador del dispositivo, formado por el driver, el MOSFET y la resistencia de cobre, dando el siguiente resultado:

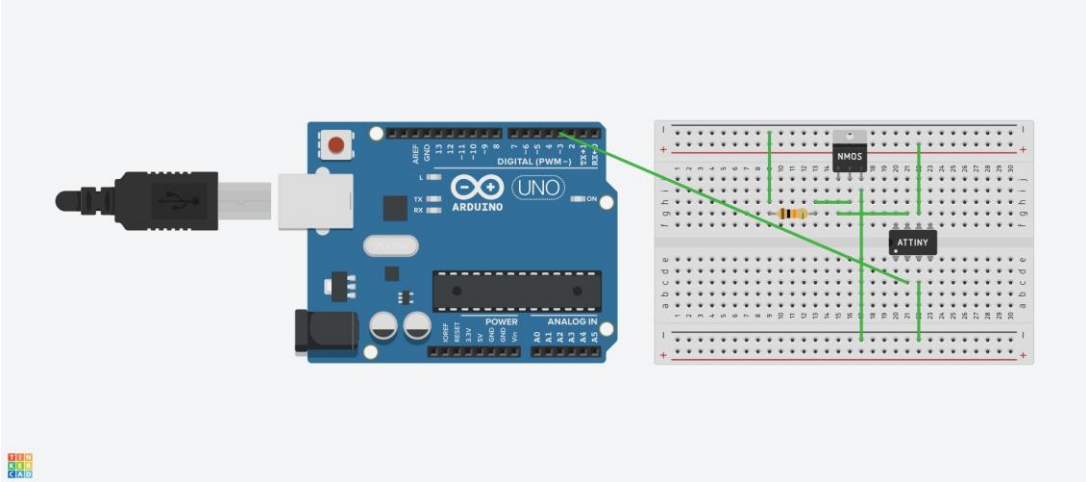

Figura 6-10. Montaje del circuito actuador

En la imagen se puede ver como se realiza el conexionado. Para simplificar los componentes, “ATTINY” representa al driver, y la resistencia representa lo que sería la resistencia de cobre que se va a calentar.

6.2.3 Circuito del amplificador operacional

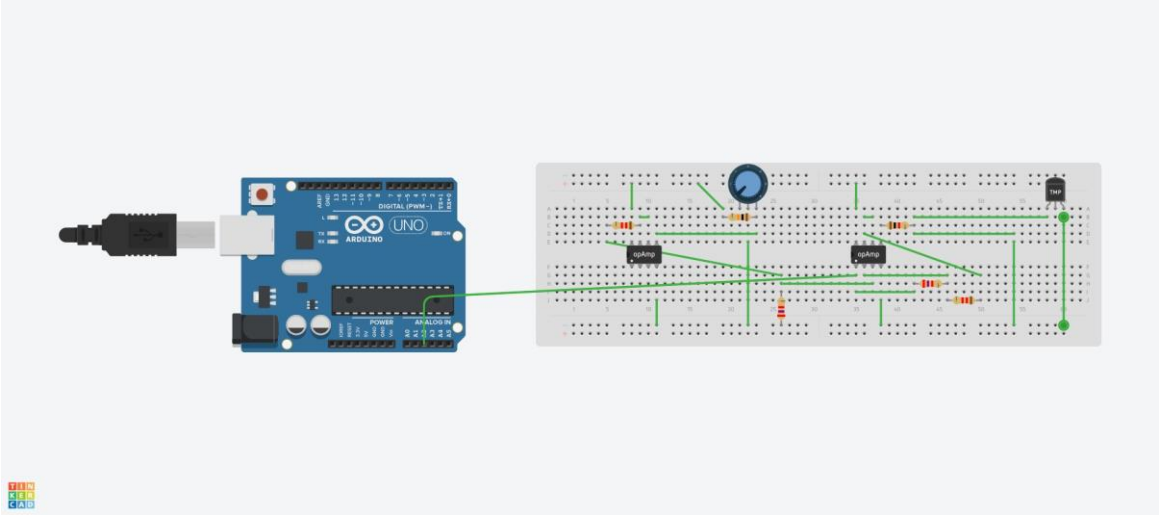

Figura 6-11. Montaje del circuito amplificador

En la imagen se puede ver como se realiza el conexionado, tal y como se ha visto en apartados anteriores. El amplificador de la izquierda se usa como seguidor de tensión y el de la derecha como otro seguidor, además de como amplificador operacional, ya que, como se ha comentado, internamente, el componente consta de dos amplificadores.

Una vez montado el circuito, se pasa a realizar las pruebas pertinentes.

6.3 Primera prueba: Resistencia LDR y LED

En primer lugar, para probar el correcto funcionamiento del PID, se sustituyeron la resistencia de cobre y el sensor de temperatura por un diodo LED y por una resistencia LDR, respectivamente.

La utilidad de esta prueba reside en que, al ser un dispositivo de control de temperatura, un exceso de la misma podría dañar los elementos del sistema. Sin embargo, un exceso de luz no tiene peligrosidad, por lo que es una manera de probar el funcionamiento del circuito y el correcto conexionado de los elementos sin miedo a que resulten dañados.

De esta manera, el funcionamiento del circuito sería el siguiente:

- Se mide la luminosidad mediante la resistencia LDR, que actúa como sensor.
- Se compara este valor con el del circuito de referencia.
- Una vez comparado, el PID actuará en consecuencia, aumentando o disminuyendo la intensidad del brillo del LED para que llegue más o menos luz a la resistencia.

Para poder realizar esta prueba y que los resultados sean satisfactorios, se distribuyeron los componentes de tal manera que el LED quedase pegado a la resistencia. Además, la prueba se realizó en un entorno oscuro para que casi la totalidad de la luz que captase la resistencia procesiese del brillo del diodo LED.

Los resultados de la prueba fueron bastante buenos, a medida que se cambiaba el valor de la resistencia de referencia variando el valor del potenciómetro, el brillo del LED variaba en consecuencia, por lo que se podía concluir que el programa del PID era funcional y ya se podía proceder a montar la resistencia de cobre con el sensor de temperatura para hacer pruebas en el laboratorio.

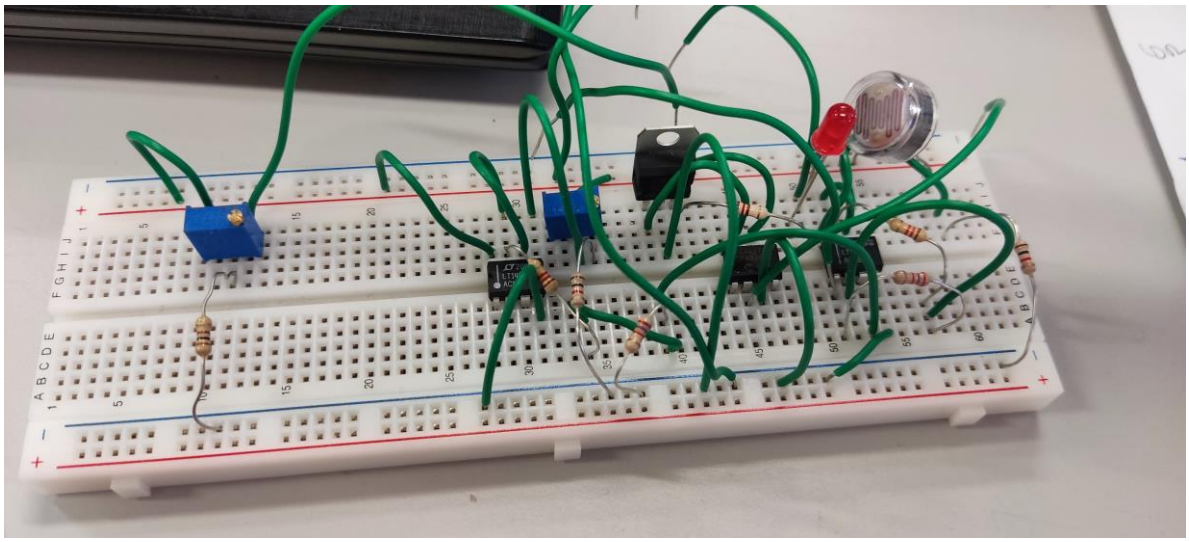

Figura 6-12. Montaje del circuito con LDR y LED en placa de pruebas

6.4 Segunda prueba: Resistencia de cobre y sensor de temperatura

El siguiente paso ya es introducir la resistencia de cobre y el sensor de temperatura en sus posiciones originales y hacer pruebas en el laboratorio.

Además de introducir ese cambio, se va a conectar la resistencia y la placa de Arduino directamente a través de la alimentación de la placa, a 12 voltios.

Con esto se consigue que el dispositivo sea autónomo y no necesite estar conectado al ordenador para funcionar, simplemente con cargar el programa una vez este se ejecutará cada vez que la placa reciba alimentación.

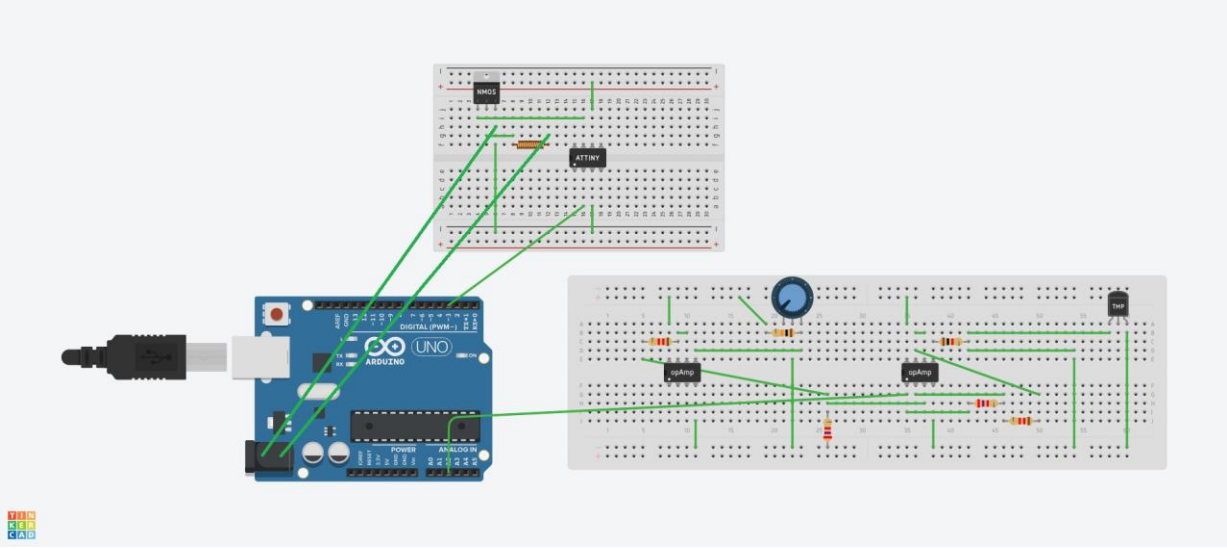

Figura 6-13. Esquema del montaje con alimentación

Al diseño final le falta todavía un componente, la pantalla.

Ya se ha visto anteriormente que esta consta de 8 pines, que se conectarían con la placa de Arduino de la siguiente manera:

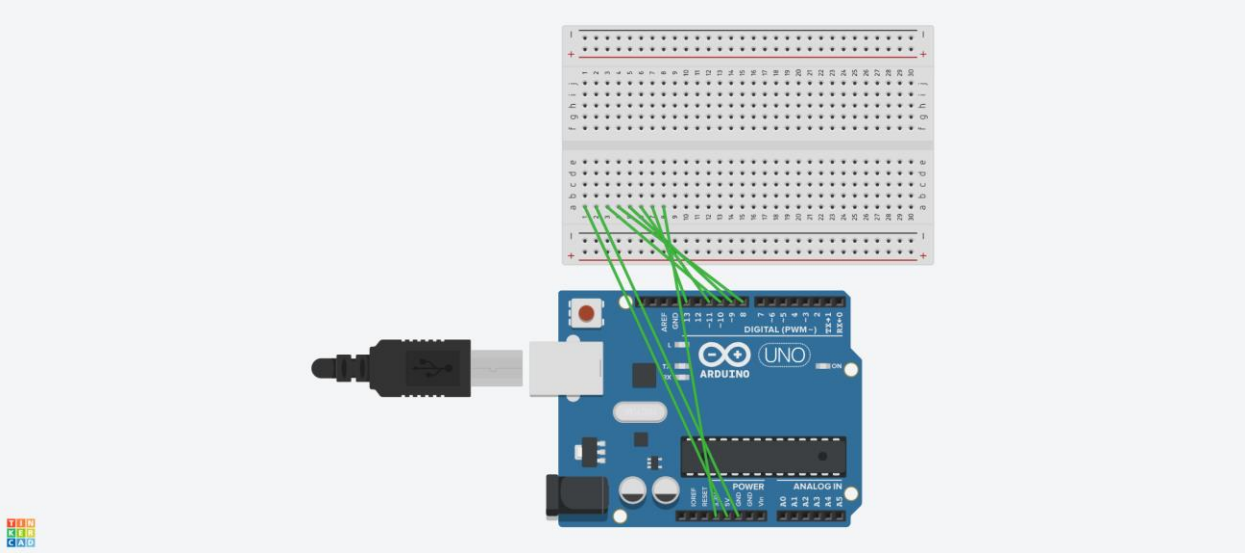

Figura 6-14. Esquema del montaje con alimentación

Este sería el montaje final del circuito con la resistencia y el sensor, a falta de representar el circuito de referencia que, como se ha visto antes, es muy simple.

Finalmente, en la siguiente imagen se puede observar como queda en la placa de pruebas, ya enchufado a la alimentación.

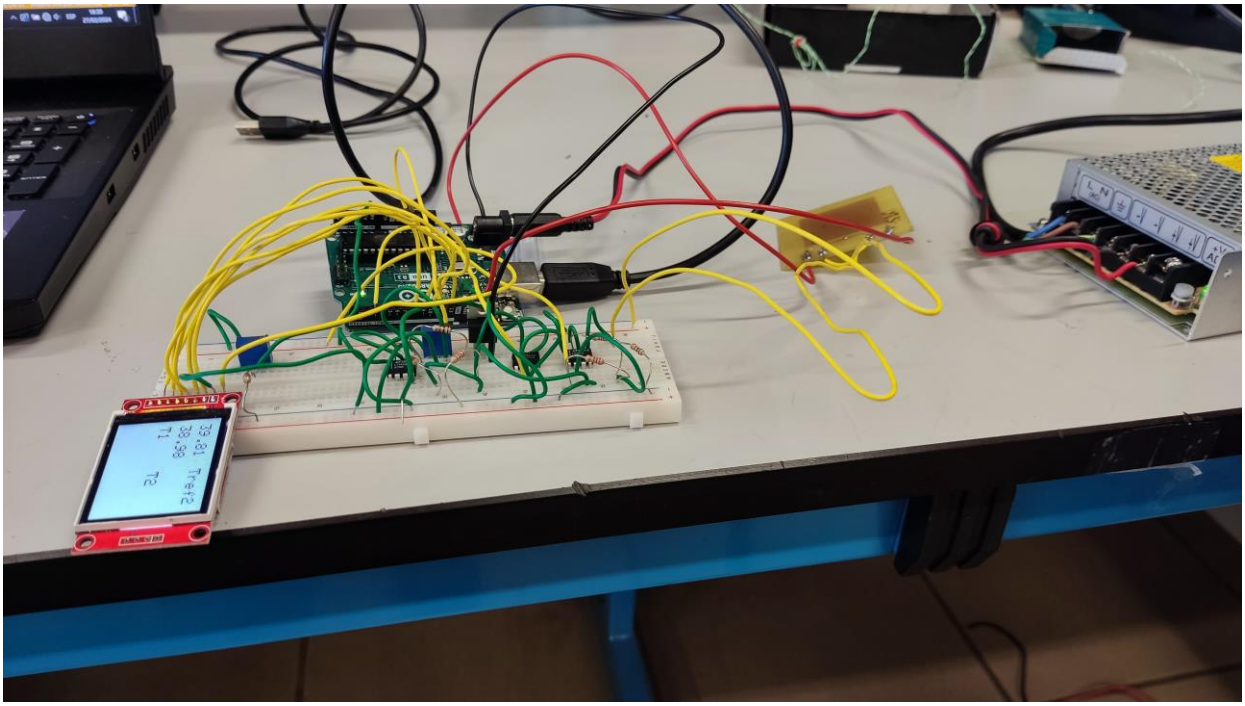

Figura 6-15. Prueba de funcionamiento

Una vez montado y conectado solo quedaba hacer las pruebas. Si estas eran buenas, ya se podría proceder con el diseño y montaje en el PCB.

Los resultados arrojados por esta prueba fueron bastante satisfactorios. Resultó en un control rápido y bastante preciso de la temperatura, teniendo sobreoscilaciones de máximo 5ºC.

Hay que tener en cuenta que estos resultados se han dado sin modificar las constantes Kp, Ki y Kd del controlador PID, por lo que ese error se puede incluso reducir bastante. Eso se ajustará cuando el PCB esté completo y se puedan realizar pruebas más sólidas.

También se ha comprobado que es posible utilizar el programa con una alimentación de 12 voltios, no teniendo que cargar el programa con un ordenador cada vez que se quiera iniciar. Todos los componentes han aguantado el nivel de tensión y los resultados son los mismos que alimentando solo a la resistencia de cobre en vez de a todo el circuito.

7 DISEÑO Y FABRICACIÓN DEL PCB

En este capítulo se va a describir el diseño y la fabricación de la placa PCB con el circuito integrado. El diseño se realiza con el programa KiCad, de software libre, y se manda a fabricar. Una vez se tenga la placa, solo hay que soldar los componentes que ya se han probado en la placa de pruebas y hacer los experimentos necesarios para comprobar que el dispositivo final funciona.

Esta placa PCB va a ir conectada directamente a Arduino, teniendo en cuenta que en la parte superior de la misma hay que colocar la pantalla y los potenciómetros del circuito de referencia, para que, cuando el dispositivo se ponga en una caja, sea más fácil operarlo.

7.1 Placa de Circuito Impreso (PCB)

Una Placa de Circuito Impreso es un dispositivo donde se fijan los componentes electrónicos de un circuito y que los conecta eléctricamente mediante pistas de cobre.

Las PCB tienen una amplia variedad de usos en la electrónica moderna. Están presentes en prácticamente todos los dispositivos electrónicos que se puedan encontrar, ya que es la manera más eficiente, en términos de espacio y aprovechamiento de materiales, en la que se puede montar un circuito electrónico.

Están formadas por capas, que siguen la siguiente estructura:

- Aislante: es una capa muy gruesa, que no conduce la electricidad, y se usa para soportar los elementos del circuito y las demás capas. Generalmente el material de fabricación de esta capa es el FR-4, una resina epoxi reforzada con vidrio.
- Cobre: Esta capa está formada por pistas creadas por el diseñador de la PCB. Estas pistas conectan eléctricamente todos los componentes del circuito. La capa de cobre solo consta de estas pistas, el resto de la capa es eliminado para conducir la corriente por donde se desea, es decir, siguiendo el camino formado por las pistas.
- Máscara: Esta capa se adhiere sobre el cobre dejando al descubierto las zonas donde se aplicará estaño para soldar los componentes. Esta capa está preparada para soportar el calor al sordar con estaño y que el cobre no se deteriore.
- Serigrafía: Es la última capa, donde se dibujan textos o símbolos identificativos para los componentes del circuito. Es opcional, puede haber placas sin serigrafía, pero es más fácil identificar los componentes de manera visual, sobre todo en placas con muchos elementos.

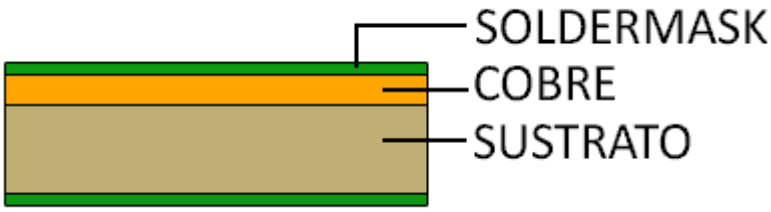

Figura 7-1. Representación gráfica de las capas de una PCB

También existen placas con dos capas de cobre, llamadas también de doble cara, que tienen una estructura simétrica, es decir que al igual que en la cara superior cuentan con una capa de cobre y una máscara en la capa inferior. La distribución de las capas sería, máscara – cobre – sustrato – cobre – máscara. También existen placas con capas de cobre intermedias para circuitos más avanzados, pero en este caso se va a usar una de doble cara.

La conexión entre las dos capas de cobre, la superior y la inferior, se realiza mediante unas perforaciones denominadas "vías", que atraviesan las capas y son recubiertas por la pared interior de la perforación de metal, para hacerlas conductoras de la electricidad.

En definitiva, para este proyecto se va a hacer uso de una PCB de doble cara. Esta va a ser diseñada con el programa KiCad, el cual se describe a continuación.

7.2 Descripción del software KiCad

KiCad es un software de diseño electrónico de código abierto que se utiliza para el diseño esquemático y de PCB (Printed Circuit Board o Placa de Circuito Impreso).

Figura 7-2. Logo de KiCad

KiCad fue fundado en 1992. Inicialmente, no estaba pensado para uso profesional, pero con el tiempo, y visto el potencial del mismo, adquirió usos educativos y profesionales. En su mayoría, el éxito que tuvo fue gracias a su licencia de código abierto, lo que lo hizo accesible para toda la comunidad de diseñadores electrónicos.

En la actualidad, KiCad es uno de los programas de diseño electrónico más populares y ampliamente utilizados. Ofrece una amplia gama de herramientas para el diseño esquemático, la creación y fabricación de PCB y la simulación de circuitos. Además, tiene una comunidad de desarrolladores bastente activa, lo que aporta mejoras constantes y corrección rápìda de errores.

En resumen, se ha escogido este software debido a su facilidad, tanto para su manejo como para encontrar información útil aportada por su gran comunidad, además que tiene la opción de exportar la huella del PCB diseñado directamente para su fabricación.

7.3 Diseño del esquemático

En el apartado 4, *"Diseño del circuito",* se diseñó el circuito de acuerdo a las especificaciones acordadas, así que, una vez probado que el circuito funciona, solo hay que replicarlo dentro del entorno de KiCad.

A continuación, se enumeran los componentes necesarios para la fabricación del PCB:

Tabla 7–1 Componentes del circuito

Componente	Cantidad
Resistencias	18
Potenciómetros	4
Transistores Mosfet	2
Amplificadores Operacionales	3
Drivers	1
Conectores Hembra 01x02	5
Pines Hembra 01x02 para LEDs	4
Pines Hembra 01x08 para Pantalla	1
Pines Macho pata insertar Arduino	1 (conjunto)

En el apartado 4, *"Diseño del circuito",* se diseñó el circuito de acuerdo a las especificaciones acordadas, así que, una vez probado que el circuito funciona, solo hay que replicarlo dentro del entorno de KiCad.

Como esta vez el diseño va a ser el definitivo, se va a realizar con los dos circuitos al completo, con la intención de no cambiar nada de cara a la fabricación.

Realizando la selección de componentes adecuada dentro del software, finalmente este es el resultado del diseño del esquemático:

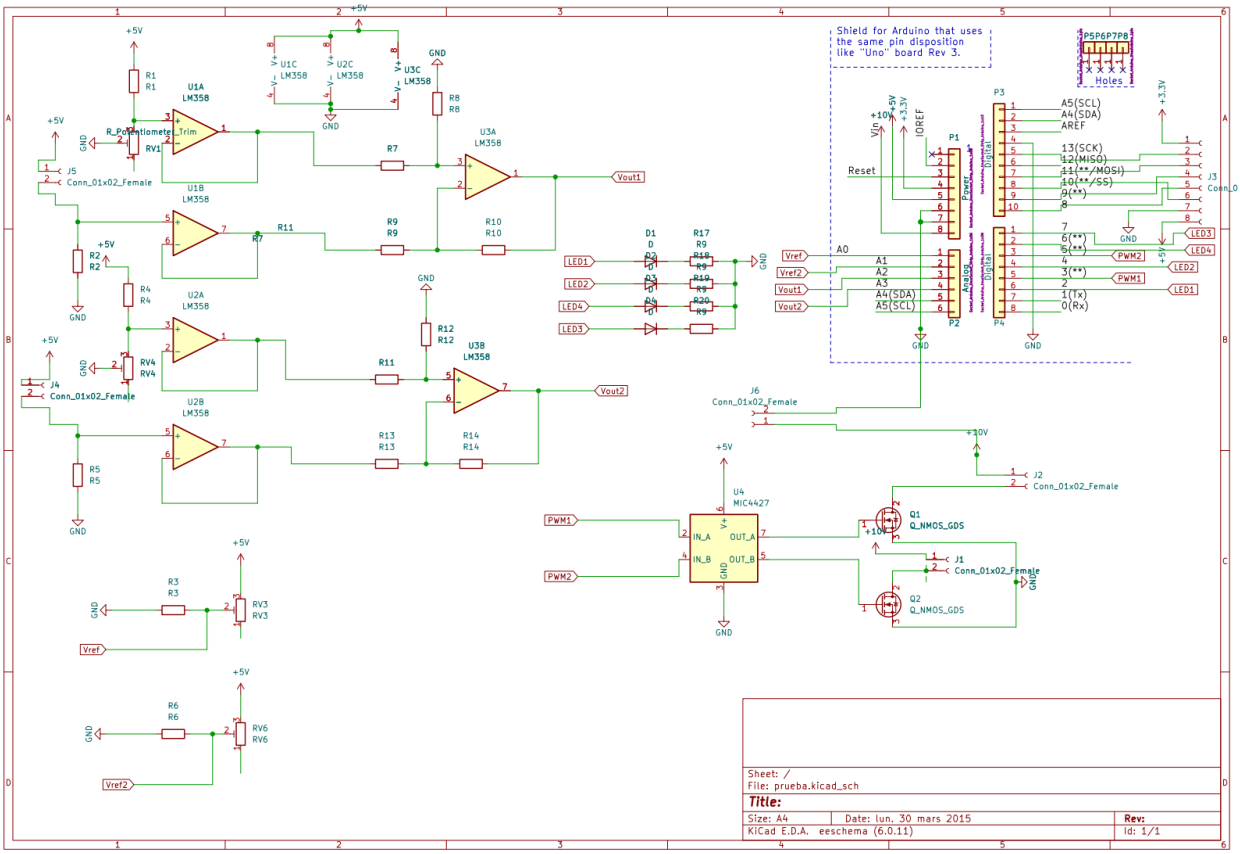

Figura 7-3. Esquemático de la PCB

En el esquema se pueden diferenciar: los dos circuitos de referencia abajo a la izquierda, los dos circuitos para medir la temperatura de los sensores arriba a la izquierda, el Arduino arriba a la derecha y el circuito actuador abajo a la derecha.

Lo siguiente sería pasar los componentes del esquemático al PCB como tal y colocarlos, pero existen unos pasos previos, como la asignación de huellas que se verán a continuación.

7.3.1 Indicadores de referencia

Esta es una herramienta principalmente de asignación de nombres automática. Es muy útil en circuitos con muchos componentes.

Estos nombres asignados se trasladarán junto con el componente hacia la placa y será más fácil orientarse para colocar correctamente los componentes, dando poco margen a errores.

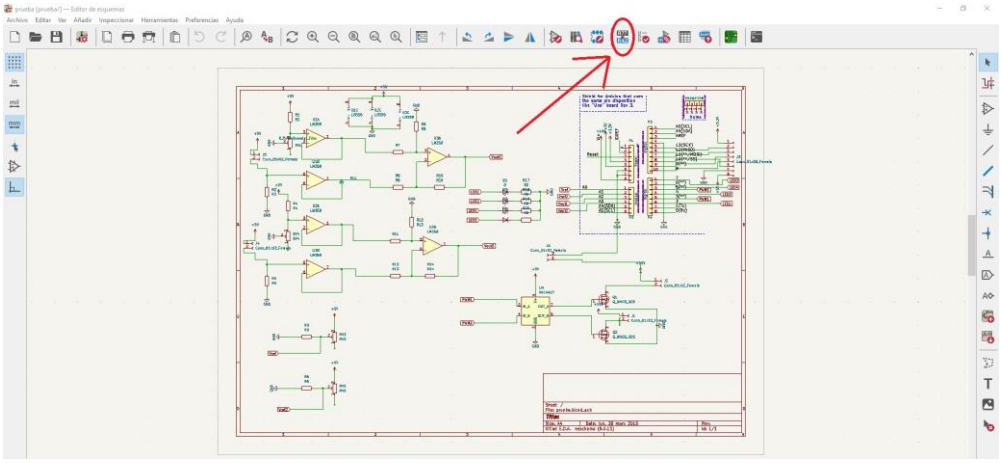

Figura 7-4. Icono de asignación automática de referencias

7.3.2 Asignación de huellas

Este paso es muy importante a la hora de pasar los componentes a la placa para realizar su diseño.

Consiste básicamente en, una vez se tenga el esquemático al completo, asignar las huellas correspondientes a cada uno de los componentes para su conexión en la placa. La huella es el espacio que va a ocupar cada componente en la placa PCB, donde van a estar los pads situados. Para asignar a cada componente una huella se mide el componente real o se miran sus medidas en el datasheet.

Para empezar con la asignación de huellas, hay que buscar el icono de “Ejecutar la herramienta de asignación de huellas” dentro del entorno de KiCad.

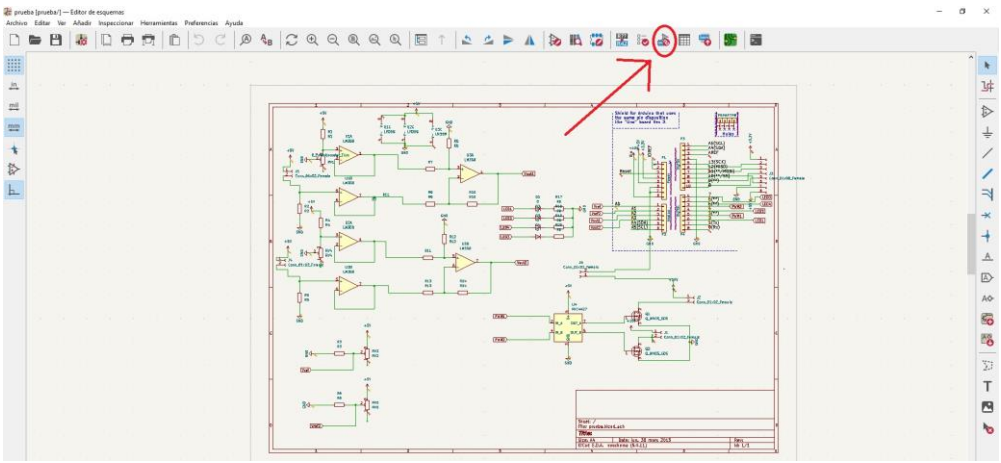

Figura 7-5. Icono de asignación de huellas

Una vez pulsado, se abre una pestaña que muestra una lista con todos los componentes y una librería con todas las huellas existentes.

El siguiente paso es asociar cada componente con la huella que se considere oportuna y que se considere que encajará con cada uno de los elementos.

A continuación, se muestra la lista de las huellas seleccionadas:

Símbolo: Asignación de huellas

1	D1	D	LED_THT:LED_D5.0mm
2	D2	D	LED_THT:LED_D5.0mm
3	D3	D	LED_THT:LED_D5.0mm
4	D4	D	LED_THT:LED_D5.0mm
5	J1	Conn_01x02_Female	TerminalBlock:TerminalBlock_bornier-2_P5.08mm
6	J2	Conn_01x02_Female	TerminalBlock:TerminalBlock_bornier-2_P5.08mm
7	J3	Conn_01x08_Female	Connector_PinHeader_2.54mm:PinHeader_1x08_P2.54mm_Vertical
8	J4	Conn_01x02_Female	TerminalBlock:TerminalBlock_bornier-2_P5.08mm
9	J5	Conn_01x02_Female	TerminalBlock:TerminalBlock_bornier-2_P5.08mm
10	J6	Conn_01x02_Female	TerminalBlock:TerminalBlock_bornier-2_P5.08mm
11	P1	Power	Socket_Arduino_Uno:Socket_Strip_Arduino_1x08
12	P2	Analog	Socket_Arduino_Uno:Socket_Strip_Arduino_1x06
13	P3	Digital	Socket_Arduino_Uno:Socket_Strip_Arduino_1x10
14	P4	Digital	Socket_Arduino_Uno:Socket_Strip_Arduino_1x08
15	P5	CONN_01X01	Socket_Arduino_Uno:Arduino_1pin
16	P6	CONN_01X01	Socket_Arduino_Uno:Arduino_1pin
17	P7	CONN_01X01	Socket_Arduino_Uno:Arduino_1pin
18	P8	CONN_01X01	Socket_Arduino_Uno:Arduino_1pin
19	Q1	Q_NMOS_GDS	Package_TO_SOT_THT:TO-220-3_Vertical
20	Q2	Q_NMOS_GDS	Package_TO_SOT_THT:TO-220-3_Vertical
21	R1	R1	Resistor_THT:R_Box_L8.4mm_W2.5mm_P5.08mm
22	R2	R2	Resistor_THT:R_Box_L8.4mm_W2.5mm_P5.08mm
23	R3	R3	Resistor_THT:R_Box_L8.4mm_W2.5mm_P5.08mm
24	R4	R4	Resistor_THT:R_Box_L8.4mm_W2.5mm_P5.08mm
25	R5	R5	Resistor_THT:R_Box_L8.4mm_W2.5mm_P5.08mm
26	R6	R6	Resistor_THT:R_Box_L8.4mm_W2.5mm_P5.08mm
27	R7	R7	Resistor_THT:R_Box_L8.4mm_W2.5mm_P5.08mm
28	R8	R8	Resistor_THT:R_Box_L8.4mm_W2.5mm_P5.08mm
29	R9	R9	Resistor_THT:R_Box_L8.4mm_W2.5mm_P5.08mm
30	R10	R10	Resistor_THT:R_Box_L8.4mm_W2.5mm_P5.08mm
31	R11	R11	Resistor_THT:R_Box_L8.4mm_W2.5mm_P5.08mm
32	R12	R12	Resistor_THT:R_Box_L8.4mm_W2.5mm_P5.08mm
33	R13	R13	Resistor_THT:R_Box_L8.4mm_W2.5mm_P5.08mm
34	R14	R14	Resistor_THT:R_Box_L8.4mm_W2.5mm_P5.08mm
35	R17	R9	Resistor_THT:R_Box_L8.4mm_W2.5mm_P5.08mm
36	R18	R9	Resistor_THT:R_Box_L8.4mm_W2.5mm_P5.08mm
37	R19	R9	Resistor_THT:R_Box_L8.4mm_W2.5mm_P5.08mm
38	R20	R9	Resistor_THT:R_Box_L8.4mm_W2.5mm_P5.08mm
39	RV1	R_Potentiometer_Trim	Potentiometer_THT:Potentiometer_Bourns_3296Y_Vertical
40	RV3	RV3	Potentiometer_THT:Potentiometer_Bourns_3296Y_Vertical
41	RV4	RV4	Potentiometer_THT:Potentiometer_Bourns_3296Y_Vertical
42	RV6	RV6	Potentiometer_THT:Potentiometer_Bourns_3296Y_Vertical
43	U1	LM358	Package_DIP:DIP-8_W7.62mm
44	U2	LM358	Package_DIP:DIP-8_W7.62mm
45	U3	LM358	Package_DIP:DIP-8_W7.62mm
46	U4	MIC4427	Package_DIP:DIP-8_W7.62mm

Figura 7-6. Huellas asignadas

7.4 Diseño de la placa

Una vez esté el circuito diseñado y todos los nombres y huellas asignados es momento de pasar todos los componentes a la placa, para lo que se usará la siguiente orden:

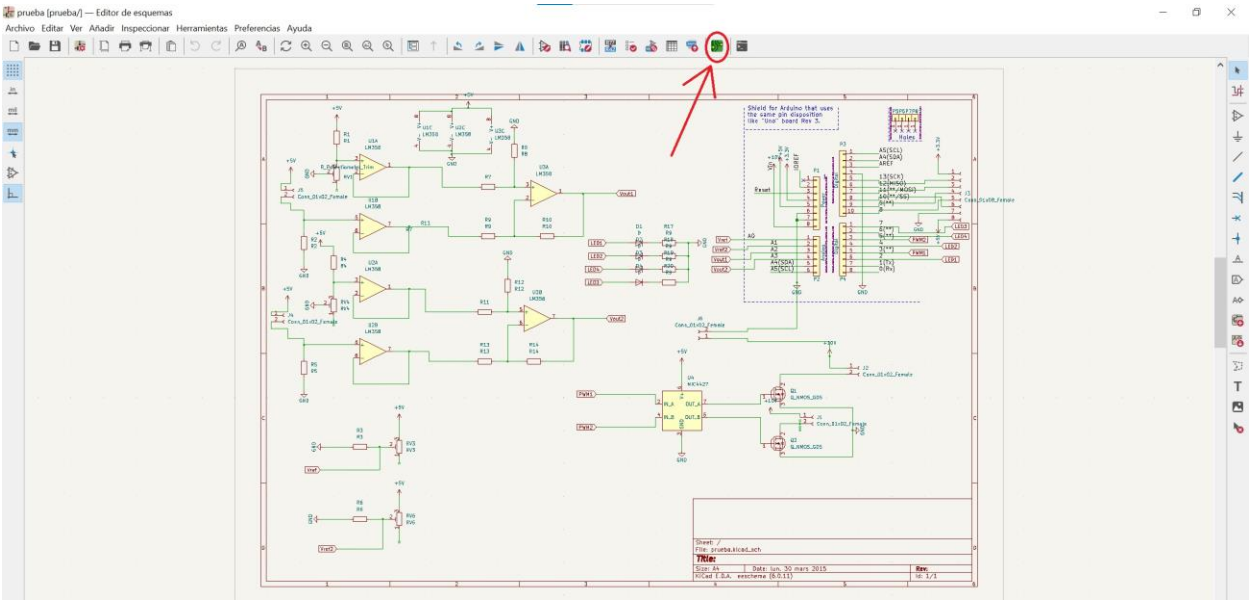

Figura 7-7. Abrir la placa

Pulsando esta función se abre el editor de placas, el cual se mostrará vacío al principio, pero habrá que cargar todos los componentes, para lo que se usa la función “Leer lista de redes y actualizar la conectividad de la placa”.

Después se selecciona la opción “Actualizar placa”.

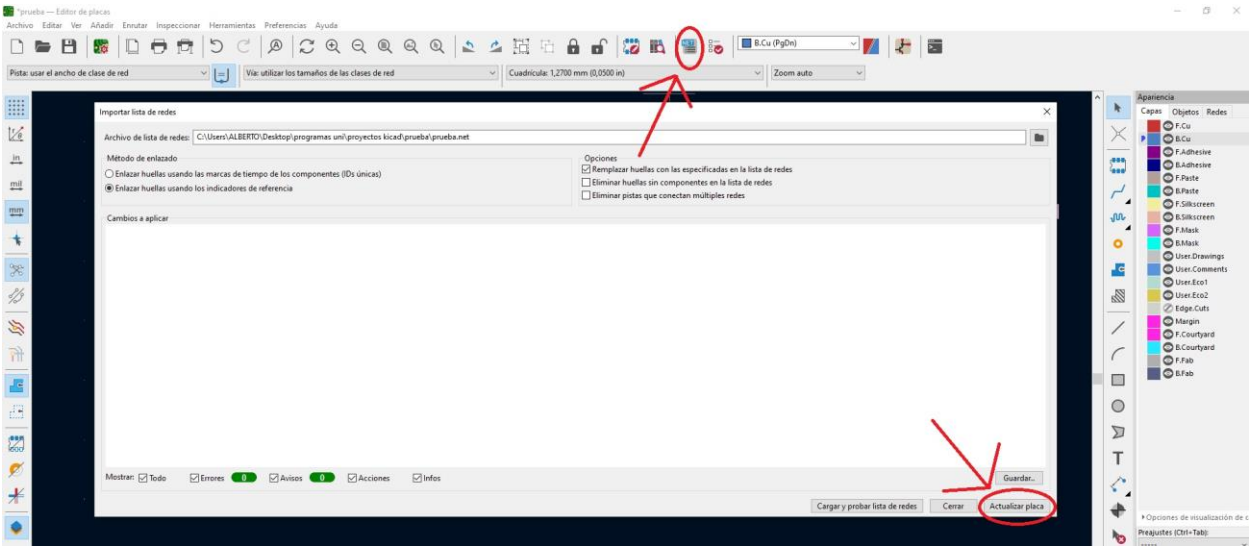

Figura 7-8. Pasos para pasar los elementos a la placa

Lo siguiente que aparece en el entorno son los componentes desordenados, así que hay que ordenarlos. Para ello se realizó el esquemático y se asignaron los nombres a cada componente.

También aparecen representadas las conexiones que hay que realizar, para que sea más fácil colocar todos los elementos de la manera más ordenada posible e intentar tener las menos pistas de cobre posibles.

Así se ve el diseño de la placa nada más cargar los componentes:

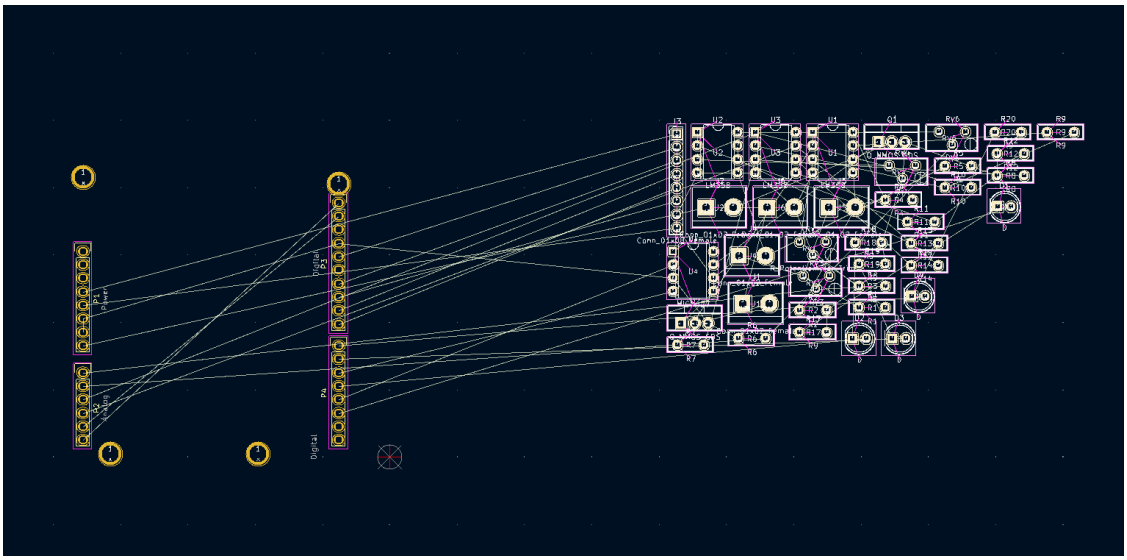

Figura 7-9. Diseño antes de la ordenación de los elementos

El siguiente paso es colocar todo de manera que quede lo más ordenadamente posible y, a ser posible, no realizar muchas perforaciones para pasar de una cara a otra, es decir, que no se realicen muchos cruzamientos de pistas que se tengan que evitar por la otra cara.

Lo más óptimo es ir realizando el circuito siguiendo el esquemático, y poner los componentes que estén conectados entre sí lo más cercanos unos de otros. También, otra restricción a tener en cuenta, es que es recomendable poner los conectores y los potenciómetros de referencia en los bordes de la placa para que sean más accesibles.

Finalmente, después de muchas pruebas, esta fue la disposición final de todos los elementos:

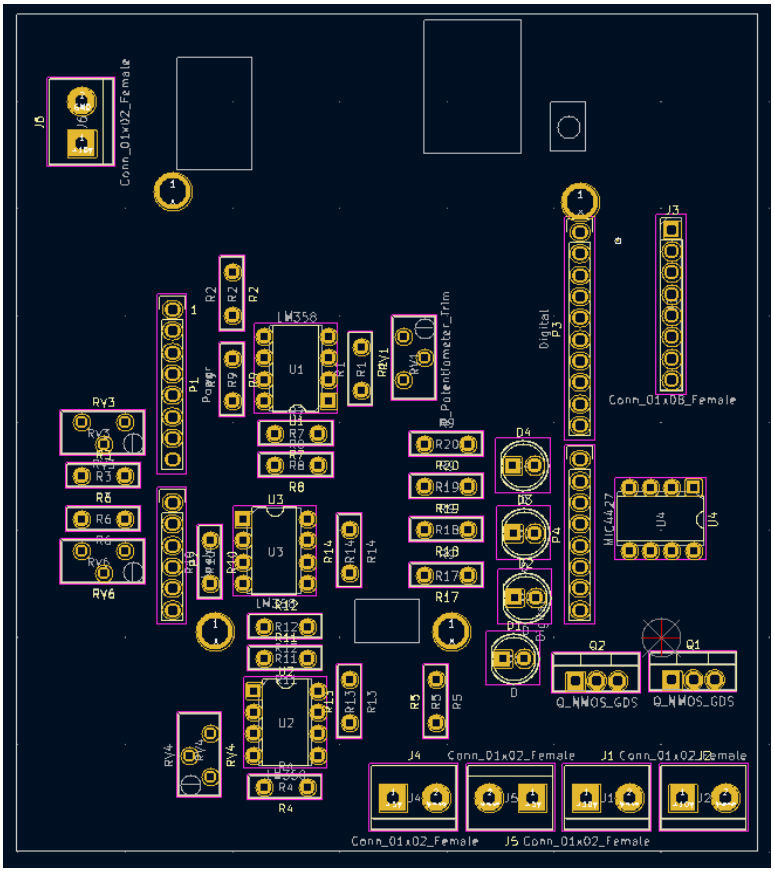

Figura 7-10. Disposición de los elementos en la PCB

Una vez colocados los componentes en sus posiciones, hay que hacer las conexiones con pistas de cobre. Estas se podrán realizar en las capas superior e inferior y no pueden cruzarse unas con otras, por que habría cortocircuitos.

Se van a utilizar dos tamaños diferentes de pistas de cobre:

- **Ancho de 0,25 mm:** Es el ancho de pista que se va a usar en la mayoría del circuito para conectar todos los componentes.
- **Ancho de 3 mm:** Estas pistas se van a usar para conectar la alimentación. Ya que la alimentación va a ser de 12 voltios, va a circular más corriente por las pistas de cobre conectadas a los 12 voltios y hay riesgo de que se fundan si son demasiado pequeñas, por lo que poniendo un ancho de 3 milímetros se previene de que esto pase.

El caso ideal sería poner todas las pistas en la cara inferior (azul), ya que los componentes irán soldados en la capa superior (rojo) y no hacer perforaciones. En este caso, se ha tenido que realizar una perforación y se han usado las dos caras, ya que hay muchos componentes.

El resultado es el siguiente:

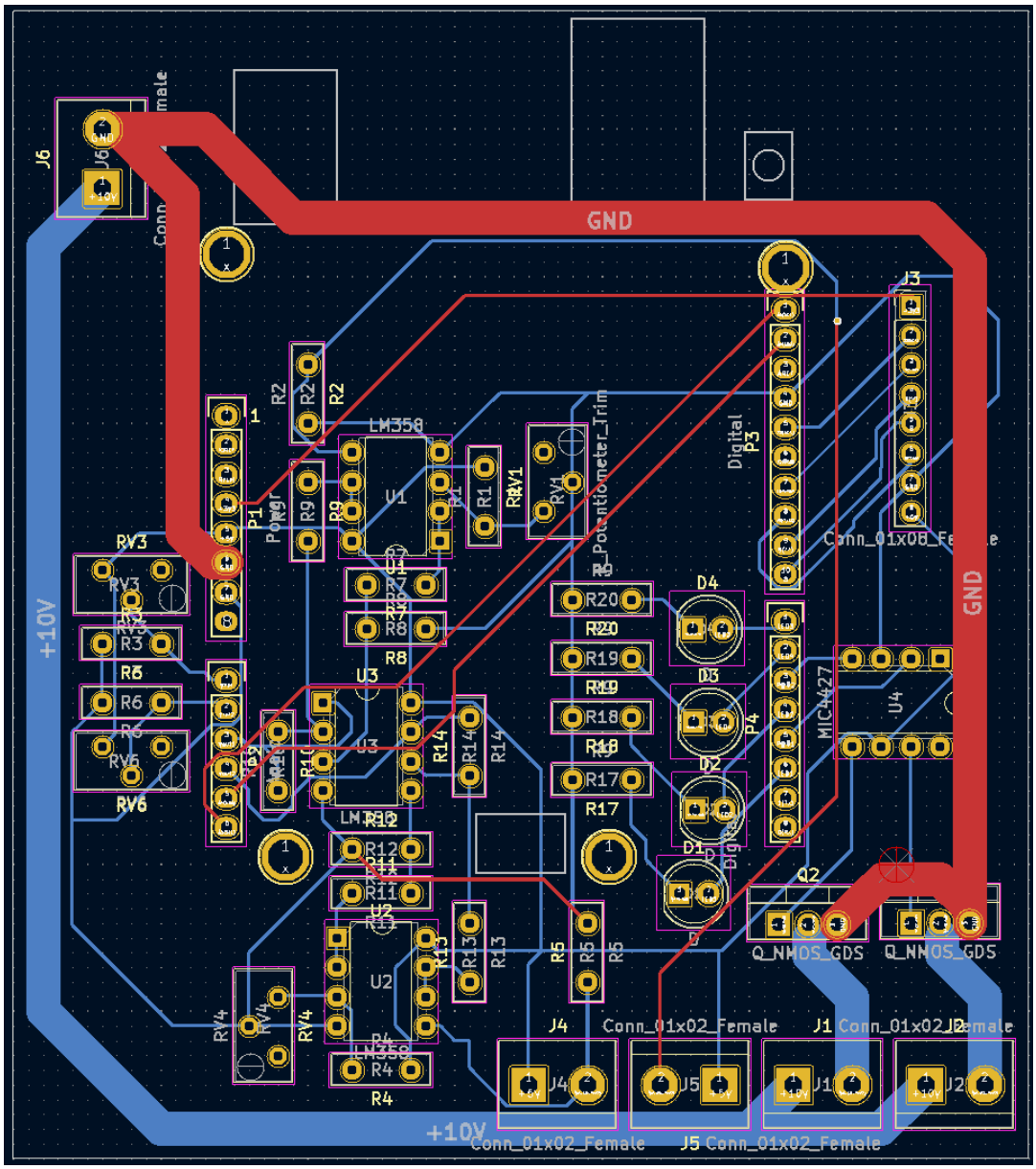

Figura 7-11. Esquema de conexionado de la PCB

El último paso antes de fabricar la placa es colocar los planos de alimentación y tierra.

Como se puede observar en la figura 7-11, se ha intentado llevar todas las tierras por una misma capa y lo mismo para las alimentaciones, lo que ha resultado en que se hayan puesto todas las alimentaciones en la capa inferior y casi todas las tierras en la capa inferior.

Las utilidades de los planos de masa en una PCB son: garantizar un funcionamiento eléctricamente estable y confiable del circuito, distribuir eficientemente la energía, aislar el ruido, mejorar la integridad de las señales y asegurar el aislamiento de las pistas de cobre.

Es decir, es ideal poner un plano de masa que ocupe el mayor espacio posible para asegurar la estabilidad del circuito. Debido a la distribución elegida para los componentes y la ordenación de las pistas, así quedarían los planos de alimentación (rojo) y tierra (azul).

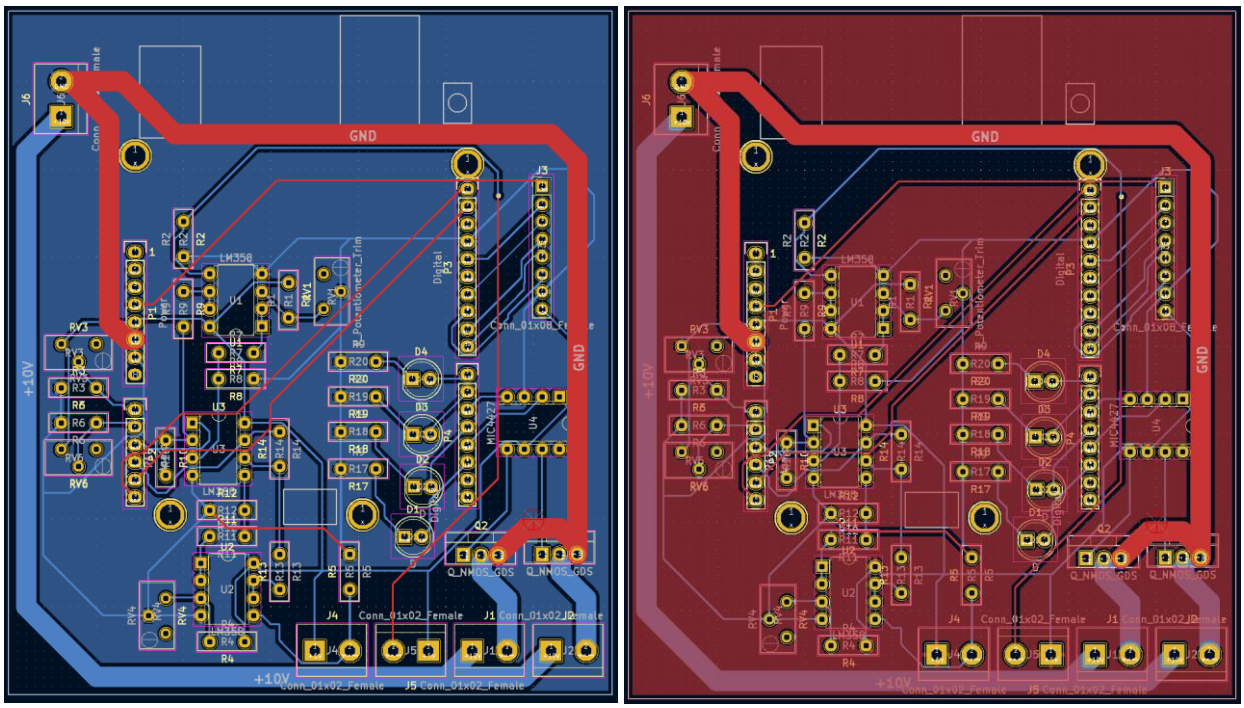

Figura 7-12. Planos de masa de la PCB

Como se puede apreciar, los planos ocupan la placa casi en su totalidad, lo que va a garantizar estabilidad al circuito.

En este punto, la placa esta ya diseñada en su totalidad, el siguiente paso es sacar los archivos y mandarla a fabricar.

7.5 Montaje físico

Una vez fabricada la placa, lo único que falta es soldar todos los componentes a la misma.

El proceso de soldadura sirve para adherir los componentes a la placa con estaño para fijarlos a ella y que a su vez puedan permitir el paso de la corriente a través del metal.

El proceso de soldadura funciona de la siguiente manera:

- Se conecta el soldador a la corriente y se espera a que esté lo suficientemente caliente

Figura 7-13. Herramienta para soldar

- Cuando esté a la temperatura adecuada para fundir el estaño, se coloca la punta sobre el pad que se quiera soldar, con el componente ya introducido.
- Cuando se caliente el pad, se acerca el hilo de estaño a este y se fundirá solo, llenando el pad y proporcionando una conexión robusta y estable

A continuación, se va a mostrar el antes y el después del soldado de los componentes a la placa:

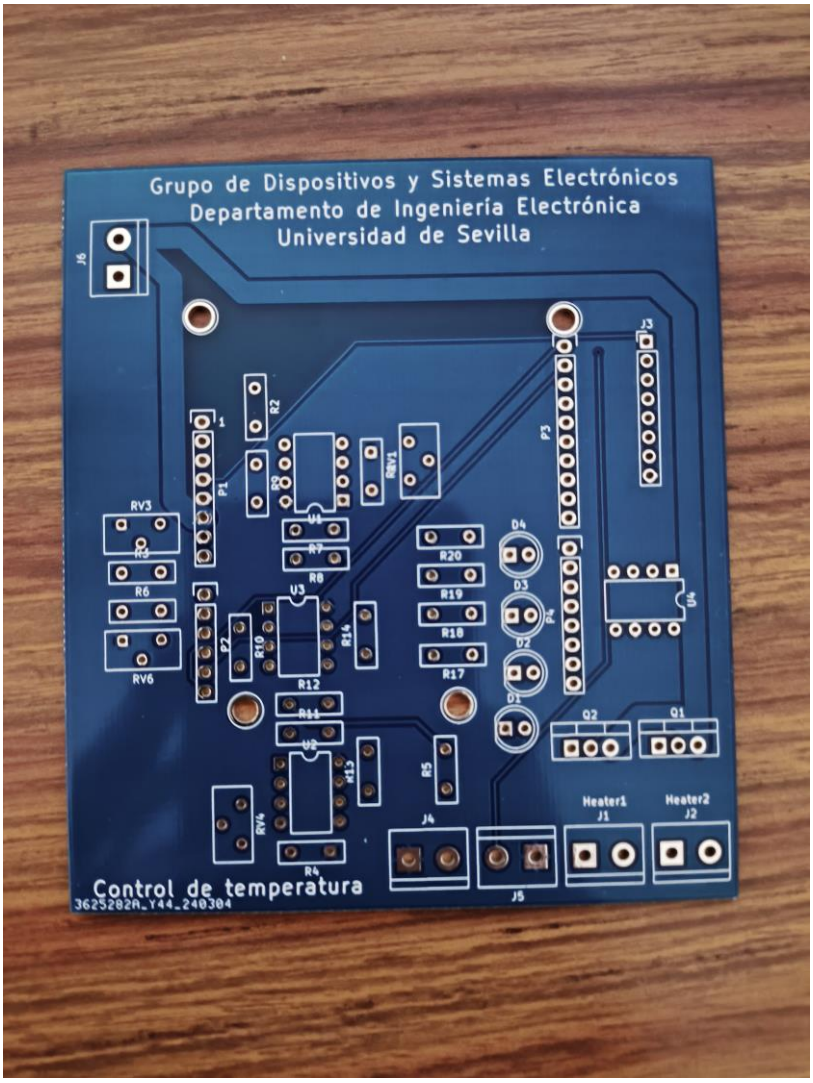

Figura 7-14. PCB fabricada sin componentes

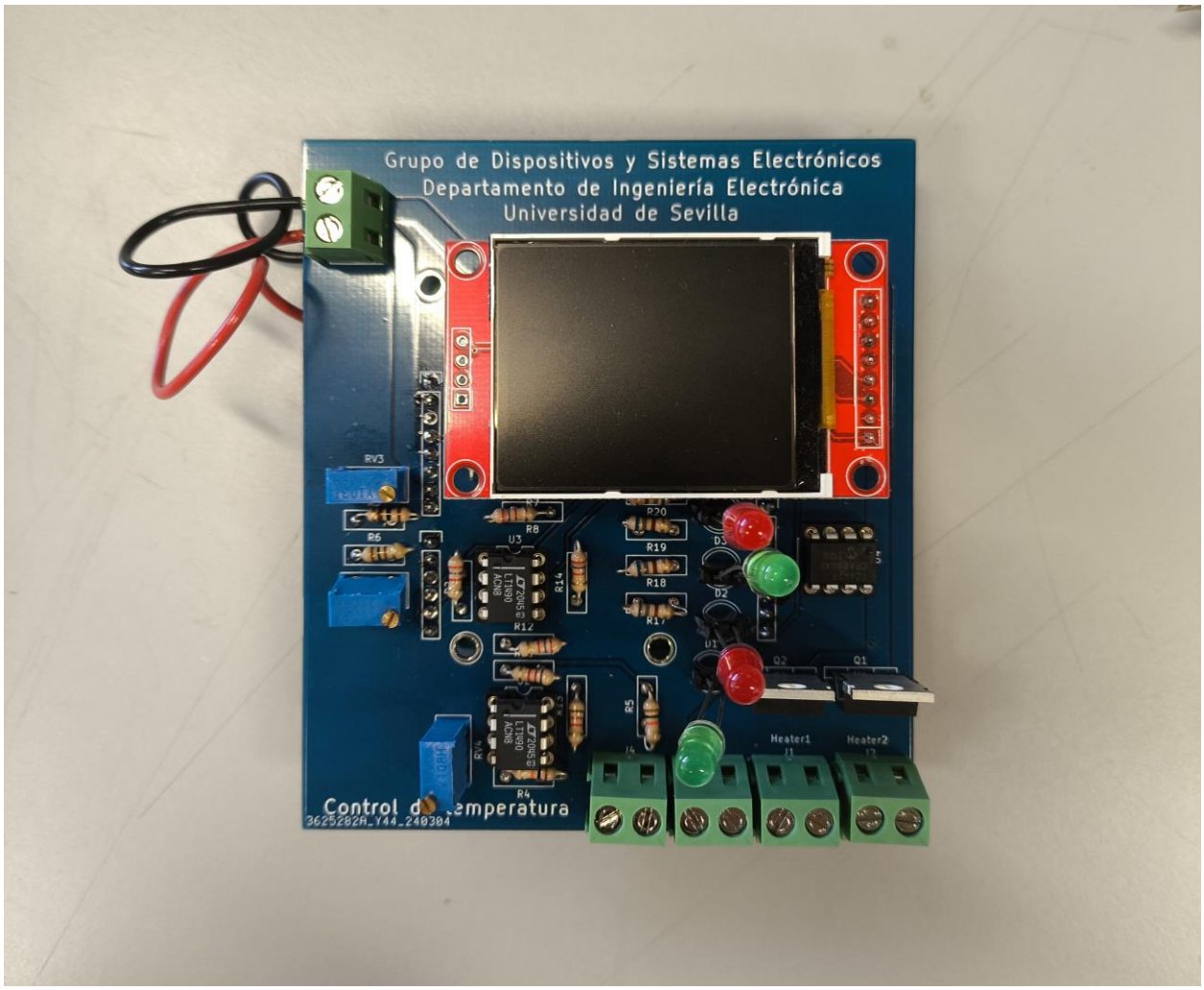

Figura 7-15. PCB final

Con esto estaría completo el dispositivo. El siguiente y último paso será hacer pruebas y recopilar los resultados del proyecto.

8 RESULTADOS EXPERIMENTALES

En este capítulo se van a describir las diferentes pruebas y resultados mostrados por el dispositivo final, tales como correcciones de errores de medida por distancia al calentador o resultados arrojados por pruebas de control.

Para la realización de las pruebas se va a utilizar una resistencia implementada en una PCB, comúnmente usada para experimentos de cultivo de retina.

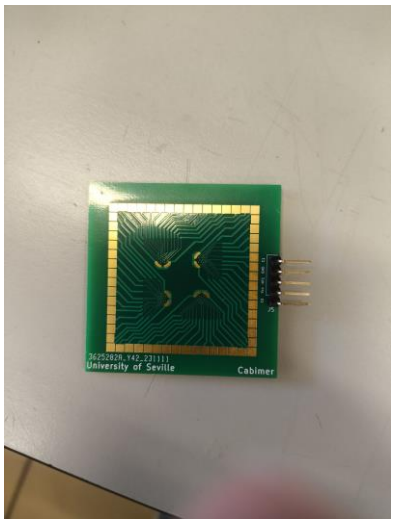

Figura 8-1. Resistencia de cobre utilizada para las pruebas

Este dispositivo lleva soldado el sensor de temperatura en la parte inferior, por lo que proporciona una muestra muy fiable de la temperatura a la que se encuentra y, por lo tanto, es ideal para analizar los resultados del proyecto.

8.1 Corrección de temperatura

Antes de comenzar con los experimentos, hay un factor importante que se debe tener en consideración. La muestra va a ir colocada encima de la placa de la resistencia, pero no va a estar en contacto con ella, por lo que hay que reducir el resultado aportado por el sensor de temperatura en algunos grados, ya que la muestra no estará a la misma temperatura que el sensor.

Para regular esto se ha usado un termómetro de laboratorio, el cual tiene un cable con un extremo que mide la

temperatura en el punto en el que se coloque. Lo que se ha hecho ha sido pegar el extremo del cable a la parte superior de la placa y medir la diferencia entre el valor devuelto por el termómetro y el de la pantalla.

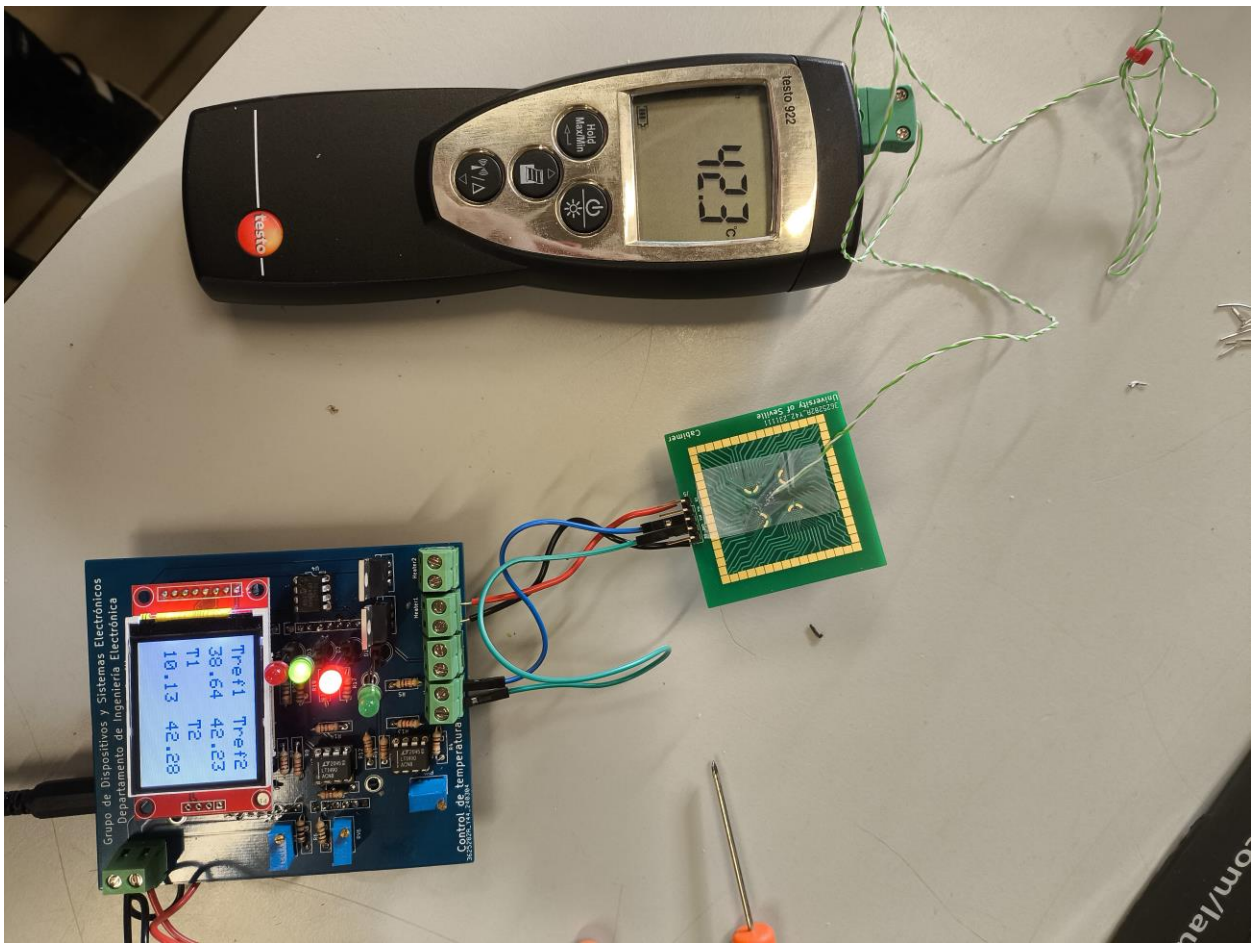

Figura 8-2. Prueba con el termómetro

Haciendo diversas pruebas y recopilando resultados, se vió que lo mostrado por la pantalla del PCB y el termómetro difería en algunos grados, por lo que se ajustó el programa de Arduino para que mostrase por la pantalla los grados de diferencia de lo que indicaba realmente el sensor de temperatura, ya que lo que interesa saber no es la temperatura del sensor, si no la de la muestra.

8.2 Ajuste de los parámetros del controlador PID

Como se ha visto antes, el controlador PID tiene 3 constantes regulables, la constante proporcional, la integral y la derivativa.

A modo de resumen, mientras más grande es la constante proporcional, menos error hay en régimen permanente, pero si es demasiado grande se producen sobreoscilaciones no deseadas. La constante integral elimina el error en régimen permanente que el proporcional no puede eliminar, por lo que se complementan. Por último, la constante derivativa, que "predice", por medio de derivar el error, los cambios que va a tener la señal y los anticipa, un valor elevado reduce la velocidad, pero genera más estabilidad, aunque si es demasiado grande también produce inestabilidad porque no le da tiempo a predecir los cambios, por eso esta constante es la más difícil de ajustar y la menos utilizada.

En un primer momento, se ha intentado estabilizar el control con solo la contante proporcional. Para un valor de Kp = 50, el error en régimen permanente entre la temperatura marcada como referencia y la que capta el sensor es de poco más de 1ºC, lo cual no es aceptable, ya que se desea que estos valores sí que coincidan casi exactamente.

Finalmente, tras varias pruebas, y añadiendo un valor de la contante integral al control, se ha llegado a que la siguiente solución cumple con las condiciones que se requieren, que no haya error en régimen permanente, y que la sobreoscilación no sea excesiva:

- **Kp = 200.**
- **Ki = 0.**
- **Kd = 0.**

Estos valores pueden ser ajustados dependiendo del tipo de prueba que se quiera realizar, si por ejemplo se desea que la temperatura vaya subiendo progresivamente de manera muy lenta, habrá que indicar un valor mucho más pequeño para Kp.

8.3 Simulación de pruebas

Para probar definitivamente que el dispositivo es funcional y que se pueden realizar pruebas biomédicas con este, se van a replicar el control de diferentes pruebas.

La primera de ellas será una prueba de control de retina. Esta consiste en que se toman muestras de la retina humana y se ponen en un calentador a 37 grados centígrados, que es la temperatura aproximada del cuerpo humano.

El obtetivo principal de este tipo de prueba es conocer y estudiar como las células de la retina se comportan y se desarrollan en un entorno similar al del cuerpo humano. Esta prueba puede ser útil para comprender mejor las enfermedades oculares y para desarrollar tratamientos potenciales que puedan reparar las células dañadas de la retina.

Para el correcto funcionamiento de esta prueba hay que asegurarse de que la muestra siempre estará a 37 grados centígrados y que la temperatura se mantiene estable sin variaciones. Esta se realizará en uno de los dos calentadores de los que consta el dispositivo.

Simultáneamente, en el otro calentador, se llevará a cabo una prueba LAMP (Loop-mediated isothermal AMPlification). Una prueba LAMP es un método sencillo, rápido y preciso de detección y ampliación de ADN. Se usa para detectar la presencia de material genético específico, como ADN de ciertos agentes patógenos.

Esta técnica se diferencia de la PCR en que, mientras la PCR realiza oscilaciones de temperatura, en la prueba LAMP, la temperatura siempre es constante. En este caso, la temperatura del experimento será de 65 grados centígrados.

El procedimiento es el mismo que para el cultivo de retina. Se va a calentar la resistencia a 65 grados para mantener encima la muestra y que esta temperatura no varíe durante todo el proceso.

Para poner en marcha el dispositivo es necesario suministrarle energía. Como ya se ha visto, esta se va a suministrar directamente por la alimentación de Arduino a 12 voltios. Para ello, se ha inyectado la energía desde la siguiente fuente:

Figura 8-3. Fuente de tensión para el dispositivo

Una vez conectado el dispositivo, solo queda modificar manualmente los potenciómetros para establecer las referencias en 37 y 65 grados centígrados para los dos calentadores.

En primer lugar, se va a ver en el Serial Plotter de Arduino como es el camino de la temperatura hasta alcanzar la referencia una vez conectados los calentadores.

Figura 8-4. Referencia de 37 grados

Como se puede observar en la imagen, para la temperatura de 37 grados no hay casi sobreoscilación y además se alcanza el valor rápidamente partiendo desde la temperatura ambiente.

A continuación, se va a ver como es el error en régimen permanente:

Figura 8-5. Error en régimen permanente a 37ºC

Las dos líneas, la de referencia y la real, están casi superpuestas, por lo que no hay error en régimen permanente.

Ahora se va a ver como se comporta la temperatura para el calentador de 65 ºC:

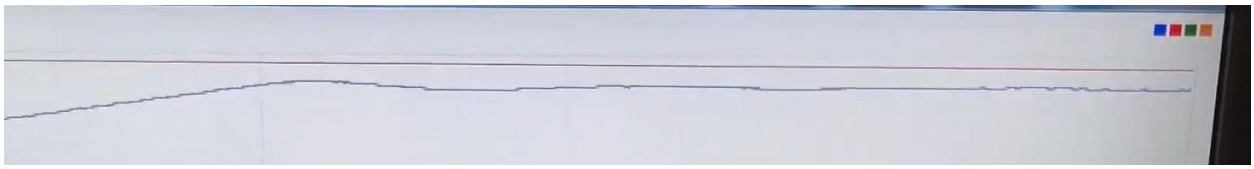

Figura 8-6. Error en régimen permanente a 65ºC

Aquí se puede observar que el error en régimen permanente es un poco mayor que el del otro calentador, casi de un grado, esto se debe a la diferencia tan grande que hay entre el ambiente y los 65 grados.

Se ha visto como es el error del controlador, pero ahora queda saber como se comporta la temperatura real, es decir, la temperatura a la que estaría realmente la muestra y no la del sensor. Para ello se ha usado un termómetro digital de nuevo, con doble salida para medir las dos temperaturas.

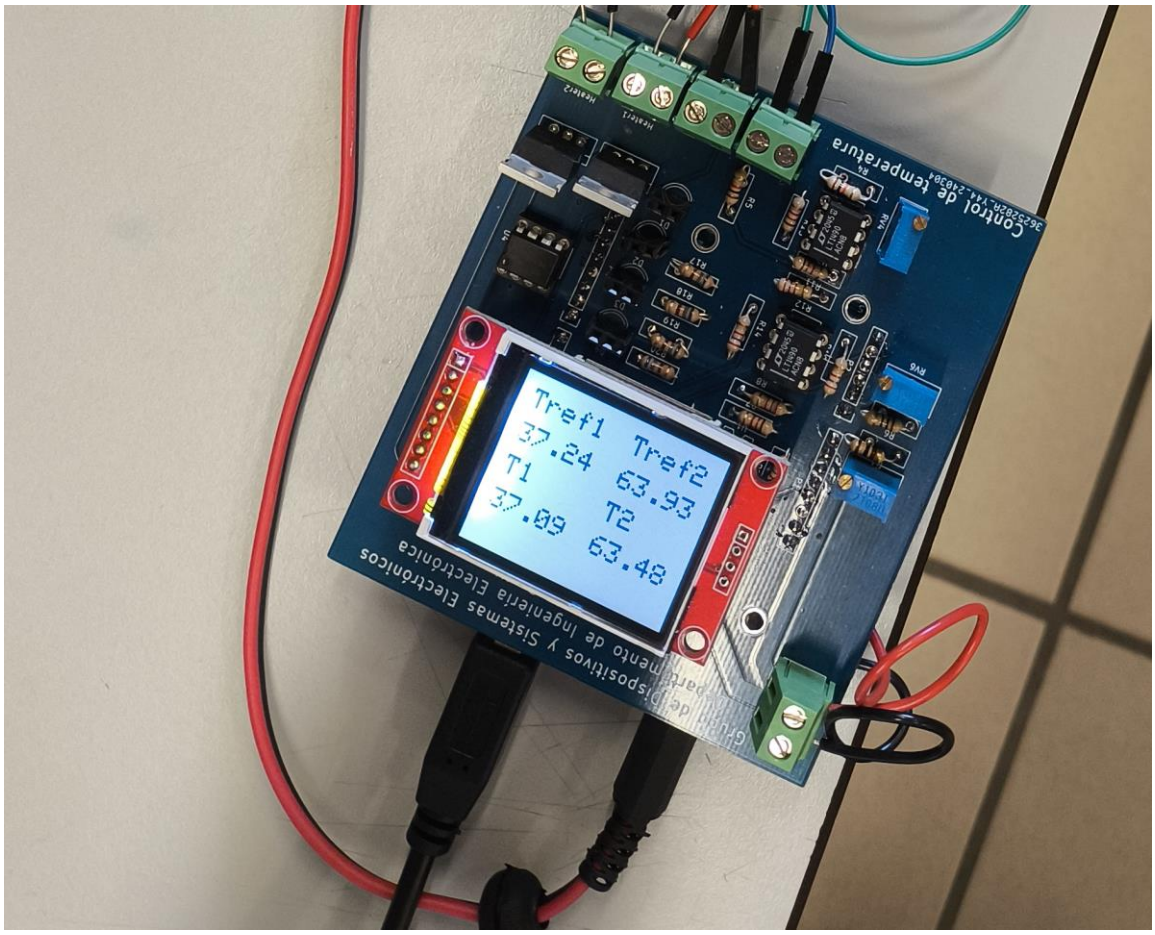

Figura 8-7. Valores de la temperatura de referencia en el experimento

Las temperaturas de referencia establecidas han sido de 37,24 ºC y de 63,93 ºC. Después de unos 3 minutos actuando se han medido las temperaturas de ambos calentadores en el lugar en el que iría ubicado la muestra y estos han sido los resultados.

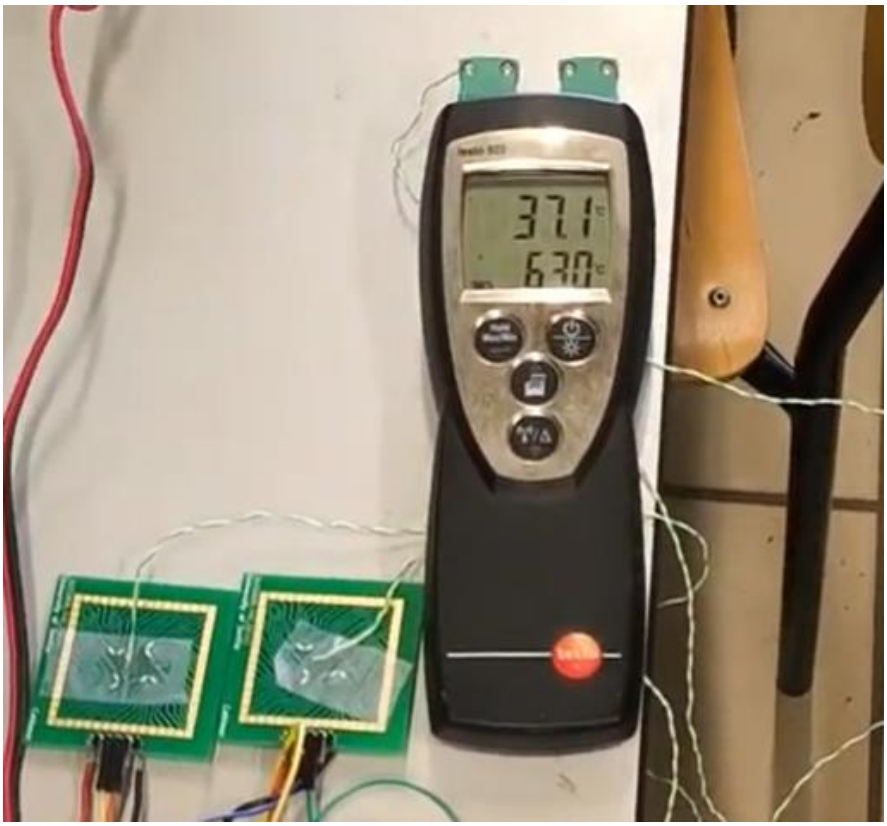

Figura 8-8. Resultados en el termómetro digital

Como podemos observar, para el calentador de 37ºC, la temperatura de la muestra sería de 37,1ºC, lo que, comparado con la referencia, da un error de 0,1ºC. Este es un resultado muy bueno que indica que la prueba de control de retina se realizaría en consiciones casi ideales. Además, en el rato que ha estado en funcionamiento, no se ha apreciado ninguna desviación de más de 0,2 grados, lo cuál indica que el control funciona perfectamente.

Para el segundo calentador se ha conseguido una temperatura de 63ºC, que comparados con los 63,9 de referencia da como resultado un error de alrededor de un grado, y teniendo en cuenta que solo por el PID se desvía 0,5 grados, se puede afirmar que el resultado es más que satisfactorio.

En resumen, se ha comprobado que el dispositivo tiene un funcionamiento lo suficientemente preciso como para realizar pruebas de cultivo de retina y pruebas LAMP fiables.

9 CONCLUSIONES

En este capítulo se van a analizar los resultados obtenidos por las pruebas realizadas y si se han cumplido con los objetivos del proyecto. Para esto se va a hacer una recapitulación de los mismos y se va a estudiar si se han alcanzado adecuadamente.

En resumen, en este proyecto se ha explorado una alternativa simple y eficiente para realizar un control de temperatura con Arduino. Los resultados obtenidos muestran que esto ha sido posible, y además con un margen mínimo de error. Ahora se va a realizar un resumen de los objetivos y alcance del proyecto y se discutirá si se han alcanzado, así como futuras ampliaciones que pueda tener el proyecto.

9.1 Consecución de objetivos

En primer lugar, se tiene que el objetivo principal del proyecto era el desarrollo de un sistema electrónico de control de temperatura para pruebas biomédicas.

En vista de las pruebas realizadas, se ha conseguido de manera más que satisfactoria. Los resultados muestran que el control de temperatura es preciso, tal y como se va a ver a continuación, enumerando los requisitos del proyecto y viendo como se comporta el dispositivo en comparación a la idea inicial:

- **Rango entre 25ºC y 100ºC:** Efectivamente, este es el rango final del control de la temperatura para el dispositivo, por lo que este requisito se ha cumplido.
- **Control de temperatura doble, con temperaturas diferentes:** En función de los experimentos realizados, se ha comprobado que es posible controlar los dos calentadores al mismo tiempo y, además, estableciendo temperaturas diferentes para cada uno de ellos, por lo que este resultado también es positivo.
- **Error menor a 0,5ºC para 37ºC:** Como se ha visto en los resultados experimentales, el error máximo que se ha observado para el control a 37ºC es de 0,2ºC. Esto comparado a los 0,5ºC marcados como objetivo, supone una mejora sustancial de la idea inicial, superando con creces lo marcado.
- **Control de la referencia con potenciómetros:** Este requisito también es una realidad, se ha conseguido instaurar que las dos temperaturas de referencia sean establecidas mediante el ajuste manual de dos potenciómetros, uno para cada calentador.
- **Uso de pantalla como interfaz con el usuario:** Se ha establecido la incorporación de una pantalla al dispositivo que muestre de manera visual la temperatura a la que se encuentra la muestra en cada instante.
- **Uso sencillo del dispositivo:** Debido a la instalación de la pantalla y los potenciómetros, el manejo del dispositivo resulta muy sencillo y compresible.

En conclusión, se puede decir que el proyecto ha cumplido con todos los objetivos y requisitos que se habían instaurado en un inicio, incluso superando las expectativas en alguno de ellos. Gracias a esto, se puede determinar que se dispone de un dispositivo sencillo capaz de realizar pruebas biomédicas con control de temperatura con éxito.

9.2 Futuras líneas de investigación

Este es un proyecto con mucho potencial que puede ser ampliado centrándose en varias partes del mismo. A continuación, se van a listar una serie de ampliaciones y mejoras que serían viables para aplicar en un futuro para aumentar las prestaciones del dispositivo:

- **Diseño de una interfaz de usuario avanzada:** Se podría diseñar una interfaz más cómoda de usar, para que, en vez de tener que ajustar la temperatura de referencia con potenciómetros, poder disponer de una pantalla táctil, o desarrollar una aplicación en el dispositivo móvil del usuario. De esta manera, el dispositivo sería operable mas fácil y cómodamente.
- **Implementación de otros algoritmos de control más desarrollados:** Este proyecto utiliza un controlador PID básico, esto se podría mejorar implementando algún tipo de control más avanzado, como el control predictivo o el control adaptativo. Alguno de estos algoritmos podría mejorar la precisión y la respuesta del sistema ante perturbaciones.
- **Optimización del diseño de la PCB:** La PCB ha quedado con un tamaño de 9x9 cm, esto se podría optimizar y poner más próximos los componentes unos de otros, ya que ha sobrado bastante espacio en el modelo final.
- **Incorporación de otras funcionalidades:** Se podrían programar más funciones para no tener que estar siempre monitoreando el dispositivo, como, por ejemplo, un apagado de emergencia por una desviación atípica de la temperatura establecida.
- **Integración de otros sensores:** Esto serviría para hacer estudios más completos de la muestra a analizar, pudiendo monitorear y controlar otras variables de la misma como el pH o los niveles de oxígeno de la muestra.

Esto son algunas ideas que podrían encajar de manera adecuada en ampliaciones del presente proyecto para darle más utilidades o, simplemente, mejorar la experiencia de uso del usuario al realizar pruebas con el dispositivo. En cualquier caso, se ve que puede ser interesante ampliar el proyecto en un futuro.

ANEXO A. CÓDIGO DE ARDUINO

```
#include <PID_v1.h>
#include <Adafruit_GFX.h>    // include Core graphics library
#include <Adafruit_ST7735.h> // include Hardware-specific library
#include <SPI.h>

#define TFT_CS     10
#define TFT_RST    8
#define TFT_DC     9
Adafruit_ST7735 tft = Adafruit_ST7735(TFT_CS,  TFT_DC, TFT_RST);
#define TFT_SCLK 13
#define TFT_MOSI 11
//Adafruit_ST7735 tft = Adafruit_ST7735(TFT_CS, TFT_DC, TFT_MOSI, TFT_SCLK,
TFT_RST);

double Input1, Output1, Setpoint1, Input2, Output2, Setpoint2;
double medida1, medida2;
double medidaf1 = 0, medidaf2 = 0;
double Kp = 200;
double Kd = 0.0;
double Ki = 0.0;
float alfa = 0.05;
float ADCref1, ADCNTC1, PWM1, Vref1, ADCref2, ADCNTC2, PWM2, Vref2, T1, T2,
Tref1, Tref2, R1, R2, alfa1, alfa2, VNTC1_f, VNTC2_f, V2_1, V2_2, error1,
error2;
float beta=3410, To=298.15, Ro=10000;
PID myPID(&Input1, &Output1, &Setpoint1, Kp, Ki, Kd, DIRECT);
PID myPID2(&Input2, &Output2, &Setpoint2, Kp, Ki, Kd, DIRECT);

int tempo=0;

void setup() {
  // put your setup code here, to run once:
  myPID.SetSampleTime(10);
  myPID.SetMode(AUTOMATIC);
  myPID2.SetSampleTime(10);
  myPID2.SetMode(AUTOMATIC);

  Serial.begin(9600);
  pinMode(2, OUTPUT);
  pinMode(4, OUTPUT);
  pinMode(6, OUTPUT);
  pinMode(7, OUTPUT);

  tft.initR(INITR_BLACKTAB);
  Serial.println(F("Initialized"));
```

```
  uint16_t time = millis();
  tft.fillScreen(ST7735_BLACK);
  time = millis() - time;
  Serial.println(time, DEC);
  delay(500);

  tft.fillScreen(ST7735_WHITE);
  tft.setRotation(3);
  testdrawtext("Tref1",ST7735_BLUE,10,10);
  testdrawtext("Tref2",ST7735_BLUE,90,10);
  testdrawtext("T1",ST7735_BLUE,10,60);
  testdrawtext("T2",ST7735_BLUE,90,60);
  testdrawnum(Tref1,ST7735_BLUE,10,35);
  testdrawnum(Tref2,ST7735_BLUE,90,35);
  testdrawnum(T1,ST7735_BLUE,10,85);
  testdrawnum(T2,ST7735_BLUE,90,85);

}

void loop() {
  // put your main code here, to run repeatedly:

  //input ntc1
  ADCNTC1 = analogRead(A2);
  VNTC1_f = ADCNTC1*5/1023;
  V2_1 = (5.625-VNTC1_f)/2.25;
  R1 = 5*(988/V2_1)-988;
  //input ntc2
  ADCNTC2 = analogRead(A3);
  VNTC2_f = (ADCNTC2*5/1023);
  V2_2 = (5.625-VNTC2_f)/2.25;
  R2 = 5*(988/V2_2)-988;

  //temperatura de la NTC1
  alfa1 = (1000/To)+((1000/beta)*(log(R1/Ro)));
  T1 = 1000/alfa1 - 278.65 + 2;
  //temperatura de la NTC2
  alfa2 = (1000/To)+((1000/beta)*(log(R2/Ro)));
  T2 = 1000/alfa2 - 278.65 - 4;

  //entrada de referencia1
  ADCref1 = analogRead(A0);
  Vref1 = ADCref1*5.0/1023;
  Tref1 = (Vref1*75/5 + 25);
  //entrada de referencia2
  ADCref2 = analogRead(A1);
  Vref2 = ADCref2*5.0/1023;
  Tref2 = (Vref2*75/5 + 25);
```

```
//salida PWM1
medida1 = T1;
medidaf1 = medida1*alfa + (1-alfa)*medidaf1;
Input1 = medidaf1;
Setpoint1 = Tref1;
error1 = Tref1 - T1;
//salida PWM2
medida2 = T2;
medidaf2 = medida2*alfa + (1-alfa)*medidaf2;
Input2 = medidaf2;
Setpoint2 = Tref2;
error2 = Tref2 - T2;

myPID.Compute();
myPID2.Compute();
analogWrite(3, Output1);
analogWrite(5, Output2);

//codigo leds

if(abs(error1)<5)
{
  digitalWrite(6, HIGH);
  digitalWrite(7, LOW);
}
else
{
  digitalWrite(6, LOW);
  digitalWrite(7, HIGH);
}

if(abs(error2)<5)
{
  digitalWrite(2, HIGH);
  digitalWrite(4, LOW);
}
else
{
  digitalWrite(2, LOW);
  digitalWrite(4, HIGH);
}


//pantalla
if (tempo>20){
  testdrawrects(ST7735_WHITE,35,50);
  testdrawrects(ST7735_WHITE,85,100);
  testdrawnum(T1,ST7735_BLUE,10,85);
  testdrawnum(Tref1,ST7735_BLUE,10,35);
```

```
    testdrawnum(T2,ST7735_BLUE,90,85);
    testdrawnum(Tref2,ST7735_BLUE,90,35);
    tempo = 0;
  }
  else{
    tempo++;
  }
}


void testdrawtext(char *text, uint16_t color, int pos_x, int pos_y) {
  tft.setCursor(pos_x, pos_y);
  tft.setTextColor(color);
  tft.setTextSize(2);
  tft.setTextWrap(true);
  tft.print(text);
}

void testdrawnum(float num, uint16_t color, int pos_x, int pos_y) {
  tft.setCursor(pos_x, pos_y);
  tft.setTextColor(color);
  tft.setTextSize(2);
  tft.setTextWrap(true);
  tft.print(num);
}

void testdrawrects(uint16_t color, int pos_y,int pos_final) {
  for (int16_t x=0; x < pos_final-pos_y; x+=1) {
    tft.drawLine(0,pos_y+x, 170, pos_y+x, color);
  }
}
```

ANEXO B. HOJA DE CARACTERÍSTICAS

En este apartado se busca ilustrar los aspectos más importantes del dispositivo para su funcionamiento. Entre ellos, las tensiones de alimentación y corrientes, entre otros.

En las siguientes tablas se pueden encontrar los valores característicos para los elementos más importantes del sistema:

Tensiones de Alimentación	
Calentador	12 V
Arduino	12 V (con limitador para 5 V)
Driver	5 V
Amplificadores operacionales	5 V
Pantalla	5 V
LED de Pantalla	3.3 V
LEDs luminosos verdes y rojos	5 V

Tabla de Anexo 1 - Tensiones de alimentación

Temperaturas		
Elemento	T. Máxima (ºC)	T. Mínima (ºC)
Calentador	100	25
Sensor de temperatura	100	25
Referencia	100	25

Tabla de Anexo 2 - Temperaturas máximas y mínimas

Dimensiones de la PCB	
Ancho (cm)	9
Profundo (cm)	9

Tabla de Anexo 3 - Dimensiones del dispositivo

Características del sensor de temperatura NTC	
Resistencia	10 kΩ
Tolerancia	± 5%
Máxima potencia nominal	100 mW
Rango de temperaturas de funcionamiento	-40ºC, 125ºC
Longitud	1 mm

Tabla de Anexo 4 - Caracteríasticas del sensor de temperatura

Pines de Arduino conectados	
13	Pantalla (SCK)
11	Pantalla (SDA)
10	Pantalla (CS)
9	Pantalla (A0)
8	Pantalla (RESET)
2	LED 1
4	LED 2
6	LED 3
7	LED 4
3	PID 1
5	PID 2
A0	Convertidor A/D (Vref1)
A1	Convertidor A/D (Vref2)
A2	Convertidor A/D (VNTC1)
A3	Convertidor A/D (VNTC2)

Tabla de Anexo 5 - Pines de Arduino utilizados

Tensiones de Alimentación	
Calentador	12 V
Arduino	12 V (con limitador para 5 V)
Driver	5 V
Amplificadores operacionales	5 V
Pantalla	5 V
LED de Pantalla	3.3 V
LEDs luminosos verdes y rojos	5 V

Tabla de Anexo 6 - Tensiones de alimentación

Corrientess de Alimentación (limitada por los transistores)	
Corriente admisible transistores	30 A

Tabla de Anexo 7 - Tensiones de alimentación

BIBLIOGRAFÍA

[1] https://es.scribd.com/document/362867715/Funcionamiento-Del-Termostato-Del-Motor

[2] https://mikai.mx/controles-y-termostatos/

[3] R. Balan, S. -D. Stan and C. Lapusan, "A model based predictive control algorithm for building temperature control," 2009 3rd IEEE International Conference on Digital Ecosystems and Technologies, Istanbul, Turkey, 2009, pp. 540-545, doi: 10.1109/DEST.2009.5276699. keywords: {Predictive models;Predictive control;Prediction algorithms;Temperature control;Computational modeling;Process control;Signal generators;Signal analysis;Optimal control;Signal processing;temperature control;model based predictive control algorithm},

[4] B. Supriyo, Dadi, S. Warjono, A. Wisaksono, S. Astuti and K. Utomo, "PID Based Air Heater Controller Implemented With Matlab/Simulink and Arduino Uno," 2018 5th International Conference on Information Technology, Computer, and Electrical Engineering (ICITACEE), Semarang, Indonesia, 2018, pp. 28-32, doi: 10.1109/ICITACEE.2018.8576955. keywords: {Heating systems;Matlab;Tools;Temperature sensors;Relays;air heater;PID;Relay Feedback;Ziegler Nichols;Matlab/Simulink;Arduino Uno},

[5] Santoso, Clarissa & Hamzah, Torib & Syaifudin, Syaifudin & Mujahid, Muhammad Umer Farooq. (2023). PID Temperature Control on Blood Warmer Equipped with Patient Temperature and Blood Temperature. Indonesian Journal of Electronics Electromedical Engineering and Medical Informatics. 5. 125-134. 10.35882/ijeeemi.v5i3.286.

[6] F. Hiltansyah and G. Kiswanto, "Development of PID Controller for Temperature Control of High Flow Nasal Cannula Humidification System Based on System Identification," *2023 13th International Conference on Power, Energy and Electrical Engineering (CPEEE)*, Tokyo, Japan, 2023, pp. 170-176.

[7] https://www.electrontools.com/Home/WP/valores-comerciales-de-resistencias/

[8] Perdigones Sánchez, F. A., Perales Esteve, M., Palomo Vázquez, B., Perales Esteve, M., & Perales Esteve, M. (2019). Apuntes de la asignatura Electrónica General. Universidad de Sevilla, Escuela Técnica Superior de Ingeniería, Sección de Publicaciones.

[9] https://www.arduino.cc/

[10] https://www.novusautomation.com/es/noticia/articulo-control-pid-rompiendo-la-barrera-del-tiempo

[11] https://www.arduino.cc/reference/en/libraries/pid/