Universidad del País Vasco Euskal Herriko Unibertsitatea
BILBOKO INGENIARITZA ESKOLA
ESCUELA DE INGENIERÍA DE BILBAO

GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

TRABAJO FIN DE GRADO

DISEÑO Y FABRICACIÓN DE UN SISTEMA DE CONTROL Y MONITORIZACIÓN DE TEMPERATURA

Alumna: España Hernández, Sara

Director: Sainz de Murieta Mangado, Joseba Andoni

Curso: 2020-2021

Fecha: Bilbao, 30 de junio de 2021

RESUMEN. El presente trabajo forma parte de una serie de trabajos en los cuales se realiza el análisis y diseño teóricos de una maqueta para el estudio de la regulación y el control de procesos térmicos. Dicha maqueta está compuesta principalmente por una resistencia calefactora, cuatro sensores de temperatura y un controlador PID implementado en Arduino.
Este trabajo en particular, tiene como finalidad adaptar y ampliar los diseños teóricos previos con el fin de obtener resultados tangibles. Además, se incorpora una aplicación informática. Esta permite controlar el equipo, representar gráficamente los datos adquiridos por los sensores y almacenar dichos datos para su posterior análisis.

PALABRAS CLAVE. Control de temperatura / Monitorización / Regulación / Control / Adquisición de datos / Software embebido / Desarrollo de aplicaciones / PID / Arduino / Matlab

LABURPENA. Lan hau lan batzuen parte da, non prozesu termikoen erregulazioa eta kontrola aztertzeko maketa baten azterketa eta diseinu teorikoak egiten diren. Maketa horren osagai nagusiak hauek dira: erresistentzia berogailua, lau tenperatura-sentsore eta Arduinon ezarritako PID kontrolatzailea.
Lan honen helburua da aldez aurreko diseinu teorikoak egokitzea eta zabaltzea, emaitza ukigarriak lortzeko. Gainera, aplikazio informatiko bat gehitu da. Honi esker, ekipoa kontrola daiteke, sentsoreek hartutako datuak grafikoki adieraz daitezke eta datu horiek gorde daitezke, ondoren aztertzeko.

HITZ GAKOAK. Tenperaturaren kontrola / Monitorizazioa / Erregulazioa / Kontrola / Datu-bilketa / Software txertatua / Aplikazioen garapena / PID / Arduino / Matlab

ABSTRACT. This project is part of a series of projects in which the theoretical analysis and design of a model for the study of the regulation and control of thermal processes is carried out. This model is mainly composed of a heating resistor, four temperature sensors and a PID controller implemented in Arduino.
This work in particular aims to adapt and expand previous theoretical designs in order to obtain tangible results. In addition, a computer application is incorporated. This allows you to control the equipment, graphically represent the data acquired by the sensors and store this data for later analysis.

KEYWORDS. Temperature control / Monitoring / Regulation / Control / Data acquisition / Embedded software / Application development / PID / Arduino / Matlab

ÍNDICE

ILUSTRACIONES

TABLAS

SIGLAS

CA: Corriente Alterna

CC: Corriente Continua

CMOS: Complementary Metal-Oxide-Semiconductor

CTC: Clear Timer on Compare Match

DC: Duty Cycle

ICSP: In Chip Serial Programmer

IDE: Entorno de Desarrollo Integrado

PWM: Pulse Width Modulation

RISC: Reduced Instruction Set Computing

RTD: Resistance Temperature Detector

TWI: Two Wire Interface

UART: Universal Asyncronos Receiver and Transmitter

Universidad del País Vasco Euskal Herriko Unibertsitatea
BILBOKO INGENIARITZA ESKOLA
ESCUELA DE INGENIERÍA DE BILBAO

GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

TRABAJO FIN DE GRADO

DISEÑO Y FABRICACIÓN DE UN SISTEMA DE CONTROL Y MONITORIZACIÓN DE TEMPERATURA

DOCUMENTO I - MEMORIA

Alumna: España Hernández, Sara

Director: Sainz de Murieta Mangado, Joseba Andoni

Curso:2020-2021

Fecha: Bilbao, 30 de junio de 2021

1. INTRODUCCIÓN

Este proyecto consiste en el diseño y fabricación de un sistema de control y regulación de temperatura y su correspondiente sistema de monitorización.

El equipo está formado principalmente por una resistencia calefactora, un sistema de control PID implementado en el microcontrolador ATmega2560, cuatro sensores de temperatura y un programa para la monitorización y el almacenamiento de datos.

El desarrollo de este proyecto tiene como finalidad englobar muchos de los conocimientos adquiridos en el grado de Ingeniería Electrónica Industrial y Automática y, al mismo tiempo, la obtención de equipamiento para la investigación y la docencia en la Escuela de Ingeniería de Bilbao.

2. CONTEXTO

El proyecto parte desde la selección de los sensores de temperatura y el diseño de los circuitos de acondicionamiento de cada uno de ellos. Continúa con la selección de los componentes electrónicos que conforman estos circuitos y el diseño de la estructura que lo contiene. También se analiza el sistema de adquisición de datos, y se realiza el tratamiento de los mismos, además de la programación del sistema de control. Por último, se desarrolla una aplicación que permite la comunicación del equipo a un PC, de manera que el proyecto finaliza con el desarrollo de un entorno gráfico que facilita el estudio y la comprensión del comportamiento de un sistema de control aplicado a un sistema térmico.

3. OBJETIVOS Y ALCANCE

Objetivos perseguidos mediante la realización del proyecto:

- Poner en práctica los conocimientos adquiridos en el grado de Ingeniería Electrónica Industrial y Automática.
- Diseñar y fabricar PCBs.
- Ampliar los conocimientos en programas de diseño 3D.
- Dominio del microcontrolador ATmega2560.
- Profundizar en las prestaciones de Matlab para el desarrollo de aplicaciones.

Objetivos finales:

- Aplicar un sistema de control de temperatura con un control PID regulable.
- Diseñar una interfaz gráfica clara e intuitiva.
- Dotar al equipo de un sistema de monitorización en tiempo real efectivo.
- Permitir el control del equipo desde el PC durante la monitorización.

4. BENEFICIOS

Las utilidades de este proyecto, al igual que los objetivos del mismo, pueden diferenciarse entre aquellas de carácter personal y las relacionadas con el uso final del equipo.

Los beneficios de carácter personal serán obtenidos durante la realización del proyecto.

Este proyecto engloba gran parte de las competencias adquiridas en el grado de Ingeniería Electrónica Industrial y Automática. Especialmente, aquellos relacionados con las asignaturas de Automatismos y Control, Electrónica Industrial, Electrónica de Potencia, Sistemas Electrónicos Digitales, Regulación Automática e Instrumentación Electrónica.

Por lo tanto, permite poner en práctica dichos conocimientos. Además, para cumplir con los objetivos planteados deberán de adquirirse nuevos conocimientos relacionados con el desarrollo de software, el manejo de nuevos programas y los procesos de fabricación.

Su realización también favorecerá el desarrollo de competencias transversales como la resolución de problemas, la gestión del tiempo, la creatividad y la responsabilidad.

En cuanto a los beneficios obtenidos del resultado final, se tiene la adquisición de nuevo equipamiento didáctico para reforzar las prácticas de laboratorio de las asignaturas de Automatismos y Control y Regulación Automática. El empleo de este equipo en la docencia permitirá aplicar los fundamentos teóricos a un caso real, asimilando así los conceptos de forma experimental.

5. ESTADO DEL ARTE

Existen numerosas empresas dedicadas a la fabricación de equipamiento de laboratorio. En ellas se puede apreciar una mayor oferta con aplicaciones en la Ingeniería Mecánica o Civil, siendo esta más escasa en el caso de la Ingeniería Electrónica o el Control de Procesos. En estos últimos casos, cabe destacar empresas como Alecop o Edibon.

Estas presentan una mayor variedad de equipos relacionados con el ámbito de la electrónica y la automatización.

En el caso de Edibon, se pueden encontrar dos modelos similares al propuesto en este proyecto.

El primero, denominado Módulo de ensayos de temperatura, consiste en un recinto semiabierto equipado con dos lámparas encargadas del calentamiento, y varios sensores de temperatura dispuestos en distintas posiciones respecto a los focos de calor. Este módulo tiene como finalidad enseñar el uso y aplicaciones de los sensores de temperatura, sin tener en cuenta el control y regulación de la misma.

Fig. 1 Módulo de ensayos de temperatura.

El segundo, denominado Módulo de ensayo de hornos, consiste en un horno provisto de una resistencia calefactora, un ventilador de velocidad variable, un control PID y distintos sensores de temperatura. En este caso, el equipo permite emplear distintos tipos de sensores, además de realizar el control de la temperatura del recinto sellado.

Fig. 2 Módulo de ensayo de hornos.

Sin embargo, estos dos módulos en sí mismos no poseen ni los respectivos circuitos de acondicionamiento de cada sensor, ni un visor en el que poder observar el resultado de las mediciones. Es por ello que el fabricante menciona la necesidad de adquirir otro de sus equipos para poder trabajar con ellos.

En este caso se trata de un equipo que contiene los circuitos de acondicionamiento de las señales de salida de los sensores, amplificadores de instrumentación, filtros, convertidores de corriente a tensión y de frecuencia a tensión, un controlador PID, etc.

Al ser este equipo compatible con varios módulos, su compra para un único equipo encarece el coste notablemente, además de no aprovechar todas sus capacidades.

Fig. 3 Módulo de ensayos.

Edibon también ofrece la posibilidad de adquirir el software para PC que controla el equipo mencionado anteriormente mediante una tarjeta de adquisición de datos. Se trata de un software genérico, que contiene los siguientes instrumentos: osciloscopio, analizador de espectro, analizador de respuesta en régimen transitorio, voltímetro y generador de funciones.

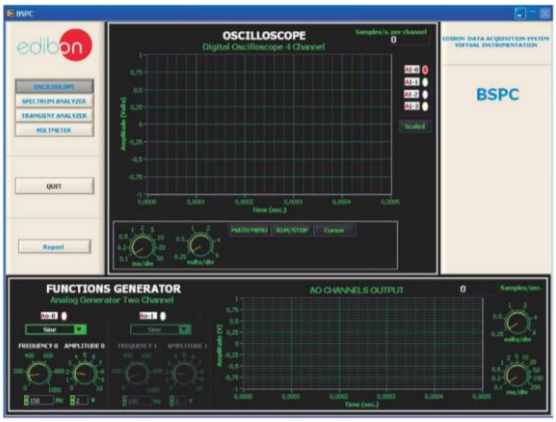


Fig. 4 Software genérico de Edibon.

Este software, por lo tanto, tampoco está optimizado para su uso con el equipo de control que resulta de interés en este caso.

En el caso de Alecop, la maqueta consiste en una resistencia calefactora, un ventilador y los diferentes captadores de temperatura. Al contrario que los modelos de Edibon, en este caso se trata de un equipo autosuficiente ya que incorpora un indicador que posibilita la lectura de la temperatura de los transductores. Por otro lado, este modelo también incluye los módulos que contienen los circuitos de acondicionamiento de cada uno de los sensores. No obstante, estos módulos únicamente permiten introducir la señal de consigna y modificar la ganancia de los circuitos de acondicionamiento, por lo que no se tiene ningún control sobre la resistencia calefactora. Por otra parte, este modelo no ofrece la posibilidad de conexión con un PC.

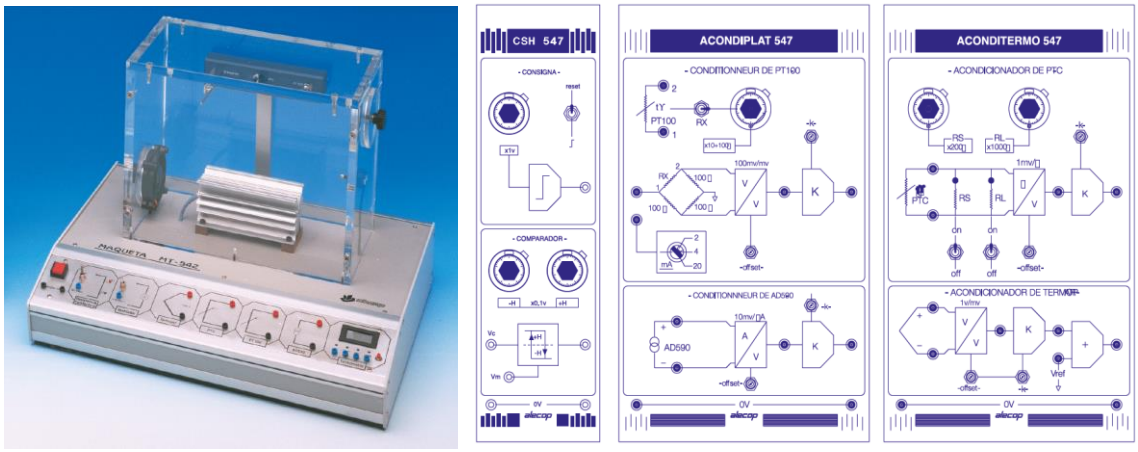


Fig. 5 Módulo de ensayos Alecop.

6. ANÁLISIS DE ALTERNATIVAS

Una vez analizados los modelos existentes actualmente en el mercado, se han tenido en cuenta las ventajas y desventajas de cada uno de ellos, para llegar a la propuesta final a desarrollar en este proyecto.

Antes, conviene recordar las necesidades que se desean satisfacer. En primer lugar, el equipo debe facilitar la enseñanza de los conceptos relacionados con el control y regulación de procesos. De igual importancia es la necesidad de que se trate de un equipo atractivo y fácil de utilizar. También, es preferible un equipo capaz de funcionar por sí mismo, si bien ofreciendo la posibilidad de conectarlo a un PC.

Finalmente, para cumplir con los requisitos anteriormente mencionados, se ha apostado por un equipo que pueda ser controlado y comprendido en su totalidad por los/las estudiantes. Esto es, que se tenga un control completo sobre el PID y, además, se tengan a disposición del alumnado que esté interesado en profundizar en el tema, cada uno de los circuitos de acondicionamiento de los sensores. Por otra parte, con la intención de crear un equipo autosuficiente, este poseerá un visor en el que se indicará la información, y contará también con la posibilidad de conectarlo al PC. A través del PC se podrá tener el control de todas las funcionalidades del equipo, además de visualizar de forma gráfica los datos adquiridos y realizar el almacenamiento de los mismos en hojas de cálculo. Todo ello, a través de un programa diseñado expresamente para el equipo en cuestión.

7. DESCRIPCIÓN DE LA SOLUCIÓN PROPUESTA

Para describir la solución propuesta, se han diferenciado tres partes del proyecto: electrónica, mecánica y software.

En la siguiente figura se muestra un diagrama representativo de la parte electrónica de la solución propuesta.

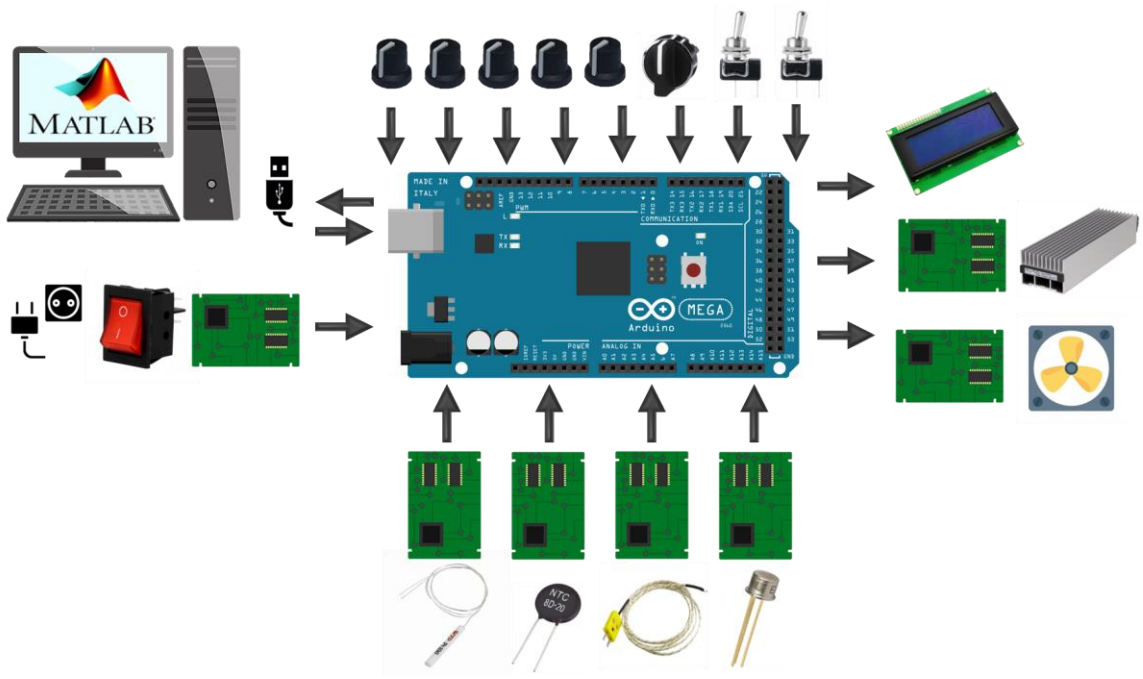


Fig. 6 Diagrama de la solución propuesta.

En él se pueden apreciar los componentes empleados para el desarrollo electrónico del equipo.

El diseño se basa en la tarjeta Arduino Mega 2560, encargada de la adquisición de datos de las entradas, y el control de los actuadores. También permite la comunicación con el PC.

En cuanto a las entradas, el diseño consta de cuatro sensores de temperatura: PT100, NTC, termopar tipo K y AD590. También se precisan cinco potenciómetros, un selector y tres interruptores. La función de los potenciómetros es permitir que la persona usuaria pueda introducir el valor de consigna deseado, modificar las constantes del PID y la velocidad del ventilador. El selector permite elegir el sensor a emplear como referencia para la acción PID. Por su parte, los interruptores posibilitan el encendido y apagado del equipo, la conmutación entre el modo manual y el modo PC y el encendido y apagado del ventilador.

En cuanto a las salidas, se tienen tres elementos a controlar: resistencia calefactora, ventilador, pantalla LCD. La resistencia calefactora es la encargada de aportar el calor al interior del horno. El ventilador permite enfriar el horno más rápidamente, o bien actúa como perturbación. Por último, la pantalla LCD visualiza la información más provechosa para la persona usuaria.

Finalmente, se tienen siete placas de circuito impreso. En el caso de las placas correspondientes a los sensores, se trata de los circuitos de acondicionamiento. Para el control de grandes cargas con Arduino, son necesarias las placas que contienen el correspondiente circuito de control. Por

último, y no menos importante, se tiene la placa de potencia, la cual permite la conexión del equipo a la red.

A continuación, se presenta la parte mecánica mediante una figura ilustrativa.

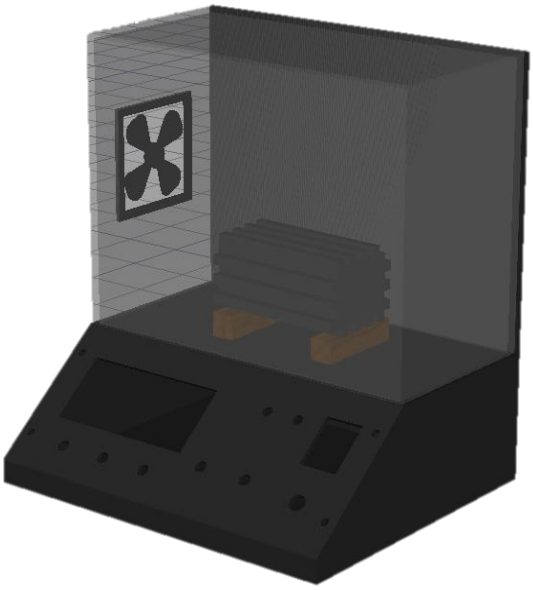


Fig. 7 Diseño de la maqueta desarrollado en AutoCAD.

El material empleado para elaborar la maqueta es acero. En los laterales se encuentran los conectores de red y USB. En su interior, se pueden hallar las placas de circuito impreso, así como el Arduino Mega 2560. El panel frontal contiene los componentes relacionados con la interacción del usuario/a. La cámara de policarbonato contiene la resistencia calefactora, el ventilador y los sensores de temperatura. Sus propiedades térmicas permiten trabajar a temperaturas de hasta 130℃.

En cuanto al software, se tienen dos programas. El primero de ellos, elaborado en el entorno de desarrollo integrado de Arduino, es ejecutado por el microcontrolador ATmega2560. Consiste en un proyecto formado por tres ficheros:

Fig. 8 Ficheros del proyecto desarrollado en Arduino.

El fichero Maqueta_Control_Temperatura.ino alberga el programa principal.

El fichero Funciones.cpp contiene la descripción de las funciones empleadas en el programa principal.

La cabecera Funciones.h enlaza el programa principal y las funciones mediante las declaraciones.

El segundo programa, realizado en el entorno de desarrollo App Designer de Matlab, es ejecutado por el PC. Su uso es opcional, ya que únicamente es necesario si se selecciona el Modo PC. En este caso, el control del equipo se realiza a través del computador. Por otra parte, se visualizan los datos adquiridos en una gráfica, y existe la opción de almacenarlos en un archivo Excel.

Finalmente, se muestra una imagen de la pantalla principal del sistema de monitorización.

Fig. 9 Interfaz gráfica desarrollada en Matlab.

Universidad del País Vasco Euskal Herriko Unibertsitatea
BILBOKO INGENIARITZA ESKOLA
ESCUELA DE INGENIERÍA DE BILBAO

GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

TRABAJO FIN DE GRADO

DISEÑO Y FABRICACIÓN DE UN SISTEMA DE CONTROL Y MONITORIZACIÓN DE TEMPERATURA

DOCUMENTO II - METODOLOGÍA

Alumna: España Hernández, Sara

Director: Sainz de Murieta Mangado, Joseba Andoni

Curso: 2020-2021

Fecha: Bilbao, 30 de junio de 2021

1. DESCRIPCIÓN DE TAREAS

En este documento se presentan y desarrollan las tareas necesarias para llegar al resultado final expuesto en la memoria.

En la siguiente ilustración se muestran las tareas que componen el proyecto, la duración de cada una de ellas y las fechas de ejecución.

Tabla 1 Descripción de tareas en Microsoft Project.

	Modo de	Nombre de tarea	Duración	Comienzo	Fin	Predecesora
1		Comienzo del TFG	0 días	lun 21/09/20	lun 21/09/20	
2		Análisis teórico	40 días	lun 21/09/20	vie 13/11/20	1
3		Diseño de los circuitos electrónicos	17 días	lun 16/11/20	mar 08/12/20	2
4		Selección de componentes	11 días	mié 09/12/20	mié 23/12/20	3
5		Diseño de las placas de circuito impreso en KiCad	20 días	jue 24/12/20	mié 20/01/21	4
6		Programación del código fuente en Arduino	16 días	jue 21/01/21	jue 11/02/21	5
7		Programación del código fuente en Matlab	33 días	vie 12/02/21	mar 30/03/21	6
8		Diseño de los planos en AutoCAD	10 días	mié 31/03/21	mar 13/04/21	7
9		Pruebas unitarias	7 días	mié 14/04/21	jue 22/04/21	8
10		Montaje	6 días	vie 23/04/21	vie 30/04/21	9
11		Pruebas generales	13 días	lun 03/05/21	mié 19/05/21	10
12		Redacción de la documentación	41 días	lun 26/04/21	lun 21/06/21	
13		Entrega del TFG	0 días	mié 30/06/21	mié 30/06/21	12

2. DIAGRAMA DE GANTT

El diagrama de Gantt es una herramienta de gestión que permite planificar y programar tareas a lo largo de un periodo determinado. Este gráfico permite también realizar el seguimiento del progreso y representa la duración y la secuencia de las tareas.

En el diagrama de Gantt se puede observar que el proyecto en cuestión contiene dos hitos: el inicio y la entrega del proyecto. Además, este segundo hito posee una fecha límite. En cuanto a las tareas, se encuentran vinculadas de forma lineal, de manera que cada una de ellas tiene como predecesora la tarea anterior. Existe una excepción, la tarea de redacción de la documentación, la cual no posee ninguna predecesora.

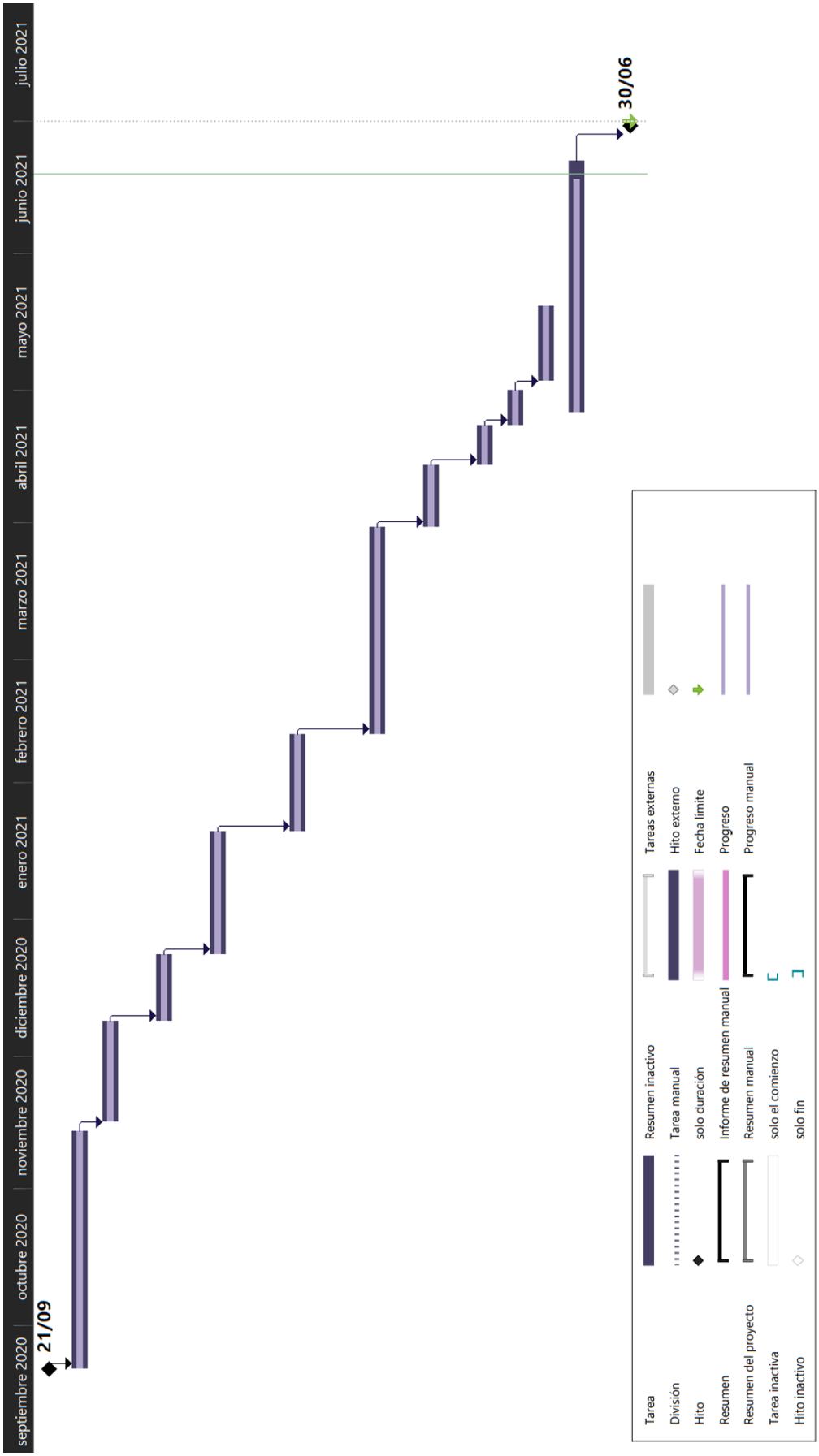


Fig. 10 Diagrama de Gantt.

3. DESARROLLO DE LAS TAREAS

3.1. ANÁLISIS TEÓRICO

3.1.1. Sensores

En este apartado se presentan los sensores más empleados en la industria para la medida de temperatura: RTD, NTC, termopar y sensores de silicio.

Para cada uno de ellos, se indican las características más notables, así como el rango de temperatura, sensibilidad, robustez y tamaño entre otros. Se presentan también las curvas características mediante las cuales se analiza la sensibilidad del sensor y el modelo matemático que lo define.

RTD

Los sensores de temperatura de resistencia metálica se basan en la variación de la resistencia eléctrica de los metales con la temperatura. Esto es, al aumentar en ellos la energía interna aumenta su resistividad.

Los metales comúnmente usados para la fabricación de RTDs son el platino, níquel, wolframio y cobre. El platino, posee un coeficiente térmico inferior al resto, de $3{,}9 \cdot 10^{-3} K^{-1}$. No obstante, su resistividad, de $10{,}6 \cdot 10^{-8} \Omega \cdot m$, es superior a las del resto de metales, lo que permite construir hilos muy finos con una elevada resistencia sin necesidad de una gran longitud. El campo de medida de las RTD suele abarcar desde los -200ºC hasta los 850ºC.

La caracterización de las RTDs se realiza mediante el valor de la resistencia frente a la temperatura.

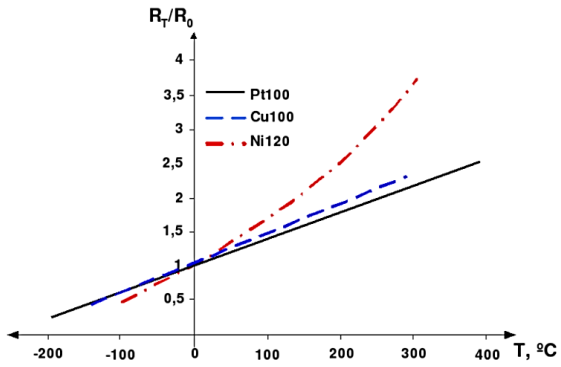


Fig. 11 Curvas características de RTDs.

Analizando las curvas de la figura, se puede observar que el metal con mayor sensibilidad es el níquel, pero también el que menos linealidad ofrece. Por esta razón, en la mayoría de las aplicaciones se utilizan RTDs de cobre o platino. Las RTDs destacan por su estabilidad y precisión aunque poseen un tamaño elevado y son más costosas que un termopar.

El modelo matemático correspondiente a la curva de calibración de una RTD es el siguiente:

$$R_T = R_0 \cdot (1 + \alpha \cdot \Delta T)$$

Donde:

- α es el coeficiente térmico de la RTD.
- R_0 es el valor de la resistencia a 0ºC.

NTC

Son sensores de temperatura de tipo resistivo. Su nombre hace referencia al signo de su coeficiente de temperatura, siendo este negativo. Es decir, están compuestas por un material semiconductor cuya resistencia disminuye cuando aumenta la temperatura. Además del coeficiente de temperatura, otra de las características destacables de estos sensores es su no linealidad. En la siguiente figura se puede observar la característica resistencia-temperatura de una NTC.

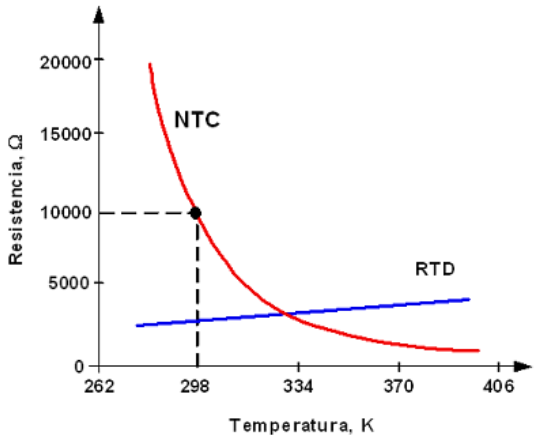


Fig. 12 Característica resistencia-temperatura de una NTC.

Una sensibilidad elevada es una característica deseable para cualquier sensor. En el caso de los termistores, la sensibilidad no es constante. Es muy grande a bajas temperaturas, si bien disminuye conforme esta aumenta. Su velocidad de respuesta no es muy elevada. Su rango de temperaturas abarca desde los -70℃ hasta 500℃, aunque la mayoría no sobrepasan los 100-150℃.

Pese a su conocida no linealidad, cuando el campo de medida es reducido, pueden emplearse circuitos de acondicionamiento con una elevada linealidad en la tensión de salida. También suelen emplearse para efectuar la compensación de temperatura de otros sensores.

El modelo matemático mediante el cual se representa la característica resistencia-temperatura se conoce como modelo exponencial y viene dado por la siguiente ecuación:

$$R_T = R_0\, e^{\beta\left(\frac{1}{T}-\frac{1}{T_0}\right)}$$

Donde:

- R_T es la resistencia del termistor a la temperatura T.
- β es el índice de sensibilidad del termistor.
- R_0 es la resistencia del termistor a la temperatura T_0.

Termopar

Los termopares son sensores de temperatura constituidos por dos metales diferentes, unidos en el punto en el que se desea medir la temperatura y con los extremos a la misma temperatura. Estas características hacen que aparezca una tensión proporcional a la diferencia de temperaturas entre los puntos de unión de ambos metales. Para comprender mejor el principio de funcionamiento, se procede a explicar los tres fenómenos en los que se basa el funcionamiento del termopar:

- Efecto Seebeck: dos conductores diferentes formando un circuito cerrado y con las uniones a temperaturas diferentes, provocan una fuerza electromotriz que da lugar a una corriente.

- Efecto Peltier: si se hace pasar una corriente por el termopar, aparece una diferencia de temperatura cuyo signo depende del sentido de la corriente.

- Efecto Thomson: si se hace circular una corriente constante por un conductor en el que los extremos están a diferente temperatura, se produce una transferencia de calor proporcional a la corriente por el gradiente de temperatura.

Por otra parte, los termopares se rigen por las denominadas leyes termoeléctricas:

- Ley de los circuitos homogéneos: la tensión generada por un termopar no depende de la temperatura a la que se encuentren los puntos intermedios.
- Ley de los metales intermedios: si se introduce un tercer metal en serie, la tensión generada por el termopar no varía siempre que los extremos del tercer metal se encuentren a la misma temperatura.
- Ley de las temperaturas intermedias: si $V_{T1,T2}$ es la tensión a la salida de un termopar cuyas uniones están a T_1 Y T_2, y $V_{T2,T3}$ es la tensión obtenida cuando están a T_2 y T_3, se conoce que la tensión de salida para las temperaturas T_1 y T_3 es igual a $V_{T1,T2}+ V_{T2,T3}$.

Estos sensores son también denominados sensores generadores ya que no es necesaria una fuente de alimentación externa.

En cuanto a los tipos de termopares, existe una gran variedad en función de los materiales que los constituyen. La importancia de esta característica radica en el efecto del ambiente sobre ellos, y el rango de medida para el cual resultan adecuados.

Los más utilizados son el Tipo J, de hierro y constantán, y el Tipo K, de cromo y aluminio. El primero posee un campo de medida menor, 0 a 760°C, frente a -200 a 1250°C del segundo. Sin embargo, el Tipo J presenta una mayor sensibilidad, siendo esta de 51,5µV/°C frente a 40,4µV/°C del tipo K.

Los termopares destacan por ser robustos y duraderos. Tienen un tamaño muy reducido y un tiempo de respuesta bajo, hasta tres veces menor que el de una RTD.

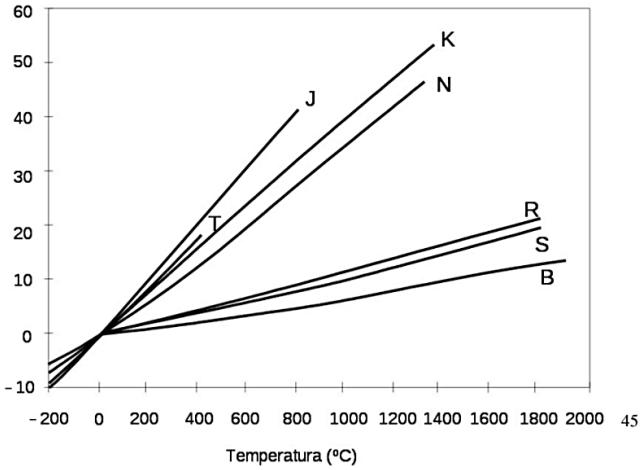


Fig. 13 Curvas de calibración de termopares.

Las curvas de calibración, en general, son bastante lineales:

En el caso de los termopares, las normas muestran la tensión que proporcionan cuando la unión de referencia está a 0℃. Aplicando la ley de las temperaturas intermedias, es posible conocer la tensión de salida para temperaturas diferentes a 0℃ en la unión de referencia.

Silicio

Los sensores de silicio, son circuitos integrados que emplean la variación predecible de la tensión de la unión base-emisor de los transistores bipolares para medir la temperatura.

Se caracterizan por su pequeño tamaño y por incluir dentro de los mismos las etapas de amplificación, linealización y compensación necesarias. Por el contrario, poseen un rango te temperatura inferior al de los sensores vistos anteriormente. La temperatura máxima que pueden alcanzar es de 150℃ debido al silicio.

La mayoría proporcionan una tensión salida que varía linealmente con la temperatura a razón de 10mV/℃. Algunos de estos circuitos integrados más conocidos son el LM34, LM35, LM135 y LM50. Otros, presentan una salida por corriente. En este caso, se tienen los modelos LM334 y AD590, cuyas sensibilidades son 1mA/K y 1µA/K.

3.1.2. Regulación y control de sistemas

Control PID

El control proporcional, integral y derivativo es un algoritmo empleado para controlar sistemas con una entrada y una salida analógicas. En la siguiente figura se puede observar una respuesta típica de estos sistemas.

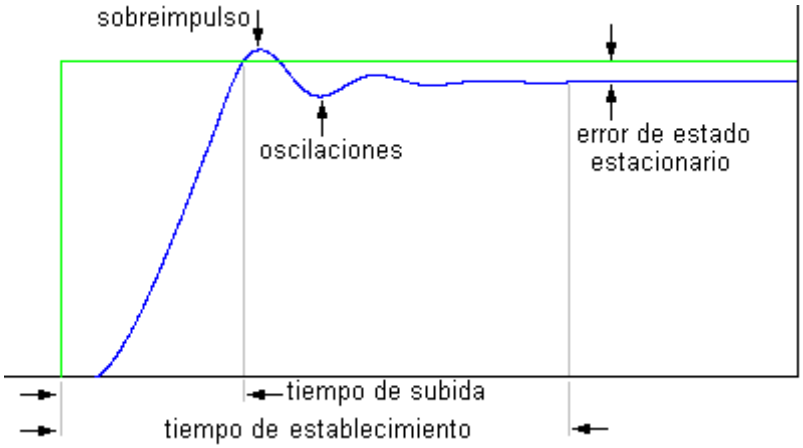


Fig. 14 Respuesta tipo de un control PID.

En ella, se muestran algunas de las características, como el rebose, las oscilaciones, el error en estado estacionario, y los tiempos de subida y de establecimiento.

Las dos primeras características provienen de la estabilidad del sistema. Los tiempos dan una idea de la velocidad de respuesta. Un sistema ideal es estable, rápido y con error nulo en el estado estacionario.

Los controladores industriales se clasifican en función de las acciones de control que aplican. Estos, pueden incluir las tres acciones (proporcional, integral y derivativa) o solo algunas de ellas.

Acción de control proporcional:

En el control proporcional la salida alcanza el valor final, pero presenta un rebose y un error en el estado estacionario.

Posee las siguientes características:

- Los cambios bruscos en la señal de referencia provocan cambios bruscos en la señal de control.
- Ante una señal de error grande, la señal de control será también grande.
- No es capaz de eliminar el error en régimen permanente.
- Los valores elevados de acción proporcional pueden provocar oscilaciones en la salida.
- A mayor acción proporcional, menor es el tiempo de subida, pero genera un aumento del tiempo de asentamiento.

Para un controlador con acción de control proporcional, la relación entre la salida del controlador u(t) y la señal de error e(t) es la siguiente:

$$u(t) = K_p e(t)$$

Aplicando la transformada de Laplace:

$$\frac{U(s)}{E(s)} = K_p$$

Donde K_p se denomina ganancia proporcional.

Acción de control integral:

La acción de control integral permite eliminar el error en régimen permanente. Su importancia es mayor a medida que pasa el tiempo y persiste la desviación. Aporta memoria al controlador. Por el contrario, valores elevados de la acción integral hacen el sistema más inestable, aumentando el tiempo de asentamiento y reduciendo el tiempo de subida.

En un controlador con acción de control integral, el valor de la salida del controlador u(t) cambia en el tiempo de forma proporcional a la señal de error e(t).

$$\frac{du(t)}{dt} = K_i\, e(t)$$
$$u(t) = K_i \int_0^t e(t)dt$$

Aplicando la transformada de Laplace:

$$\frac{U(s)}{E(s)} = \frac{K_i}{s}$$

Acción de control proporcional-integral:

Este modelo de control reúne las ventajas de las acciones de control proporcional e integral.

Se define mediante la siguiente ecuación:

$$u(t) = K_p e(t) + \frac{K_p}{T_i} \int_0^t e(t)dt$$

Aplicando la transformada de Laplace:

$$\frac{U(s)}{E(s)} = K_p \left(1 + \frac{1}{T_i s}\right)$$

Donde Ti se denomina tiempo integral.

Acción de control proporcional-derivativa:

La combinación de las acciones proporcional y derivativa resulta interesante para eliminar el rebose y mejorar la estabilidad del sistema. Para ello, es necesario predecir dicho rebose. Esto se consigue mediante la derivada de la señal de error. La acción derivativa se opone a las desviaciones respecto del valor de la señal de referencia. Como ventajas, mejora el amortiguamiento y reduce el rebose. No obstante, la acción derivativa resulta incapaz de detectar ni de corregir el error en régimen permanente puesto que este es constante y, por lo tanto, su derivada es nula. Posee otras desventajas como la ralentización del sistema, aumentando el tiempo de subida y reduciendo el de asentamiento, y la amplificación de las señales de alta frecuencia (ruido).

Se define mediante la siguiente ecuación:

$$u(t) = K_p e(t) + K_p T_d \frac{de(t)}{dt}$$

$$\frac{U(s)}{E(s)} = K_p(1 + T_d s)$$

Donde T_d, el tiempo derivativo, afecta en la señal del controlador junto con la variación de la señal de error y la ganancia proporcional.

En la función de transferencia se puede observar que posee un cero y ningún polo, lo que hace que este controlador sea físicamente irrealizable, si bien existen algunas alternativas.

Acción de control proporcional-integral-derivativa:

La combinación de las ventajas de las acciones de control proporcional, derivativa e integral da lugar a la siguiente ecuación que describe el comportamiento del controlador:

$$u(t) = K_p e(t) + \frac{K_p}{Ti}\int_0^t e(t)dt + K_p T_p \frac{de(t)}{dt}$$

Aplicando la transformada de Laplace se obtiene la función de transferencia:

$$\frac{U(s)}{E(s)} = K_p\left(1 + \frac{1}{T_i s} + T_d s\right)$$

Mediante este controlador es posible reducir el tiempo de subida y eliminar tanto el rebose como el error en régimen permanente.

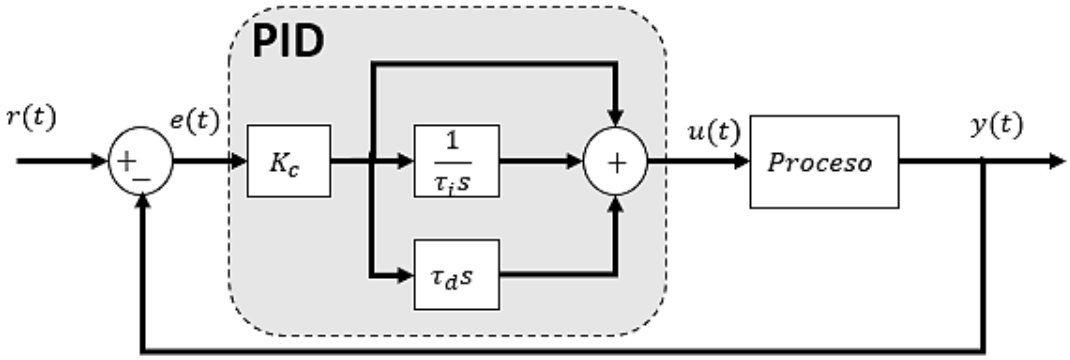


Fig. 15 Diagrama de bloques de un control PID.

Modulación por ancho de pulsos

La modulación por ancho de pulsos o PWM es una técnica ampliamente utilizada para controlar la cantidad de energía que se envía a una carga.

Consiste en modificar el ciclo de trabajo de una señal periódica, generalmente cuadrada.

El ciclo de trabajo se define como el tiempo en que la señal posee un valor positivo en relación a su periodo. Se expresa matemáticamente mediante la siguiente ecuación:

$$DC = \frac{t_{on}}{t_{on} + t_{off}} \cdot 100 = \frac{t_{on}}{T_c} \cdot 100$$

Siendo:

DC: ciclo de trabajo en porcentaje.

t_{on}: tiempo en el que la señal está en el nivel alto.

t_{off}: tiempo en el que la señal está en el nivel bajo.

T_c: tiempo de ciclo.

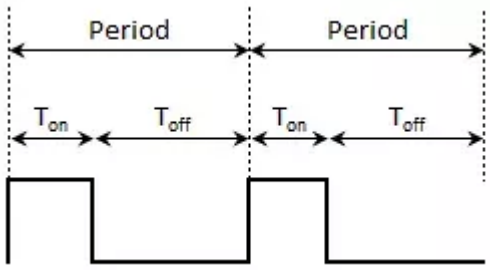


Fig. 16 Ciclo de trabajo.

De manera que modificar el ciclo de trabajo consiste en modificar el tiempo que la señal permanece en el nivel alto. Por lo tanto, a mayor ciclo de trabajo, mayor será la cantidad de energía transmitida a la carga.

La construcción típica de un circuito PWM se lleva a cabo mediante un comparador con dos entradas y una salida. A una de las entradas se le aplica una señal en dientes de sierra, mientras que a la otra se le aplica la señal moduladora. En la salida la frecuencia es igual a la de la señal dientes de sierra y el ciclo de trabajo varía en función de la portadora.

La importancia de esta técnica radica en la necesidad de poder general tensiones analógicas mediante dispositivos digitales, como es el caso de los microcontroladores. Mediante la modulación por ancho de pulsos es posible emplear una salida que únicamente puede tomar dos valores de tensión (nivel alto y nivel bajo) para generar una señal analógica, ya que al modificar la anchura del pulso se está modificando la tensión media.

3.1.3. Arduino

Arduino es una plataforma de código abierto basada en hardware y software libre. En cuanto al software, ofrece la plataforma Arduino IDE donde se pueden crear programas para las placas Arduino. En cuanto al hardware, es posible encontrar una gran variedad de placas. En la siguiente imagen se muestran algunas de ellas.

Fig. 17 Placas Arduino.

En general, todas las placas poseen una interfaz de entrada, lo que permite la conexión a la placa de diferentes tipos de periféricos tales como sensores, interruptores, teclados, etc. La información de estos se trasladará al microcontrolador integrado en la placa. De igual manera, poseen una interfaz de salida, encargada de transmitir la información procesada por en microcontrolador a otros periféricos, como por ejemplo pantallas, motores, altavoces … o incluso otras placas.

A continuación, se realiza una síntesis de las características técnicas de la placa Arduino Mega 2560.

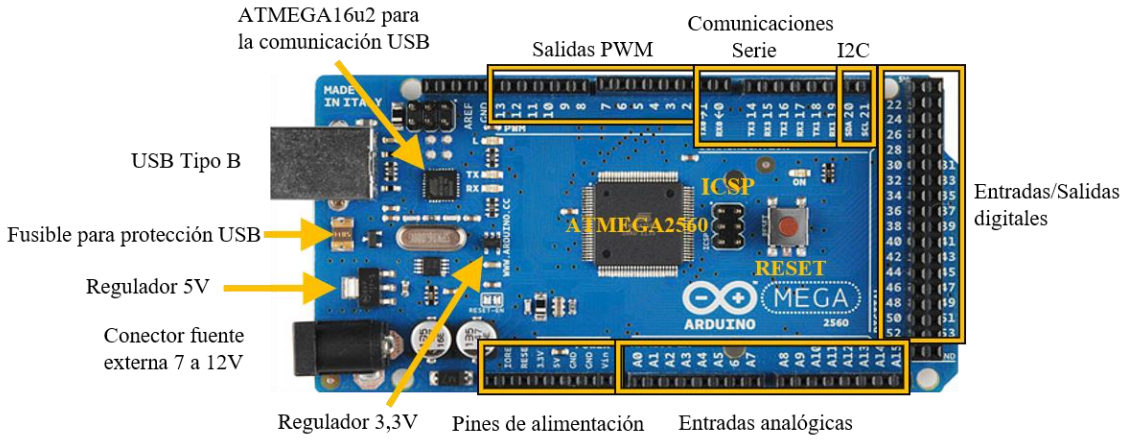


Fig. 18 Arduino Mega 2560.

Se trata de una placa basada en el microcontrolador ATmega2560. Dispone de 54 entradas y salidas digitales, de las cuales 15 pueden ser usadas como salidas PWM, 16 entradas analógicas, 4 puertos seriales por hardware (UART), un cristal de 16MHz, una conexión USB, un conector de alimentación, un conector ICSP para programación y un botón de reset.

Microcontrolador

Fabricado con tecnología CMOS de 8 bits y baja potencia, basado en la arquitectura RISC de AVR.

Alimentación

La placa Arduino Mega 2560 puede alimentarse a través de la conexión USB o con una fuente de alimentación externa, ya sea un convertidor CA/CC o una batería. El suministro externo de energía puede ser de 6V a 20V, aunque el rango de voltaje recomendado es de 7V a 12V.

Los pines de alimentación disponibles son los siguientes:

Vin: voltaje de entrada a la placa cuando se emplea una fuente de alimentación externa. Se puede alimentar la placa desde este pin o, si se utiliza el conector de alimentación, se puede acceder a la fuente de alimentación a través de este pin.

5V: suministra 5V desde el regulador de la placa. Suministrar energía a través de los pines de 5V o 3.3V evita el uso del regulador incorporado en la placa.

3.3V: fuente de 3.3V generada por el regulador de la placa. El consumo máximo de corriente es de 50mA.

GND: pines de referencia de tensión negativa.

IOREF: proporciona la referencia de voltaje con la que funciona el microcontrolador.

Memoria

La placa cuenta con una memoria FLASH de 256KB para almacenar el código del programa, de los cuales 8KB se utilizan para el gestor de arranque (bootloader), 8KB de SRAM y 4KB DE EEPROM.

Entradas y salidas

Los 54 pines digitales de la placa pueden funcionar como entradas o salidas. Cada pin funciona a 5V y puede proporcionar o recibir 20mA. La carga máxima admisible es de 40mA en periodos cortos de tiempo. Poseen una resistencia interna de pull-up de 20-50kΩ, desconectada por defecto.

Cuenta con 16 entradas analógicas, cada una de las cuales proporciona una resolución de 10 bits (1024 valores digitales diferentes). Por defecto, miden voltaje desde tierra (GND) a 5V, aunque es posible cambiar el extremo superior de su rango usando el pin AREF.

Pines con funciones especiales

Comunicación serial: se tienen cuatro puertos seriales correspondientes a los pines 0 y 1, 19 y 18, 17 y 16, 15 y 14 para la recepción (RX) y transmisión (TX) de datos serie respectivamente.

Interrupciones: hasta cinco interrupciones, en los pines 2, 3, 21, 20, 19 y 18. Es posible configurar estos pines para activar una interrupción en nivel bajo, en flanco ascendente o descendente, o un cambio en el nivel.

PWM: los pines 2 a 13 y 44 a 46 proporcionan una salida PWM de 8 bits.

SPI: los pines 50 (MISO), 51 (MOSI), 52 (SCK), 53 (SS) soportan la comunicación SPI utilizando para ello la librería SPI.

I2C: los pines 20 (SDA) y 21 (SCL) soportan la comunicación I2C utilizando la librería Wire.

Reset: se trata de un pin de entrada. Su conexión a GND reinicia el microcontrolador.

3.1.4. Comunicación serial

La comunicación serial permite el intercambio de información entre dos dispositivos digitales. Para ello emplea dos conexiones denominadas RX y RT destinadas a la recepción y transmisión de datos respectivamente. La serialización consiste en convertir un dato (byte) en un conjunto de pulsos que puedan ser recibidos y enviados por una única línea de transmisión. Su principal ventaja es la sencillez del protocolo de comunicación, si bien tiene desventajas como que solo se pueden comunicar dos dispositivos. Existen tres modos de comunicación: Full-duplex, Duplex y Simplex.

Full-duplex: es posible recibir y enviar información simultáneamente.

Duplex o Half-duplex: solo permite realizar una acción a la vez, la transmisión o la recepción.

Simplex: solo es posible recibir o transmitir.

Al módulo serial también se le conoce como UART.

A continuación, se muestran las conexiones básicas para dos sistemas con el mismo voltaje lógico.

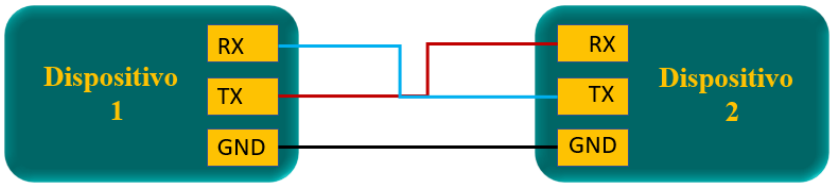


Fig. 19 Comunicación serial.

Protocolo

El protocolo serial opera mediante tres condiciones digitales básicas:

IT: inicio de transmisión.

P: paridad.

FT: fin de transmisión.

Estas condiciones son sincronizadas mediante un oscilador interno. La velocidad se mide en baudios.

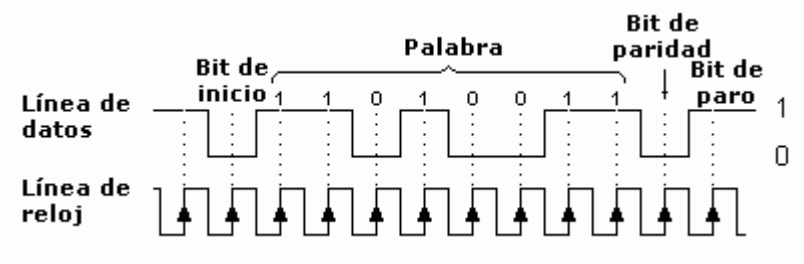


Fig. 20 Protocolo serie.

Estructura interna y configuración

Su estructura interna consiste en un generador de paridad, registros de desplazamiento, un oscilador variable, verificador de las tres condiciones y lógica de control. En la siguiente figura se muestra un diagrama de bloques general de un puerto serial.

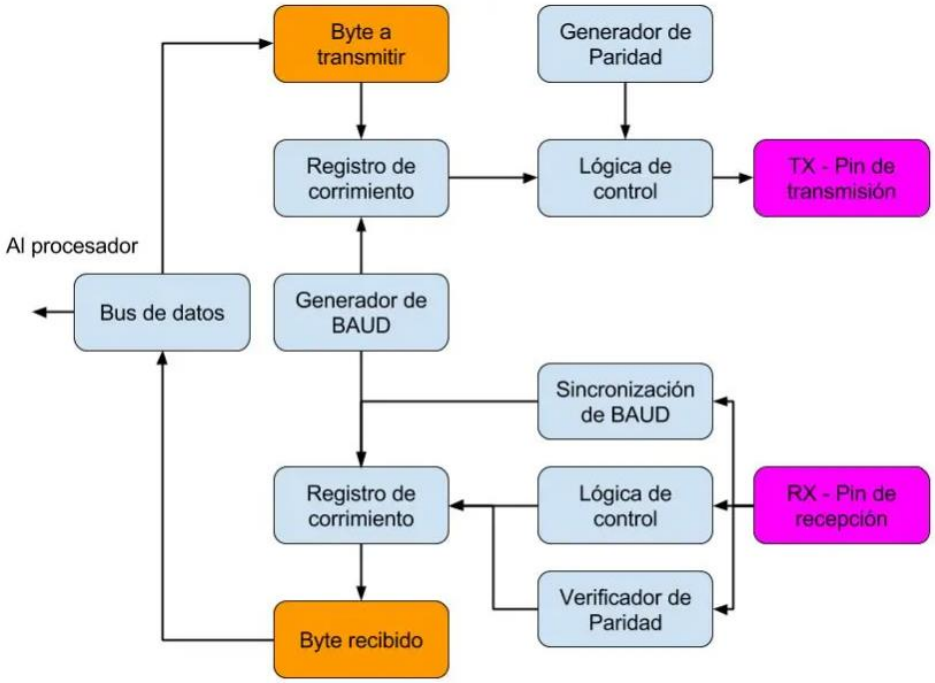


Fig. 21 Diagrama de bloques de la comunicación serial.

Tal y como se puede observar en el diagrama de bloques, los datos se transmiten a través del registro de desplazamiento. A su vez, la velocidad a la que se transmiten los datos depende del

generador de baudios. Este consiste en un divisor de frecuencia que genera la tasa de baudios de acuerdo con la siguiente ecuación:

$$Baud = \frac{f_{CK}}{16\ (UBRR + 1)}$$

Donde:

- Baud: tasa de baudios.
- f_{CK}: frecuencia del reloj.
- UBRR: registro de 8 bits (0-255).

Finalmente, la lógica de control se encarga de agregar los bits de inicio, paridad y fin de transmisión. En el caso de la recepción, el proceso sería inverso.

En cuanto a la configuración, es necesario elegir cuántos bits de parada y de inicio hay, si hay o no bit de paridad, y cuál es la velocidad de operación. Los baudios son una medida de los bits por segundo que se van a transmitir. Para que exista una sincronización de los datos, ambos dispositivos deben tener la misma configuración.

La trama de datos, es decir, los bits digitales necesarios para transmitir un byte de información, puede variar en función del número de bits de inicio, parada y paridad seleccionados. Por ejemplo, si se configura el puerto serial a 9600 baudios, 8 bits de datos, 1 bit de parada y sin bit de paridad, es necesario enviar 10 bits digitales por cada 8 bits de datos. De tal manera, que la velocidad real a la que se transmite la información útil es de 7680bits/s.

3.1.5. Comunicación I2C

La comunicación I2C, al igual que el caso anterior, consiste en el intercambio de bits de forma serial entre dispositivos digitales. En cuanto a las características, destaca la capacidad de conectar hasta 127 dispositivos esclavos con tan solo 2 líneas de comunicación: SDA y SCL. Se pueden alcanzar velocidades de 100kbits/s, 400kbits/s y 1000kbits/s. También es conocido como TWI.

El protocolo I2C es uno de los más utilizados para la comunicación con sensores digitales, ya que a diferencia del puerto Serial, su arquitectura permite tener una confirmación de los datos recibidos.

Las tramas enviadas mediante un puerto I2C, incluyen, además del byte de información, una dirección tanto del registro como del sensor.

Esquema de comunicación y elementos

En la comunicación I2C se diferencian dos elementos básicos: el maestro y el esclavo.

El maestro se encarga de controlar la señal de reloj y del inicio y parada de la comunicación. En cada puerto I2C solo puede existir un maestro. Puede funcionar como transmisor o como receptor.

A continuación, se indican las funciones correspondientes al maestro:

- Enviar 1 bit de inicio.
- Enviar los 7 bits de dirección.
- Generar 1 bit de lectura o escritura.
- Enviar 8 bits de dirección de memoria.
- Transmitir 8 bits de datos.

- Confirmar la recepción de datos.
- Confirmar la no recepción de datos.
- Enviar 1 bit de parada.

Los esclavos generalmente son sensores. Estos, suministran la información al maestro. Pueden actuar como emisor como receptor.

Sus funciones principales son las siguientes:

- Enviar información en paquetes de 8 bits.
- Enviar confirmaciones de recepción.

Las líneas SDA (Serial Data) y SCL (Serial Clock) realizan la transmisión de la información y la señal de reloj. También se conectan resistencias de pull-up de 2.2kΩ o 10kΩ para mantener las líneas a nivel alto cuando el puerto está disponible.

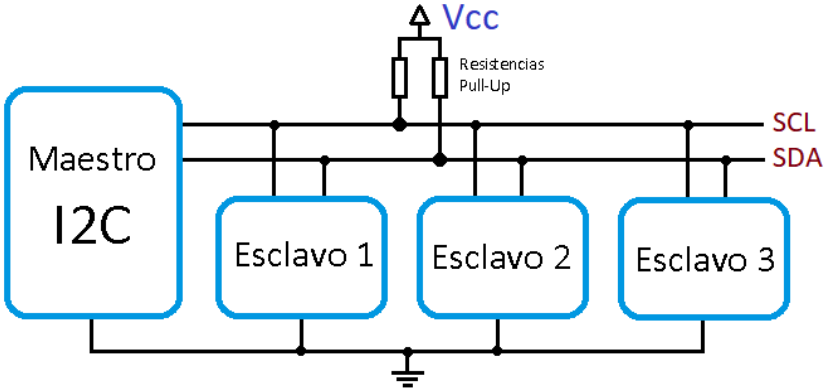


Fig. 22 Comunicación I2C.

Condiciones eléctricas

Los bits anteriormente mencionados (inicio, parada, recepción y no recepción) representan cambios en la tensión del bus I2C. A continuación, se describen las condiciones de voltaje correspondientes a cada bit:

- **Inicio**: este bit solo puede ser generado por el maestro. La condición de inicio se genera cuando, estando el bus disponible, la línea SCL está a nivel alto y la línea SDA pasa de nivel alto a nivel bajo.
- **Paro**: solo puede ser generado por el maestro. Se produce cuando hay un flanco de subida en la línea de datos estando la línea de reloj a nivel alto.

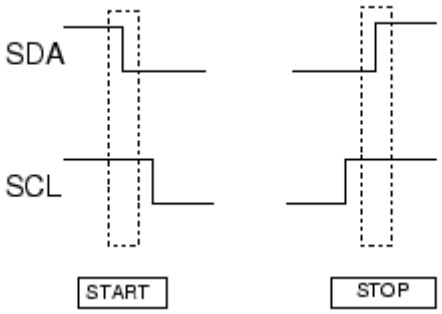


Fig. 23 Condiciones de inicio y parada.

- **ACK**: la confirmación de recepción puede ser transmitida por parte del maestro o del esclavo. Para ello, la línea SCL debe estar en nivel alto y la línea SDA en nivel bajo.
- **NACK**: la confirmación de no recepción solo puede ser generada por el maestro ya que se emplea cuando en una sola transmisión se quieren leer varios bytes de un esclavo. Cuando ya no se quieren leer más bytes, se utiliza el bit NACK. Para ello, al contrario que en el caso anterior, estando la línea SCL a nivel alto, la línea SDA también está a nivel alto.

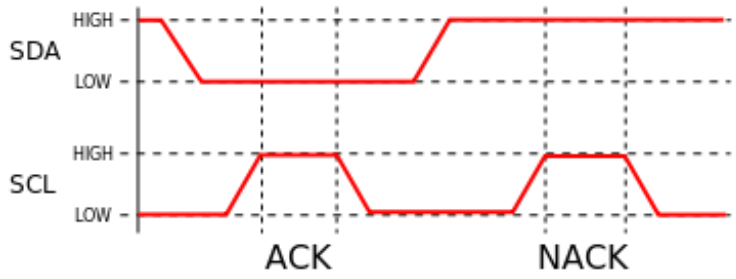


Fig. 24 Condiciones de recepción y no recepción.

Modos de comunicación

Los modos de comunicación en I2C se refieren a las distintas tramas que pueden formarse en el bus. Existen principalmente dos modos de comunicación:

- **Maestro-transmisor y esclavo-receptor:** se usa para configurar un registro del esclavo.

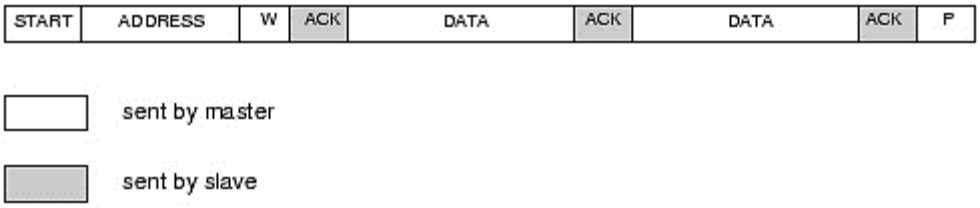


Fig. 25 Comunicación maestro transmisor.

- **Maestro-receptor y esclavo-transmisor:** se usa para leer información del esclavo.

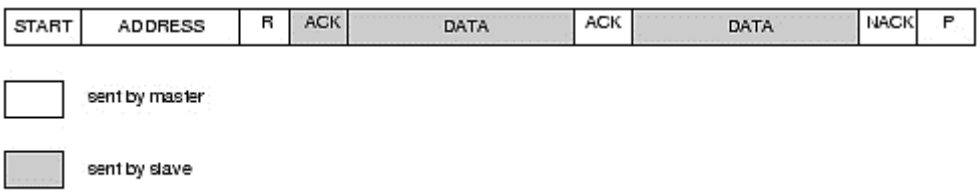


Fig. 26 Comunicación maestro receptor.

3.1.6. Desarrollo de aplicaciones en MATLAB

Entre la gran variedad de productos que ofrece MATLAB, existen varios especializados en el desarrollo de aplicaciones. Las aplicaciones se desarrollan en un lenguaje de programación propio. Este lenguaje permite operaciones de vectores y matrices, funciones y programación orientada a objetos.

App Designer

Se trata de una herramienta que permite crear aplicaciones. Cuenta con una librería de componentes visuales que facilitan el diseño de la interfaz gráfica, así como un editor de código integrado desde el cual programar el comportamiento de dichos componentes.

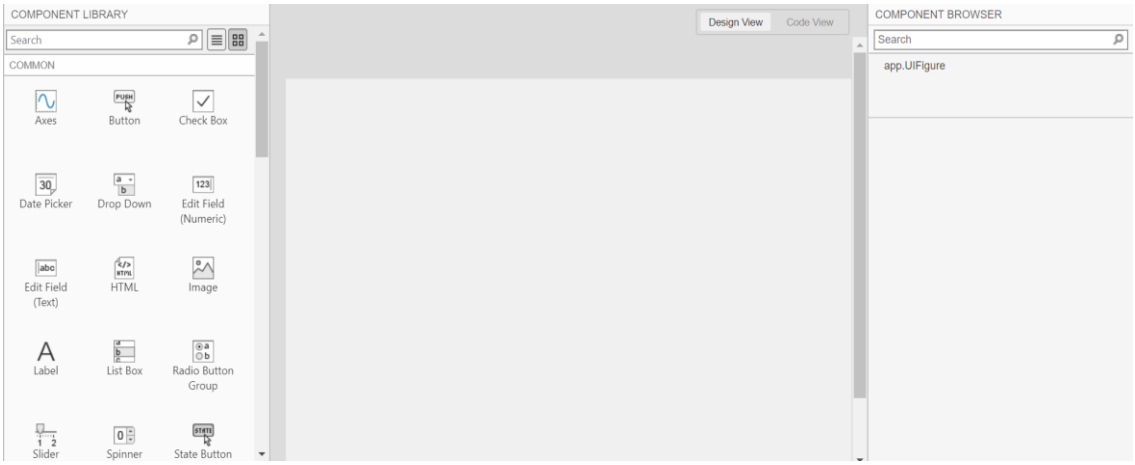


Fig. 27 Vista de inicio en App Designer.

Al iniciar App Designer, se observa en el centro la vista del panel frontal de la aplicación. A su izquierda, aparecen una serie de componentes predefinidos por Matlab. A la derecha del panel frontal, aparecen los componentes colocados en la aplicación. Desde esta ventana es posible modificar las propiedades de los componentes y crear callbacks.

En la esquina superior derecha del panel frontal, se selecciona entre dos vistas: la vista del diseño gráfico y la vista del código fuente.

En la librería de componentes se pueden encontrar componentes comunes como botones, casillas de verificación y listas desplegables. También se tienen elementos de control como medidores, indicadores luminosos o conmutadores.

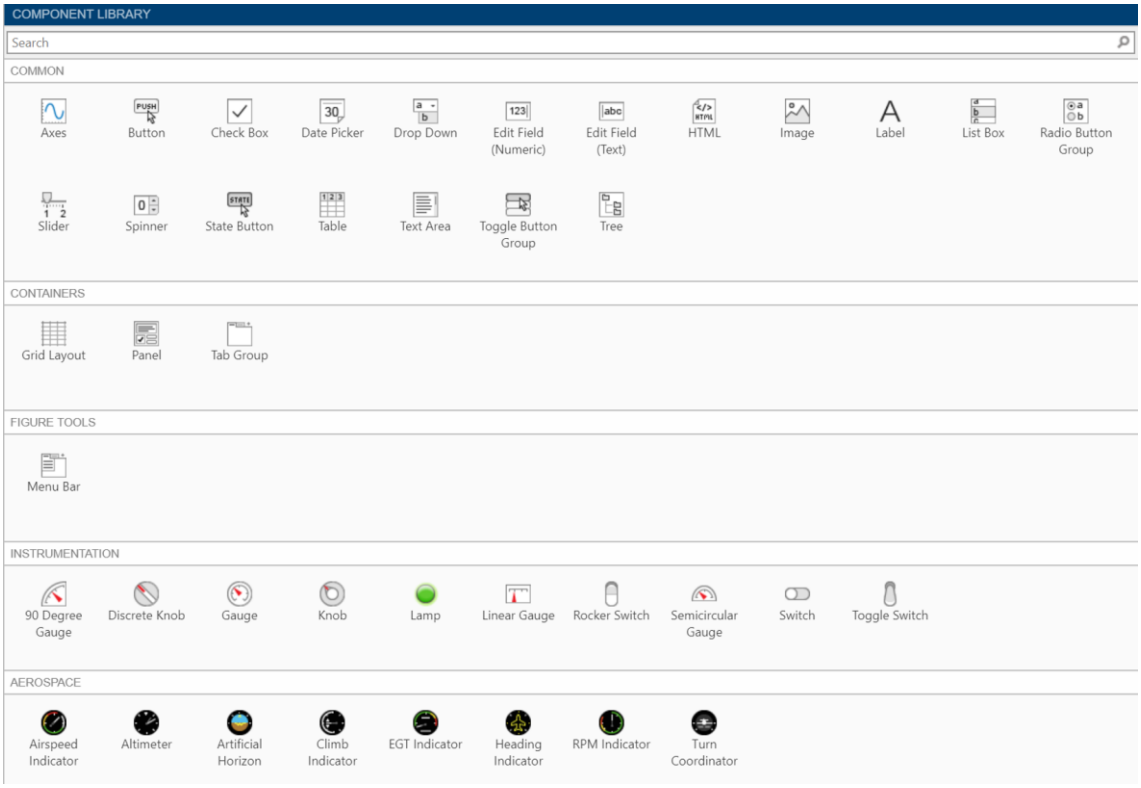


Fig. 28 Librería de componentes.

En cuanto al diseño de la interfaz gráfica, basta con arrastrar y colocar los componentes en el área de diseño. App Designer facilita la tarea ya que genera automáticamente parte del código donde se especifica la distribución y el diseño de la aplicación. No obstante, es posible modificar o añadir código para editar cada una de las propiedades de los objetos.

La interacción con el ratón o el teclado se gestiona a partir de callbacks que se ejecutan cuando la persona usuaria interactúa con la aplicación.

El editor de código permite definir el comportamiento de la aplicación. Además, cuenta con las comprobaciones automáticas de Code Analyzer que puede detectar errores de codificación y visualizar mensajes de advertencia a medida que se genera el código.

Al crear una aplicación, automáticamente se define una clase con el nombre de la aplicación. En ella, App Designer define las propiedades de acceso público correspondientes a los objetos colocados en el panel frontal de la aplicación. Además, genera una función denominada *createComponents* donde se definen todas las propiedades de dichos objetos, así como el lugar que ocupan en el panel frontal, el tamaño y tipo de letra, la imagen asociada, etc. También se definen dos métodos relacionados con la creación y el borrado de la aplicación, los cuales permiten gestionar el inicio y cierre de la misma.

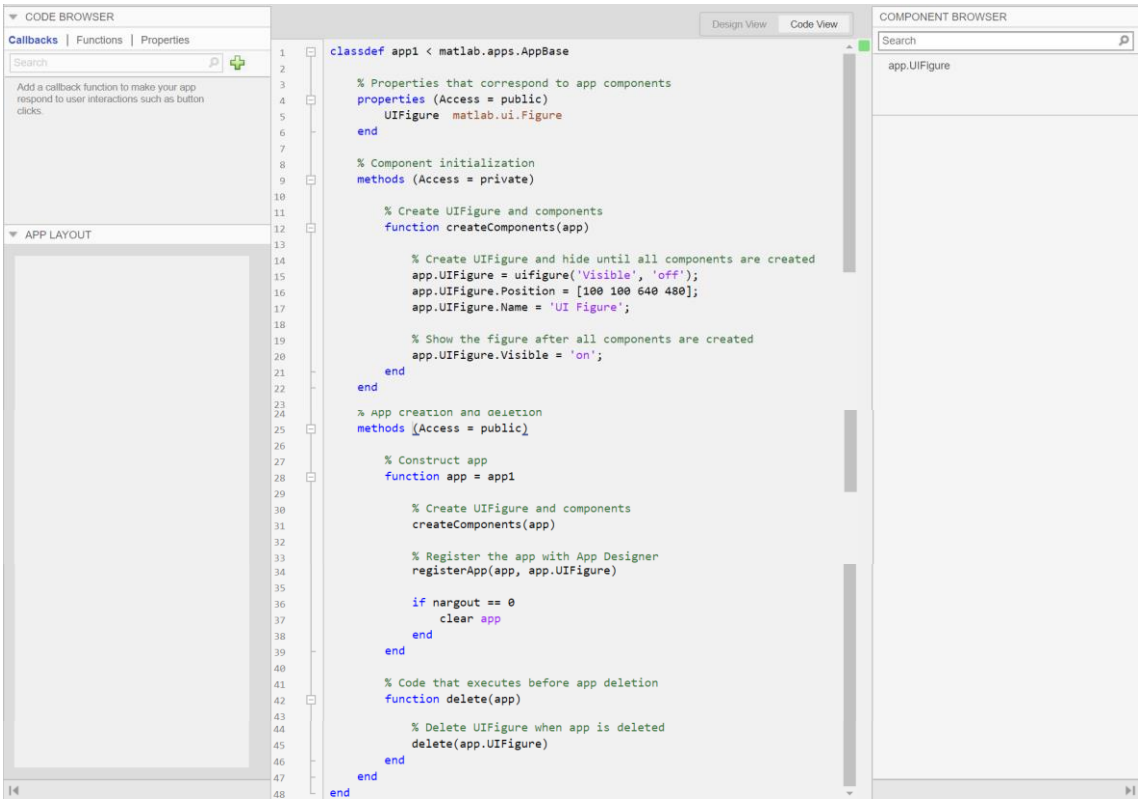


Fig. 29 Vista de código en app Designer.

En la vista de código, se mantiene la información del panel derecho. Sin embargo, en este caso el panel izquierdo está relacionado con la generación de código. Desde este se pueden generar callbacks, funciones o propiedades. Las callbacks son funciones relacionadas con la interacción de la persona usuaria. Es decir, están relacionadas con los componentes del panel frontal. Las funciones son estructuras internas del código que permiten mejorar la organización del mismo. Las propiedades son variables globales de la aplicación y, como tales, pueden ser compartidas entre las funciones o callbacks que la componen.

MATLAB Compiler

Permite crear aplicaciones independientes a partir de aplicaciones diseñadas en App Designer o ejecutables de línea de comandos.

Además de poder crear ejecutables independientes, también es posible empaquetar las aplicaciones como aplicaciones web interactivas. De esta manera, los usuarios finales pueden ejecutarlas desde el navegador sin necesidad de instalar software adicional.

MATLAB Web App Server

Gestiona el alojamiento de aplicaciones web. Los usuarios finales pueden ejecutar las aplicaciones web a través de una URL. Emplea protocolos de control de acceso y autenticación.

3.2. SELECCIÓN DE COMPONENTES

3.2.1. Control y adquisición de datos

En primer lugar, se ha seleccionado la placa de Arduino ya que se trata del elemento principal del equipo, encargado de la adquisición de los datos de los sensores, el control de los actuadores y la comunicación con el PC.

Dentro de las numerosas placas de Arduino, existen algunas dirigidas a fines más concretos, como es el caso de las placas Arduino Robot, LilyPad, Esplora, etc. Por otro lado, existen otros modelos más genéricos y por lo tanto más adecuados para proyectos como este.

Una de las primeras características a tener en cuenta a la hora de seleccionar una placa es el microcontrolador. Existen algunos modelos que ofrecen una gran potencia de cómputo gracias a microcontroladores de 32 bits. En este caso, es suficiente con los modelos que incluyen uno de 8 bits. Por lo tanto, dentro de los modelos genéricos, entre los que incluyen microcontroladores de 8 bits, destacan por ser los más populares las placas Arduino Uno, Leonardo y Mega2560.

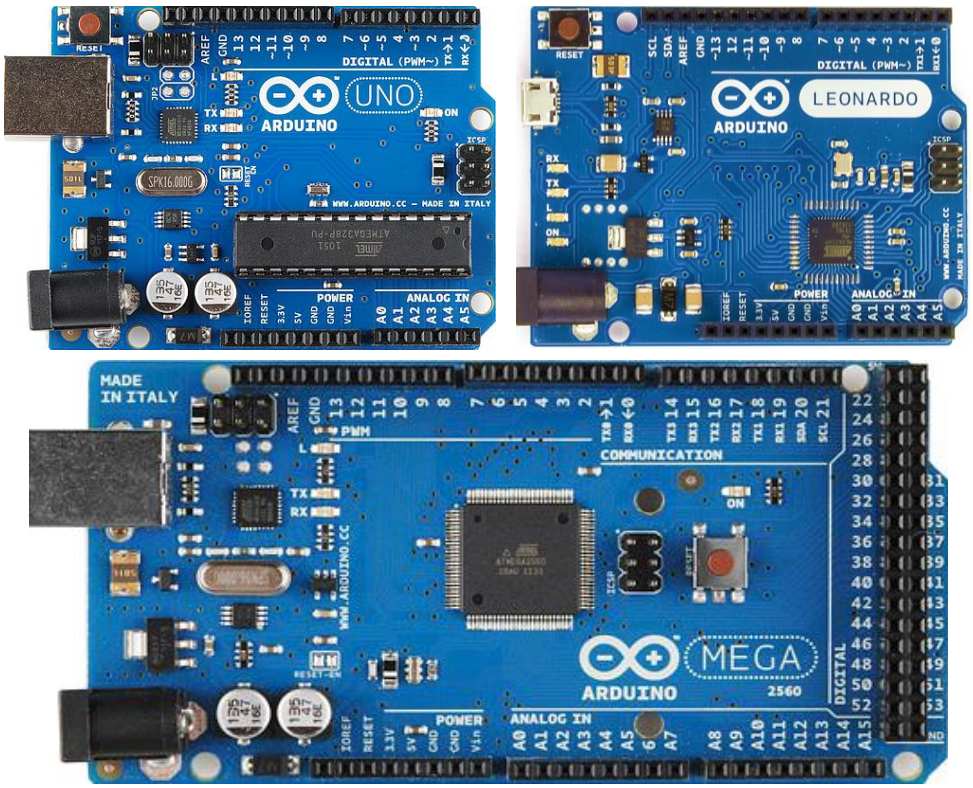


Fig. 30 Placas Arduino.

Estos modelos tienen algunas características en común. Todos ellos trabajan a una frecuencia de 16MHz, se alimentan a una tensión de 7 a 12V y su voltaje de operación es de 5V.

Sin embargo, son bastante diferentes en cuanto al número de pines de entrada/salida e interrupciones, y la posibilidad de comunicación que tiene cada uno.

Arduino UNO posee un microcontrolador ATmega328. Tiene 20 pines de entrada/salida disponibles, de los cuales 6 admiten salidas PWM. No obstante, cuenta con tan solo 6 entradas analógicas, lo que hace que esta placa sea insuficiente para este proyecto.

Arduino Leonardo contiene un microcontrolador ATmega32u4. Cuenta con 20 pines de entrada/salida, 7 de ellos admiten PWM y 5 pueden ser utilizados para realizar interrupciones. En cuanto a las entradas analógicas, cuenta con 6 específicas y 6 incluidas en los 20 pines digitales. Si bien esta cantidad de pines podría ser suficiente, es bastante limitada.

Arduino Mega ofrece una cantidad de pines considerablemente mayor en comparación con el modelo anterior, así como una mayor memoria FLASH y SRAM.

Por ello, se ha optado por la placa Arduino Mega, cuyas características se explicaron más detenidamente en el apartado anterior.

3.2.2. Sensores

PT100

La resistencia de platino se ha implementado por su presencia en numerosas aplicaciones industriales. Se ha seleccionado una PT100 genérica, cuyo coeficiente de temperatura es de 0,00385Ω/Ω/°C.

Fig. 31 PT100.

NTC

Para seleccionar la NTC a emplear se han realizado los cálculos correspondientes a la tensión de salida alimentando la NTC y su resistencia de linealización con una fuente de corriente de 1mA. Para ello se han empleado NTC de 1KΩ, 10KΩ y 100KΩ. En los resultados se ha observado que las tensiones obtenidas en el caso de 1KΩ eran muy pequeñas, mientras que las de 100KΩ superaban los 5V. Por ello, se decidió emplear una NTC de 10KΩ a 25°C.

El modelo seleccionado es una NTC de uso general en forma de disco, con una tolerancia de ±10%. Su factor de disipación térmico es de 5mW/°C y su índice de sensibilidad es de 4080K. Posee un tiempo de respuesta inferior a 3s.

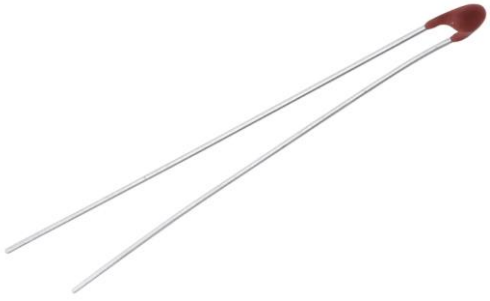


Fig. 32 NTC.

Termopar

El termopar seleccionado es un termopar tipo K. Posee una tolerancia de ±1,5℃ y precisa 0,7 segundos para alcanzar el 63% del valor final.

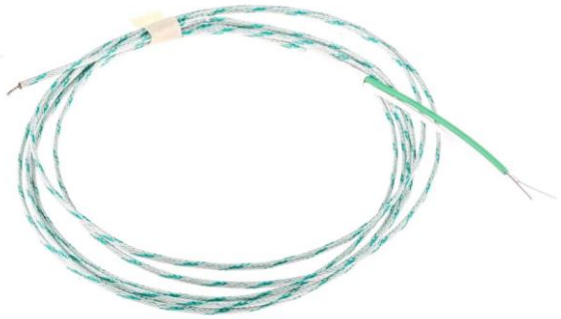


Fig. 33 Termopar.

AD590

Se ha seleccionado este sensor por las características que ofrece. Entre ellas, destacan su salida lineal que varía a razón de 1μA/K, su tolerancia de ±3℃ y su amplio rango de alimentación de 4V a 30V.

Fig. 34 AD590.

3.2.3. Actuadores

Ventilador

Los principales criterios que se han tenido en cuenta a la hora de seleccionar el ventilador han sido sus dimensiones y el rango de voltaje de alimentación.

Fig. 35 Ventilador.

Resistencia calefactora

En este caso las prioridades han sido la potencia y el tamaño del componente. Finalmente, se ha seleccionado un calentador de cartucho de 200W y con un alcance de hasta 450℃.

Fig. 36 Resistencia calefactora.

Pantalla LCD

Para poder mostrar la información en el panel frontal mediante un visor se ha seleccionado una pantalla LCD de 4 filas y 20 columnas de manera que de un solo vistazo se pueda ver la mayor cantidad de información posible. Además, se ha optado por un modelo con comunicación I2C ya integrada ya que se reduce significativamente la cantidad de cables necesarios para su conexión con la placa Arduino.

Fig. 37 Pantalla LCD.

3.3. DISEÑO ELECTRÓNICO

3.3.1. Diseño de los circuitos de acondicionamiento

Una vez seleccionados los sensores de temperatura, los actuadores y la placa Arduino, se procede al diseño de los circuitos de acondicionamiento. En el caso de los sensores, estos son los encargados de acondicionar su salida de manera que la señal se adapte a la entrada de los convertidores ADC del Arduino. Mientras que, en el caso de los actuadores, estos circuitos llevan a cabo el control de los mismos. Finalmente, se diseña también la alimentación de la placa Arduino Mega2560.

Se ha empleado el programa de software libre KiCad para el diseño de los esquemas.

Potencia

El circuito de potencia es el encargado de proporcionar la energía a la resistencia calefactora y realizar un control de la misma.

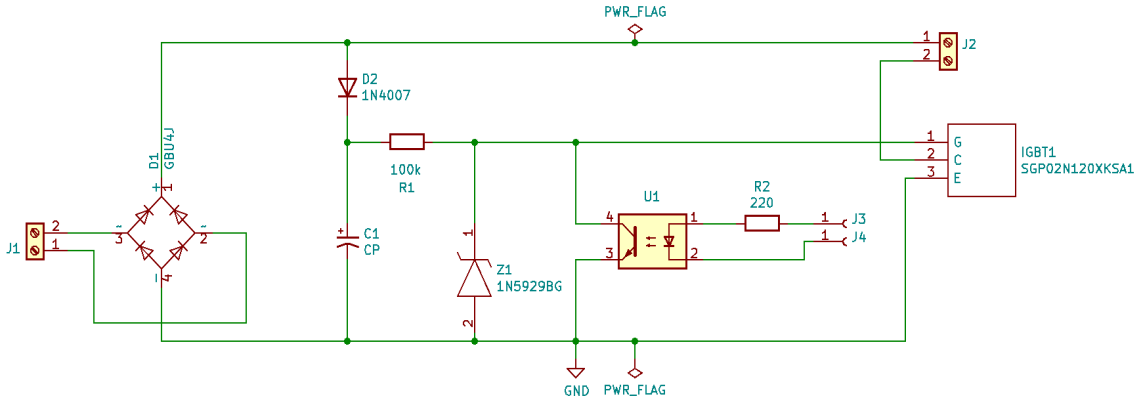


Fig. 38 Circuito de potencia.

El funcionamiento es el siguiente: el conector de la izquierda se conecta a la red. A continuación, se obtiene corriente continua mediante un rectificador de onda completa. El diodo protege el circuito de corrientes inversas, mientras que el condensador electrolítico mejora el rizado de la salida del rectificador. Mediante la resistencia elevada, de 100kΩ, se obtiene una pequeña corriente que circula a través del diodo Zener de 15V. Por otro lado, la resistencia calefactora está conectada al conector de la derecha y su conexión a la red depende del transistor IGBT. A su vez, la conmutación on/off de este transistor depende del control PWM realizado desde el Arduino. Este está conectado a los pines hembra del circuito. Cuando la señal PWM se encuentra a nivel alto, el fototransistor del optoacoplador cortocircuita el Zener, o lo que es lo mismo, V_{GS}=0V por lo que el IGBT no conduce corriente y la resistencia calefactora se encuentra desconectada. En cambio, cuando la señal PWM del Arduino se encuentra a nivel bajo, no circula corriente a través del fototransistor y por lo tanto V_{GS}=15V debidos al Zener. Este voltaje es el necesario para excitar el IGBT de forma que permita el paso de corriente y, por lo tanto, la resistencia calefactora se encuentre alimentada.

Control del ventilador

El control de la velocidad de giro del ventilador se lleva a cabo también mediante el control de la alimentación del mismo. Esto es, se emplea una señal PWM que, mediante un transistor bipolar, conecta o desconecta el ventilador de la alimentación en función de la tensión media que se desea aplicar. La alimentación necesaria varia proporcionalmente con la velocidad deseada. Para lograr que el ventilador gire a una mayor velocidad, la tensión media aplicada deberá ser mayor.

El diodo flyback protege al transistor de los picos de tensión generados por el motor durante el arranque y la parada.

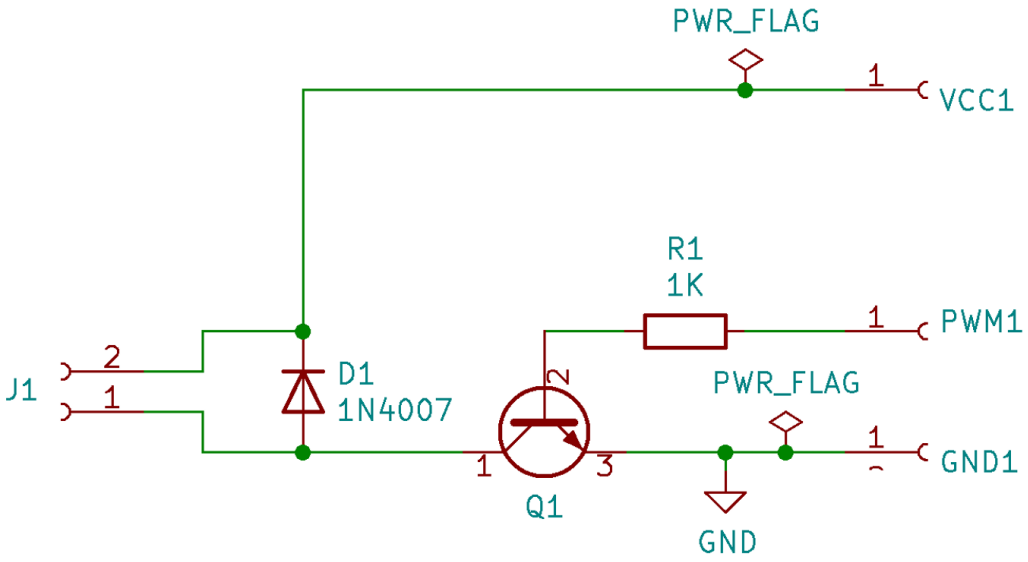


Fig. 39 Circuito de control del ventilador.

Alimentación del Arduino Mega 2560

Para alimentar la placa Arduino Mega 2560 se emplea un módulo de alimentación de Az-delivery.

Este módulo admite una entrada de corriente alterna de 100-240V_{AC} a 50-60Hz y ofrece una salida de corriente continua de 5V_{DC} y 3W.

Fig. 40 Módulo de alimentación.

Acondicionamiento del AD590

El circuito de acondicionamiento del sensor AD590 cumple tres objetivos:

- Convertir la corriente en tensión.
- Corregir el error de calibración.
- Amplificar la señal de salida.

Para convertir la corriente en tensión, basta con colocar una resistencia y medir la tensión en sus bornas. En este caso, con una resistencia de 1kΩ se puede convertir la señal de 1µA/K generada por el sensor en 1mV/K.

El error de calibración es un error constante que aparece cuando existe una diferencia entre la temperatura que marca el sensor y la temperatura real. Para corregirlo, se añade un potenciómetro que permite ajustar la función de transferencia actual de manera que coincida con la ideal. Una vez realizado el ajuste para una temperatura, queda ajustado para todo el rango de temperaturas.

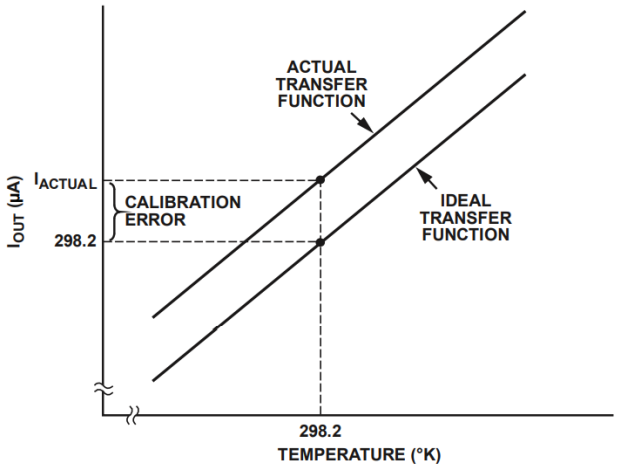


Fig. 41 Error de calibración.

Como se puede observar en la imagen, el circuito de acondicionamiento propuesto en las hojas de características del sensor cumple con los dos primeros objetivos.

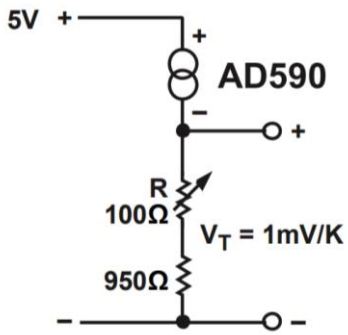


Fig. 42 Circuito de acondicionamiento propuesto.

Sin embargo, con este circuito a 150ºC la tensión de salida no llega a alcanzar los 0,5V. Con el fin de no perder precisión, se utiliza el rango de 0 a 5V disponible en las entradas de la placa

Arduino Mega 2560. Para ello, se emplea un amplificador no inversor de ganancia 11. Por último, se incluye un filtro paso bajo para eliminar las señales de alta frecuencia.

Finalmente, se muestra el circuito de acondicionamiento completo y los cálculos correspondientes.

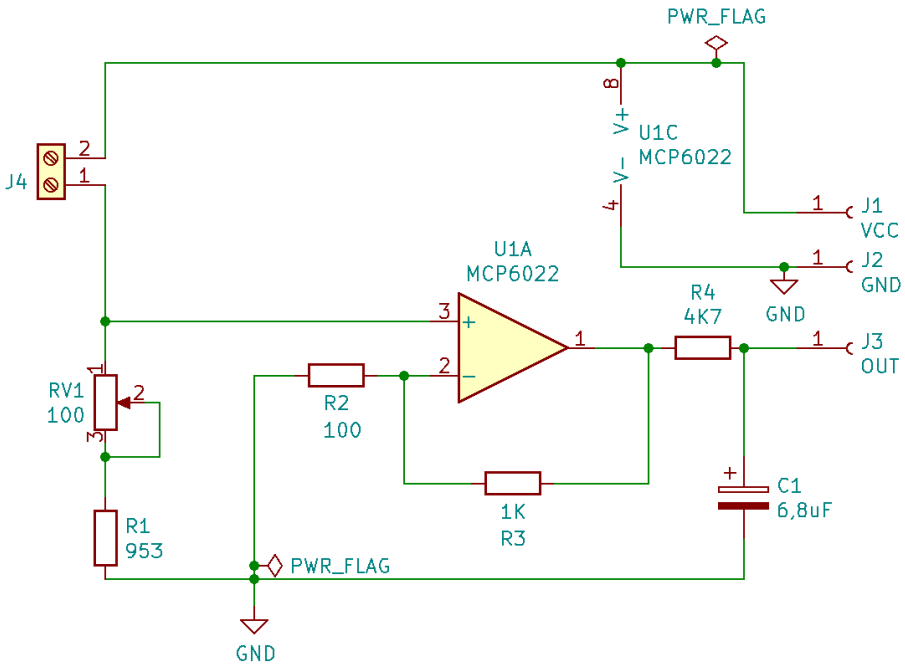


Fig. 43 Circuito de acondicionamiento del AD590.

Siendo V_T la señal correspondiente al AD590 que, conectado al terminal positivo del amplificador, varía según la siguiente expresión:

$$V_T = 1\text{mV/K}$$

Según el circuito, la salida del amplificador:

$$V_{o_{AMP}} = V_T \cdot T \cdot G$$

Siendo G ganancia del amplificador operacional.

Se desea obtener una salida de 5V cuando la temperatura del interior del horno alcance los 150ºC.

$$5V = 1mV/K \cdot (150 + 273)(K) \cdot G \cdot 10^{-3}$$

De donde se obtiene la ganancia del amplificador:

$$G = 11{,}82$$

Para un amplificador no inversor, la ganancia sigue la siguiente expresión:

$$G = 1 + \frac{R_3}{R_2} = 11{,}82$$

Para valores normalizados de resistencias, si R_3=1kΩ, entonces, R_2=100Ω y G=11. Finalmente, la tensión de salida del amplificador para cualquier valor de temperatura:

$$V_{o_{AMP}} = 1mV/K \cdot T \cdot 11$$

Acondicionamiento del termopar tipo K

Debido a que el voltaje de salida de los termopares tipo K es una señal muy pequeña ($40{,}4\mu V/°C$), es necesario amplificar la señal con una ganancia elevada. Además, como ya se mencionó en el fundamento teórico correspondiente a los termopares, para conocer la temperatura del termopar cuando la unión fría es distinta de 0ºC es necesario conocer también esta temperatura y realizar una compensación de la unión fría:

$$V_{T,UF} = V_{T,0°C} + V_{UF,0°C}$$

Es posible realizar la compensación de la unión fría empleando para ello otro sensor de temperatura, como una NTC o un LM35.

Analog Devices dispone de circuitos integrados especialmente diseñados para realizar el acondicionamiento de termopares. En concreto, el AD594 está calibrado para realizar el acondicionamiento de termopares tipo K.

El AD594 permite amplificar la señal de $40{,}4\mu V/°C$ del termopar a una de $10mV/ºC$. También realiza la compensación de la unión fría y rechaza el ruido en modo común de la conexión del termopar. Además, incluye una alarma para indicar la desconexión del termopar.

Al contrario que la salida del termopar, la salida del AD594 sí es lineal. Según la hoja de características, presenta la siguiente función de transferencia:

$$V_{o_{AD594}} = 247{,}3 \cdot (V_{T_J} + 11\mu V)$$

Donde 247,3 es la ganancia del amplificador y 11µV el offset de entrada.

El diseño del circuito realizado se basa en el circuito propuesto por la hoja de características en caso de emplear una fuente única de alimentación:

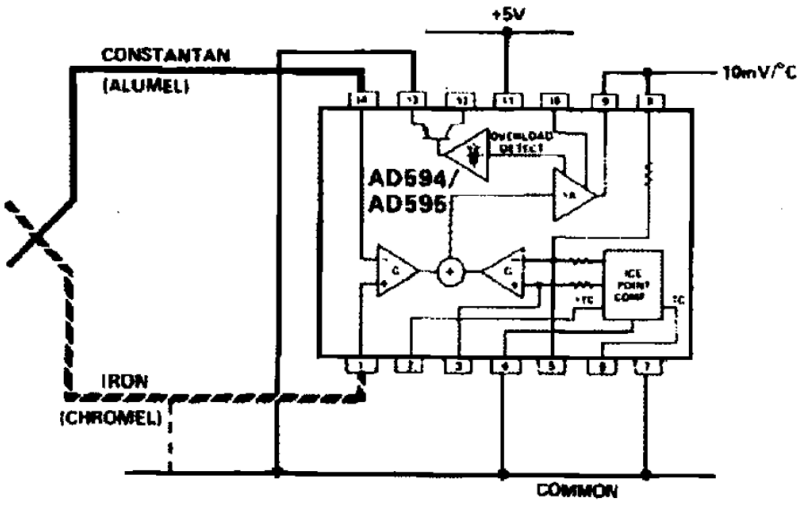


Fig. 44 Circuito propuesto del termopar tipo K.

Finalmente, se muestra el circuito de acondicionamiento diseñado:

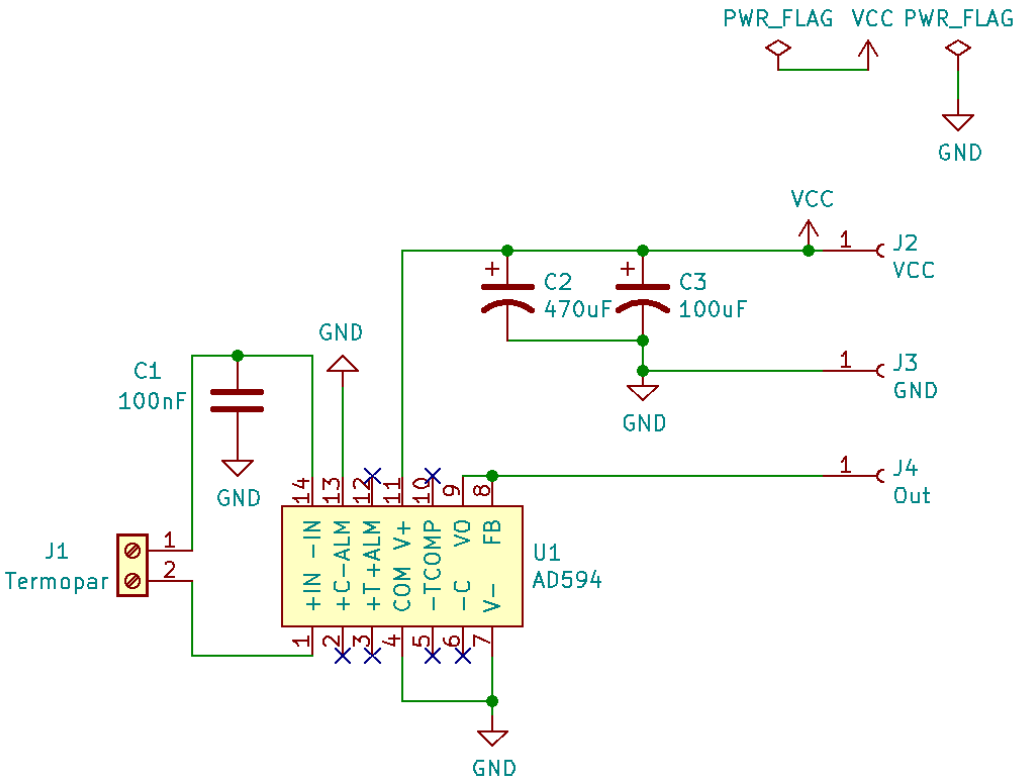


Fig. 45 Circuito de acondicionamiento del termopar tipo K.

Como se puede observar en el esquema del circuito, se han añadido condensadores para filtrar las señales de ruido de frecuencias altas o medias provenientes de las líneas de potencia y del propio termopar.

Acondicionamiento de la NTC

Para realizar el acondicionamiento de la NTC es necesario tener en cuenta dos factores importantes: el autocalentamiento y la linealidad.

La linealización de la NTC puede realizarse en el dominio analógico o en el digital. En el dominio analógico se tienen dos posibles circuitos: mediante un divisor resistivo o mediante paralelizado. Esto es, la linealización consiste en colocar una resistencia, ya sea en serie o en paralelo con la NTC. De esta manera, debido al carácter lineal de la resistencia de linealización, se obtiene una salida más lineal.

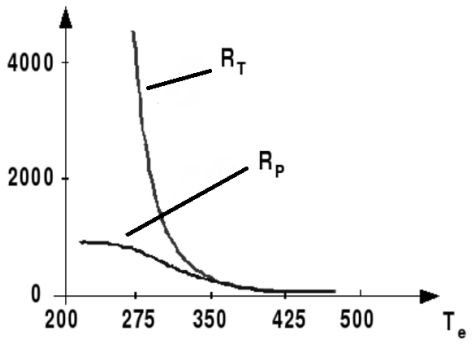


Fig. 46 Linealización NTC.

El autocalentamiento, por su parte, introduce errores en la medida. Por ello, deberá ser eliminado o reducido tanto como sea posible. Este está relacionado con la fuente de alimentación aplicada al circuito.

En primer lugar, se diseñará el circuito de linealización de la NTC. En este caso, se opta por una linealización por paralelizado.

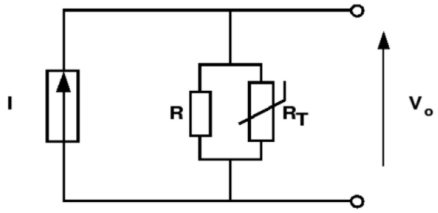


Fig. 47 Circuito de linealización.

Para ello, se debe hallar la resistencia que, en paralelo con la NTC, proporcione una linealidad óptima para un rango de temperaturas determinado. El método del punto de inflexión consiste en hacer coincidir el punto de inflexión de la curva de salida con la temperatura central del rango de temperatura seleccionado. De donde se obtiene la siguiente expresión para la resistencia de paralelizado:

$$R = \frac{\beta - 2T_C}{\beta + 2T_c} R_{T_c}$$

Donde:

- R es la resistencia de linealización.
- β es el índice de sensibilidad del termistor.
- T_C es la temperatura central del rango de temperatura.
- R_{TC} es la resistencia del termistor a la temperatura T_C.

Para un rango de temperaturas de 20ºC a 100ºC:

$$T_C = \frac{293K + 373K}{2} = 333K$$

$$R_{T_C} = R_0 \, e^{\beta\left(\frac{1}{T_C} - \frac{1}{T_0}\right)}$$

$$R_{T_C} = 10K\Omega \, e^{4080K \cdot \left(\frac{1}{333K} - \frac{1}{298K}\right)} = 2.37K\Omega$$

$$R = \frac{\beta - 2T_C}{\beta + 2T_c} \, R_{T_c}$$

$$R = \frac{4080K - 2 \cdot 333K}{4080K + 2 \cdot 333K} \cdot 2.37K\Omega = 1.7\text{K}\Omega$$

El valor de resistencia de linealización normalizado más próximo es de 1.74KΩ.

En segundo lugar, se diseña la fuente de alimentación del circuito. En el caso de la linealización por paralelizado, es habitual emplear una fuente de corriente. Tal y como se ha mencionado anteriormente, la importancia del diseño de la fuente de alimentación reside en su efecto sobre el autocalentamiento de la NTC.

Una mayor corriente de alimentación ofrece una mayor sensibilidad en la medida. Sin embargo, el autocalentamiento es también mayor, lo que afecta negativamente a la precisión de la medida.

Es por ello que se debe buscar un compromiso entre sensibilidad y precisión.

El incremento máximo de temperatura por autocalentamiento provoca la máxima disipación de potencia:

$$P_{max} = \Delta T \cdot \delta$$

Donde:

- P_{max} es la máxima potencia disipada.
- ΔT es el error máximo de temperatura permitido.
- δ es el factor de disipación térmico.

De otra manera, la potencia máxima disipada:

$$P_{max} = I^2 \cdot R$$

De la relación entre las dos ecuaciones anteriores se obtiene la máxima corriente aplicable:

$$\Delta T \cdot \delta = I^2 \cdot R$$

$$I_{max} = \sqrt{\frac{\Delta T \cdot \delta}{R_{max}}}$$

Para una precisión de 2ºC, se fija un error máximo por autocalentamiento diez veces menor, es decir, de 0,2ºC.

La resistencia máxima se obtiene cuando la temperatura del horno es mínima:

$$R_{NTC(20ºC)} = 10K\Omega\, e^{4080K\cdot\left(\frac{1}{293K}-\frac{1}{298K}\right)} = 12{,}632K\Omega$$

$$R_{max} = \frac{R \cdot R_{NTC}}{R + R_{NTC}} = \frac{1740\Omega \cdot 12632\Omega}{1740\Omega + 12632\Omega} = 926{,}64\Omega$$

$$I_{max} = \sqrt{\frac{0{,}2ºC \cdot 5 \cdot 10^{-3} W/ºC}{926{,}64\Omega}} = 1mA$$

Una vez seleccionada la alimentación y la resistencia de linealización, se diseña el circuito de acondicionamiento completo. Lo cual consiste en el diseño de la fuente de corriente de 1mA y el acondicionamiento de la salida para la lectura de los datos a través del Arduino Mega 2560.

A continuación, se muestra el esquema del circuito completo:

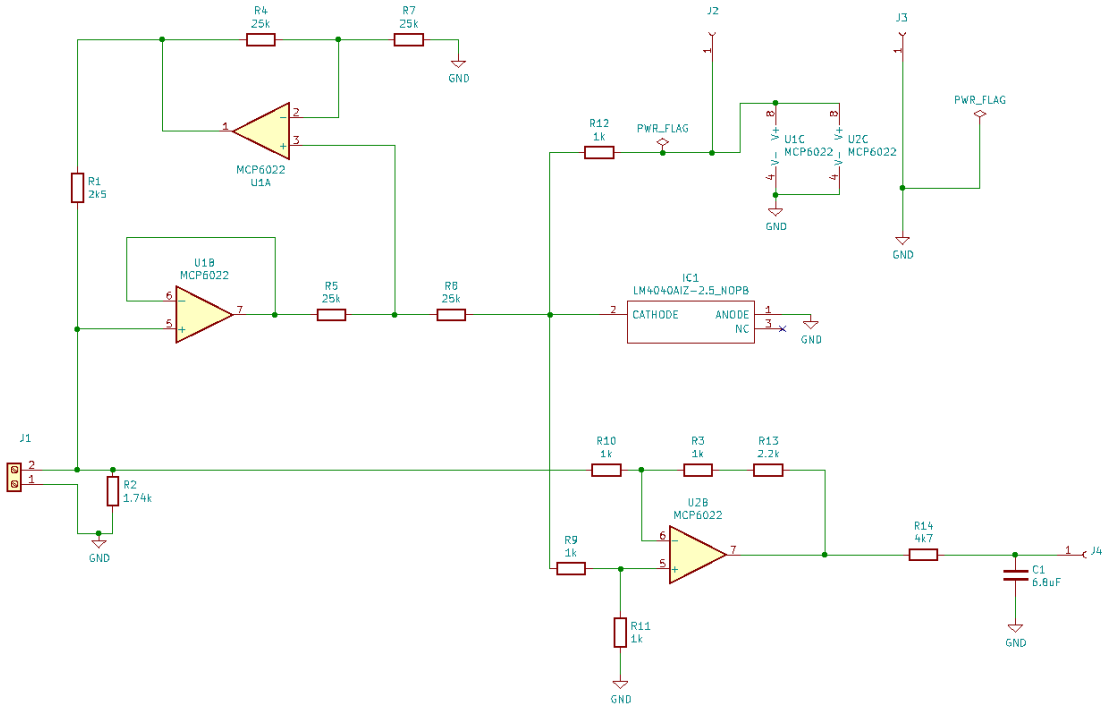


Fig. 48 Circuito de acondicionamiento de la NTC.

La fuente de corriente genera una intensidad constante en la resistencia R1 que viene determinada por la siguiente ecuación:

$$I = \frac{Vo_{U1A} - V^{+}_{U1B}}{R1}$$

Analizando la tensión de salida del amplificador U_{1A}:

$$Vo_{U1A} = \left(1 + \frac{25K}{35K}\right) V_{U1A}^{+} = 2V_{U1A}^{+}$$

Teniendo en cuenta que la corriente que atraviesa las resistencias R5 y R6 es la misma y que ambas tienen el mismo valor óhmico:

$$\frac{V_{U1B}^{+} - V_{U1A}^{+}}{R5} = \frac{V_{U1A}^{+} - 2.5V}{R6}$$
$$R5 = R6$$
$$2.5V = 2V_{U1A}^{+} - V_{U1B}^{+}$$

$$2.5V = Vo_{U1A} - V_{U1B}^{+}$$

Finalmente, se puede concluir que la corriente generada depende únicamente de la resistencia R1, por lo que se debe seleccionar el valor adecuado para una corriente de 1mA.

$$I = \frac{2.5V}{R1}$$
$$1mA = \frac{2.5V}{R1}$$
$$R1 = 2.5K$$

En este caso también se toma el rango de temperaturas posibles del horno de 0℃ a 150℃, aunque el rango para el cual la precisión de la NTC será mayor es de 20℃ a 100℃.

Se calcula la ganancia necesaria para amplificar la tensión $V_{o_{Rp}}$ en el rango de temperaturas:

$$R_{NTC(0℃)} = 10K\Omega\, e^{4080K\cdot\left(\frac{1}{273K} - \frac{1}{298K}\right)} = 35K\Omega$$
$$R_{NTC(150℃)} = 10K\Omega\, e^{4080K\cdot\left(\frac{1}{423K} - \frac{1}{298K}\right)} = 0{,}175K\Omega$$

$$V_{o_{Rp(0℃)}} = R_{p(0℃)} \cdot I = \frac{35K \cdot 1{,}74K}{35K + 1{,}74K} \cdot 1mA = 1{,}66V$$
$$V_{o_{Rp(150℃)}} = R_{p(150℃)} \cdot I = \frac{0{,}175K \cdot 1{,}74K}{0{,}175K + 1{,}74K} \cdot 1mA = 0{,}16V$$
$$G = \frac{5 - 0}{0{,}16 - 1{,}66} = -3{,}33$$

Para amplificar e invertir la señal de entrada $V_{o_{Rp}}$ se aplica en el terminal inversor del amplificador.

Finalmente, es necesario ajustar el rango de tensiones de salida. Para ello, se emplea un amplificador diferencial cuya ecuación de salida es la siguiente:

$$V_{O_{AMP}} = \frac{(R_3 + R_{13} + R_{10}) \cdot R_{11}}{(R_{11} + R_9) \cdot R_{10}} \cdot V_{ref} - \frac{R_3 + R_{13}}{R_{10}} \cdot V_{o_{Rp}}$$

De la ganancia anteriormente calculada se obtienen R_3, R_{13} y R_{10}:

$$\frac{R_3 + R_{13}}{R_{10}} = 3{,}33$$

Si R_{10} =1K y R_3=1K:

$$R_{13} = 2{,}2K$$

Finalmente, se ajusta el rango de tensiones de salida de manera que se tengan 5V a 150℃:

$$5V = \frac{4{,}2K \cdot R_{11}}{(R_{11} + R_9) \cdot 1K} \cdot 2{,}5V \ - 3{,}2 \cdot 0{,}16V$$

Si R_9=1K entonces R_{11}=1K.

Se obtiene la siguiente expresión:

$$V_{O_{AMP}} = 5{,}25 \ - 3{,}2 \cdot V_{o_{Rp}}$$

Se aplica la ley de corrientes de Kirchhoff y la ley de Ohm para expresar la tensión V_{ORp} en función de R_p.

$$\begin{cases} I = \dfrac{V_{O_{Rp}}}{R_P} + \dfrac{V_{O_{Rp}} - V^-}{R_{10}} \\ V^- = V^+ = \dfrac{V_{ref} - GND}{R_9 + R_{11}} \cdot R_9 = 1{,}25V \end{cases}$$

$$1mA = \frac{V_{O_{Rp}}}{R_P} + \frac{V_{O_{Rp}} - 1{,}25V}{1K\Omega}$$

Finalmente:

$$V_{O_{Rp}} = \frac{2{,}25R_p}{R_p + 1}$$

Por último, se obtiene la expresión de salida del circuito de acondicionamiento en función de R_p.

$$V_{O_{AMP}} = 5{,}25 \ - 7{,}2 \cdot \frac{R_p}{R_p + 1}$$

Acondicionamiento de la PT100

El circuito de acondicionamiento seleccionado para la PT100 consiste en la alimentación de esta por la fuente de corriente descrita en el apartado anterior, la compensación de la resistencia de los hilos de la PT100 y la amplificación de la salida adaptada al rango del convertidor analógico digital de la placa Arduino.

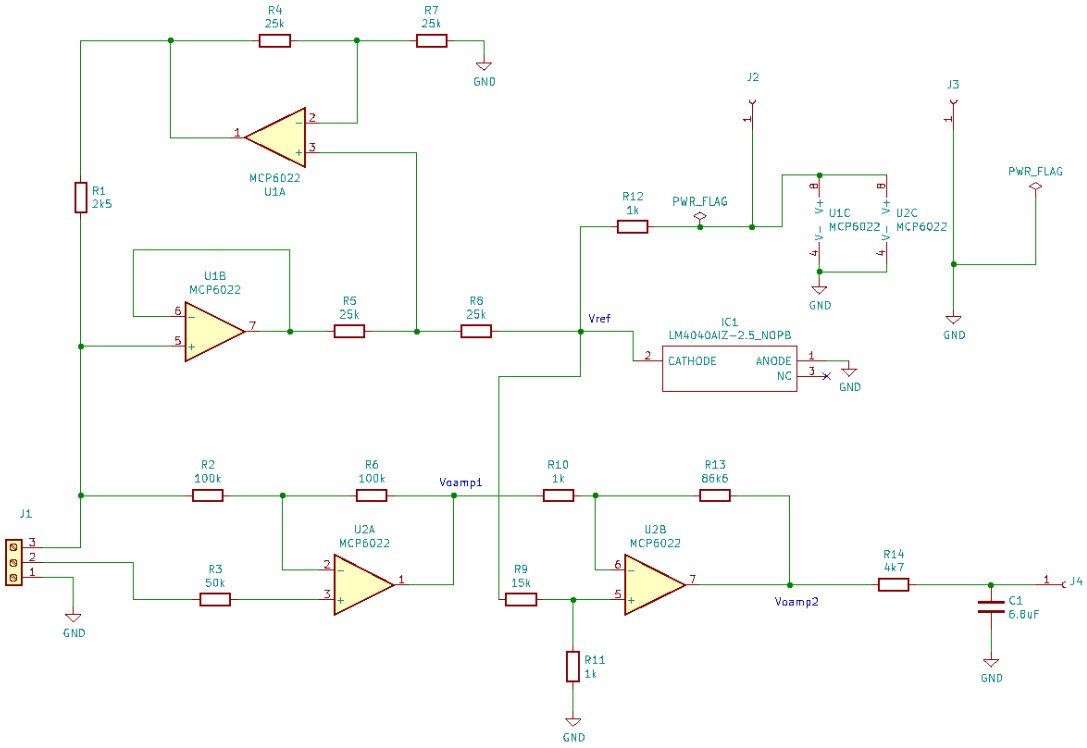


Fig. 49 Circuito de acondicionamiento de la PT100.

La resistencia de los hilos que conectan el sensor de temperatura y su circuito de acondicionamiento es especialmente importante cuando la distancia entre ellos es elevada ya que pueden introducir un error en la medida. Para evitar estos errores es necesario compensar estas resistencias.

Para la compensación de la resistencia de los hilos de la PT100 se ha empleado un amplificador operacional cuya función de transferencia es la siguiente:

$$V_{O_{AMP1}} = 2\,V^{+} - V^{-}$$

Teniendo en cuenta las resistencias de los hilos y suponiendo que estas poseen el mismo valor óhmico y que las corrientes de polarización de entrada de amplificador son nulas, se obtienen las señales de entrada del $V_{O_{AMP1}}$:

$$V^{+} = (R_h + R_{PT100}) \cdot I$$
$$V^{-} = (2R_h + R_{PT100}) \cdot I$$

Sustituyendo estos valores en la función de transferencia:

$$V_{O_{AMP1}} = 2 \cdot (R_h + R_{PT100}) \cdot I - (2R_h + R_{PT100}) \cdot I$$
$$V_{O_{AMP1}} = 2 \cdot R_h \cdot I + 2 \cdot R_{PT100} \cdot I - 2 \cdot R_h \cdot I - R_{PT100} \cdot I$$
$$V_{O_{AMP1}} = R_{PT100} \cdot I$$

Se deduce que la tensión de salida del circuito de compensación es independiente de las resistencias de los hilos de la PT100.

Por último, se ajusta la tensión de salida anterior para obtener el rango de tensión de lectura deseado:

Para ello, se calcula la ganancia necesaria para amplificar la tensión $V_{O_{AMP1}}$:

$$R_{PT100(T)} = R_O \cdot (1 + \alpha T)$$

$$R_{PT100(0℃)} = 100 \cdot \left(1 + 0{,}00385 \frac{\Omega}{\Omega℃} \cdot 0℃\right) = 100\Omega$$

$$R_{PT100(150℃)} = 100 \cdot \left(1 + 0{,}00385 \frac{\Omega}{\Omega℃} \cdot 150℃\right) = 158\Omega$$

$$V_{o_{PT100(0℃)}} = R_{PT100(0℃)} \cdot I = 100\Omega \cdot 1mA = 0{,}1V$$

$$V_{o_{PT100(150℃)}} = R_{PT100(150℃)} \cdot I = 158\Omega \cdot 1mA = 0{,}158V$$

$$G = \frac{0 - 5}{0{,}158 - 0{,}1} = -86{,}2$$

Finalmente, es necesario ajustar el rango de tensiones de salida. Para ello, se emplea un amplificador diferencial cuya ecuación de salida es la siguiente:

$$V_{O_{AMP2}} = \frac{(R_{13} + R_{10}) \cdot R_{11}}{(R_{11} + R_{15}) \cdot R_{10}} \cdot V_{ref} - \frac{R_{13}}{R_{10}} \cdot V_{O_{AMP1}}$$

De la ganancia anteriormente calculada se obtienen R_{13} y R_{10}:

$$\frac{R_{13}}{R_{10}} = 86{,}2$$

Si R_{10} =1KΩ, para valores normalizados:

$$R_{13} = 86{,}6K\Omega$$

Finalmente, se ajusta el rango de tensiones de salida de manera que se tengan 5V a 0℃:

$$5V = \frac{87{,}6K \cdot R_{11}}{(R_{11} + R_{15}) \cdot 1K} \cdot 2{,}5V - 86{,}6 \cdot 0{,}1V$$

Si R_{11}=1KΩ entonces R_{15}=15KΩ.

De modo que la señal a leer desde la placa Arduino sigue la siguiente expresión:

$$V_{O_{AMP2}} = 13{,}7V - 86{,}6 \cdot V_{O_{AMP1}}$$

3.3.2. Diseño de las placas de circuito impreso

Una vez diseñados los esquemas de los circuitos, se han diseñado las placas de circuito impreso para su posterior fabricación.

En esta tarea también se ha empleado el programa KiCad.

El procedimiento a seguir consiste en asignar la huella correspondiente a cada componente del esquema según se indique en la hoja de características. A continuación, se genera una lista de redes. Estas redes conectan los componentes entre sí según el esquema, e impiden que se realicen conexiones incorrectas. Con ello, en el editor de PCB de KiCad, denominado Pcbnew, se organizan los componentes de forma que se optimice el espacio requerido y la longitud de las conexiones. Una vez colocados los componentes, se procede al enrutado de las pistas. Para facilitar la fabricación de las PCB, se ha empleado una única capa de cobre y una anchura de pista de 2mm. Finalmente, se añade la zona de relleno conectada a la red GND.

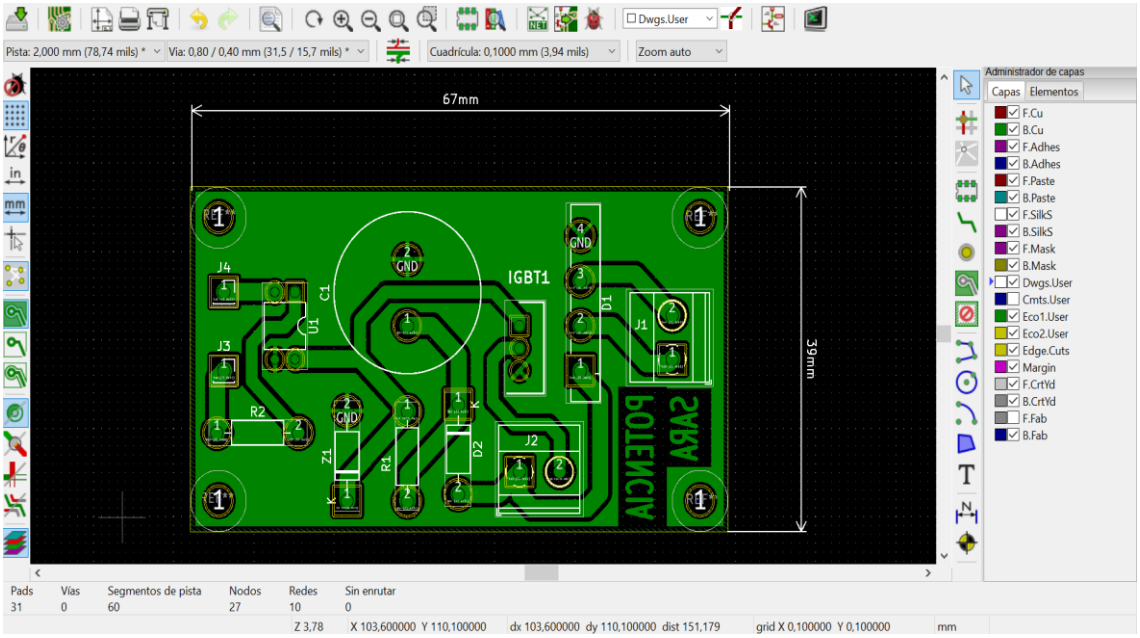


Fig. 50 Diseño de PCB en KiCAD.

A continuación, se muestra la placa de circuito impreso para el circuito de acondicionamiento de la NTC como ejemplo de los resultados obtenidos. El resto de PCB se pueden consultar en el Anexo II.

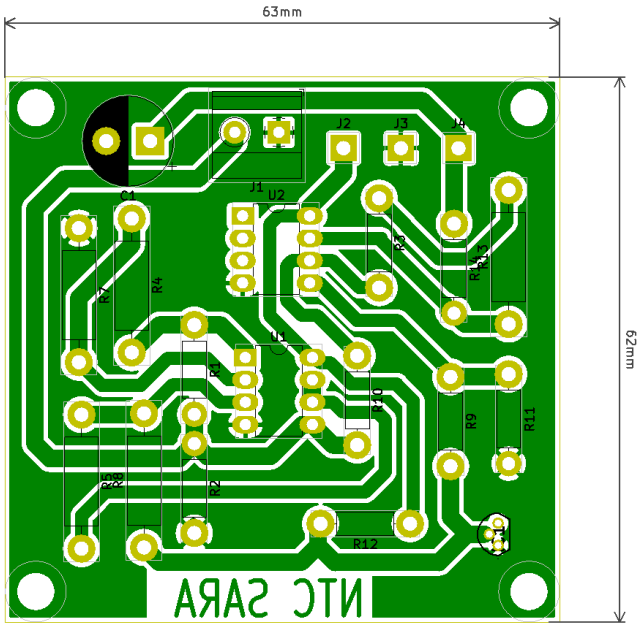


Fig. 51 Placa de circuito impreso de la NTC.

3.3.3. Fabricación de las placas de circuito impreso

Una vez diseñadas las placas de circuito impreso, se ha procedido a su fabricación. Después del tratamiento químico de las placas, se tiene la capa de cobre diseñada en Pcbnew lista para ser taladrada. El siguiente paso consiste en soldar los componentes. Una vez soldados, se han comprobado las conexiones con un multímetro.

A continuación, se muestran los resultados obtenidos para la PCB del circuito de acondicionamiento de la PT100.

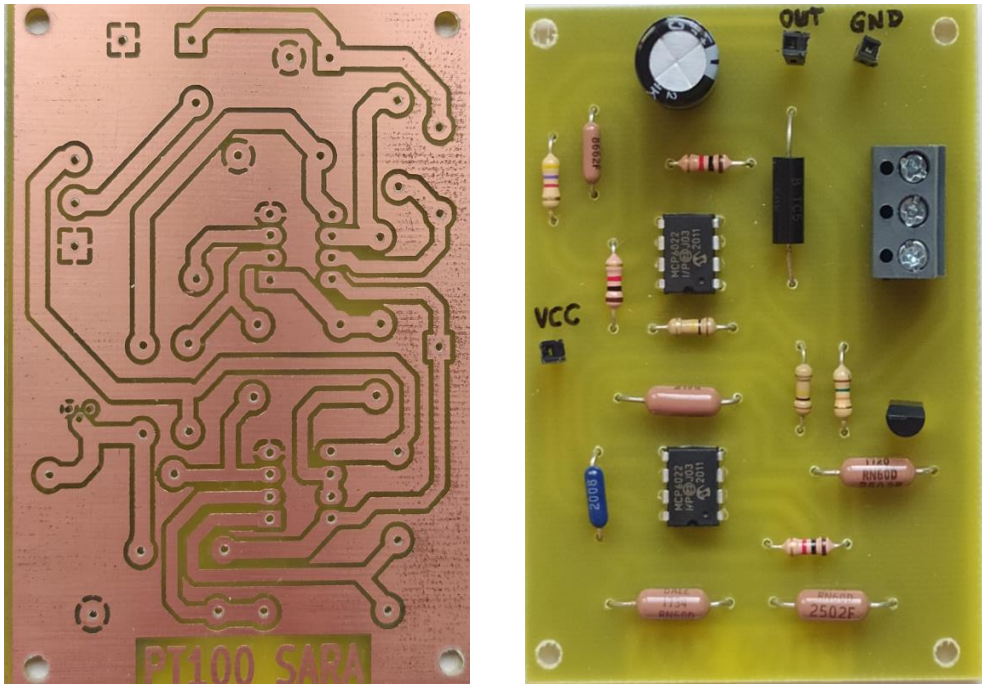


Fig. 52 Resultado final de la PCB de la PT100.

3.4. PROGRAMACIÓN DEL CÓDIGO FUENTE

3.4.1. Arduino

En este apartado se desarrolla el código a ejecutar en el microcontrolador ATmega2560 de la placa Arduino.

Este proyecto, realizado en el Entorno de Desarrollo Integrado de Arduino, consiste en tres archivos principales: Maqueta_Control_Temperatura.ino, Funciones.cpp y Funciones.h. También hace uso de la librería LiquidCrystal_I2C.

El programa Maqueta_Control_Temperatura.ino contiene el código principal. Este, como todos los programas de Arduino se divide en dos partes claramente diferenciadas por las funciones *setup* y *loop*.

La función *setup* solo se ejecuta una vez cuando se alimenta la placa, se carga un nuevo programa o se pulsa el botón de reset. Generalmente se emplea para inicializar pines y variables.

La función *loop* se ejecuta de forma cíclica. La velocidad de ejecución depende del microcontrolador. En el caso del ATmega2560, se tiene una frecuencia de reloj de 16MHz. Es decir, cada ciclo dura 62,5ns. Sin embargo, no todas las instrucciones se ejecutan en un único ciclo ya que pueden requerir varias operaciones y es necesario un ciclo para ejecutar cada una de ellas.

setup

```
void setup() {
  pinMode(pinIntModo,INPUT_PULLUP); //configura el pin como entrada con R pull-up
  pinMode(pinIntVentilador,INPUT_PULLUP);
  pinMode(pinSelec_AD590,INPUT_PULLUP);
  pinMode(pinSelec_Termopar,INPUT_PULLUP);
  pinMode(pinSelec_PT100,INPUT_PULLUP);
  pinMode(pinSelec_NTC,INPUT_PULLUP);
  pinMode(pinPWMVentilador,OUTPUT); //configura el pin como salida
  pinMode(pinPWM,OUTPUT);
  initFiltros(); //inicia el filtrado de temperaturas
  init_lcd(); //inicia el display LCD
  imprimir_saludo(); //imprime mensaje en pantalla
  imprimir_titulos(); //imprime los nombres en pantalla
  configTimerPWM(); //Configura el temporizador 5 para generar la señal PWM
  configTimer4Interrupt(); //Configura el temporizador 4 para realizar interrupciones
}
```

Fig. 53 Setup.

Uno de los cometidos de la función *setup* de este proyecto es configurar los pines de entrada y salida de la tarjeta Arduino Mega 2560. Esto solo es necesario en el caso de los pines digitales, ya que los pines analógicos por defecto son entradas. De manera que no es necesario configurar los pines destinados a la lectura de los sensores de temperatura ni de los potenciómetros. En el caso de los pines digitales, pueden trabajar como entradas o salidas por lo que debe ser indicado. En este caso, los pines digitales se emplean para los interruptores y el selector.

Para indicar el modo de funcionamiento de un pin se emplea la función *pinMode* incluida en la librería Arduino. En el primer argumento se indica el número del pin. En el segundo, el modo. Se tienen tres posibles modos: entrada (INPUT), salida (OUTPUT) y entrada con resistencia de pull-up (INPUT_PULLUP). En el último caso, el valor lógico leído por defecto es un 1 ya que la

entrada se encuentra conectada a nivel alto a través de una resistencia. Este modo se emplea para evitar la lectura de valores erróneos en estado de reposo provocados por el ruido.

Además, dentro de la función *setup* se llama también a otras funciones.

La función *initFiltros* es la encargada de inicializar los buffers empleados para el filtrado de las lecturas de temperatura.

Para interactuar con la persona usuaria se inicia la pantalla LCD mediante *init_LCD*. A través de *imprimir_saludo* se muestra un mensaje por pantalla. Seguido, se imprimen los títulos o nombres de los datos que posteriormente se muestran en el visor. Esta última tarea se realiza en la función *imprimir_titulos.*

En la función *configTimerPWM* se configura el temporizador 5 para generar la señal PWM periódica de frecuencia 500Hz.

De manera similar, la función *configTimer4Interrupt* es la encargada de configurar los registros del temporizador 4 con el fin de generar interrupciones cada 500ms.

Una vez realizadas estas tareas, el programa está listo para comenzar.

loop

La funcion loop contiene las instrucciones principales del programa que son ejecutadas de forma cíclica. Para facilitar la comprensión del funcionamiento del programa completo, se muestra a continuación el diagrama de flujo correspondiente a esta función.

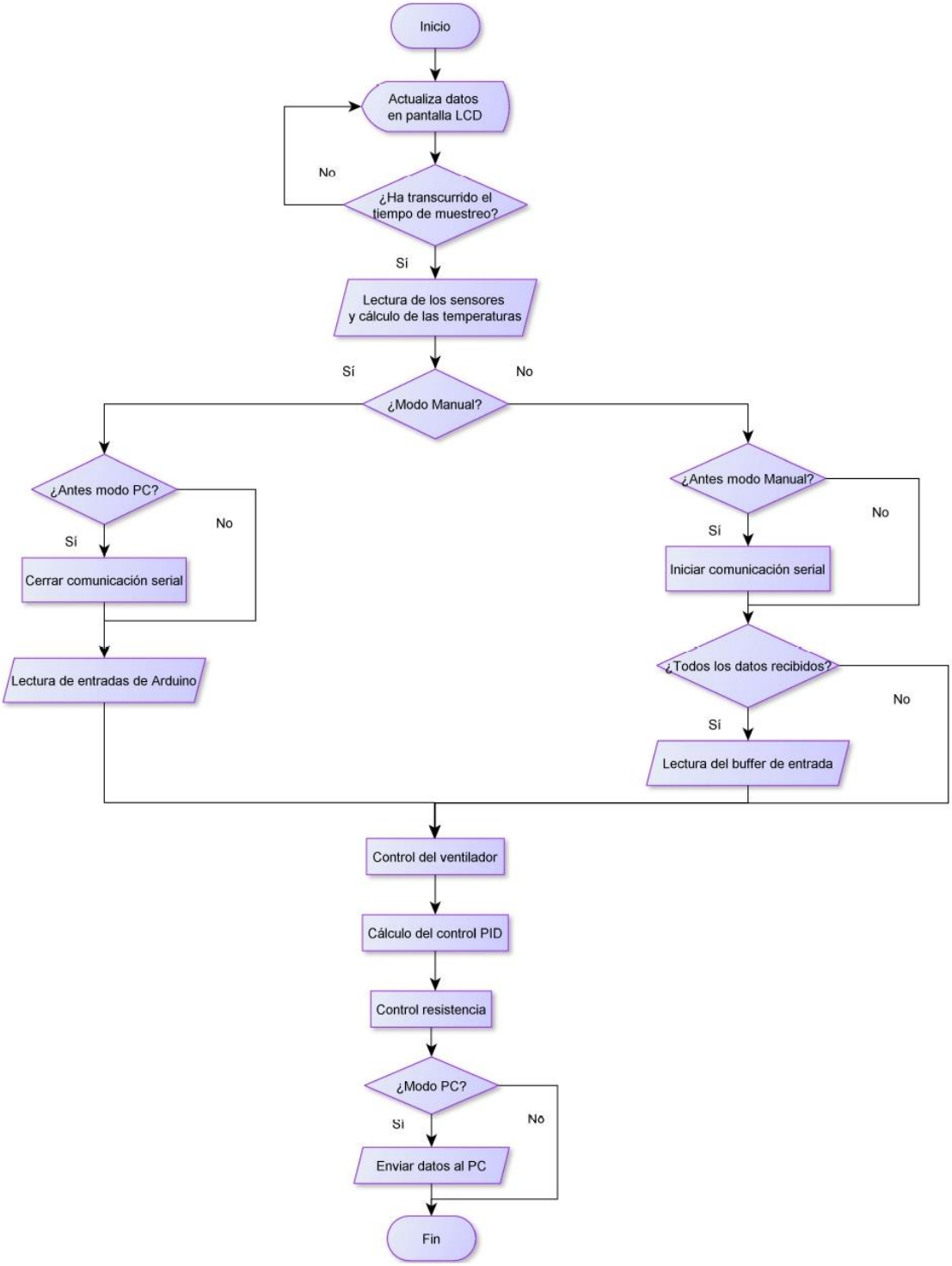


Fig. 54 Diagrama de flujo de la función loop.

Este diagrama de flujo da una idea del funcionamiento general del programa. A continuación, se analizan más detalladamente las instrucciones de la función loop.

```
void loop() {
  //Actualiza los datos en pantalla
  imprimir_temp();
  imprimir_consig();
  imprimir_error();
  imprimir_modo();
  imprimir_Kp();
  imprimir_Ki();
  imprimir_Kd();
  imprimir_DC();
  if(Tmuestreo==1){ //Flag interrupción
    //Detiene el contador
    TCCR4B = (0<<ICNC4)|(0<<ICES4)|(0<<5)|(0<<WGM43)|(1<<WGM42)|(0<<CS42)|(0<<CS41)|(0<<CS40);
    TIMSK4 = (0<<ICIE4)|(0<<OCIE4B)|(0<<OCIE4A)|(0<<TOIE4);//Deshabilita la interrupción
    if(Temperatura > TempMax){ //La temperatura supera el límite
      AlertaEnfriamiento(); //Enfriamiento forzado
    }

    Calc_Temp_ad590(pinAD590);
    FiltradoAD();
    Calc_Temp_ntc(pinNTC);
    FiltradoNTC();
    Calc_Temp_termopar(pinTermopar);
    FiltradoK();
    Modo=digitalRead(pinIntModo); //Lectura del interruptor de Modo
    if(Modo==1){//Modo Manual
      if(inicio_PC==1){ //Estaba en modo PC
        inicio_PC=0; //Limpia el flag
        Serial.end(); //Cierra la comunicación serial
      }
      Lect_Entradas_Manual(); //Lectura de las entradas del panel frontal
    }else{//Modo PC
      if(inicio_PC==0){ //Estaba en modo Manual
        Serial.begin(115200); //Inicia comunicación serial
        inicio_PC=1; //Activa el flag del modo PC
      }
      if (Serial.available() >= 6) { //Se han recibido todos los datos
        Lect_Entradas_PC(); //Lectura del buffer de entrada
      }
    }
    Control_Ventilador(); //Controla el ventilador
    Control_PID();//Calcula el control PID
    Resistencia_PWM(); //Aplica el control PID
    if(Modo==0){ //Modo PC
      Enviar_Datos_PC(); //Envía los datos al PC
    }
    Tmuestreo=0; //Resetea flag interrupción
    TCNT4 = 0; //Resetea el contador
    //Activa el contador
    TCCR4B = (0<<ICNC4)|(0<<ICES4)|(0<<5)|(0<<WGM43)|(1<<WGM42)|(0<<CS42)|(1<<CS41)|(1<<CS40);
    TIMSK4 = (0<<ICIE4)|(0<<OCIE4B)|(1<<OCIE4A)|(0<<TOIE4);//Habilita la interrupción
  }
}
```

Fig. 55 Loop.

En el *loop* se llaman a las distintas funciones de impresión de manera que continuamente se refrescan los valores de los datos que se muestran en la pantalla.

Dado que la subrutina de atención a la interrupción tiene un espacio limitado de memoria reservada, todas las tareas a realizar una vez transcurrido el tiempo de muestreo se realizan en el loop. Para ello, se emplea el flag Tmuestreo. Su valor cambia a 1 cada 500ms a través de la interrupción, tal y como se verá más adelante. De esta manera se asegura que la lectura de los

sensores y el control de los actuadores se realiza periódicamente siendo este intervalo de tiempo conocido e igual al tiempo de muestreo.

En primer lugar, se detiene el contador y se deshabilita la interrupción para que la ejecución de estas instrucciones no sea interrumpida. En caso de que en el ciclo anterior la temperatura superase la máxima establecida, se ejecuta la función *AlertaEnfriamiento.* A continuación, se procede a la lectura de los sensores y el cálculo de las temperaturas. Cada una de las lecturas es filtrada de manera que se eliminan las variaciones instantáneas en la entrada. Seguido, comienza la lectura del resto de las entradas. Para ello, se comienza con la lectura del interruptor del modo de funcionamiento ya que el modo seleccionado condiciona la lectura de las demás variables de entrada. En caso de tratarse del modo manual, se comprueba si el estado anterior era el modo PC. De ser así, se cierra la comunicación serial con el ordenador mediante el método *end* y se guarda el estado actual. En el modo manual se realiza la lectura de las entradas situadas en el panel frontal del horno. Por el contrario, en el modo PC se comprueba si el estado anterior era el modo manual ya que, en ese caso, es necesario iniciar la comunicación serial mediante el método *begin* y guardar el nuevo estado de funcionamiento. La velocidad establecida para la comunicación serial es de 115200 baudios. Antes de proceder a la lectura de las entradas, que en este caso se realizará desde el buffer serial, se comprueba que en el buffer existan al menos los seis datos necesarios. Para ello se emplea el método *available,* el cual devuelve el número de bytes disponibles para leer del buffer de entrada. Una vez almacenadas las entradas, se realiza el control de los actuadores en base a estas. Para ello, se emplean las funciones *Control_Ventilador, Control_PID* y *Resistencia_PWM*. En el modo PC, se realiza el envío de los datos a través de la comunicación serial. Finalmente, se resetea y activa el contador y se habilita la interrupción.

En la cabecera Funciones.h se encuentran las declaraciones de las funciones que se definen en el archivo Funciones.cpp. También se han definido en el archivo Funciones.h los valores constantes mediante #define. De esta manera, se asocia un nombre a estas constantes y no ocupan espacio en la memoria de programa del microcontrolador ya que en la compilación se sustituyen los nombres por el valor asignado. Al comienzo de las cabeceras se encuentra la condición #ifndef #define la cual indica que, si no se ha definido previamente la cabecera, debe definirse. La definición finaliza tras el #endif.

Finalmente, se analizan las funciones vistas anteriormente cuyas definiciones se encuentran en el archivo Funciones.cpp.

configTimer4Interrupt

La función, como se ha comentado anteriormente, tiene la tarea de configurar el temporizador 4 para realizar una interrupción cada 500ms.

El temporizador 4, de 16 bits, es configurable a través de los registros: TCCR4A, TCCR4B, OCR4A, TCNT4 y TIMSK4.

Para realizar interrupciones se emplea el modo CTC. En este modo, cuando el valor del contador indicado en el registro TCNT4 coincide con el valor almacenado en el registro OCR4A, se resetea el contador. El registro TCNT4 indica el valor del contador en cada momento. En el registro OCR4A se almacena el valor máximo que alcanza el contador. De forma que, cuanto mayor sea el valor almacenado en OCR4A, mayores serán el periodo y la resolución del contador.

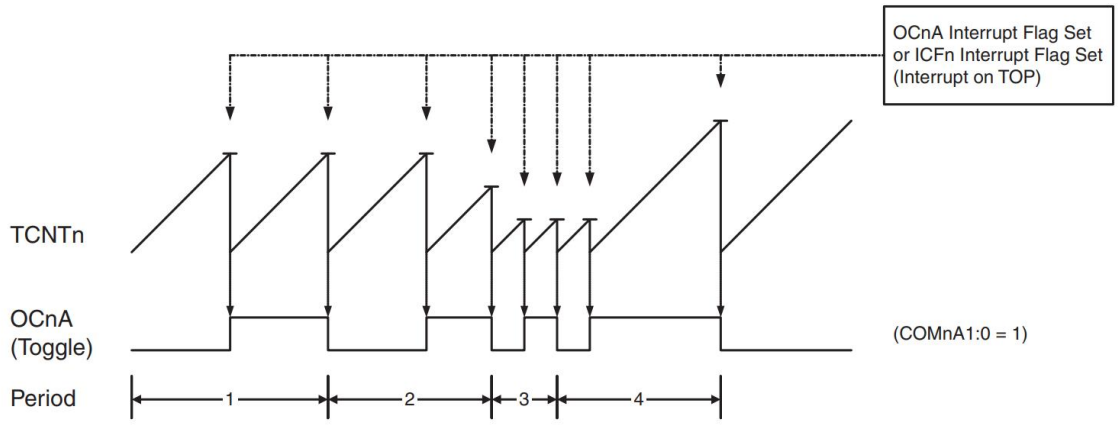


Fig. 56 Diagrama del modo CTC.

En la figura se observa el diagrama de tiempo correspondiente al modo CTC.

Cuando el contador alcanza el valor máximo, se activa el flag de interrupción OCF4A. Si las interrupciones están activadas, además, se ejecuta la rutina de manejo de interrupciones correspondiente.

Dicha rutina se desea ejecutar con un periodo de 500ms, por lo que es necesario que el contador tarde 500ms en alcanzar su valor máximo.

Según la siguiente expresión:

$$T_{OC4A} = \frac{2 \cdot N \cdot (1 + OCR4A)}{f_{clk}}$$

Donde:

- T_{OC4A}: en este caso es el periodo deseado para la ejecución de la rutina.
- N: valor del divisor de frecuencia.
- OCR4A: registro de almacenamiento del valor máximo.
- f_{clk}: frecuencia del reloj.

El registro OCR4A es también un registro de 16 bits por lo que su valor máximo es de:

$$2^{16} = 65536$$

El divisor de frecuencia del temporizador reduce la velocidad de este de manera que se puedan realizar interrupciones con periodos más largos sin que ocurra un desbordamiento del contador. Es posible seleccionar frecuencias 8, 64, 256 o 1024 veces inferiores, aunque se debe tener en cuenta que a mayor frecuencia la precisión es mejor.

Finalmente, para un periodo de 500ms y una frecuencia de reloj de 16MHz:

$$500 \cdot 10^{-3} s = \frac{2 \cdot N \cdot (1 + OCR4A)}{16 \cdot 10^6\ Hz}$$

Sin divisor de frecuencia (N=1):

$$OCR4A = 3999999$$

Dado que supera el valor máximo del registro, es necesario aplicar un divisor de frecuencia mínimo de 64.

$$500 \cdot 10^{-3} s = \frac{2 \cdot 64 \cdot (1 + OCR4A)}{16 \cdot 10^{6}\ Hz}$$

$$OCR4A = 62499$$

Una vez determinadas las características requeridas, se configuran en los registros TCCR4A, TCCR4B y TIMSK4.

TCCR4A – Timer/Counter 4 Control Register A

Bit	7	6	5	4	3	2	1	0	
(0xA0)	COM4A1	COM4A0	COM4B1	COM4B0	COM4C1	COM4C0	WGM41	WGM40	TCCR4A
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

TCCR4B – Timer/Counter 4 Control Register B

Bit	7	6	5	4	3	2	1	0	
(0xA1)	ICNC4	ICES4	–	WGM43	WGM42	CS42	CS41	CS40	TCCR4B
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Fig. 57 Registros de configuración del contador.

Los bits COM4A1:0, COM4B1:0 y COM4C1:0 establecen el funcionamiento de los pines OC4A, OC4B y OC4C. Indicando un 1 el pin deja de funcionar como entrada o salida y su funcionamiento depende del modo seleccionado. En este caso se mantienen a 0 ya que se trata de una interrupción interna y no se precisa ningún pin.

En los bits WGM43:0 se puede seleccionar el modo de funcionamiento. Tal y como se indica en la tabla, existen dos opciones para el modo CTC. La diferencia entre ellas es el registro en el cual se almacena el valor máximo a alcanzar por el contador. En este caso se ha seleccionado el registro OCR4A, por lo tanto, se trata del modo 4 y los bits correspondientes son 0100.

Mode	WGMn3	WGMn2 (CTCn)	WGMn1 (PWMn1)	WGMn0 (PWMn0)	Timer/Counter Mode of Operation	TOP	Update of OCRnx at	TOVn Flag Set on
0	0	0	0	0	Normal	0xFFFF	Immediate	MAX
1	0	0	0	1	PWM, Phase Correct, 8-bit	0x00FF	TOP	BOTTOM
2	0	0	1	0	PWM, Phase Correct, 9-bit	0x01FF	TOP	BOTTOM
3	0	0	1	1	PWM, Phase Correct, 10-bit	0x03FF	TOP	BOTTOM
4	0	1	0	0	CTC	OCRnA	Immediate	MAX
5	0	1	0	1	Fast PWM, 8-bit	0x00FF	BOTTOM	TOP
6	0	1	1	0	Fast PWM, 9-bit	0x01FF	BOTTOM	TOP
7	0	1	1	1	Fast PWM, 10-bit	0x03FF	BOTTOM	TOP
8	1	0	0	0	PWM, Phase and Frequency Correct	ICRn	BOTTOM	BOTTOM
9	1	0	0	1	PWM,Phase and Frequency Correct	OCRnA	BOTTOM	BOTTOM
10	1	0	1	0	PWM, Phase Correct	ICRn	TOP	BOTTOM
11	1	0	1	1	PWM, Phase Correct	OCRnA	TOP	BOTTOM
12	1	1	0	0	CTC	ICRn	Immediate	MAX
13	1	1	0	1	(Reserved)	–	–	–
14	1	1	1	0	Fast PWM	ICRn	BOTTOM	TOP
15	1	1	1	1	Fast PWM	OCRnA	BOTTOM	TOP

Fig. 58 Bits de configuración de los registros.

El bit ICNC4 a nivel alto activa el cancelador de sonido para la captura de entradas. De nuevo, se mantiene a nivel bajo por no precisar entradas externas.

El bit ICES4, también relacionado con la captura de entradas, se emplea para seleccionar el flanco de subida (1) o bajada (0).

Mediante los bits CS42:0, se selecciona el divisor de frecuencia. En este caso, según indica la tabla, para configurar N=64 los bits deben tomar el valor 011.

CSn2	CSn1	CSn0	Description
0	0	0	No clock source. (Timer/Counter stopped)
0	0	1	$clk_{I/O}$/1 (No prescaling
0	1	0	$clk_{I/O}$/8 (From prescaler)
0	1	1	$clk_{I/O}$/64 (From prescaler)
1	0	0	$clk_{I/O}$/256 (From prescaler)
1	0	1	$clk_{I/O}$/1024 (From prescaler)
1	1	0	External clock source on Tn pin. Clock on falling edge
1	1	1	External clock source on Tn pin. Clock on rising edge

Fig. 59 Descripción de los bits del divisor de frecuencia.

TIMSK4 – Timer/Counter 4 Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0	
(0x72)	–	–	ICIE4	–	OCIE4C	OCIE4B	OCIE4A	TOIE4	TIMSK4
Read/Write	R	R	R/W	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Fig. 60 Registro de interrupciones.

En el registro TIMSK4 se realizan las configuraciones correspondientes a la interrupción. El bit ICIE4, a nivel alto activa la interrupción por captura de entrada. Los bits OCIE4A, OCIE4B y OCIE4C habilitan la interrupción que se activa cuando el flag OCF4A, OCF4B o OCF4C está a nivel alto. En el modo seleccionado el flag OCF4A se activa cuando el contador alcanza el valor máximo, por lo tanto, para que la interrupción se ejecute en ese instante el bit OCIE4A debe estar a nivel alto.

A continuación, se muestran las instrucciones que forman la función *configTimer4Interrupt*.

```
void configTimer4Interrupt(){
  cli(); //Desactiva interrupciones
  TCCR4A = (0<<COM4A1)|(0<<COM4A0)|(0<<COM4B1)|(0<<COM4B0)|(0<<COM4C1)|(0<<COM4C0)|(0<<WGM41)|(0<<WGM40);
  //Divisor de frecuencia 64 y modo CTC
  TCCR4B = (0<<ICNC4)|(0<<ICES4)|(0<<5)|(0<<WGM43)|(1<<WGM42)|(0<<CS42)|(1<<CS41)|(1<<CS40);
  OCR4A = 62499; // Límite 500ms
  TCNT4 = 0;// Resetea el contador
  TIMSK4 = (0<<ICIE4)|(0<<OCIE4B)|(1<<OCIE4A)|(0<<TOIE4); //Interrupción por límite de cuenta
  sei(); //Activa interrupciones
}
```

Fig. 61 Configuración de la interrupción.

En primer lugar, se desactivan las interrupciones mediante la función *cli* para evitar que se active antes de finalizar la configuración de la misma. A continuación, se configuran los registros antes mencionados. También se resetea el contador. Finalmente, se activan las interrupciones con la función *sei*. Tanto *cli* como *sei* son funciones incluidas en la librería Arduino.

Calc_Temp_ad590

```
void Calc_Temp_ad590(int pinAD590){
  int AD590; //valor digital AD590
  float Vad590; //tensión
  AD590=analogRead(pinAD590);//lee valor digital del pin AD590
  Vad590=(float(AD590)*5)/1023; //conversión del valor digital a tensión AD590
  Tad590=(Vad590/0.011)-273; //obtiene la temperatura del AD590
}
```

Fig. 62 Cálculo de la temperatura del AD590.

En esta función se emplean dos variables locales. La primera, AD590, es de tipo entero ya que almacena el valor digital obtenido en la lectura del pin. Para ello, se emplea la función *analogRead* la cual recibe como argumento el número del pin a leer.

El convertidor analógico digital de la placa Arduino Mega posee una resolución de 10 bits, lo que significa que puede tomar 2^{10} valores. Esto es, realiza la lectura de una tensión analógica de 0V a 5V y devuelve su valor digital correspondiente que será un número entre 0 y 1023. La resolución

de este convertidor expresada en voltaje es de 4,88mV lo cual significa que, para apreciar un cambio en el valor digital, la tensión analógica de entrada debe variar al menos en 4,88mV.

Una vez se tiene el valor digital correspondiente a la tensión de salida del circuito de acondicionamiento del AD590, se calcula su valor analógico teniendo en cuenta la equivalencia entre 5V y su valor digital 1023. El resultado se almacena en la variable local Vad590 de tipo coma flotante para no perder la información de los decimales.

Finalmente, para conocer el valor de la temperatura que corresponde a dicha tensión, se recurre a la ecuación obtenida en el apartado correspondiente al acondicionamiento del AD590:

$$V_O = 1mV/K \cdot T \cdot 11$$

Donde V_O es la tensión almacenada en la variable Vad590 y T la temperatura que se desea almacenar en la variable Tad590. Se despeja la incógnita de la temperatura y se convierte a grados centígrados:

$$Tad590 = \frac{V_{AD590}}{11 \cdot 10^{-3} \frac{V}{°C}} - 273$$

La variable Tad590, a diferencia de las anteriores, es una variable global. Las variables locales existen únicamente en la función en la que se definen. Al finalizar, estas se eliminan. En cambio, las variables globales pueden emplearse en cualquier parte del archivo en el que se han definido y mantienen su valor. Esto permite que más adelante, se pueda recuperar el valor calculado de temperatura para operar con él, enviarlo o mostrarlo por pantalla.

Calc_Temp_termopar

```
void Calc_Temp_termopar(int pinTermopar){
  int Termopar; //valor digital Termopar
  float Vtermopar; //tensión
  Termopar=analogRead(pinTermopar);//lee valor digital Termopar
  //conversión del valor digital a tensión Termopar
  Vtermopar=(float(Termopar)*5)/1023;
  Ttermopar=100*Vtermopar; //obtiene la temperatura del Termopar
}
```

Fig. 63 Cálculo de la temperatura del termopar.

En el caso del termopar y también para el resto de sensores, las primeras operaciones a realizar son análogas al caso del AD590. La única diferencia es que cada sensor emplea sus propias variables.

Para el cálculo de la temperatura, de nuevo, se recurre al análisis del circuito de acondicionamiento, en este caso del termopar tipo K, de donde se obtiene la expresión:

$$V_O = \frac{10mV}{°C} \cdot T$$

Se obtiene la temperatura del termopar:

$$Ttermopar = 100 \cdot V_{termopar}$$

Calc_Temp_ntc

```
void Calc_Temp_ntc(int pinNTC){
  int NTC; //lectura valor digital NTC
  float Vntc; //tensión
  NTC=analogRead(pinNTC);//lee valor digital NTC
  Vntc=(float(NTC)*5)/1023; //conversión del valor digital a tensión NTC
  //obtiene la temperatura de la ntc
  Tntc=((7.08*Vntc+6.606)/(0.0152639*Vntc+0.029764605))-273;
}
```

Fig. 64 Cálculo de la temperatura de la NTC.

En este caso, además de emplear la expresión de salida del circuito de acondicionamiento, es necesario aplicar el método del punto de inflexión para obtener la ecuación lineal que representa la variación del conjunto de resistencias en paralelo en función de la temperatura:

$$R_p(T) = R_p(T_c) + S_{Tc}(T - T_C)$$

Siendo:

- R_p el valor de la NTC y la resistencia de linealización en paralelo.
- T_c la temperatura central del rango de linealización (60ºC).
- S_{Tc} la sensibilidad a la temperatura central.

Donde:

$$R_{T_C=60ºC} = 10K\Omega\, e^{4080K\cdot\left(\frac{1}{333K}-\frac{1}{298K}\right)} = 2{,}37K\Omega$$

$$R_P(T_P) = \frac{2{,}37 \cdot 1{,}74}{2{,}37 + 1{,}74} = 1K\Omega$$

$$S(T_c) = -\frac{1}{4}\, R_{T_C}\frac{(\beta - 2\,T_C)^2}{\beta\, T_c^2}$$

$$S(T_c) = -\frac{1}{4}\, 2{,}37K\Omega\frac{(4080 - 2\cdot 333)^2}{4080\cdot 333^2} = -0{,}0152639\frac{K\Omega}{\mathrm{K}}$$

Sustituyendo los valores obtenidos en la ecuación anterior:

$$R_p(T) = 1K\Omega - 0{,}0152639\frac{K\Omega}{\mathrm{K}}(T - 333)$$

De la misma manera, se sustituye la expresión de Rp(T) en la ecuación de salida del circuito de acondicionamiento:

$$V_{O_{AMP}} = 5{,}25\ - 7{,}2\cdot\frac{R_p}{R_p + 1}$$

$$V_{O_{AMP}} = 5{,}25\ - 7{,}2\cdot\frac{1K\Omega - 0{,}0152639\frac{K\Omega}{\mathrm{K}}(T - 333)}{2K\Omega - 0{,}0152639\frac{K\Omega}{\mathrm{K}}(T - 333)}$$

Despejando la temperatura, se obtiene finalmente la ecuación implementada en el código:

$$Tntc = \left(\frac{7{,}08\cdot V_{O_{AMP}} + 6{,}606}{0{,}0152639\cdot V_{O_{AMP}} + 0{,}029764605}\right) - 273$$

Calc_Temp_pt100

```
void Calc_Temp_pt100(int pinPT100){
  int PT100; //lectura valor digital PT100
  float Vpt100; //tensión
  PT100=analogRead(pinPT100);//lee valor digital PT100
  Vpt100=(float(PT100)*5)/1023; //conversión del valor digital a tensión PT100
  Tpt100=(5.04-Vpt100)/(0.033341); //obtiene la temperatura de la PT100
}
```

Fig. 65 Cálculo de la temperatura de la PT100.

Para el cálculo de la temperatura indicada por la PT100 se procede de la misma manera que en los casos anteriores.

Se tiene la salida del circuito de acondicionamiento expresada en función de V_{OAMP1}.

$$V_{O_{AMP2}} = 13{,}7V - 86{,}6 \cdot V_{O_{AMP1}}$$

Donde V_{OAMP1}:

$$V_{O_{AMP1}} = R_{PT100} \cdot I$$

En cuanto al valor de la resistencia de la PT100, esta varía en función de la temperatura:

$$R_{PT100(T)} = R_O \cdot (1 + \alpha T)$$

De modo que se obtiene la expresión de salida del circuito de acondicionamiento en función de la temperatura:

$$V_{O_{AMP2}} = 13{,}7V - 86{,}6 \cdot R_O \cdot (1 + \alpha T) \cdot I$$

Donde Ro=100Ω y α=0,00385.

La tensión V_{OAMP1} se almacena en la variable Vpt100 por lo que, para obtener la temperatura, basta con despejarla de la ecuación anterior e implementar el resultado en el código.

$$V_{O_{AMP2}} = 13{,}7 - 86{,}6 \cdot 100 \cdot (1 + 0{,}00385 \cdot T) \cdot 1$$

$$Tpt100 = \frac{5{,}05 - V_{O_{AMP2}}}{0{,}033341}$$

initFiltros

```
void initFiltros(){
  for(int i=0;i<ventana;i++){ //Recorre los buffers
    Calc_Temp_ad590(pinAD590); //Obtiene la temperatura del sensor
    bufferCircularAD[i]=Tad590; //La almacena en el buffer
    sumaBufferCircularAD+=bufferCircularAD[i]; //Actualiza la suma
    Calc_Temp_ntc(pinNTC);
    bufferCircularNTC[i]=Tntc;
    sumaBufferCircularNTC+=bufferCircularNTC[i];
    Calc_Temp_pt100(pinPT100);
    bufferCircularPT[i]=Tpt100;
    sumaBufferCircularPT+=bufferCircularPT[i];
    Calc_Temp_termopar(pinTermopar);
    bufferCircularK[i]=Ttermopar;
    sumaBufferCircularK+=bufferCircularK[i];
    delay(10); //Espera 10ms entre muestras
  }
```

Fig. 66 Inicialización de los filtros digitales.

Esta función, empleada en el *setup*, es la encargada de inicializar los buffers empleados en el filtrado de las lecturas de los sensores. Para ello, se recorren mediante un bucle *for* y en cada elemento se almacena una lectura de la temperatura, llamando para ello a las funciones vistas anteriormente. También se calcula la suma del buffer. De esta manera, se evita el efecto de valores nulos al comenzar la ejecución del programa.

Filtrado

```
void FiltradoAD(){ //Filtrado de la temperatura del AD590
   sumaBufferCircularAD-=bufferCircularAD[indiceAD]; //Resta la lectura más antigua
   bufferCircularAD[indiceAD]=Tad590; //sustituye la lectura más antigua por la actual
   sumaBufferCircularAD+=bufferCircularAD[indiceAD]; //Actualiza la suma
   indiceAD++; //Actualiza el índice
   if(indiceAD==ventana){ //Valor máximo del índice alcanzado
    indiceAD=0; //Resetea el índice
   }
   //El valor filtrado es la media de los valores del buffer
   Tad590=sumaBufferCircularAD/ventana;
}
```

Fig. 67 Filtrado de temperaturas.

Se ha definido una función de filtrado para cada sensor, ya que cada uno posee su propio buffer en el que se almacenan las últimas diez lecturas. En la imagen se muestra el filtro del sensor AD590, siendo la única diferencia entre los filtros el nombre de las variables. El funcionamiento del filtrado es el siguiente: en primer lugar, se resta el valor de la lectura más antigua del total de las lecturas almacenadas en el vector. Este valor se encuentra en la variable sumaBufferCircularX. A continuación, se sustituye esta misma lectura en el buffer circular por la lectura más reciente. También se actualiza la suma de las lecturas del buffer. Se emplea un índice para recorrer el vector. Este índice se actualiza en cada ejecución de la función. Su valor máximo viene definido por la constante Ventana. Una vez alcanzada la última posición del vector, se vuelve a la primera

posición del mismo, cuyo índice es el valor cero. Finalmente, en la variable que almacena la temperatura del sensor, se escribe el valor correspondiente a la media de los valores almacenados en vector, calculada a partir de la suma y el número de lecturas almacenadas en este.

Además de la lectura de los sensores, también es necesario leer las entradas de la persona usuaria. En el caso de que el modo manual se encuentre activo, estas se realizan a través del de los potenciómetros e interruptores situados en el panel frontal del horno. De lo contrario, en el modo PC la persona usuaria interactúa con el horno a través del programa.

Para cada caso, se define una función de lectura de entradas.

Lect_Entradas_Manual

```
void Lect_Entradas_Manual(){
  if(digitalRead(pinSelec_AD590)==LOW){ //Lectura del selector
    Sensor=1; //Asignación del número correspondiente al sensor
  }else if(digitalRead(pinSelec_Termopar)==LOW){
    Sensor=2;
  }else if(digitalRead(pinSelec_PT100)==LOW){
    Sensor=3;
  }else if(digitalRead(pinSelec_NTC)==LOW){
    Sensor=4;
  }
  Consigna=analogRead(pinConsigna); //Lectura potenciómetro Consigna
  Consigna=map(Consigna,0,1023,0,150); //Ajusta el valor digital al rango 0-150°C
  //Lectura potenciómetros Kp, Ki y Kd y ajuste de rango
  Kp=analogRead(pinKp);
  Kp=map(Kp,0,1023,0,100);
  Ki=analogRead(pinKi);
  Ki=map(Ki,0,1023,0,100);
  Kd=analogRead(pinKd);
  Kd=map(Kd,0,1023,0,50);
  Ventilador=digitalRead(pinIntVentilador); //Lectura interruptor Ventilador
}
```

Fig. 68 Lectura de las entradas en modo manual.

En primer lugar, se realiza la lectura del selector. En este caso, dado que se trata de entradas digitales, la función a emplear es *digitalRead*. Esta devuelve un 0 en caso de que la entrada esté a nivel bajo y un 1 si se encuentra a nivel alto. Dado que todos los pines de la placa Arduino relacionados con el selector son entradas con resistencia de pull-up, el sensor seleccionado es aquél cuya entrada se encuentre a nivel bajo. Una vez encontrado el sensor seleccionado, se le asigna el número correspondiente a dicho sensor a la variable Sensor.

A continuación, se procede a la lectura de los potenciómetros mediante los cuales la persona usuaria introduce los valores de consigna de temperatura y constantes del PID deseados. Todos ellos se encuentran conectados a entradas analógicas de la placa que serán leídas a través de la función *analogRead*. Tal y como se ha mencionado anteriormente, esta función siempre devuelve un valor de 0 a 1023. Sin embargo, este rango de valores no es el rango de valores que pueden tomar las variables en cuestión. Por ejemplo, en el caso de la consigna de temperatura, su valor puede ir de 0 a 150. Para realizar esta conversión se emplea la función *map* también incluida en la librería Arduino. Esta recibe como argumentos el valor

a modificar, el rango de valores en el que se encuentra, y el rango de valores al cual se quiere convertir. El valor de la variable se sustituye por el resultado de la asignación.

Por último, se lee el interruptor asociado al encendido y apagado del ventilador.

Lect_Entradas_PC

```
//Almacena los 6 bytes recibidos en el buffer de entrada
void Lect_Entradas_PC(){
    Sensor=Serial.read();
    Consigna=Serial.read();
    Kp=Serial.read();
    Ki=Serial.read();
    Kd=Serial.read();
    Ventilador=Serial.read();
}
```

Fig. 69 Lectura de las entradas en modo PC.

Si el horno se encuentra conectado con el ordenador, las entradas de la persona usuaria se reciben a través de la comunicación serial. Para realizar la lectura se emplea la función *Serial.read*. Esta función lee un byte del buffer serial.

El buffer serial tiene capacidad para almacenar 64 bytes. Se trata de una pila FIFO (First In First Out), lo que significa que el primer byte recibido es el primer byte leído. A medida que llegan los datos, el buffer se va llenando. Mientras que a medida que se leen bytes con la función *Serial.read*, esta se va vaciando. Si el buffer se llena, los datos recibidos se pierden.

En este caso no es necesario realizar ningún tratamiento a los datos recibidos ya que llegan con las características deseadas.

Además de la lectura de las entradas, la subrutina de atención a la interrupción también realiza el control de las salidas del sistema. A continuación, se analizan las funciones dedicadas a ello.

Control_Ventilador

```
void Control_Ventilador(){
  if(Ventilador==1){ //Ventilador OFF
    analogWrite(pinPWMVentilador,0); //Apaga el ventilador
  }else{ //Ventilador ON
    //Lectura de la consigna de velocidad
    VelVentilador=analogRead(pinVelVentilador);
    //Ajuste de la precisión de  10 bits a 8 bits
    VelVentilador=map(VelVentilador,0,1023,0,255);
    //Control de velocidad según consigna
    analogWrite(pinPWMVentilador,VelVentilador);
  }
}
```

Fig. 70 Control del ventilador.

Se trata de la función encargada del control del encendido, apagado y velocidad del ventilador.

La lectura del interruptor del ventilador, almacenada en la variable Ventilador indica si se debe encender el ventilador con un 0 o con un 1 en el caso contrario.

Tal y como se muestra en el código, la escritura de un 0 en la función *analogWrite* equivale a un ciclo de trabajo del 0% por lo que el ventilador permanece desconectado.

En cambio, si el interruptor del ventilador se encuentra en la posición de encendido, se realiza la lectura del potenciómetro del panel frontal destinado a controlar la velocidad del ventilador.

Las entradas analógicas de la placa Arduino Mega poseen una precisión de 16 bits. Sin embargo, las salidas PWM son de 8 bits. Esto significa que en la lectura del potenciómetro se puede recibir un valor digital de 0 a 1023 mientras que en la salida PWM solo se puede indicar un valor digital de 0 a 255. Para realizar este ajuste se emplea de nuevo la función *map*.

Una vez obtenido el valor adecuado a la señal PWM, se emplea la función *analogWrite* para modificar la señal de salida. A mayor valor, mayor es la velocidad del ventilador ya que el ciclo de trabajo aumenta y con ello la tensión media de alimentación aplicada al ventilador.

Params_PID

```
void Params_PID(){
  switch(Sensor){  //Asigna la temperatura de referencia según sensor seleccionado
    case 1:
      Temperatura=Tad590;
      break;
    case 2:
      Temperatura=Ttermopar;
      break;
    case 3:
      Temperatura=Tpt100;
      break;
    case 4:
      Temperatura=Tntc;
      break;
    default:
      Temperatura=Tad590;
  }
}
```

Fig. 71 Temperatura de referencia en el control PID.

Para poder aplicar la acción de control PID, se precisa, además de una consigna que indique el valor deseado a alcanzar por la variable a controlar, el valor actual de dicha variable.

En el horno, la variable a controlar es la temperatura y esta es medida por cuatro sensores. Bien mediante el selector o bien desde la aplicación es posible seleccionar cuál de esos cuatro sensores se desea utilizar como referencia para el control PID. Ahora, mediante la función *switch* se asigna la lectura del sensor a la variable Temperatura. Si por alguna razón la lectura del selector no es correcta, se emplea el caso por defecto, donde se utiliza como referencia la temperatura del AD590. En caso de haber seleccionado el sensor desde el ordenador, esto es, en el modo PC, se habrá recibido el número del sensor seleccionado

Control_PID

```
void Control_PID(){
  Params_PID();
  ErrorAnterior=ErrorActual; //Almacena error anterior
  ErrorActual=(Consigna-Temperatura); //Calcula error actual
  //Término Proporcional
  pTerm=Kp*ErrorActual; //Calcula el término proporcional
  //Término Integral
  if(abs(ErrorActual)>LimInt ) { //Error mayor que el límite
    iError=((float(ErrorActual)+float(ErrorAnterior))/2)*Tm; //Calcula error integral
    iTerm=Ki*iError; //Calcula el término integral
  }
  else {
    iTerm = 0; //No se aplica acción integral
  }
  if (iTerm < 0){
    iTerm = 0; //Mínimo valor del término integral
  }
  //Término Derivativo
  dError=(float(ErrorActual)-float(ErrorAnterior))/Tm; //Calcula error derivativo
  dTerm=Kd*dError; //Calcula el término derivativo
  if(dTerm<0){
    dTerm=0; //Mínimo valor del término derivativo
  }
  //Acción PID
  Salida_PID=pTerm+iTerm+dTerm; //Cálculo de la señal PID
  if (Salida_PID > Salida_PID_max){
    Salida_PID = Salida_PID_max; //Máximo valor de la salida
  }
  if (Salida_PID < 0){
    Salida_PID = 0; //Mínimo valor de la salida
  }
  Salida_PID=map(Salida_PID,0,Salida_PID_max,0,16000); //Ajuste del rango
}
```

Fig. 72 Control PID.

Se trata de la función principal del control PID, donde, finalmente, se obtiene la señal de control.

En primer lugar, se llama a la función anterior para obtener los parámetros necesarios para el cálculo de la nueva señal de control. A continuación, se almacena el error anterior y se calcula el error entre el valor actual de temperatura y el valor deseado de la misma. Este dato será mostrado en pantalla posteriormente. Seguido, se calcula cada uno de los términos de la señal de control. El término proporcional es proporcional a la diferencia entre el valor actual y el valor de consigna. El término integral se aplica tanto si el error es positivo como negativo. Al tratarse de la discretización de una integral, se calcula el área que conforman el error y el tiempo de muestreo. Para obtener una mayor resolución, se emplea el valor medio entre el error anterior y el actual. Para calcular el término derivativo, al tratarse de la discretización de una derivada, se emplea la diferencia entre el error actual y el anterior obtenidos en una diferencia de tiempo igual al tiempo de muestreo. Finalmente, la señal de control se obtiene a partir de la suma de los tres términos anteriores. El resultado de la suma se mantiene dentro del rango definido por los valores máximo y mínimo de 255 y 0 respectivamente. Por último, se emplea la función *map* para ajustar el rango al rango del registro del contador, cuyo valor máximo es de 16000. La razón por la que no se ha empleado este límite desde el comienzo, es debido a que los valores necesarios para las constantes del PID serían excesivamente elevados.

configTimerPWM

Para configurar el temporizador encargado de la señal PWM de la resistencia calefactora, se configuran los registros TCCR5A, TCCR5B y OCR5A vistos en la configuración del temporizador 4, aunque en este caso corresponden al temporizador 5. Además, esta vez también es necesario configurar el registro ICR5.

El temporizador debe funcionar en modo control de fase y frecuencia. En él, el contador funciona con doble pendiente. Es decir, cuenta de cero al valor máximo y del valor máximo a cero. El funcionamiento a doble pendiente permite una frecuencia de la salida inferior a la de pendiente única. En el modo no inversor, la salida se pone a nivel bajo cuando se alcanza el valor máximo en la cuenta ascendente y a nivel alto en la cuenta descendente.

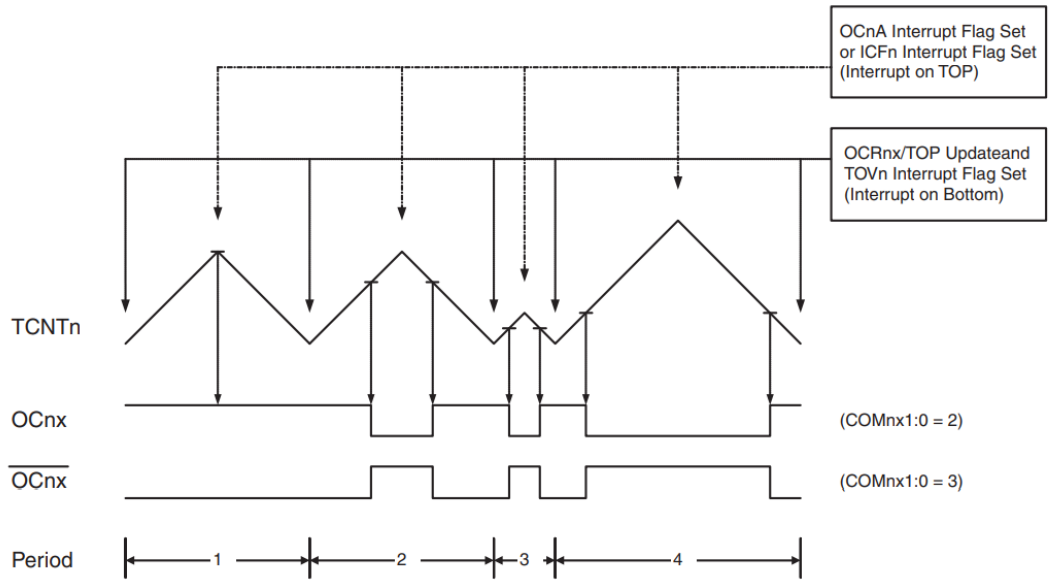


Fig. 73 Diagrama del contador en modo control de fase y frecuencia.

```
void configTimerPWM(){
  //Salida PWM en OC5A
  TCCR5A=(1<<COM5A1)|(0<<COM5A0)|(0<<COM5B1)|(0<<COM5B0)|(0<<COM5C1)|(0<<COM5C0)|(0<<WGM51)|(0<<WGM50);
  //Modo control de fase y frecuencia, prescaler 1
  TCCR5B=(0<<ICNC5)|(0<<ICES5)|(0<<5)|(1<<WGM53)|(0<<WGM52)|(0<<CS52)|(0<<CS51)|(1<<CS50);
  ICR5=16000; //Frecuencia PWM 500Hz
  TCNT5=0;//Resetea Timer 5
  OCR5A=16000;//registro para controlar el ciclo de trabajo
}
```

Fig. 74 Configuración del contador.

En primer lugar, se configuran los registros TCCR5A y TCCR5B.

En los bits COM5A1:0, COM5B1:0 y COM5C1:0 se configura el pin OC5A como salida de la señal PWM y se mantiene el funcionamiento normal de los pines OC5B y OC5C. El pin OC5A equivale al pin número 46 de la placa Arduino. En el modo COM5A1=1 y COM5A0=0 el pin 46 se pone a nivel bajo cuando se alcanza el valor máximo en la cuenta ascendente y a nivel alto en la cuenta descendente.

El modo de funcionamiento seleccionado en los bits WGM53:0 es el 8, dedicado al control de fase y frecuencia. Se emplea el registro ICR5 para almacenar el valor máximo.

En este caso, los bits ICNC5 e ICES5 también se mantienen a nivel bajo por no precisar entradas externas.

Mediante los bits CS52:0, se selecciona el divisor de frecuencia. En este caso, no es necesario reducir la frecuencia por lo que N=1.

El valor almacenado en el registro ICR5 determina la frecuencia de la señal PWM. A continuación, se muestran los cálculos para una frecuencia de 500Hz:

$$F_{PWM} = \frac{F_{CLK}}{2 \cdot N \cdot ICR5}$$
$$500Hz = \frac{16 \cdot 10^6 Hz}{2 \cdot 1 \cdot ICR5}$$
$$ICR5 = 16000$$

Dado que el valor obtenido para el registro ICR5 es inferior al valor máximo que admite el registro de 16 bits, no es necesario emplear la división de frecuencia.

Una vez reseteado el temporizador mediante el registro TCNT5, se establece el valor máximo de 16000 para el registro OCR5A. Con ello, se selecciona un ciclo de trabajo del 100%. Es decir, en el modo no inversor se tiene inicialmente la salida a nivel alto. De esta manera se desconecta la resistencia calefactora de la alimentación.

Resistencia_PWM

```
void Resistencia_PWM(){
  DC=map(Salida_PID,0,16000,16000,0);//Invierte la salida
  Duty=100-(float(DC)/16000)*100; //Duty Cycle en %
  OCR5A=DC; //Actualiza el registro del contador 5
}
```

Fig. 75 Control de la resistencia calefactora.

Se trata de la función encargada del control de la alimentación aplicada a la resistencia calefactora.

Para ello, se emplea la salida del control PID, donde se ha empleado la lógica positiva. Esto es, un valor alto en la salida está relacionado con una mayor alimentación aplicada a la carga a controlar. Sin embargo, tal y como se ha visto en la función anterior, en este caso el valor máximo equivale a la desconexión de la alimentación aplicada a la resistencia calefactora. Por lo tanto, se emplea la función *map* para invertir la salida.

En la variable Duty se almacena el valor del ciclo de trabajo en porcentaje para mostrarlo posteriormente por pantalla.

Por último, se actualiza el registro OCR5A con el nuevo valor calculado del ciclo de trabajo.

AlertaEnfriamiento

```
void AlertaEnfriamiento(){
  OCR5A=16000; //Desconecta la resistencia calefactora
  analogWrite(pinPWMVentilador,255); //Enciende el ventilador a su velocidad máxima
  //Muestra mentase de alerta en pantalla y temperatura del horno
  lcd.setCursor(0,0);
  lcd.print("                    ");
  lcd.setCursor(0,1);
  lcd.print("ALERTA ENFRIAMIENTO ");
  lcd.setCursor(0,2);
  lcd.print("      TEMP:         ");
  lcd.setCursor(0,3);
  lcd.print("                    ");
  do{
     Calc_Temp_ad590(pinAD590); //Calcula la temperatura
     lcd.setCursor(11,2);
     lcd.print(Tad590); //Imprime la temperatura
     lcd.print("C");
     lcd.setCursor(11,2);
     lcd.print("     ");
   }while(Tad590 > 100);
   //El bucle finaliza cuando la temperatura es igual o inferior a 100C
   analogWrite(pinPWMVentilador,0); //Apaga el ventilador
   //Reset pantalla a estado anterior
   lcd.clear();
   imprimir_titulos();
 }
```

Fig. 76 Alerta de enfriamiento.

Este conjunto de instrucciones lleva a cabo un enfriamiento forzado del horno. El objetivo del mismo es evitar que el equipo funcione a temperaturas excesivas. Tal y como se ha visto en apartados anteriores, los sensores pueden alcanzar temperaturas de hasta 150ºC y las paredes de policarbonato del horno hasta 130ºC. Para garantizar la seguridad y el correcto funcionamiento del horno, se ha reducido el límite a 110ºC. En caso de alcanzar dicha temperatura, se ejecutan las instrucciones de esta función.

En primer lugar, se desconecta la resistencia calefactora de la alimentación para detener el suministro de calor en el interior del horno. Seguido, se enciende el ventilador a su velocidad máxima para acelerar el proceso de enfriamiento. A continuación, se vacía la pantalla LCD y se muestra el mensaje "ALERTA ENFRIAMIENTO". En este momento, comienza la ejecución de un bucle *do…while* que será ejecutado al menos una vez. En él, se calcula la temperatura detectada por el sensor AD590 y se imprime su valor en la pantalla. Para finalizar el bucle, es necesario que la temperatura leída por el sensor sea inferior a 100ºC. Entonces, se detiene el ventilador y se devuelve la pantalla a su estado anterior. El programa podrá continuar su funcionamiento normal.

Finalmente, se tienen las funciones encargadas del control de la pantalla LCD situada en el panel frontal del horno.

init_lcd

```
void init_lcd(){
  lcd.begin();// Inicializar el display LCD
  lcd.backlight();//Encender la luz de fondo
}
```

Fig. 77 Inicialización de la pantalla LCD.

Se emplea para iniciar la comunicación I2C entre la placa Arduino y la pantalla LCD. Una vez inicializada la pantalla queda con la luz encendida.

imprimir_saludo

```
void imprimir_saludo(){ //Imprime mensaje de inicio
  lcd.setCursor(7,1); //Coloca el cursor
  //Imprime mensaje
  lcd.print("CONTROL");
  lcd.setCursor(3,2);
  lcd.print("DE TEMPERATURA");
  delay(3000); //Espera 3s
  lcd.clear(); //Limpia la pantalla
}
```

Fig. 78 Imprimir saludo.

Muestra un mensaje en la pantalla cada vez que se inicia el equipo. Este mensaje se mantiene durante tres segundos y posteriormente se vacía la pantalla.

imprimir_modo

```
void imprimir_modo(){ //Imprime el modo de funcionamiento
  lcd.setCursor(0,3);
  lcd.print("           "); //Borra modo anterior
  lcd.setCursor(0,3);
  if(Modo==1){  //Modo manual activo
    lcd.print("/manual/");
  }else{ //Modo PC activo
    lcd.print("/pc/");
  }
}
```

Fig. 79 Imprimir modo.

En la esquina inferior izquierda de la pantalla se muestra el modo de funcionamiento seleccionado.

imprimir_titulos

```
void imprimir_titulos(){ //Imprime los nombres de las variables
  lcd.setCursor(0,1);
  lcd.print("Consig:");
  lcd.setCursor(0,2);
  lcd.print("Error:");
  lcd.setCursor(11,0);
  lcd.print("Kp:");
  lcd.setCursor(11,1);
  lcd.print("Ki:");
  lcd.setCursor(11,2);
  lcd.print("Kd:");
  lcd.setCursor(11,3);
  lcd.print("DC:");
}
```

Fig. 80 Imprimir títulos.

En esta función se imprimen los nombres de los datos a mostrar por pantalla que se mantienen constantes.

imprimir_temp

```
//Imprime el nombre del sensor seleccionado y su temperatura
void imprimir_temp(){
  lcd.setCursor(0,0);
  lcd.print("           ");
  lcd.setCursor(0,0);
  switch(Sensor){
    case 1:
      lcd.print("AD590:");
      break;
    case 2:
      lcd.print("TIPOK:");
      break;
    case 3:
      lcd.print("PT100:");
      break;
    case 4:
      lcd.print("NTC:");
      break;
    default:
      lcd.print("AD590:");
  }
  lcd.print(Temperatura);
  lcd.print("C");
}
```

Fig. 81 Imprimir temperatura.

El nombre del sensor que se imprime en la pantalla varía en función del sensor seleccionado. Es por ello que se emplea una función independiente encargada de imprimir el nombre y el valor correspondiente a la temperatura.

imprimir_consig, imprimir_error, imprimir_Kp, imprimir_Ki, imprimir_Kd e imprimir_DC

```
void imprimir_consig(){
  lcd.setCursor(7,1);
  lcd.print("    ");
  lcd.setCursor(7,1);
  lcd.print(Consigna);
  lcd.print("C");
}

void imprimir_Kp(){
  lcd.setCursor(14,0);
  lcd.print("      ");
  lcd.setCursor(14,0);
  lcd.print(Kp);
}

void imprimir_Kd(){
  lcd.setCursor(14,2);
  lcd.print("      ");
  lcd.setCursor(14,2);
  lcd.print(Kd);
}

void imprimir_error(){
  lcd.setCursor(6,2);
  lcd.print("     ");
  lcd.setCursor(6,2);
  lcd.print(error);
  lcd.print("C");
}

void imprimir_Ki(){
  lcd.setCursor(14,1);
  lcd.print("      ");
  lcd.setCursor(14,1);
  lcd.print(Ki);
}

void imprimir_DC(){
  lcd.setCursor(14,3);
  lcd.print("      ");
  lcd.setCursor(14,3);
  lcd.print(Duty);
}
```

Fig. 82 Imprimir datos.

Son las funciones encargadas de imprimir los datos de las variables.

ISR

```
ISR(TIMER4_COMPA_vect) {
  //Detiene el contador
  TCCR4B = (0<<ICNC4)|(0<<ICES4)|(0<<5)|(0<<WGM43)|(1<<WGM42)|(0<<CS42)|(0<<CS41)|(0<<CS40);
  TIMSK4 = (0<<ICIE4)|(0<<OCIE4B)|(0<<OCIE4A)|(0<<TOIE4);//Deshabilita la interrupción
  Tmuestreo=1; //Flag tiempo de muestreo
  TCNT4 = 0; //Resetea el contador
  //Activa el contador
  TCCR4B = (0<<ICNC4)|(0<<ICES4)|(0<<5)|(0<<WGM43)|(1<<WGM42)|(0<<CS42)|(1<<CS41)|(1<<CS40);
  TIMSK4 = (0<<ICIE4)|(0<<OCIE4B)|(1<<OCIE4A)|(0<<TOIE4);//Habilita la interrupción
}
```

Fig. 83 Subrutina de atención a la interrupción.

La subrutina de atención a la interrupción programada para el temporizador 4 es un conjunto de instrucciones que se ejecutan cada 500ms. Por tratarse de una interrupción, es necesario primeramente deshabilitarla y detener el contador. Para ello, se modifican los registros vistos anteriormente. Esta interrupción se emplea para poner a nivel alto el flag Tmuestreo cada 500ms. Este valor del flag permite que se ejecuten las instrucciones relacionadas con la lectura de los sensores y el control de los actuadores. Estas se encuentran en el bucle principal del programa. De esta manera se tiene una breve subrutina de atención a la interrupción, que puede ser almacenada en el espacio de la memoria reservado para ella y que permite controlar el periodo con el que se ejecuta un conjunto de instrucciones más extenso. Antes de finalizar la subrutina, es necesario resetear el contador y activar tanto el contador como su interrupción, de manera que la interrupción suceda de forma cíclica.

3.4.2. Matlab

El sistema de monitorización se desarrolla en Matlab App Designer.

Antes de proceder al análisis del código, se describe el diseño final de la interfaz gráfica ya que gran parte del código se encarga de gestionar el comportamiento de los componentes que se encuentran en ella.

El diseño de la aplicación es el resultado de la implementación de los componentes necesarios para lograr que se puedan realizar tres tareas. Concretamente: permitir a la persona usuaria controlar el equipo desde el PC, visualizar los datos de forma gráfica y almacenar la información para su posterior análisis. Además, el diseño de la interfaz gráfica debe ser visualmente atractivo e intuitivo.

Fig. 84 Diseño final de la interfaz gráfica.

Elementos de control

Por elementos de control se entienden aquellos componentes que permiten a la usuaria modificar alguna de las variables de entrada del equipo o de la aplicación.

Puerto Serie

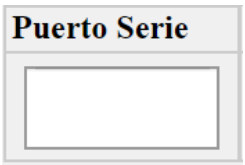


Fig. 85 Puerto serie.

Se trata de un área de texto en el que se debe introducir por teclado el nombre del puerto COM al que se encuentra conectada la placa Arduino.

Encendido/apagado

Fig. 86 Botones de inicio y parada.

El botón de encendido/apagado de la aplicación es un botón de estado. Esto es, al pulsar en él conmuta entre los dos estados. En el estado "apagado" la aplicación se encuentra detenida, por lo que no se intercambian datos con la placa Arduino. En este caso, el valor del botón es 0. En el estado "encendido" la aplicación se encuentra en ejecución e intercambia datos con el Arduino. El valor del botón en este estado es un 1.

```
% Value changed function: Iniciar
function IniciarValueChanged(app, event)
    if app.Iniciar.Value==0 %Apagar
       app.Iniciar.Icon = 'ON3.png'; %Icono botón verde
       app.ImagenPuerto.Visible='off'; %Oculta icono puerto COM conectado
       app.PuertoSerie.Visible='on'; %Muestra área de escritura puerto COM
       app.ExcelDatos.Visible='on'; %Muestra área de escritura nombre fichero
       app.Exportar.Visible='on'; %Muestra botón exportar
       app.MostrarTabla.Visible='on'; %Muestra botón tabla
       write(app.s,1,"uint8"); %SensorPID=1
       write(app.s,0,"uint8"); %Consigna=0
       write(app.s,0,"uint8"); %Kp=0
       write(app.s,0,"uint8"); %Ki=0
       write(app.s,0,"uint8"); %Kd=0
       write(app.s,1,"uint8"); %Ventilador=1
       app.s= []; %cierra la comunicación
       app.ImagenAviso.Visible='on';%Muestra aviso equipo desconectado
    else
       app.Iniciar.Icon = 'OFF3.png'; %Icono botón rojo
       Puerto=app.PuertoSerie.Value; %Guarda el nombre del puerto
       if strcmp(Puerto,'') %No se ha indicado ningún nombre
         mensaje=sprintf(['Puerto COM no definido. ' ...
         '\n\nIntroduzca el nombre del puerto al que se desea conectar' ...
         ' (COM4,COM5...).']); %Mensaje de error
         uialert(app.UIFigure,mensaje,'Error'); %Muestra el mensaje de error
         app.Iniciar.Value=0; %Vuelve al estado apagado
         app.Iniciar.Icon = 'ON3.png'; %Icono botón verde
       else
         app.PuertoSerie.Visible='off'; %Oculta área de escritura puerto COM
         app.ImagenPuerto.Visible='on'; %Muestra icono puerto COM conectado
         app.ExcelDatos.Visible='off'; %Oculta área de escritura nombre fichero
         app.Exportar.Visible='off'; %Oculta botón exportar
         app.MostrarTabla.Value=0; %Desactiva botón tabla
         app.MostrarTabla.Visible='off'; %Oculta botón tabla
         app.Tabla.Visible='off'; %Oculta tabla
         app.s = serialport(char(Puerto),115200,"Timeout",5); %Inicia el puerto serial
         app.ImagenAviso.Visible='off'; %Oculta aviso equipo desconectado
         %Configura callback cada 5 bytes recibidos
         configureCallback(app.s,"byte",5,@app.BytesAvailableFcn);
         flush(app.s);%Limpia buffer entrada y salida
```

```
        %Inicializa propiedades y restaura indicadores
        app.muestra=[];
        app.contador=1;
        app.AD590=[];
        app.TERMOPAR=[];
        app.PT100=[];
        app.NTC=[];
        app.grafAD590=[];
        app.grafTERMOPAR=[];
        app.grafPT100=[];
        app.grafNTC=[];
        app.pos=1;
        app.error=[];
        app.datos=[];
        app.Tabla.Data=app.datos';
        plot(app.UIAxes,0);
        app.Kp=0;
        app.kp.Value=0;
        app.KpIndicador.Value=0;
        app.Ki=0;
        app.ki.Value=0;
        app.KiIndicador.Value=0;
        app.Kd=0;
        app.kd.Value=0;
        app.KdIndicador.Value=0;
        app.Ventilador=1;
        app.ventilador.Value='Off';
        app.sensorPID=1;
        app.Sensor.Value='AD590';
        app.Consigna=0;
        app.consigna.Value=0;
        app.DutyCycleGauge.Value=0;
        %Define ejes iniciales para la gráfica
        app.UIAxes.YLim=[0 150]; %Límites eje Y
        app.UIAxes.YTick=0:10:150; %Líneas eje Y
        app.UIAxes.YTickLabel=0:10:150; %Números eje Y
        app.UIAxes.XLim=[app.xLimInf app.xLimSup]; %Límites eje X
    end

end
```

Fig. 87 Inicio y paro de la ejecución.

En la función que controla el cambio de estado del botón, en primer lugar, se comprueba el valor del botón. Si el nuevo estado es “apagado”, entonces el icono del botón será verde ya que pulsando en él se pasa al estado “encendido”. Oculta el icono de puerto serie conectado ya que en este estado la comunicación serial permanece desactivada. En su lugar, aparece el área de escritura del puerto serie. También se muestran el área de escritura para el nombre del fichero y el botón “Exportar”. Para realizar estas acciones se modifica la propiedad “Visible” de los objetos “Imagen Puerto”, “Puerto Serie”, “ExcelDatos” y “Exportar”. A continuación, por razones de seguridad, se envían los datos correspondientes para apagar tanto la resistencia calefactora como el ventilador y se cierra la comunicación serial. Por la misma razón, se muestra un aviso indicando que el equipo y el ordenador no se encuentran conectados.

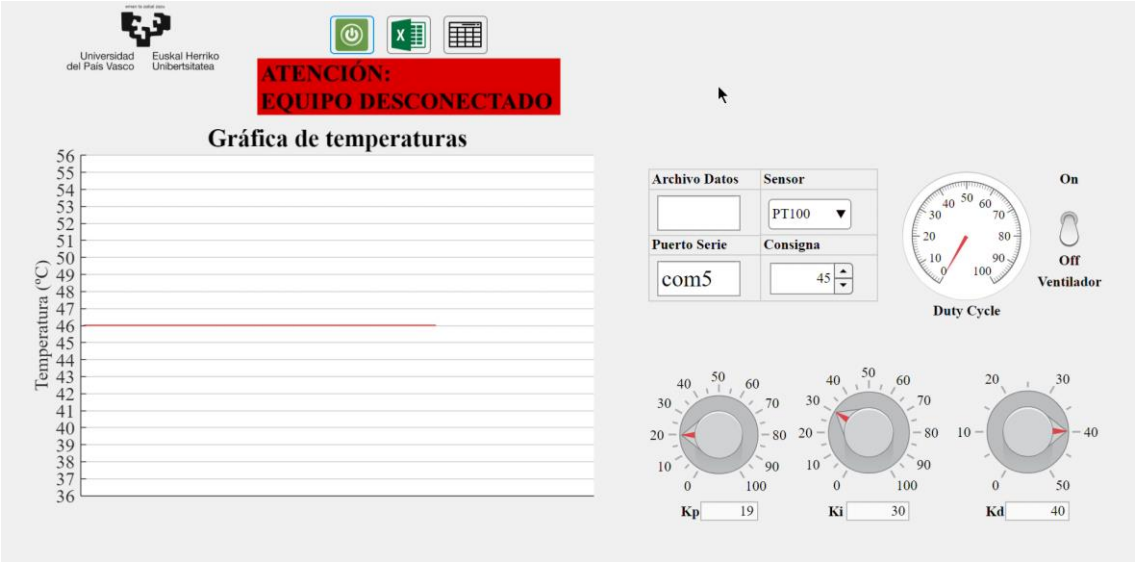


Fig. 88 Estado de desconexión al detener la ejecución.

En este estado, la persona usuaria puede visualizar la tabla con los datos o generar el fichero Excel con los mismos.

En caso de conmutar al estado "encendido", el icono del botón pasa a ser rojo. Seguido, se recoge el nombre del puerto serie y se compara mediante la función *strcmp* con una cadena de caracteres vacía para comprobar que se ha introducido un nombre. En caso de no contener ningún nombre, se emplean las funciones *sprintf* y *uialert* para imprimir el siguiente mensaje de error:

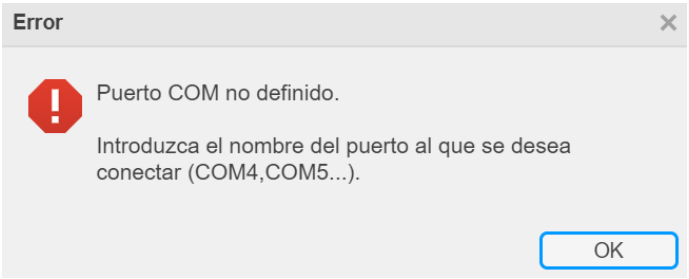


Fig. 89 Mensaje de error.

A continuación, se regresa al estado "apagado" forzando el valor de la propiedad "Iniciar" a 0. En caso de tratarse de una cadena no vacía, se oculta el cuadro de texto y aparece la imagen del puerto serie. También se ocultan los objetos relacionados con el almacenamiento de datos y el mensaje de aviso. A continuación, mediante la función *serialport* se configura el puerto serial a una velocidad de 115200 baudios y un tiempo de respuesta máximo de 5 segundos. También se configura el callback denominado *BytesAvailableFcn*. Esta función se ejecutará cuando en el buffer de entrada se hayan recibido al menos 5 bytes. Se vacía el buffer de entrada y salida mediante la función *flush* para asegurar que los datos leídos no son anteriores al inicio de la comunicación actual. Finalmente, se definen los valores por defecto de cada una de las propiedades y se restauran los indicadores.

Mostrar Tabla

Este botón permite mostrar u ocultar la tabla con los datos almacenados.

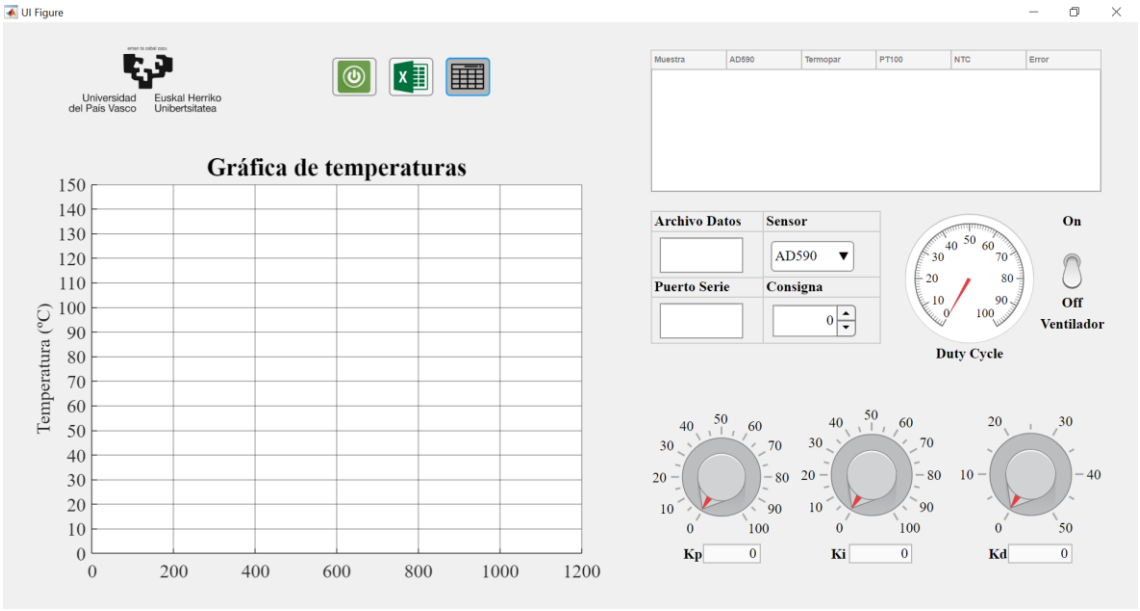


Fig. 90 Interfaz gráfica con tabla de datos visible.

```
    % Value changed function: MostrarTabla
    function MostrarTablaValueChanged(app, event)
        if app.MostrarTabla.Value==1 %Mostrar tabla
            app.Tabla.Visible='on';
        else %Ocultar tabla
            app.Tabla.Visible='off';
        end
    end
end
```

Fig. 91 Control de la visibilidad de la tabla de datos.

Al pulsar el botón, se ejecuta la función asociada a este evento. Si el valor del botón es un 1, se modifica la propiedad “visible” de la tabla, donde se indicará con la cadena ‘on’ que esta debe mostrarse. De lo contrario, se indica con ‘off’ que la tabla debe ocultarse.

Archivo Datos

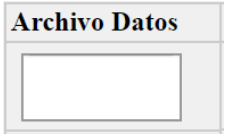


Fig. 92 Campo para el nombre del fichero a exportar.

El texto introducido en este campo determina el nombre del fichero Excel generado con los datos recibidos.

Exportar

Fig. 93 Botón para exportar datos en fichero Excel.

El botón "Exportar" es el encargado de generar el archivo Excel con los datos del ensayo y con el nombre indicado por la usuaria en el campo Archivo Datos. Los datos a escribir en el archivo Excel mantienen la estructura de la tabla que se muestra en la aplicación. Además, indica con un mensaje el éxito de la operación o, en su caso, un mensaje de error.

```
% Button pushed function: Exportar
function ExportarButtonPushed(app, event)
    Nombre=app.ExcelDatos.Value;
    if strcmp(Nombre,'') %Campo vacío
        uialert(app.UIFigure,'Nombre del archivo no definido.','Error'); %Mensaje error
    else
        NombreXLS=strcat(Nombre,'.xls'); % Concatena extensión de archivo
        writematrix(app.datos',NombreXLS); % Escribe los datos en el fichero
        app.ExcelDatos.Value=''; %Vacía campo Nombre
        %Mensaje operación completada
        uialert(app.UIFigure,'Datos correctamente exportados.','Info','Icon','success');
    end
end
```

Fig. 94 Gestión del fichero a exportar.

Presionar el botón "Exportar" llama directamente a la función. En ella, se recoge en la variable local "Nombre" el nombre del archivo Excel a generar. Este nombre debe haber sido escrito anteriormente en el área de texto denominado "Archivo Datos".

Para comprobar que se ha indicado un nombre para el archivo, se emplea la función *strcmp*. Esta función compara las dos cadenas de caracteres que recibe como argumentos. Se compara la variable "Nombre" con una cadena vacía. En caso de que estas coincidan, muestra un mensaje de error indicando que debe definirse el nombre previamente.

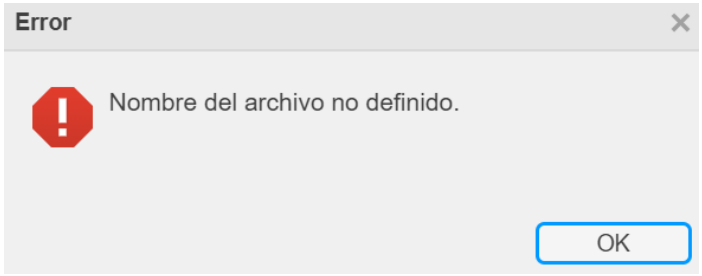


Fig. 95 Mensaje de error.

Al aceptar el mensaje de error, la aplicación vuelve al estado anterior permitiendo así la escritura del nombre.

En caso de acceder a la función con el nombre previamente definido, se emplea la función *strcat* para concatenar la extensión correspondiente al archivo Excel. A continuación, la función *writematrix* realiza la escritura de una matriz en un fichero. En este caso, la matriz es la traspuesta de la matriz "Datos" y el nombre completo del fichero a generar se encuentra en la variable

NombreXLS. Una vez generado el fichero, se vacía el campo de escritura del nombre y finalmente se muestra el mensaje indicando que la operación se ha realizado.

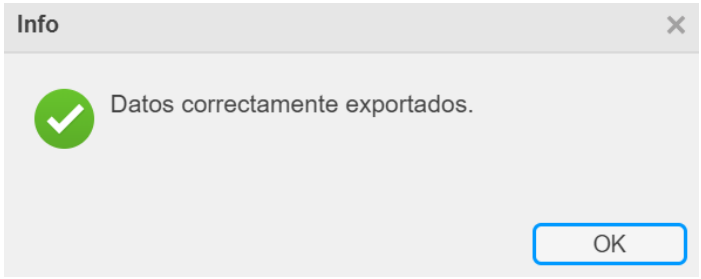


Fig. 96 Mensaje informativo.

Sensor

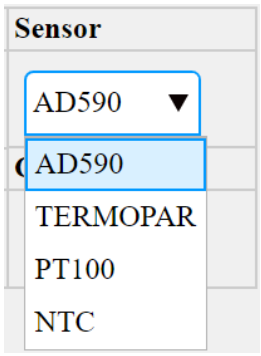


Fig. 97 Menú desplegable para la selección del sensor.

Al pulsar en él se despliega una lista con los nombres de los sensores disponibles. La temperatura del sensor seleccionado se emplea como entrada de referencia en el control PID. La selección del sensor también determina la traza a visualizar en la gráfica de temperaturas.

```
% Value changed function: Sensor
function SensorValueChanged(app, event)
    switch app.Sensor.Value %Asigna el número correspondiente al sensor
        case 'AD590'
            app.sensorPID=1;
        case 'TERMOPAR'
            app.sensorPID=2;
        case 'PT100'
            app.sensorPID=3;
        case 'NTC'
            app.sensorPID=4;
        otherwise
            app.sensorPID=1;
     end
end
```

Fig. 98 Asignación del sensor seleccionado.

El código asociado se ejecuta cuando se detecta un cambio en el valor del objeto “Sensor”. Este valor es una cadena de caracteres. El sensor seleccionado será enviado mediante comunicación serial a la placa Arduino en forma de byte por lo que se asigna el número correspondiente mediante la sentencia *switch* y se almacena el valor en la variable “sensorPID”.

Consigna

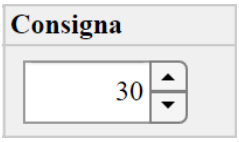


Fig. 99 Campo para el valor de consigna.

Este elemento permite introducir por teclado el valor numérico de la consigna de temperatura. También es posible incrementar o decrementar dicho valor mediante los cursores del mismo.

```
% Value changed function: consigna
function consignaValueChanged(app, event)
    app.Consigna = app.consigna.Value; %Almacena el valor de consigna
end
```

Fig. 100 Almacenamiento del valor de consigna.

La función se ejecuta al modificar el valor de consigna. En ella, se almacena el valor en la propiedad "Consigna" de la aplicación para posteriormente enviar la información a la placa Arduino.

Constantes PID

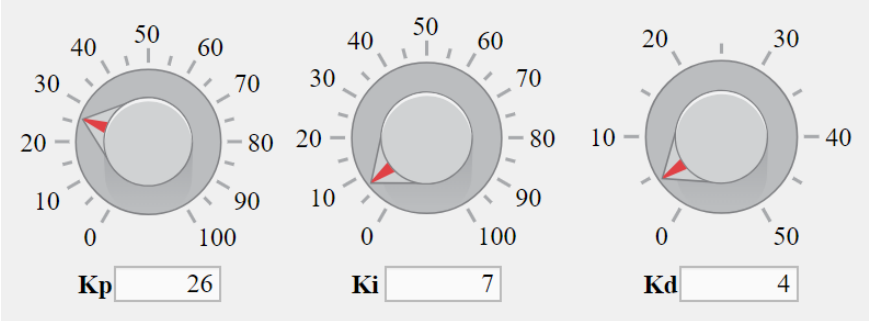


Fig. 101 Mandos para el control PID.

Mediante estos mandos de control se introducen las constantes proporcional, integral y derivativa. Con el fin de mejorar la precisión, cada mando se acompaña de un indicador numérico donde se muestra el valor asignado.

```
% Value changed function: kp
function kpValueChanged(app, event)
    app.Kp = app.kp.Value; %Almacena el valor
    app.KpIndicador.Value=app.Kp; %Muestra el valor en el indicador
end

% Value changed function: ki
function kiValueChanged(app, event)
    app.Ki = app.ki.Value; %Almacena el valor
    app.KiIndicador.Value=app.Ki; %Muestra el valor en el indicador
end

% Value changed function: kd
function kdValueChanged(app, event)
    app.Kd =app.kd.Value; %Almacena el valor
    app.KdIndicador.Value=app.Kd; %Muestra el valor en el indicador
end
```

Fig. 102 Actualización de las variables e indicadores asociados a los mandos de control PID.

Cada mando de control está asociado a una función. En cada una de ellas se almacena el valor en la propiedad correspondiente y se actualiza el valor del indicador.

Ventilador

Fig. 103 Interruptor asociado al ventilador.

Se trata de un interruptor que permite el encendido y apagado del ventilador.

```
% Value changed function: ventilador
function ventiladorValueChanged(app, event)
    if strcmp(app.ventilador.Value,'On')
        app.Ventilador=0; %Encender ventilador
    else
        app.Ventilador=1; %Apagar ventilador
    end
end
```

Fig. 104 Control del ventilador.

Al detectar un cambio en el valor del interruptor, se ejecuta la función asociada a dicho evento. El valor del interruptor es una cadena de caracteres que puede tomar los valores ‘On’ u ‘Off’ por

lo que para conocer el estado del interruptor se ha de comparar el valor del interruptor con alguno de los valores posibles. En caso de que el valor coincida con la cadena 'On', el ventilador debe encenderse por lo que en la propiedad "Ventilador" se almacena un 0. De lo contrario, se almacena un 1. Estos valores lógicos serán enviados a la placa Arduino y procesados por esta, por lo que se emplea lógica inversa ya que el pin asociado a este interruptor se define como INPUT_PULLUP.

Monitorización del sistema

Los componentes relacionados con la monitorización del sistema son aquellos que permiten observar el estado en el que se encuentra. La información del estado del sistema se obtiene de los datos recibidos desde la placa Arduino. Por lo tanto, estos elementos son los encargados de representar de forma visual dichos datos.

Fig. 105 Gráfica de temperaturas.

La gráfica de temperaturas muestra la temperatura en el momento del muestreo del sensor seleccionado. En el eje vertical se tiene la temperatura, cuyos valores están comprendidos en el rango de 0 a 150ºC inicialmente, si bien durante la ejecución del programa se ajusta en rango de manera que se pueda observar con mayor precisión la temperatura indicada por el sensor.

En el eje horizontal no se muestran los valores del número de muestra asociado a la temperatura ya que ello implica trazar una gran cantidad de datos y se ha optado por ofrecer una representación gráfica de la temperatura sin que afecte al rendimiento del programa. No obstante, es posible conocer el número de muestra asociado a cada temperatura y, conocido el tiempo de muestreo, el tiempo a través de los datos almacenados.

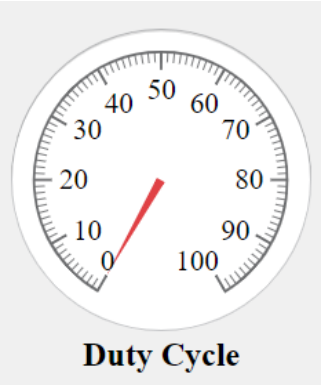


Fig. 106 Indicador del ciclo de trabajo.

Además de las temperaturas, también se recibe información acerca del ciclo de trabajo aplicado en la señal PWM encargada de controlar la resistencia calefactora. Se trata de un componente visual en el que se muestra la información en porcentaje, por lo que puede tomar valores de 0 a 100.

Gestión de la aplicación

La función *startupFcn* es opcional. Se ejecuta automáticamente al iniciar la aplicación, una vez creados los componentes. En este caso, se ha empleado para gestionar la visibilidad de algunas imágenes y componentes.

```
% Code that executes after component creation
function startupFcn(app)
    app.PuertoSerie.Visible='on'; %Muestra campo escritura puerto serie
    app.ImagenPuerto.Visible='off'; %Oculta imagen puerto serie conectado
    app.ImagenAviso.Visible='off'; %Oculta imagen aviso equipo desconectado
end
```

Fig. 107 Gestión de la visibilidad previo al inicio de la aplicación.

Gestión de los datos

```
function BytesAvailableFcn(app,~,~)
    %Lectura y almacenamiento de los bytes enviados por Arduino
    app.AD590(app.contador)=read(app.s,1,"uint8");
    app.TERMOPAR(app.contador)=read(app.s,1,"uint8");
    app.PT100(app.contador)=read(app.s,1,"uint8");
    app.NTC(app.contador)=read(app.s,1,"uint8");
    app.DutyCycleGauge.Value=read(app.s,1,"uint8");
    %Selección valores a mostrar
    app.grafAD590(app.pos)=app.AD590(app.contador);
    app.grafTERMOPAR(app.pos)=app.TERMOPAR(app.contador);
    app.grafPT100(app.pos)=app.PT100(app.contador);
    app.grafNTC(app.pos)=app.NTC(app.contador);
    switch app.sensorPID %Muestra datos del sensor seleccionado en la gráfica
        case 1
            app.error(app.contador)=app.Consigna-app.AD590(app.contador); %Calcula el error
            %Ajuste del eje vertical. Muestra datos en el centro
            yLimInf=app.AD590(app.contador)-10;
            yLimSup=app.AD590(app.contador)+10;
            app.UIAxes.YLim=[yLimInf yLimSup];
            app.UIAxes.YTick=yLimInf:yLimSup;
            app.UIAxes.YTickLabel=yLimInf:yLimSup;
            plot(app.UIAxes,app.grafAD590,'m'); %Traza línea magenta en la gráfica
        case 2
            app.error(app.contador)=app.Consigna-app.TERMOPAR(app.contador); %Calcula el error
            %Ajuste del eje vertical. Muestra datos en el centro
            yLimInf=app.TERMOPAR(app.contador)-10;
            yLimSup=app.TERMOPAR(app.contador)+10;
            app.UIAxes.YLim=[yLimInf yLimSup];
            app.UIAxes.YTick=yLimInf:yLimSup;
            app.UIAxes.YTickLabel=yLimInf:yLimSup;
            plot(app.UIAxes,app.grafTERMOPAR,'b'); %Traza línea azul en la gráfica
        case 3
            app.error(app.contador)=app.Consigna-app.PT100(app.contador); %Calcula el error
            %Ajuste del eje vertical. Muestra datos en el centro
            yLimInf=app.PT100(app.contador)-10;
            yLimSup=app.PT100(app.contador)+10;
            app.UIAxes.YLim=[yLimInf yLimSup];
            app.UIAxes.YTick=yLimInf:yLimSup;
            app.UIAxes.YTickLabel=yLimInf:yLimSup;
            plot(app.UIAxes,app.grafPT100,'r'); %Traza línea roja en la gráfica
```

```
        case 4
            app.error(app.contador)=app.Consigna-app.NTC(app.contador); %Calcula el error
            %Ajuste del eje vertical. Muestra datos en el centro
            yLimInf=app.NTC(app.contador)-10;
            yLimSup=app.NTC(app.contador)+10;
            app.UIAxes.YLim=[yLimInf yLimSup];
            app.UIAxes.YTick=yLimInf:yLimSup;
            app.UIAxes.YTickLabel=yLimInf:yLimSup;
            plot(app.UIAxes,app.grafNTC,'k'); %Traza línea negra en la gráfica
        otherwise
            app.error(app.contador)=app.Consigna-app.AD590(app.contador); %Calcula el error
            %Ajuste del eje vertical. Muestra datos en el centro
            yLimInf=app.AD590(app.contador)-10;
            yLimSup=app.AD590(app.contador)+10;
            app.UIAxes.YLim=[yLimInf yLimSup];
            app.UIAxes.YTick=yLimInf:yLimSup;
            app.UIAxes.YTickLabel=yLimInf:yLimSup;
            plot(app.UIAxes,app.grafAD590,'m'); %Traza línea magenta en la gráfica
    end
    app.pos=app.pos+1; %Actualiza posición
    if app.pos>240 %Supera el rango del eje X
        app.pos=1; %Resetea posición
        %Vacía arrays de temperaturas para gráfica
        app.grafAD590=[];
        app.grafTERMOPAR=[];
        app.grafPT100=[];
        app.grafNTC=[];
    end
    app.muestra(app.contador)=app.contador; %Guarda número de muestra
    %Genera matriz Datos
    app.datos=[app.muestra;app.AD590;app.TERMOPAR;app.PT100;app.NTC;app.error];
    app.Tabla.Data=app.datos'; %Guarda en la tabla la traspuesta de Datos
    app.contador=app.contador+1; %Actualiza valor del contador
    %Envio de la información de las entradas a Arduino
    write(app.s,app.sensorPID,"uint8");
    write(app.s,app.Consigna,"uint8");
    write(app.s,app.Kp,"uint8");
    write(app.s,app.Ki,"uint8");
    write(app.s,app.Kd,"uint8");
    write(app.s,app.Ventilador,"uint8");
end
```

Fig. 108 Comunicación serie y almacenamiento de datos.

La función *BytesAvailableFcn* es la encargada de gestionar todo lo relacionado con el intercambio de datos entre la placa Arduino y la interfaz gráfica. Esta función se ejecuta cuando en el buffer serial se hayan recibido al menos cinco bytes. En primer lugar, se realiza la lectura de dicha información. Para ello, se emplea la función *read* a la cual se le indica el puerto serial, el número de datos a leer y el tipo de dato. En este caso, se trata de datos de tipo entero sin signo de 8 bits que serán leídos uno a uno y almacenados en sus correspondientes propiedades. En el caso de las temperaturas, estas se almacenan en vectores donde cada temperatura se relaciona con el número de muestra. El ciclo de trabajo es una variable numérica que no se almacena, sino que se representa directamente en el indicador accediendo a la propiedad "Value" de este. A continuación, se guardan las lecturas en otros vectores. Se trata de vectores con una capacidad de 240 celdas. Estos vectores son los que posteriormente se trazan en la gráfica 2D ya que de esta manera se obtiene un mayor rendimiento de programa. Mediante la sentencia *switch…case* se determina el sensor seleccionado y posteriormente se calcula el error. Esto es, la diferencia entre la temperatura deseada (consigna) y la temperatura de dicho sensor. Seguido, se realiza el ajuste

del eje vertical en función del valor a mostrar en la gráfica, de manera que este se sitúe en el centro. Para mostrar los datos en la gráfica de temperaturas se emplea la función *plot* en la cual se indica el gráfico, los datos y el color de la traza.

De cara a la siguiente ejecución de la función, es necesario incrementar la posición del vector en la que se almacena el valor recibido. En caso de que la posición supere el valor máximo, se comienza de nuevo desde la posición 1 y se borran los vectores a mostrar en la gráfica.

La variable "contador" contiene el número de muestra. Esta se almacena en el vector "muestra" que formará parte de la información contenida en la tabla de datos. Una vez recopilados los datos, se genera la matriz "datos". En cada fila de esta matriz se almacenan los vectores "muestra", "AD590", "TERMOPAR", "PT100", "NTC" y "error". En la tabla se almacena la traspuesta de la matriz datos de forma que la información aparece por columnas. Seguido, se actualiza la variable "contador". Finalmente, se realiza el envío de los datos recopilados a partir de los componentes de la interfaz gráfica. Para ello, la función a emplear es *write*. En ella se debe indicar el puerto serial, la variable a escribir y el tipo de dato en el que se desea enviar, independientemente del tipo de dato de la variable.

3.5. PROCESO DE FABRICACIÓN Y MONTAJE FINAL

Una de las partes principales es la base del horno, formada por una estructura de acero. En las paredes de esta se encuentran los conectores de red y USB, así como la toma de tierra. En su interior se sitúan las placas de circuito impreso y la Arduino Mega 2560.

Fig. 109 Estructura de acero.

Para colocar dichos componentes en las paredes de la estructura y fijar las placas de circuito impreso a la base de la misma, se han realizado los correspondientes orificios. A continuación, se han colocado las placas en el interior de la estructura.

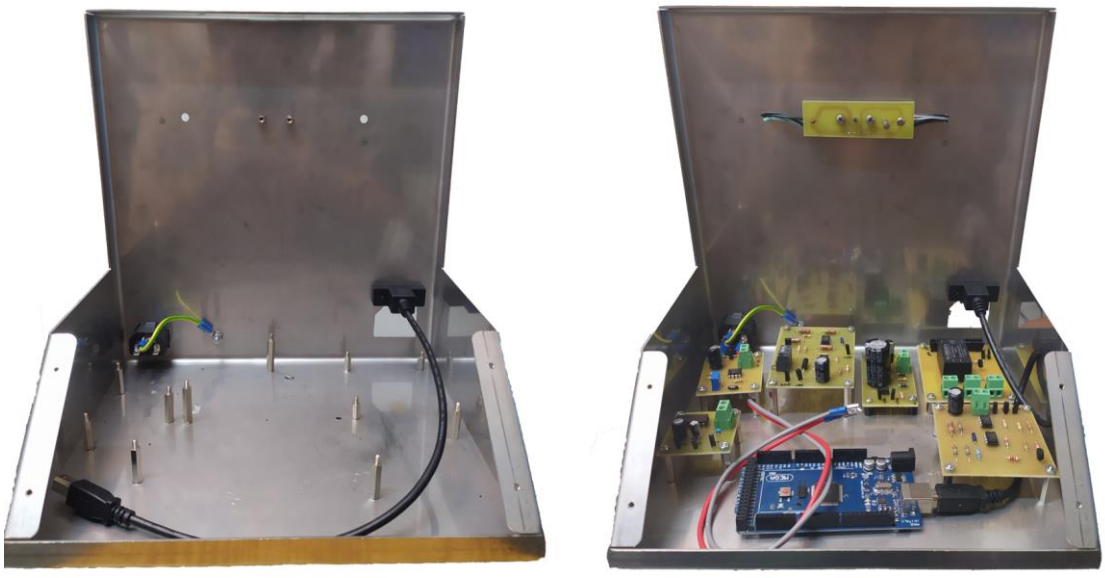


Fig. 110 Fijación de las placas de circuito impreso.

Una vez colocadas las placas, se han realizado las conexiones entre las placas y entre los sensores de temperatura y su circuito de acondicionamiento.

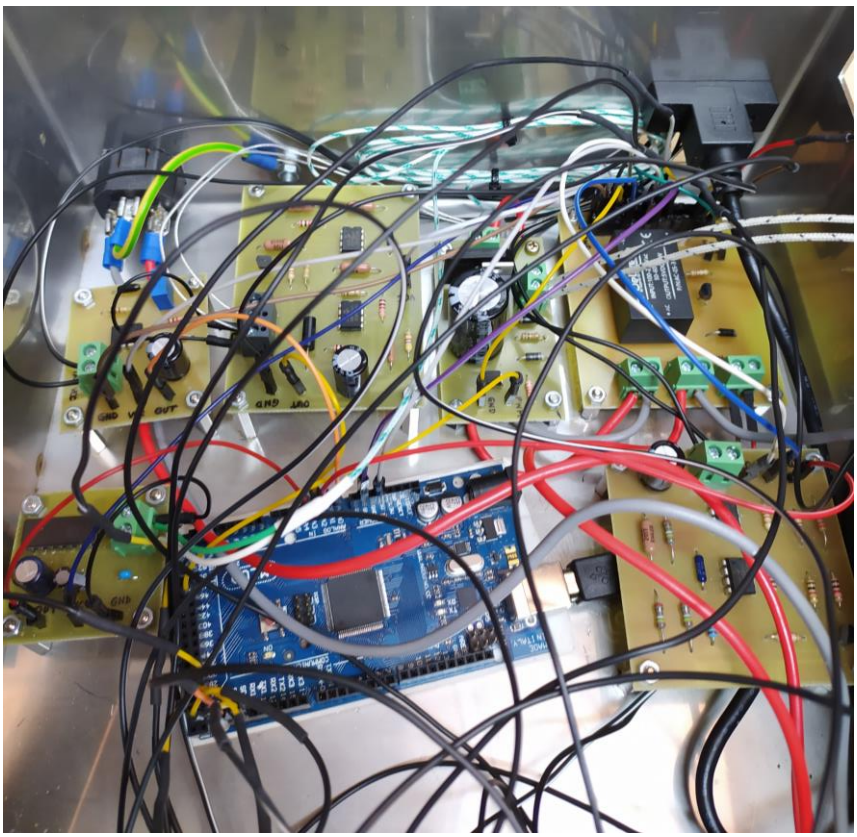


Fig. 111 Conexiones eléctricas.

A continuación, se han colocado los componentes del panel frontal y se han soldado los cables para posteriormente realizar las conexiones entre estos y la placa Arduino.

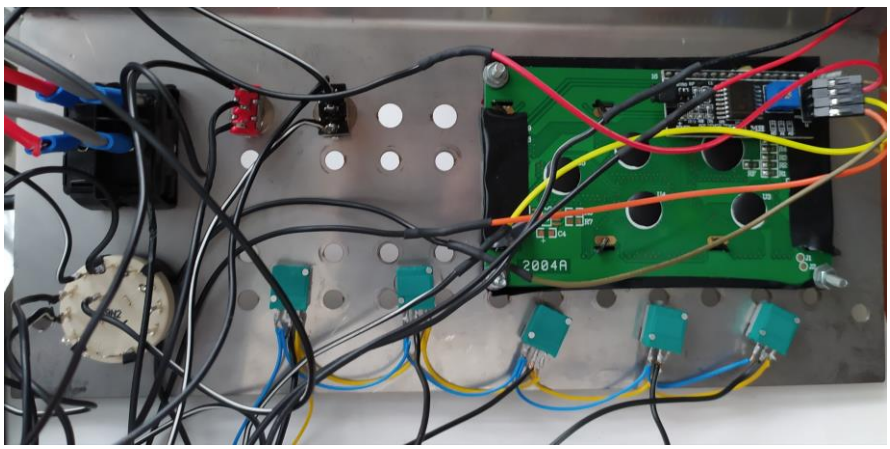


Fig. 112 Fijación y conexiones de los componentes del panel frontal.

El diseño del panel frontal se ha realizado en el programa Fusion 3D. Posteriormente, se ha fabricado en una impresora 3D. Con intención de facilitar el manejo del equipo, se han colocado nombres junto a los elementos. Para ello, se ha realizado un primer sólido de color blanco donde únicamente se han colocado los orificios para los componentes y la sujeción de la misma. El segundo sólido, en color gris, además de los orificios anteriores contiene los correspondientes a los nombres.

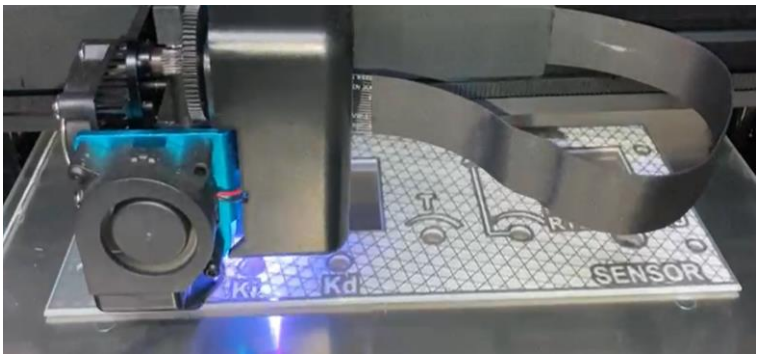


Fig. 113 Fabricación del panel frontal en impresora 3D.

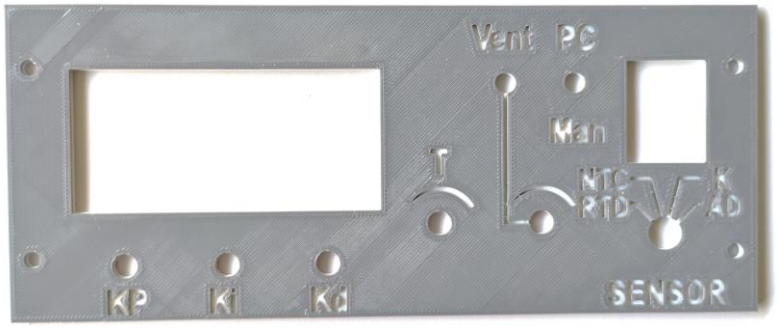


Fig. 114 Resultado final del panel frontal.

En la parte superior de la estructura de acero se encuentra la resistencia calefactora envuelta por la cámara de policarbonato. La cámara sostiene el ventilador que acelera la refrigeración del horno. Más tarde, se han sustituido las maderas que sujetan la resistencia calefactora por dos tubos de aluminio que cubren los tornillos que fijan la resistencia a la estructura de acero.

Fig. 115 Cámara de policarbonato.

Por último, se muestra una imagen representativa del resultado final.

Fig. 116 Resultado final del montaje.

4. RESULTADOS OBTENIDOS

En este apartado se presentan los resultados obtenidos en una prueba realizada con el modo PC. En ella, el horno se encuentra inicialmente a 31ºC y se introduce una consigna de 45ºC. Se selecciona la NTC como sensor de referencia, y los valores introducidos para el control PID son: 19 para la constante proporcional, 49 para la constante integral y 40 para la derivativa.

En esta primera imagen se observan las variables de entrada introducidas desde la aplicación. También se pueden apreciar en la gráfica las lecturas de temperatura recibidas desde el Arduino. Dado que la diferencia entre la temperatura del horno y la consigna es elevada, el ciclo de trabajo es del 100%.

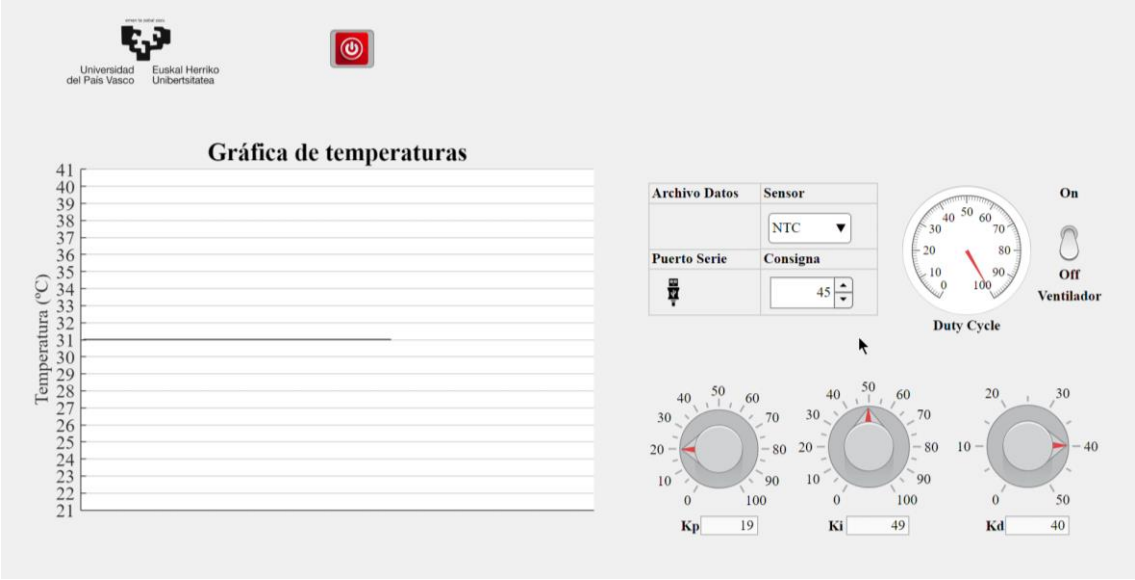


Fig. 117 Inicio del ensayo.

Esta vez se muestra el equipo, donde se puede ver cómo las entradas de la aplicación han sido correctamente recibidas y los datos calculados por el programa de la placa Arduino concuerdan con los datos recibidos en la aplicación.

Fig. 118 Comunicación serie con Arduino.

A continuación, se observa cómo la temperatura aumenta hacia el valor de consigna y el ciclo de trabajo disminuye.

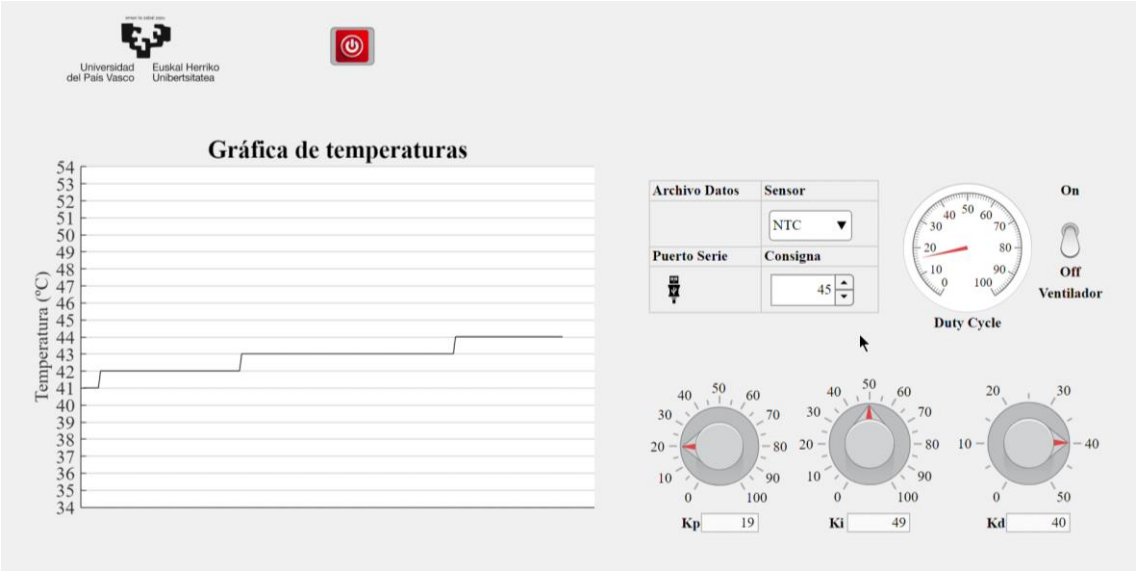


Fig. 119 Control del ciclo de trabajo.

De nuevo, se muestran también los datos en la pantalla del equipo, donde se indica que el error es de 1ºC y el ciclo de trabajo del 16%.

Fig. 120 Datos en el panel frontal de la maqueta.

Una vez alcanzado el valor de consigna, el ciclo de trabajo es del 0% por lo que la resistencia calefactora queda desconectada.

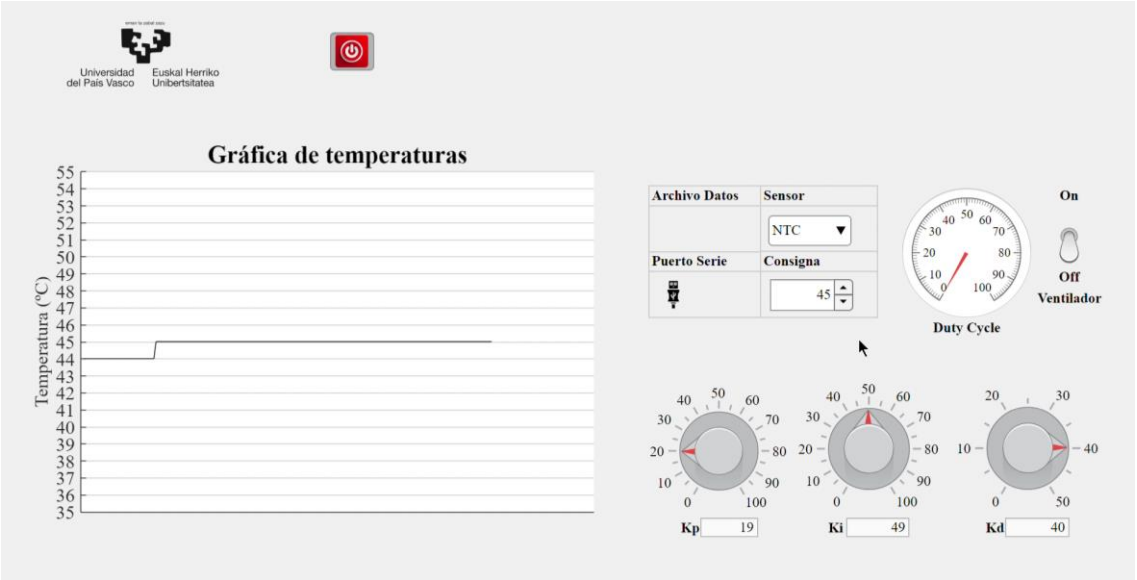


Fig. 121 Alcance del valor de consigna.

Transcurrido un tiempo, se observa que la temperatura del horno ha descendido hasta 2ºC por debajo de la consigna, y que el ciclo de trabajo ha aumentado para evitar que la temperatura siga descendiendo.

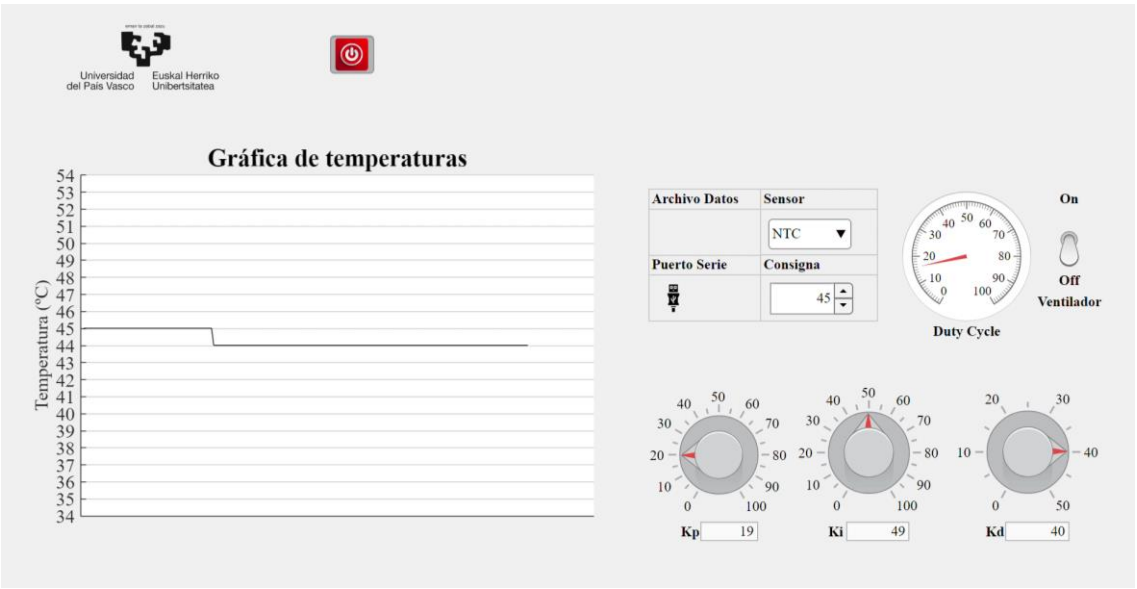


Fig. 122 Enfriamiento y respuesta del sistema.

Para finalizar la prueba, se ha detenido la conexión y se han exportado los datos a un archivo Excel.

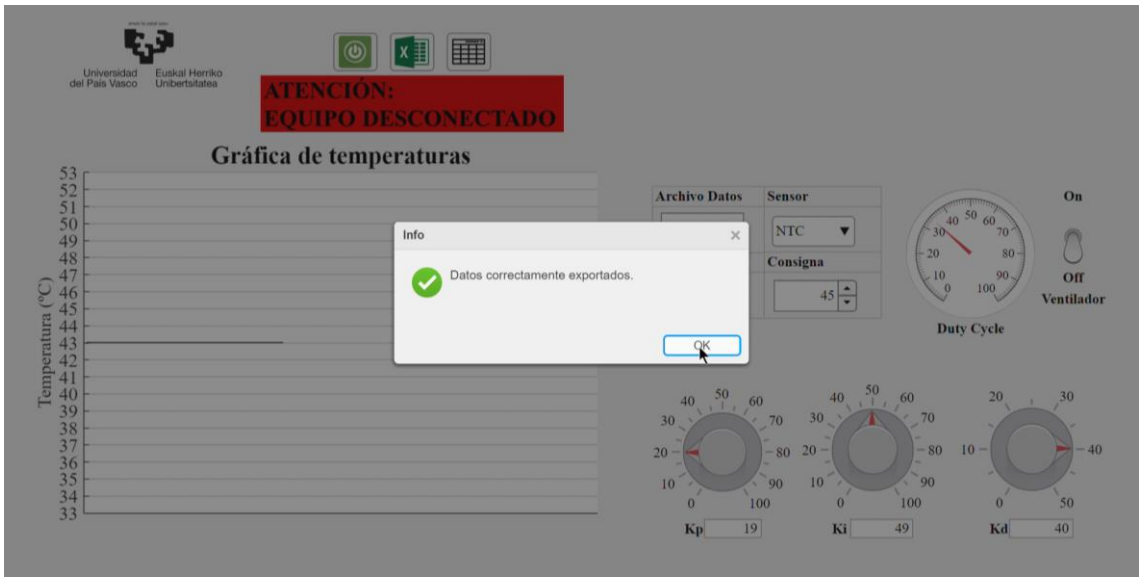


Fig. 123 Datos exportados.

En el archivo generado se pueden analizar los datos recogidos. Se ha realizado una gráfica en la que se muestra la temperatura en el eje vertical y el tiempo en el eje horizontal. Además de las lecturas, también se ha trazado el valor de consigna.

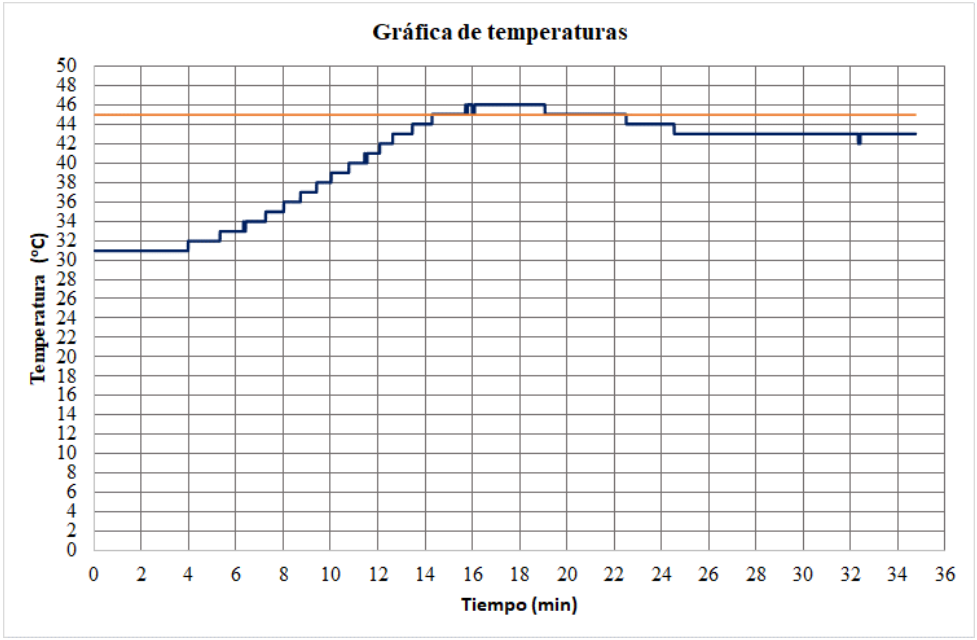


Fig. 124 Representación gráfica del resultado en Excel.

Tal y como se observa en la gráfica, el resultado obtenido presenta un rebose de 1ºC. Esto se debe a que, a pesar de que el ciclo de trabajo se ha comenzado a reducir significativamente antes de alcanzar el valor de consigna, la resistencia calefactora posee una gran inercia térmica. Esto es, pese a estar desconectada, el cuerpo sigue conservando calor. También se aprecia un error de 2ºC aunque no se puede determinar si se trata de las oscilaciones propias del tiempo de establecimiento, o de un error que permanecerá en el estado estacionario. Para ello, sería necesario observar el comportamiento durante un periodo mayor de tiempo.

Universidad del País Vasco Euskal Herriko Unibertsitatea
BILBOKO INGENIARITZA ESKOLA
ESCUELA DE INGENIERÍA DE BILBAO

GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

TRABAJO FIN DE GRADO

DISEÑO Y FABRICACIÓN DE UN SISTEMA DE CONTROL Y MONITORIZACIÓN DE TEMPERATURA

DOCUMENTO III - ASPECTOS ECONÓMICOS

Alumna: España Hernández, Sara

Director: Sainz de Murieta Mangado, Joseba Andoni

Curso: 2020-2021

Fecha: Bilbao, 30 de junio de 2021

En este documento se exponen los costes asociados a la realización del proyecto. Para el planteamiento del presupuesto se han tenido en cuenta el importe relacionado con los componentes electrónicos y la propiedad intelectual. No se han incluido los gastos correspondientes a los materiales reutilizados ni a las licencias de Autocad y Matlab a las que se ha tenido acceso como estudiante.

En la siguiente tabla se presenta el presupuesto de los componentes electrónicos, distribuidos en su mayoría por RS Components.

Tabla 2 Presupuesto.

CONCEPTO	PRECIO UNITARIO IVA INCLUIDO	UNIDADES	PRECIO TOTAL
Calentador de cartucho	38,12 €	1	38,12 €
Diodo	0,27 €	1	0,27 €
Condensador 100uF 400V	0,89 €	1	0,89 €
Rectificador puente de diodos	1,28 €	1	1,28 €
Resistencia 100k 0,5W	0,01 €	3	0,03 €
Diodo zener 15V 3W	0,26 €	1	0,26 €
Optoacoplador	0,29 €	1	0,29 €
Bloque de terminal (2)	0,36 €	6	2,13 €
Resistencia 220 0,5W	0,05 €	1	0,05 €
Conector hembra	0,33 €	16	5,33 €
IGBT	2,19 €	1	2,19 €
OpAmps MCP6022	1,34 €	5	6,68 €
Referencia 2,5V	1,88 €	1	1,88 €
PT100	49,66 €	1	49,66 €
Resistencia 1k	0,15 €	9	1,36 €
Resistencia 25k	1,57 €	4	6,28 €
Resistencia 4k7	0,15 €	3	0,45 €
Resistencia 15k	0,17 €	1	0,17 €
Resistencia 2k5	2,43 €	1	2,43 €
Resistencia 50k	3,87 €	1	3,87 €
Bloque de terminal (3)	0,59 €	1	0,59 €
Resistencia 86k6	1,57 €	1	1,57 €
Condensador 6,8uF	0,24 €	3	0,71 €
NTC 10K 25°C	0,73 €	1	0,73 €
Resistencia 25k	1,57 €	4	6,28 €
Resistencia 2k5	2,43 €	1	2,43 €
Referencia 2,5V	1,88 €	1	1,88 €
Resistencia 5k	0,45 €	1	0,45 €
Resistencia 1,74k	0,07 €	1	0,07 €
AD595	33,98 €	1	33,98 €
Termopar K	8,81 €	1	8,81 €
Condensador 100nF	0,17 €	1	0,17 €
Condensador 470uF	0,10 €	1	0,10 €
Condensador 100uF	0,14 €	1	0,14 €
AD590	20,64 €	1	20,64 €
Resistencia 100Ω	0,10 €	1	0,10 €
Potenciómetro 100Ω	1,83 €	1	1,83 €
Resistencia 953Ω	0,57 €	1	0,57 €
Conector USB	6,10 €	1	6,10 €
Cable USB	3,56 €	1	3,56 €
Interruptores modo y ventilador	1,91 €	2	3,82 €
Conector alimentación	1,37 €	1	1,37 €
Potenciómetros 10k	3,52 €	5	17,60 €
BJT PN2222	0,18 €	1	0,18 €
Ventilador	9,56 €	1	9,56 €
Cable alimentación	3,85 €	1	3,85 €
Interruptor de palanca	1,91 €	2	3,82 €
Interruptor balancín	4,80 €	1	4,80 €
Cables M-M	4,00 €	1	4,00 €
Módulo fuente de alimentación	5,99 €	1	5,99 €
Display LCD	11,24 €	1	11,24 €
Arduino Mega 2560	35,00 €	1	35,00 €
TOTAL			**315,55 €**

En cuanto al presupuesto de la propiedad intelectual y mano de obra, se han tenido en cuenta el número total de horas empleadas en las tareas de análisis teórico, selección de componentes, diseño electrónico, programación del código fuente, realización de pruebas, montaje y redacción de la documentación.

Tabla 3 Propiedad intelectual.

TAREA	TIEMPO DEDICADO h	PRECIO/HORA €/h	IMPORTE TOTAL
Análisis teórico	90	15	1.350 €
Selección de componentes	25	12	300 €
Diseño electrónico	75	18	1.350 €
Programación del código fuente	120	20	2.400 €
Realización de pruebas	45	15	675 €
Montaje	20	10	200 €
Redacción de la documentación	100	15	1.500 €
TOTAL			7.775 €

Con todo, el coste total del proyecto es de OCHO MIL NOVENTA CON CINCUENTA Y CINCO EUROS.

Coste total del proyecto= 8.090,55€

CONCLUSIONES

Con todo lo anterior se puede concluir que este documento describe detalladamente el proceso de diseño y fabricación de la maqueta de control de temperatura y su sistema de monitorización.

El proyecto posee un enfoque tanto teórico como práctico, lo que supone la necesidad de partir del análisis teórico hasta llegar a los resultados finales. Durante el proceso, ha sido necesario realizar tareas de investigación, decidir entre varias alternativas, desarrollar nuevas habilidades y hacer frente a problemas.

Se puede atestiguar que los objetivos inicialmente planteados han sido cumplimentados, consiguiendo un equipo y una aplicación funcionales. Además, se han desarrollado más prestaciones de las que se tenía previsto. En un principio, el cometido del sistema de monitorización no era más que representar gráficamente la lectura de los sensores. Sin embargo, al conocer las prestaciones ofrecidas por Matlab, se decidió profundizar más y sacar el mayor partido posible a la aplicación. Con ello, se ha logrado finalmente desarrollar una aplicación que, además de mostrar gráficamente los datos, permite tener un control total del equipo y almacenar los datos recibidos para su posterior análisis. Esta última característica resulta especialmente interesante para el uso del equipo en la docencia o en la investigación.

Gracias a este proyecto se han conocido nuevas prestaciones de MATLAB que, junto con las placas de desarrollo de Arduino, permiten desarrollar proyectos mucho más completos.

Como mejoras realizables, sería conveniente mejorar el aislamiento de la cámara de policarbonato. Igualmente, para evitar el paso del calor al interior del horno, se debería aislar completamente la sujeción de la resistencia calefactora de la estructura de acero.

BIBLIOGRAFÍA

[1] Instrumentación electrónica. Miguel A. Pérez García ... [et al.]. 2ª ed., 4ª reimp. Madrid: International Thomson Editores Spain Paraninfo, 2008

[2] K. Ogata, Ingeniería de control moderna, Pearson Educación, 2003.

[3] M. Á. P. García, Instrumentación electrónica, Ediciones Paraninfo, 2014.

[4] Implementación de maqueta para el estudio de la regulación y control de procesos térmicos - https://addi.ehu.es/handle/10810/43111

[5] Estudio de procesos de control de temperatura. https://www.alecop.com/equipamiento-didactico/areas/captadores-regulacion-de-procesos-y-automatas-programables/estudio-de-procesos-de-control-de-temperatura-serie-540/

[6] Módulo de ensayos de temperatura. https://www.edibon.com/es/modulo-de-ensayos-de-temperatura#descripciongeneral

[7] Puerto Serial – protocolo y su teoría. https://hetpro-store.com/TUTORIALES/puerto-serial/

[8] Puerto I2C – Puerto, introducción, trama y protocolo. https://hetpro-store.com/TUTORIALES/i2c/

[9] Setting up Library Loader for use with KiCad - https://www.samacsys.com/kicad/

[10] Academia ZerüsS. 2017. Diseño componente para KiCad, Símbolo, footprint y modelo 3D |#1. Youtube. https://www.youtube.com/watch?app=desktop&v=-RchMlOK_2A&t=210s

[11] Academia ZerüsS. 2017. Diseño componente para KiCad, Símbolo, footprint y modelo 3D |#2. Youtube. https://www.youtube.com/watch?app=desktop&v=Gfh8pHBVq6g

[12] Getting Started in KiCad - https://docs.kicad.org/4.0/en/getting_started_in_kicad/getting_started_in_kicad.pdf

[13] ATmega640/V-1280/V-1281/V-2560/V-2561/V [DATASHEET] - https://ww1.microchip.com/downloads/en/devicedoc/atmel-2549-8-bit-avr-microcontroller-atmega640-1280-1281-2560-2561_datasheet.pdf

[14] Diseño, desarrollo e implementación de una maqueta para el estudio de la regulación y el control de procesos térmicos

https://addi.ehu.es/bitstream/handle/10810/29285/TFG_IbarrondoIbai.pdf?sequence=1&isAllowed=y

[15] Características de las placas de Arduino - https://docplayer.es/70502740-Caracteristicas-de-las-placas-arduino.html

[16] Arduino Mega 2560 - https://proyectoarduino.com/arduino-mega-2560/

[17] Administrar código en la vista de código de App Designer - https://www.mathworks.com/help/matlab/creating_guis/app-designer-code-generation.html

[18] Serial Port Interface - https://www.mathworks.com/help/instrument/serial-port-interface.html

[19] configureCallback - https://www.mathworks.com/help/instrument/serialport.configurecallback.html

[20] UIAxes Properties - https://la.mathworks.com/help/matlab/ref/matlab.ui.control.uiaxes-properties.html?searchHighlight=ui%20axes&s_tid=srchtitle#responsive_offcanvas

[21] Uialert - https://la.mathworks.com/help/matlab/ref/uialert.html

[22] Serial available - https://www.arduino.cc/reference/en/language/functions/communication/serial/available/

[23] https://www.arduino.cc/reference/en/language/functions/communication/serial/begin/

[24] https://www.arduino.cc/reference/en/language/functions/communication/serial/read/

[25] https://www.arduino.cc/reference/en/language/functions/communication/serial/println/

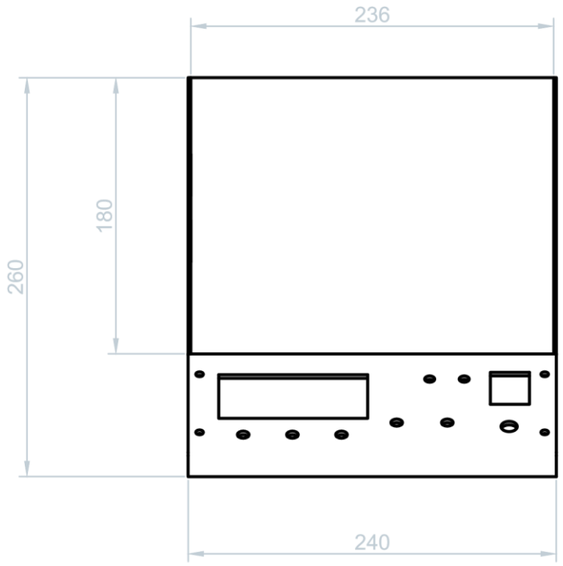
[26] https://www.arduino.cc/reference/en/language/functions/communication/serial/write/

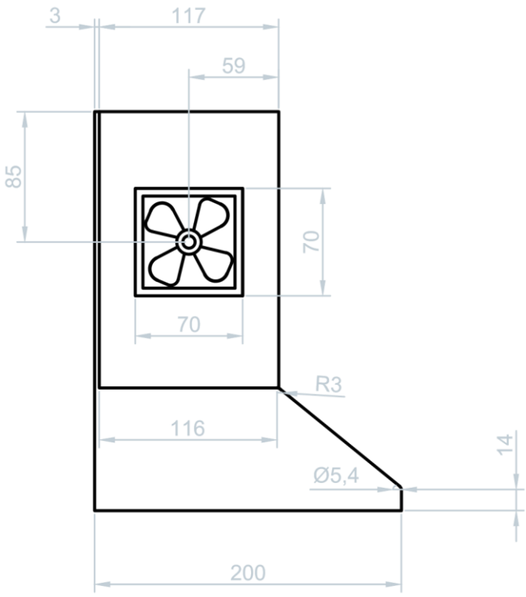
[27] https://www.arduino.cc/reference/en/language/functions/communication/serial/end/

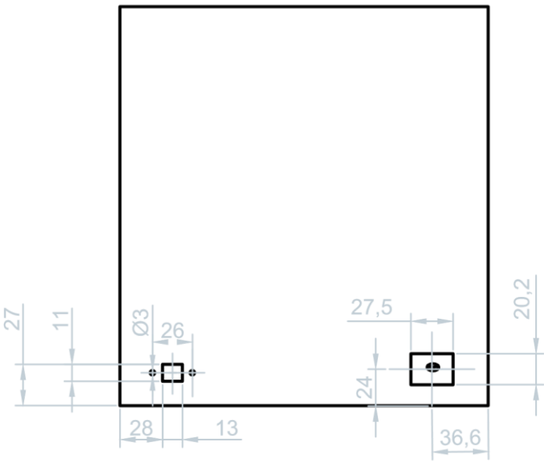
[28] https://www.arduino.cc/reference/en/language/functions/communication/serial/write/

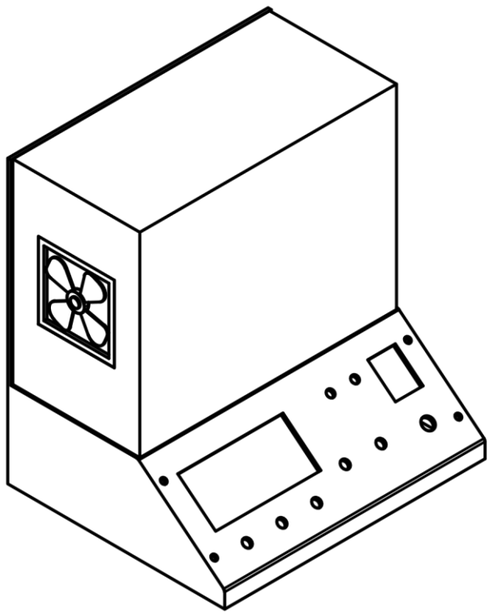
[29] https://www.arduino.cc/reference/en/language/functions/math/map/

ANEXO I: PLANOS









	Fecha/Data:	Nombre/Izena:	Firma/Signa.:
Dibujado/Marraztuko:	8/04/2021	SARA ESPAÑA HERNÁNDEZ	
Comprob./Egiaztatu:			
Dirigido/Zuzenduta:			

UNIVERSIDAD DE PAIS VASCO
EUSKAL HERRIKO UNIBETSITATEA
ESCUELA DE INGENIERÍA DE BILBAO
BILBOKO INGENIARITZA ESKOLA
GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA
INDUSTRIA ELEKTRONIKAREN ETA AUTOMATIKAREN INGENIARITZAKO GRADUA

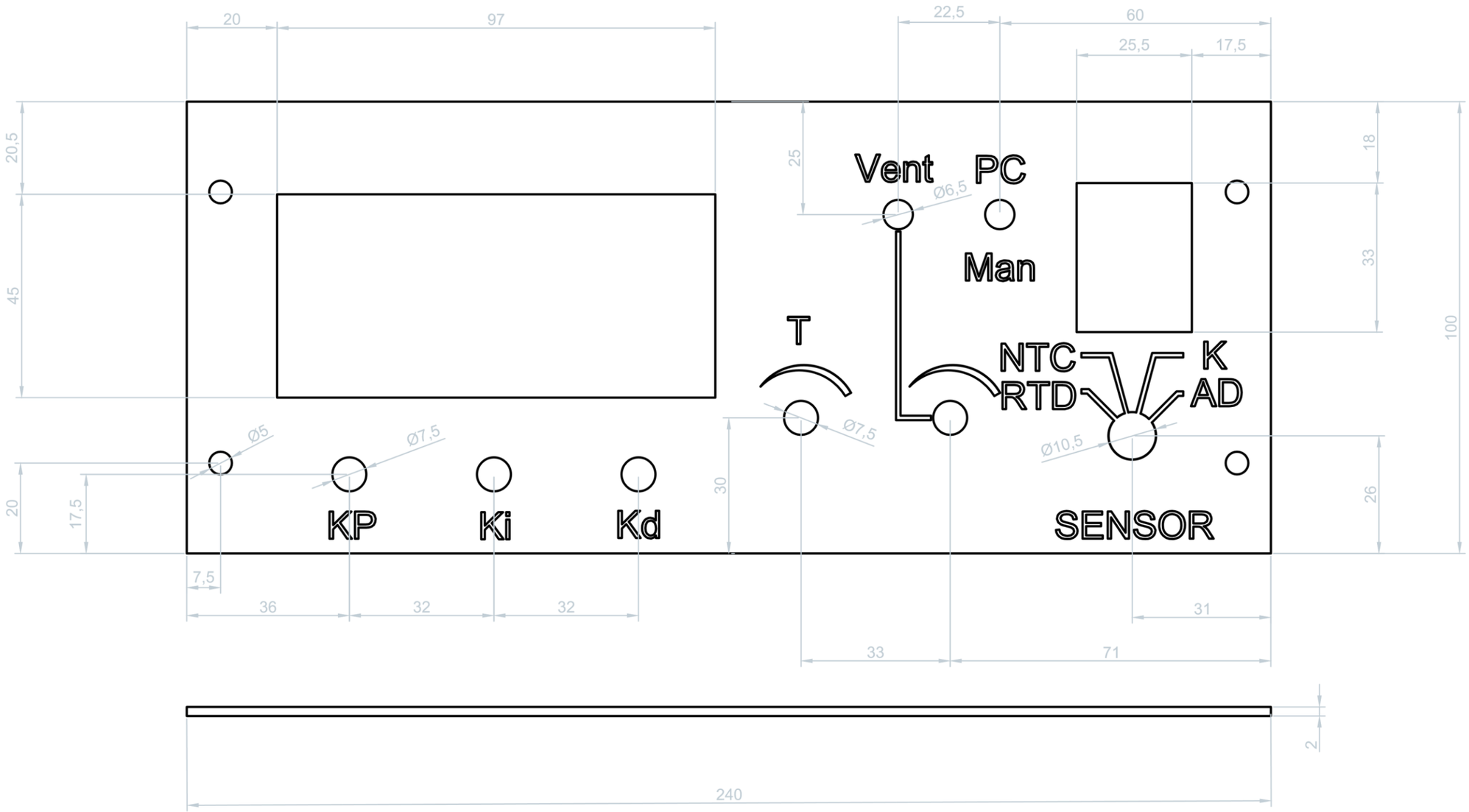
Toleran. gen.:
Perdoiak gen.:
m

Escala/Eskala:
1:4

ESTRUCTURA GENERAL
EGITURA OROKORRA

CONTROL DE TEMPERATURA
TEMPERATURAREN KONTROLA

Plano Nº/Zkia.: 1
Plano Cant./Kop.: 2
Calificación/Kalifikazioa:



	Fecha/Data:	Nombre/Izena:	Firma/Signa.:
Dibujado/Marraztuko:	8/04/2021	SARA ESPAÑA HERNÁNDEZ	
Comprob./Egiaztatu:			
Dirigido/Zuzenduta:			

UNIVERSIDAD DE PAIS VASCO
EUSKAL HERRIKO UNIBETSITATEA
ESCUELA DE INGENIERÍA DE BILBAO
BILBOKO INGENIARITZA ESKOLA
GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA
INDUSTRIA ELEKTRONIKAREN ETA AUTOMATIKAREN INGENIARITZAKO GRADUA

Toleran. gen.:
Perdoiak gen.:
m

Escala/Eskala: 1:1

PANEL FRONTAL
AURREKO PANELA

CONTROL DE TEMPERATURA
TEMPERATURAREN KONTROLA

Plano Nº/Zkia.: 2
Plano Cant./Kop.: 2
Calificación/Kalifikazioa:

ANEXO II: CIRCUITOS Y PCB

Potencia

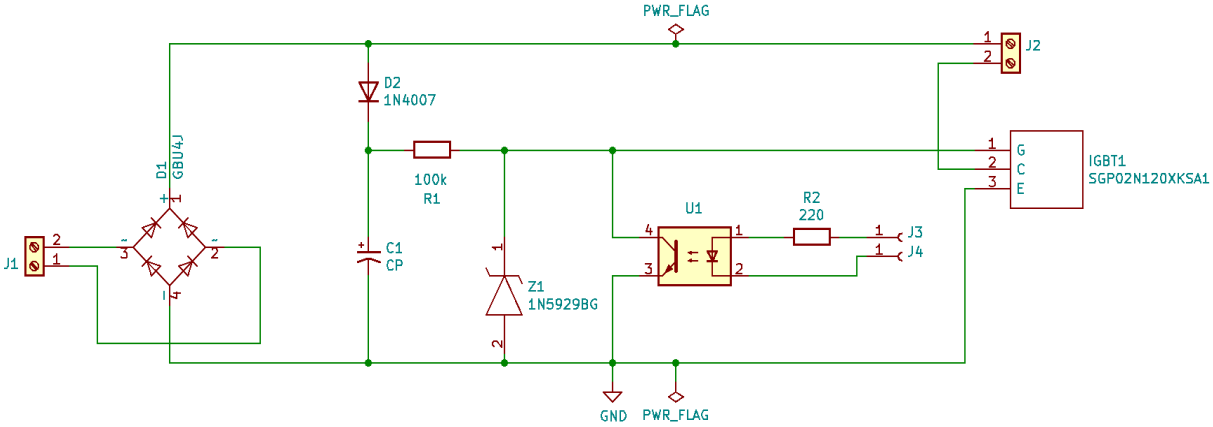


Fig. 125 Circuito de potencia.

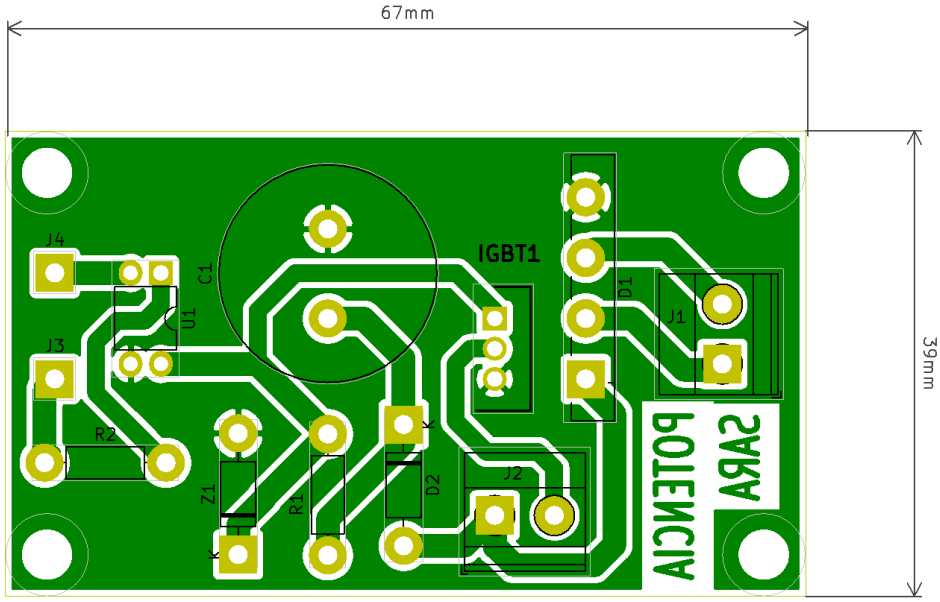


Fig. 126 PCB de potencia.

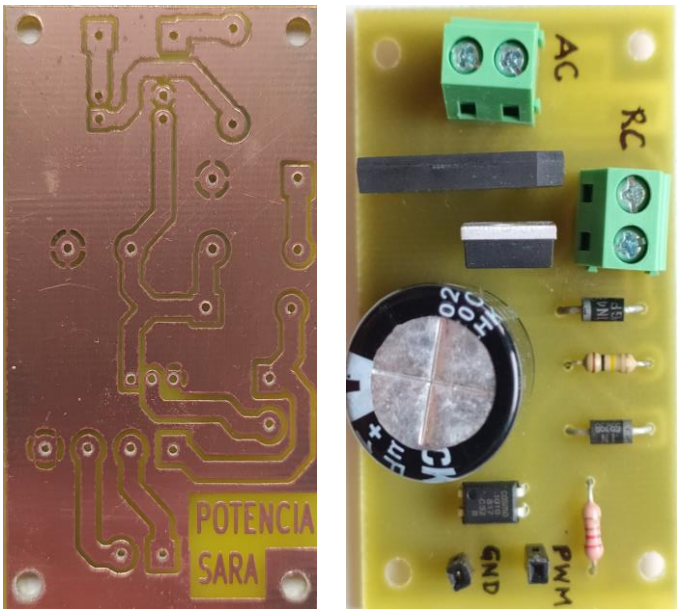


Fig. 127 Resultado final de la placa de potencia.

Control del ventilador y alimentación

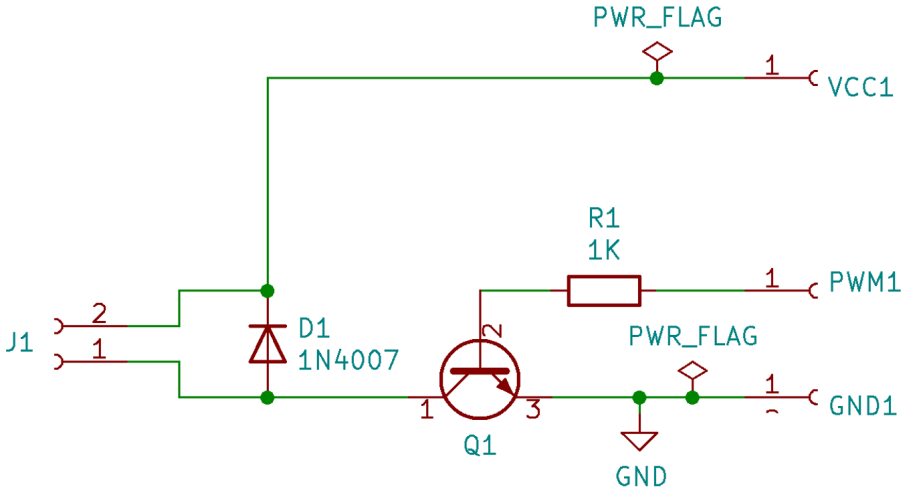


Fig. 128 Circuito de control del ventilador y alimentación.

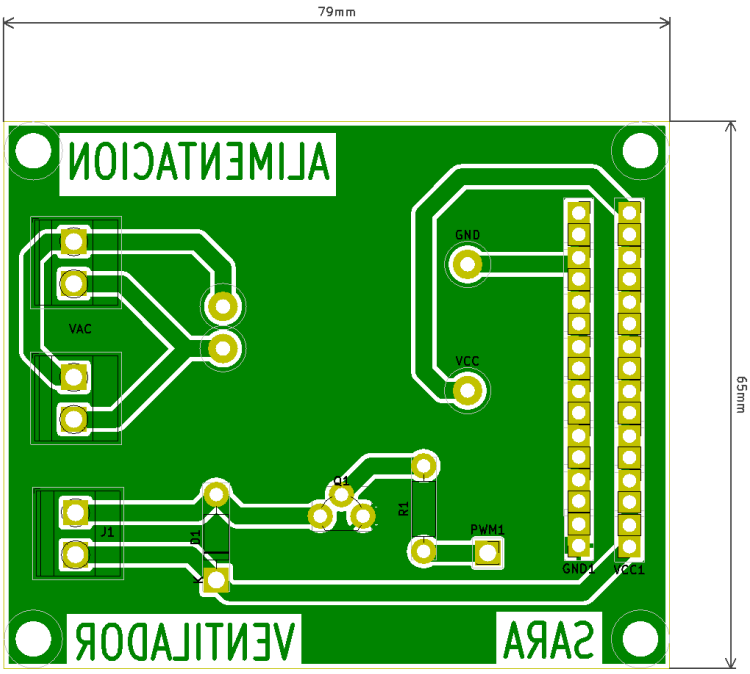


Fig. 129 PCB de alimentación y control del ventilador.

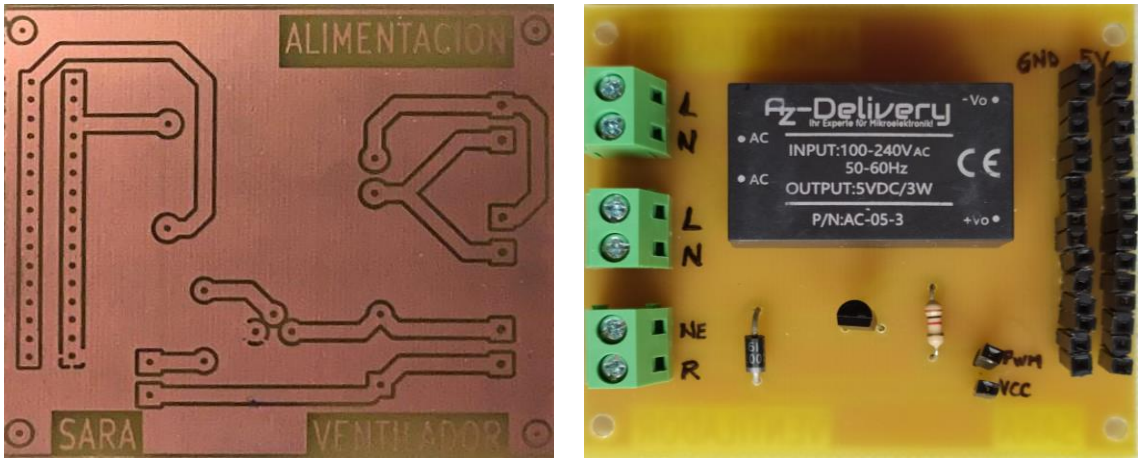


Fig. 130 Resultado final de la PCB de control del ventilador y alimentación.

AD590

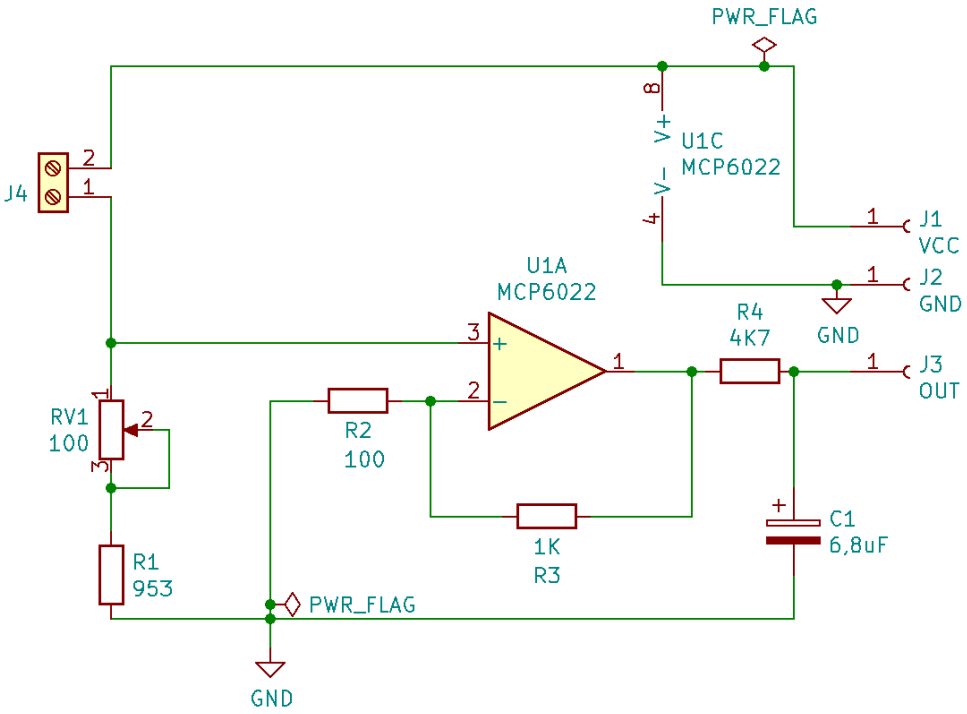


Fig. 131 Circuito del AD590.

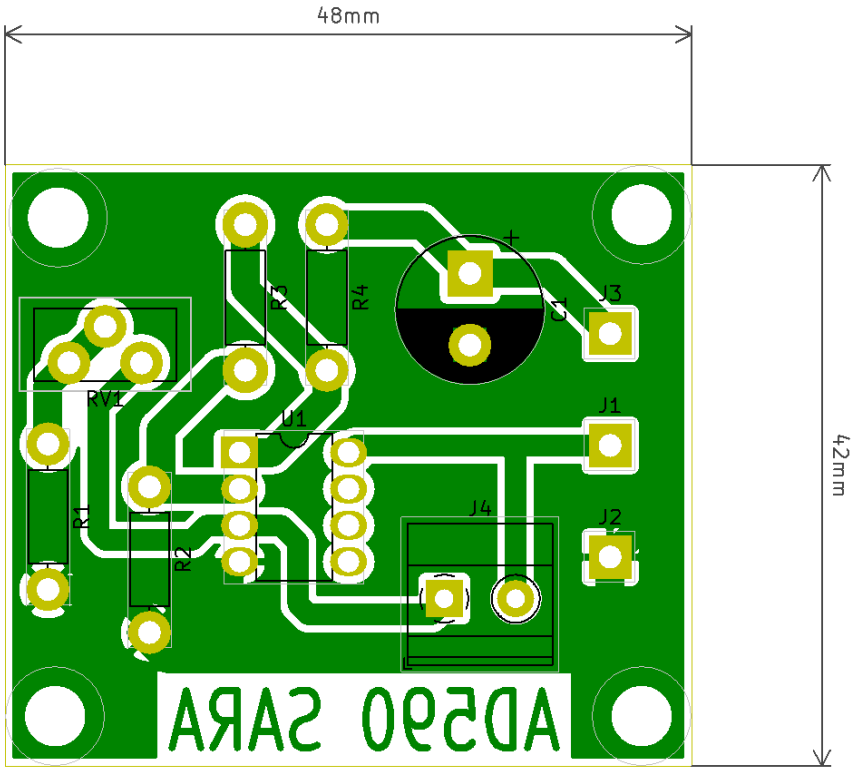


Fig. 132 PCB del AD590.

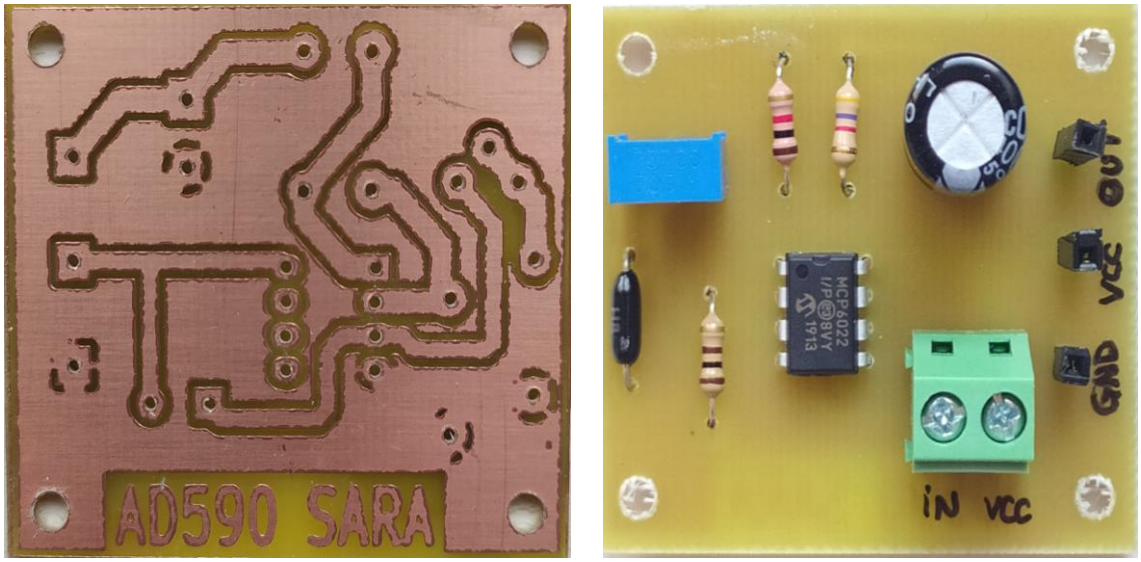


Fig. 133 Resultado final de la PCB del AD590.

Termopar

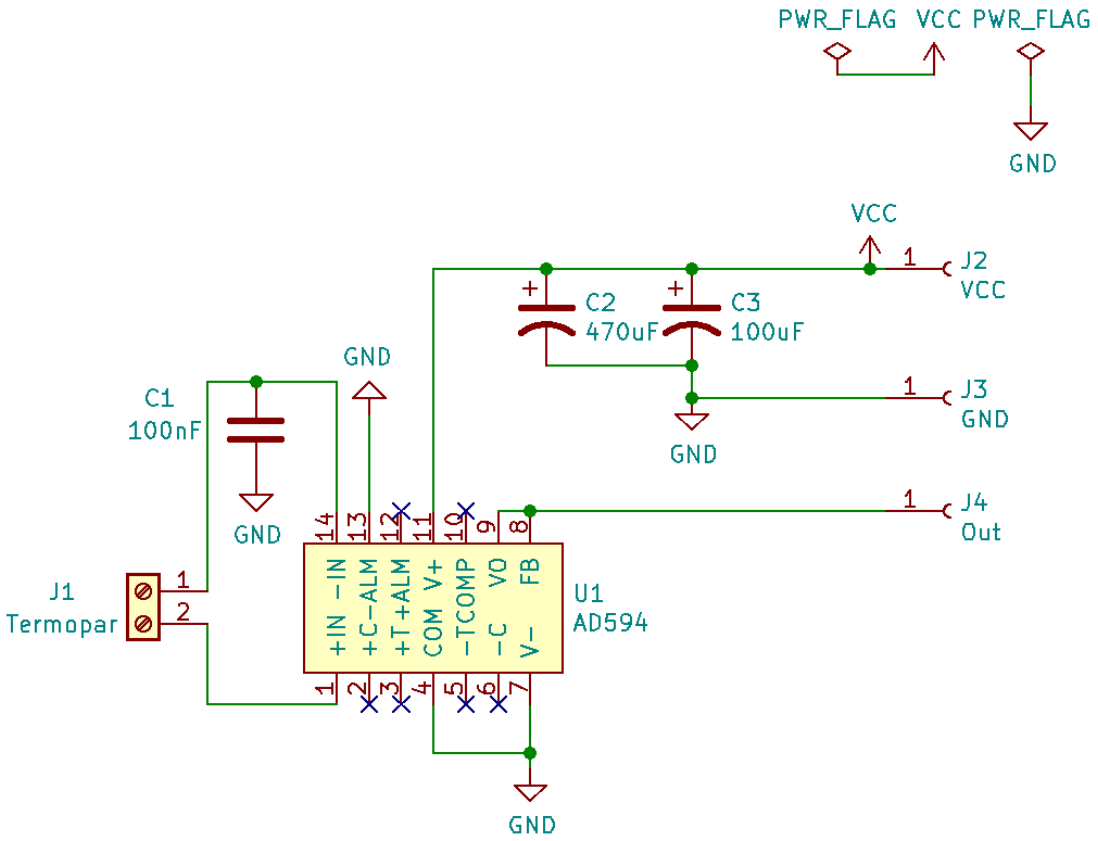


Fig. 134 Circuito del termopar.

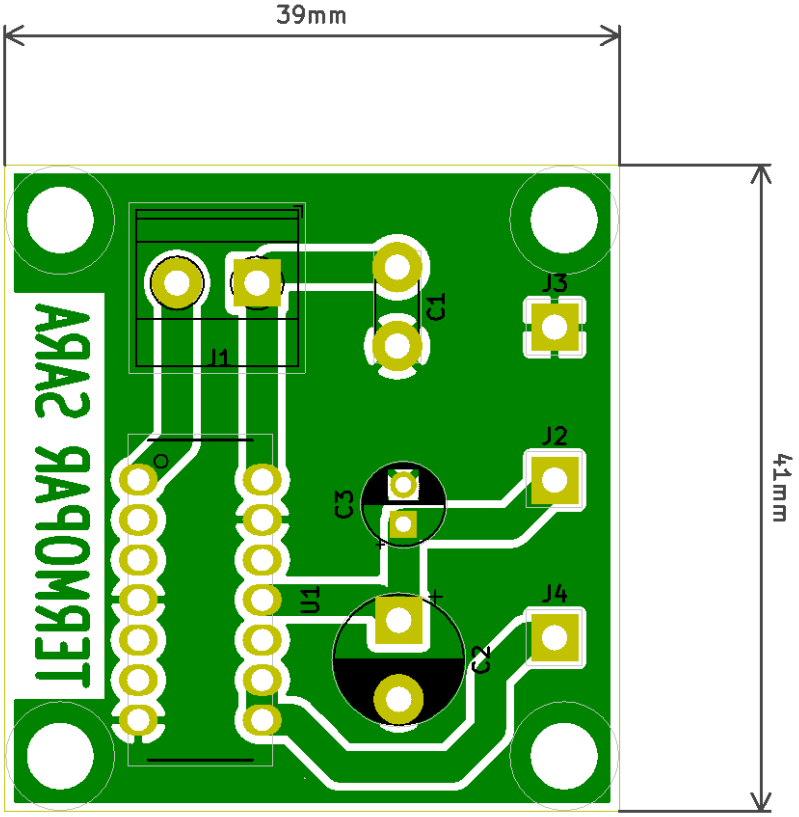


Fig. 135 PCB del termopar.

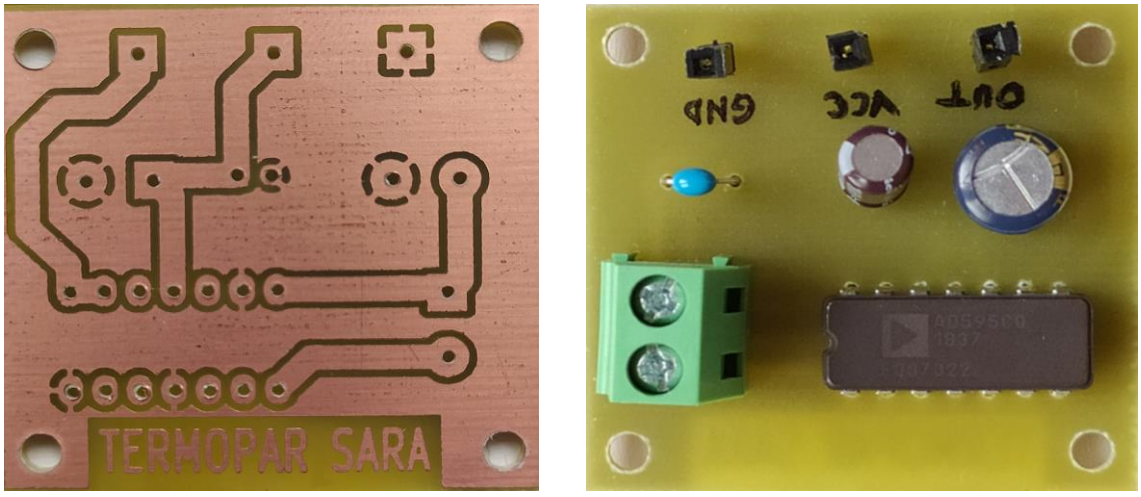


Fig. 136 Resultado final de la PCB del termopar.

NTC

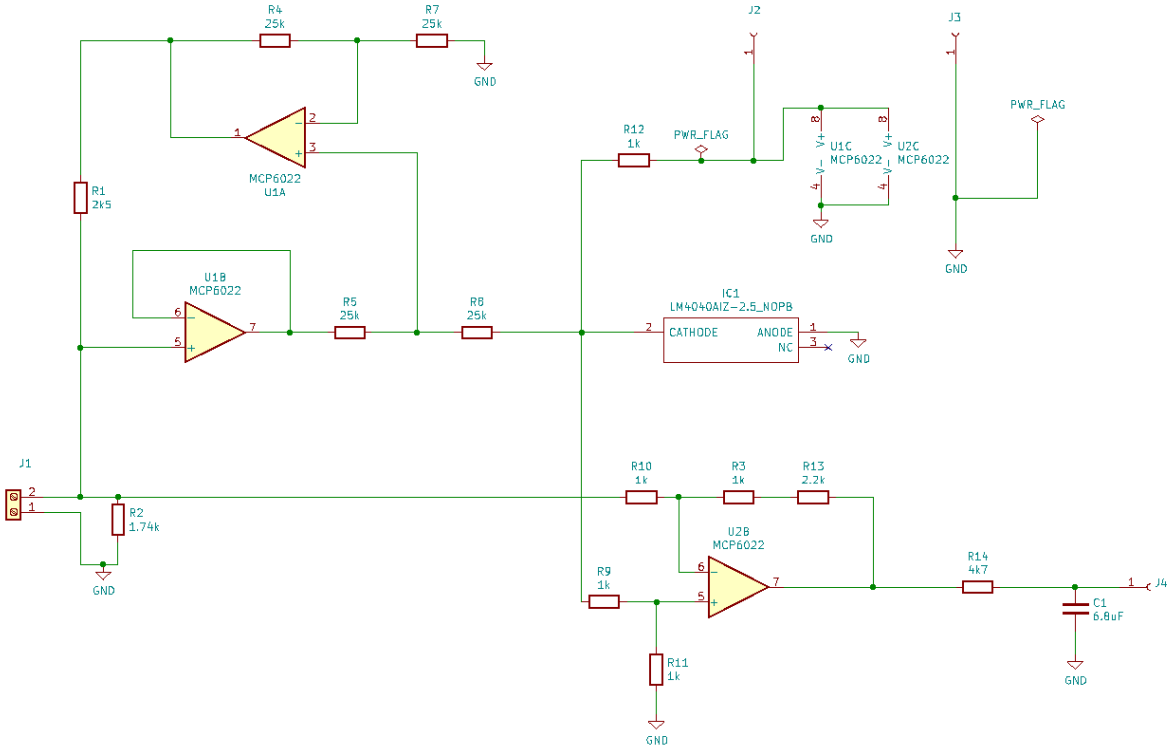


Fig. 137 Circuito de la NTC.

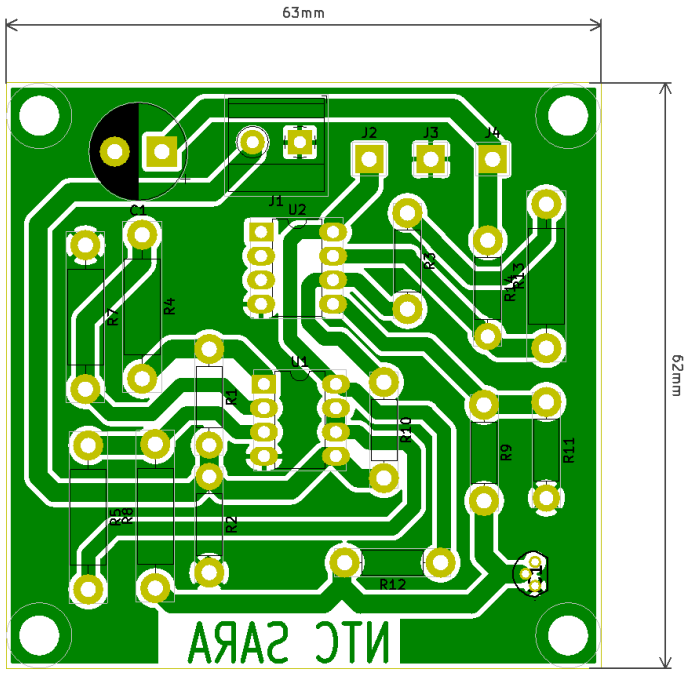


Fig. 138 PCB de la NTC.

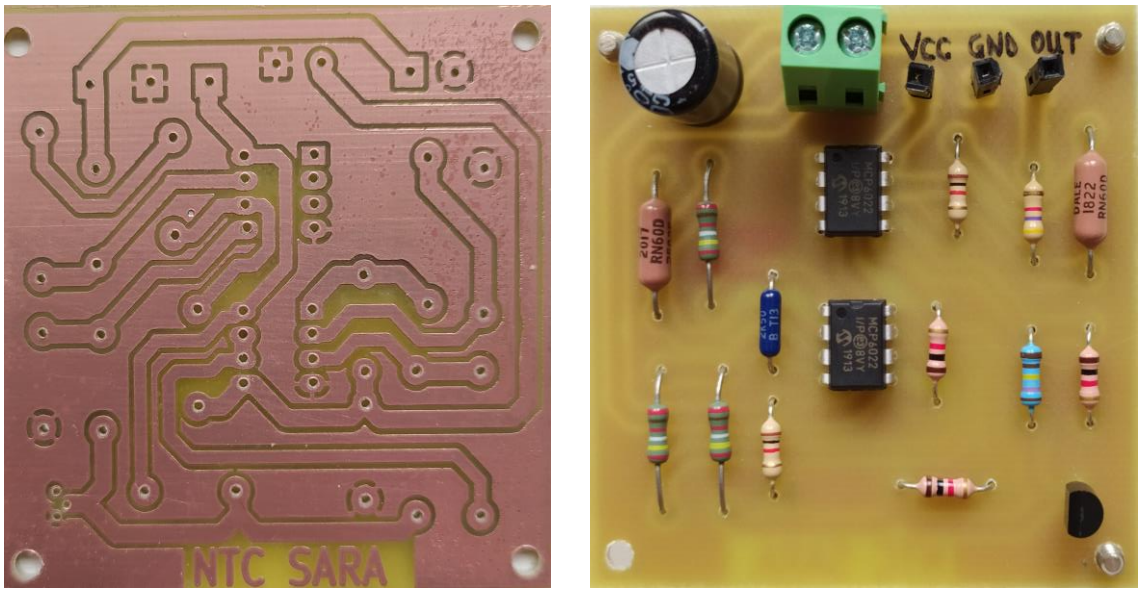


Fig. 139 Resultado final de la PCB de la NTC.

PT100

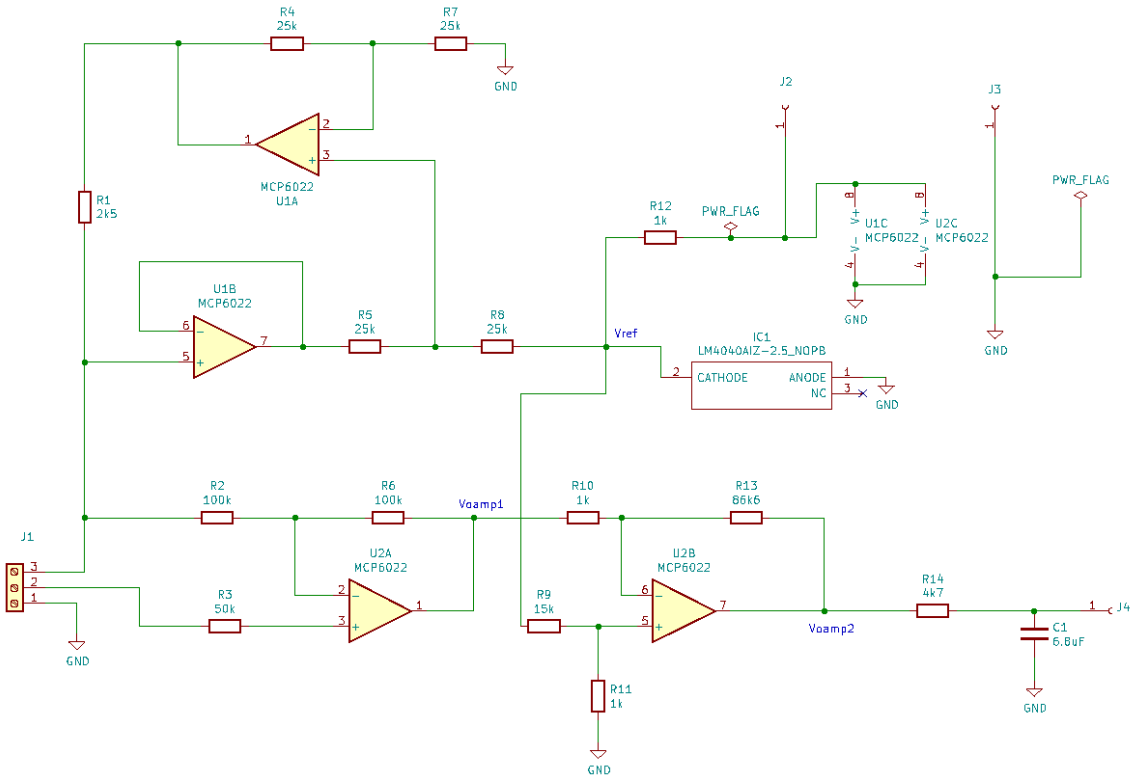


Fig. 140 Circuito de la PT100.

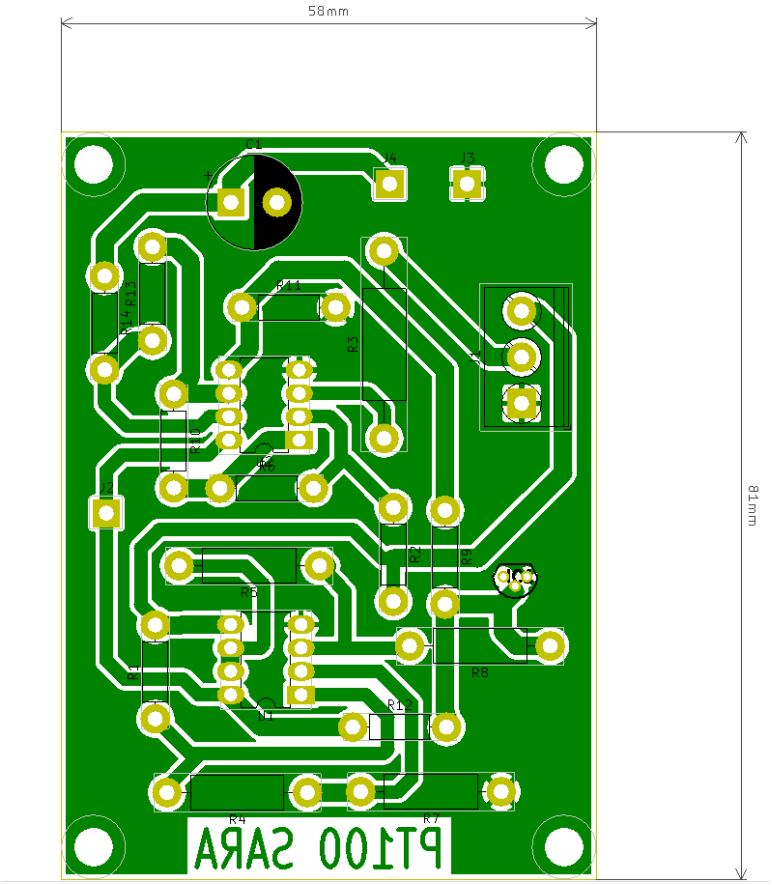
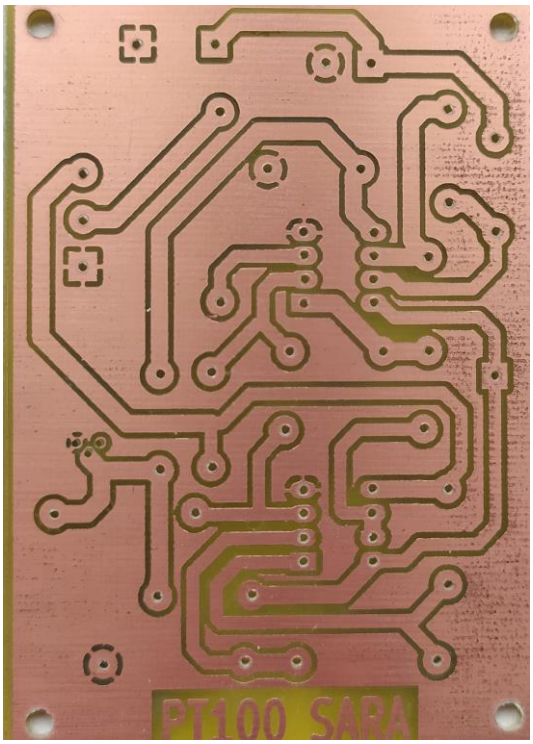


Fig. 141 PCB de la PT100.



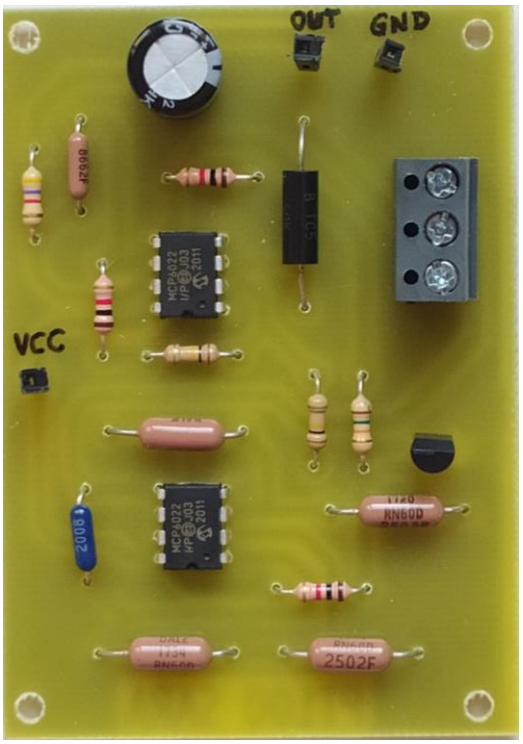


Fig. 142 Resultado final de la PCB de la PT100.

ANEXO III: CÓDIGO ARDUINO

Funciones.h

```
/*HARDWARE:
 * Timer 4: interrupción cada 100ms
 * Timer 5: señal PWM de frecuencia 500Hz
 * A0: pin AD590
 * A1: pin NTC
 * A2: pin PT100
 * A3: pin Termopar
 * A5: pin Consigna
 * A6: pin Kp
 * A7: pin Ki
 * A8: pin Kd
 * A9: pin VelVentilador
 * 46: pin salida señal PWM resistencia
 * 50: pin interruptor modo
 * 52: pin interruptor ventilador
 * 04: pin PWM Ventilador
 * 22: pin selector AD590
 * 24: pin selector Termopar
 * 26: pin selector PT100
 * 28: pin selector NTC
*/
#ifndef Funciones_h
#define Funciones_h

#define pinAD590 A0
#define pinNTC A1
#define pinPT100 A2
#define pinTermopar A3
#define pinConsigna A5
#define pinKp A6
#define pinKi A7
#define pinKd A8
#define pinVelVentilador A9
```

```
#define pinIntModo 50
#define pinPWMVentilador 4
#define pinIntVentilador 52
#define pinPWM 46
#define pinSelec_AD590 22
#define pinSelec_Termopar 24
#define pinSelec_PT100 26
#define pinSelec_NTC 28

#define TempMax 110

//VARIABLES EXTERNAS
extern int Temperatura;
extern int Tmuestreo;
extern int Modo;
extern int inicio_PC;


//INTERRUPCIÓN 500ms
void configTimer4Interrupt();

//SENSORES
void FiltradoAD();
void FiltradoNTC();
void FiltradoPT();
void FiltradoK();
void Calc_Temp_ad590(int pinAD590);
void Calc_Temp_ntc(int pinNTC);
void Calc_Temp_pt100(int pinPT100);
void Calc_Temp_termopar(int pinTermopar);
void initFiltros();

//ENTRADAS
void Lect_Entradas_Manual();
void Lect_Entradas_PC();
```

```
void Enviar_Datos_PC();

//VENTILADOR
void Control_Ventilador();

//CONTROL PID
void Params_PID();
void Control_PID();

//RESISTENCIA PWM
void configTimerPWM();
void Resistencia_PWM();
void AlertaEnfriamiento();

//LCD
void init_lcd();
void imprimir_saludo();
void imprimir_modo();
void imprimir_titulos();
void imprimir_temp();
void imprimir_consig();
void imprimir_error();
void imprimir_Kp();
void imprimir_Ki();
void imprimir_Kd();
void imprimir_DC();

#endif //Funciones_h
```

Funciones.cpp

```
#include "Funciones.h"//Cabecera con las declaraciones
#include <Arduino.h>//Incluye la librería para emplear funciones propias de arduino
#include <LiquidCrystal_I2C.h>//Incluye la librería para LCD controlada por I2C

//SENSORES
int Tad590=0;//temperatura AD590
int Tntc=0; //temperatura NTC
int Tpt100=0;//temperatura PT100
int Ttermopar=0;//temperatura Termopar

//FILTROS
const int ventana=10;

int bufferCircularAD[ventana];
int sumaBufferCircularAD=0;
int indiceAD=0;

int bufferCircularNTC[ventana];
int sumaBufferCircularNTC=0;
int indiceNTC=0;

int bufferCircularPT[ventana];
int sumaBufferCircularPT=0;
int indicePT=0;

int bufferCircularK[ventana];
int sumaBufferCircularK=0;
int indiceK=0;

//ENTRADAS
int Modo=1; //1=Manual 0=PC
int Sensor=1; //1=AD590 2=Termopar 3=PT100 4=NTC
int Ventilador=1; //1=OFF 0=ON
```

```
int VelVentilador=255;

//FLAGS
int inicio_PC=0; //Flag para detectar el cambio de modo
int Tmuestreo=0; //Flag tiempo de muestreo

//CONTROL PID
int Consigna=0,Kp=0,Ki=0,Kd=0;
int ErrorActual=0;
int ErrorAnterior=0;
float iError=0;
float dError=0;
int pTerm=0;
int iTerm=0;
int dTerm=0;
int LimInt=0;
int Temperatura=0;
int Salida_PID=0;
const int Salida_PID_max=255;
float Tm=0.5;

//RESISTENCIA PWM
int DC=0;
int Duty=0;

//LCD
LiquidCrystal_I2C lcd(0x3F,20,4);

//INTERRUPCIÓN
//Configura el Timer 4 para interrupción cada 500ms
void configTimer4Interrupt(){
  cli(); //Desactiva interrupciones
  TCCR4A = (0<<COM4A1)|(0<<COM4A0)|(0<<COM4B1)|(0<<COM4B0)|(0<<COM4C1)|(0<<COM4C0)|(0<<WGM41)|(0<<WGM40);
```

```
 //Divisor de frecuencia 64 y modo CTC
 TCCR4B = (0<<ICNC4)|(0<<ICES4)|(0<<5)|(0<<WGM43)|(1<<WGM42)|(0<<CS42)|(1<<CS41)|(1<<CS40);
 OCR4A = 62499; // Límite 500ms
 TCNT4 = 0;// Resetea el contador
 TIMSK4 = (0<<ICIE4)|(0<<OCIE4B)|(1<<OCIE4A)|(0<<TOIE4); //Interrupción por límite de cuenta
 sei(); //Activa interrupciones
}

//Función a ejecutar tras interrupción 500ms
ISR(TIMER4_COMPA_vect) {
 //Detiene el contador
 TCCR4B = (0<<ICNC4)|(0<<ICES4)|(0<<5)|(0<<WGM43)|(1<<WGM42)|(0<<CS42)|(0<<CS41)|(0<<CS40);
 TIMSK4 = (0<<ICIE4)|(0<<OCIE4B)|(0<<OCIE4A)|(0<<TOIE4);//Deshabilita la interrupción
 Tmuestreo=1; //Flag tiempo de muestreo
 TCNT4 = 0; //Resetea el contador
 //Activa el contador
 TCCR4B = (0<<ICNC4)|(0<<ICES4)|(0<<5)|(0<<WGM43)|(1<<WGM42)|(0<<CS42)|(1<<CS41)|(1<<CS40);
 TIMSK4 = (0<<ICIE4)|(0<<OCIE4B)|(1<<OCIE4A)|(0<<TOIE4);//Habilita la interrupción
}

//SENSORES
void FiltradoAD(){ //Filtrado de la temperatura del AD590
  sumaBufferCircularAD-=bufferCircularAD[indiceAD]; //Resta la lectura más antigua
  bufferCircularAD[indiceAD]=Tad590; //sustituye la lectura más antigua por la actual
  sumaBufferCircularAD+=bufferCircularAD[indiceAD]; //Actualiza la suma
  indiceAD++; //Actualiza el índice
  if(indiceAD==ventana){ //Valor máximo del índice alcanzado
   indiceAD=0; //Resetea el índice
  }
  //El valor filtrado es la media de los valores del buffer
```

```
  Tad590=sumaBufferCircularAD/ventana;
}

void Calc_Temp_ad590(int pinAD590){
 int AD590; //valor digital AD590
 float Vad590; //tensión
 AD590=analogRead(pinAD590);//lee valor digital del pin AD590
 Vad590=(float(AD590)*5)/1023; //conversión del valor digital a tensión AD590
 Tad590=(Vad590/0.011)-273; //obtiene la temperatura del AD590
}

void FiltradoNTC(){
 sumaBufferCircularNTC-=bufferCircularNTC[indiceNTC];
 bufferCircularNTC[indiceNTC]=Tntc;
 sumaBufferCircularNTC+=bufferCircularNTC[indiceNTC];
 indiceNTC++;
 if(indiceNTC==ventana){
  indiceNTC=0;
 }
 Tntc=sumaBufferCircularNTC/ventana;
}

void Calc_Temp_ntc(int pinNTC){
 int NTC; //lectura valor digital NTC
 float Vntc; //tensión
 NTC=analogRead(pinNTC);//lee valor digital NTC
 Vntc=(float(NTC)*5)/1023; //conversión del valor digital a tensión NTC
 //obtiene la temperatura de la ntc
 Tntc=((7.08*Vntc+6.606)/(0.0152639*Vntc+0.029764605))-273;
}

void FiltradoPT(){
  sumaBufferCircularPT-=bufferCircularPT[indicePT];
  bufferCircularPT[indicePT]=Tpt100;
  sumaBufferCircularPT+=bufferCircularPT[indicePT];
```

```
  indicePT++;
  if(indicePT==ventana){
   indicePT=0;
  }
  Tpt100=sumaBufferCircularPT/ventana;
}

void Calc_Temp_pt100(int pinPT100){
 int PT100; //lectura valor digital PT100
 float Vpt100; //tensión
 PT100=analogRead(pinPT100);//lee valor digital PT100
 Vpt100=(float(PT100)*5)/1023; //conversión del valor digital a tensión PT100
 Tpt100=(5.04-Vpt100)/(0.033341); //obtiene la temperatura de la PT100
}

void FiltradoK(){
  sumaBufferCircularK-=bufferCircularK[indiceK];
  bufferCircularK[indiceK]=Ttermopar;
  sumaBufferCircularK+=bufferCircularK[indiceK];
  indiceK++;
  if(indiceK==ventana){
   indiceK=0;
  }
  Ttermopar=sumaBufferCircularK/ventana;
}

void Calc_Temp_termopar(int pinTermopar){
 int Termopar; //valor digital Termopar
 float Vtermopar; //tensión
 Termopar=analogRead(pinTermopar);//lee valor digital Termopar
 //conversión del valor digital a tensión Termopar
 Vtermopar=(float(Termopar)*5)/1023;
 Ttermopar=100*Vtermopar; //obtiene la temperatura del Termopar
}
```

```
void initFiltros(){
  for(int i=0;i<ventana;i++){ //Recorre los buffers
    Calc_Temp_ad590(pinAD590); //Obtiene la temperatura del sensor
    bufferCircularAD[i]=Tad590; //La almacena en el buffer
    sumaBufferCircularAD+=bufferCircularAD[i]; //Actualiza la suma
    Calc_Temp_ntc(pinNTC);
    bufferCircularNTC[i]=Tntc;
    sumaBufferCircularNTC+=bufferCircularNTC[i];
    Calc_Temp_pt100(pinPT100);
    bufferCircularPT[i]=Tpt100;
    sumaBufferCircularPT+=bufferCircularPT[i];
    Calc_Temp_termopar(pinTermopar);
    bufferCircularK[i]=Ttermopar;
    sumaBufferCircularK+=bufferCircularK[i];
    delay(10); //Espera 10ms entre muestras
  }
}
//ENTRADAS
void Lect_Entradas_Manual(){
  if(digitalRead(pinSelec_AD590)==LOW){ //Lectura del selector
    Sensor=1; //Asignación del número correspondiente al sensor
  }else if(digitalRead(pinSelec_Termopar)==LOW){
    Sensor=2;
  }else if(digitalRead(pinSelec_PT100)==LOW){
    Sensor=3;
  }else if(digitalRead(pinSelec_NTC)==LOW){
    Sensor=4;
  }
  Consigna=analogRead(pinConsigna); //Lectura potenciómetro Consigna
  Consigna=map(Consigna,0,1023,0,150); //Ajusta el valor digital al rango 0-150ºC
  //Lectura potenciómetros Kp, Ki y Kd y ajuste de rango
  Kp=analogRead(pinKp);
  Kp=map(Kp,0,1023,0,100);
  Ki=analogRead(pinKi);
  Ki=map(Ki,0,1023,0,100);
```

```
  Kd=analogRead(pinKd);
  Kd=map(Kd,0,1023,0,50);
  Ventilador=digitalRead(pinIntVentilador); //Lectura interruptor Ventilador
}
//Almacena los 6 bytes recibidos en el buffer de entrada
void Lect_Entradas_PC(){
    Sensor=Serial.read();
    Consigna=Serial.read();
    Kp=Serial.read();
    Ki=Serial.read();
    Kd=Serial.read();
    Ventilador=Serial.read();
}

//Escritura de los 5 bytes a enviar en el buffer de salida
void Enviar_Datos_PC(){
  Serial.write(Tad590);
  Serial.write(Ttermopar);
  Serial.write(Tpt100);
  Serial.write(Tntc);
  Serial.write(Duty);
}

//VENTILADOR
void Control_Ventilador(){
  if(Ventilador==1){ //Ventilador OFF
    analogWrite(pinPWMVentilador,0); //Apaga el ventilador
  }else{ //Ventilador ON
    //Lectura de la consigna de velocidad
    VelVentilador=analogRead(pinVelVentilador);
    //Ajuste de la precisión de  10 bits a 8 bits
    VelVentilador=map(VelVentilador,0,1023,0,255);
    //Control de velocidad según consigna
    analogWrite(pinPWMVentilador,VelVentilador);
  }
```

```
}

//CONTROL PID
void Params_PID(){
  switch(Sensor){  //Asigna la temperatura de referencia según sensor seleccionado
    case 1:
      Temperatura=Tad590;
      break;
    case 2:
      Temperatura=Ttermopar;
      break;
    case 3:
      Temperatura=Tpt100;
      break;
    case 4:
      Temperatura=Tntc;
      break;
    default:
      Temperatura=Tad590;
  }
}

void Control_PID(){
  Params_PID();
  ErrorAnterior=ErrorActual; //Almacena error anterior
  ErrorActual=(Consigna-Temperatura); //Calcula error actual
  //Término Proporcional
  pTerm=Kp*ErrorActual; //Calcula el término proporcional
  //Término Integral
  if(abs(ErrorActual)>LimInt ) { //Error mayor que el límite
    iError=((float(ErrorActual)+float(ErrorAnterior))/2)*Tm; //Calcula error integral
    iTerm=Ki*iError; //Calcula el término integral
  }
  else {
    iTerm = 0; //No se aplica acción integral
```

```
 }
 if (iTerm < 0){
  iTerm = 0; //Mínimo valor del término integral
 }
 //Término Derivativo
 dError=(float(ErrorActual)-float(ErrorAnterior))/Tm; //Calcula error derivativo
 dTerm=Kd*dError; //Calcula el término derivativo
 if(dTerm<0){
  dTerm=0; //Mínimo valor del término derivativo
 }
 //Acción PID
 Salida_PID=pTerm+iTerm+dTerm; //Cálculo de la señal PID
 if (Salida_PID > Salida_PID_max){
  Salida_PID = Salida_PID_max; //Máximo valor de la salida
 }
 if (Salida_PID < 0){
  Salida_PID = 0; //Mínimo valor de la salida
 }
 Salida_PID=map(Salida_PID,0,Salida_PID_max,0,16000); //Ajuste del rango
}



//RESISTENCIA PWM
void configTimerPWM(){
 //Salida PWM en OC5A

TCCR5A=(1<<COM5A1)|(0<<COM5A0)|(0<<COM5B1)|(0<<COM5B0)|(0<<COM5C1)|(0<
<COM5C0)|(0<<WGM51)|(0<<WGM50);
 //Modo control de fase y frecuencia, prescaler 1

TCCR5B=(0<<ICNC5)|(0<<ICES5)|(0<<5)|(1<<WGM53)|(0<<WGM52)|(0<<CS52)|(0<<CS5
1)|(1<<CS50);
 ICR5=16000; //Frecuencia PWM 500Hz
 TCNT5=0;//Resetea Timer 5
 OCR5A=16000;//registro para controlar el ciclo de trabajo
}
```

```
void Resistencia_PWM(){
  DC=map(Salida_PID,0,16000,16000,0);//Invierte la salida
  Duty=100-(float(DC)/16000)*100; //Duty Cycle en %
  OCR5A=DC; //Actualiza el registro del contador 5
}

void AlertaEnfriamiento(){
  OCR5A=16000; //Desconecta la resistencia calefactora
  analogWrite(pinPWMVentilador,255); //Enciende el ventilador a su velocidad máxima
  //Muestra mentase de alerta en pantalla y temperatura del horno
  lcd.setCursor(0,0);
  lcd.print("                ");
  lcd.setCursor(0,1);
  lcd.print("ALERTA ENFRIAMIENTO ");
  lcd.setCursor(0,2);
  lcd.print("    TEMP:       ");
  lcd.setCursor(0,3);
  lcd.print("                ");
  do{
    Calc_Temp_ad590(pinAD590); //Calcula la temperatura
    lcd.setCursor(11,2);
    lcd.print(Tad590); //Imprime la temperatura
    lcd.print("C");
    lcd.setCursor(11,2);
    lcd.print("    ");
  }while(Tad590 > 100);
  //El bucle finaliza cuando la temperatura es igual o inferior a 100C
  analogWrite(pinPWMVentilador,0); //Apaga el ventilador
  //Reset pantalla a estado anterior
  lcd.clear();
  imprimir_titulos();
}

//LCD
```

```
void init_lcd(){
  lcd.begin();// Inicializar el display LCD
  lcd.backlight();//Encender la luz de fondo
}

void imprimir_saludo(){ //Imprime mensaje de inicio
  lcd.setCursor(7,1); //Coloca el cursor
  //Imprime mensaje
  lcd.print("CONTROL");
  lcd.setCursor(3,2);
  lcd.print("DE TEMPERATURA");
  delay(3000); //Espera 3s
  lcd.clear(); //Limpia la pantalla
}

void imprimir_modo(){ //Imprime el modo de funcionamiento
  lcd.setCursor(0,3);
  lcd.print("          "); //Borra modo anterior
  lcd.setCursor(0,3);
  if(Modo==1){  //Modo manual activo
    lcd.print("/manual/");
  }else{ //Modo PC activo
    lcd.print("/pc/");
  }
}
void imprimir_titulos(){ //Imprime los nombres de las variables
  lcd.setCursor(0,1);
  lcd.print("Consig:");
  lcd.setCursor(0,2);
  lcd.print("Error:");
  lcd.setCursor(11,0);
  lcd.print("Kp:");
  lcd.setCursor(11,1);
  lcd.print("Ki:");
  lcd.setCursor(11,2);
```

```
  lcd.print("Kd:");
  lcd.setCursor(11,3);
  lcd.print("DC:");
}
//Imprime el nombre del sensor seleccionado y su temperatura
void imprimir_temp(){
  lcd.setCursor(0,0);
  lcd.print("          ");
  lcd.setCursor(0,0);
  switch(Sensor){
    case 1:
      lcd.print("AD590:");
      break;
    case 2:
      lcd.print("TIPOK:");
      break;
    case 3:
      lcd.print("PT100:");
      break;
    case 4:
      lcd.print("NTC:");
      break;
    default:
      lcd.print("AD590:");
  }
  lcd.print(Temperatura);
  lcd.print("C");
}
void imprimir_consig(){ //Imprime el valor de consigna
  lcd.setCursor(7,1);
  lcd.print("    ");
  lcd.setCursor(7,1);
  lcd.print(Consigna);
  lcd.print("C");
}
```

```
void imprimir_error(){ //Imprime el valor del error actual
  lcd.setCursor(6,2);
  lcd.print("     ");
  lcd.setCursor(6,2);
  lcd.print(ErrorActual);
  lcd.print("C");
}

void imprimir_Kp(){ //Imprime el valor de la constante Kp
  lcd.setCursor(14,0);
  lcd.print("      ");
  lcd.setCursor(14,0);
  lcd.print(Kp);
}

void imprimir_Ki(){ //Imprime el valor de la constante Ki
  lcd.setCursor(14,1);
  lcd.print("      ");
  lcd.setCursor(14,1);
  lcd.print(Ki);
}
void imprimir_Kd(){ //Imprime el valor de la constante Kd
  lcd.setCursor(14,2);
  lcd.print("      ");
  lcd.setCursor(14,2);
  lcd.print(Kd);
}

void imprimir_DC(){ //Imprime el valor del ciclo de trabajo
  lcd.setCursor(14,3);
  lcd.print("      ");
  lcd.setCursor(14,3);
  lcd.print(Duty);
}
```

Maqueta_Control_Temperatura.ino

```
#include "Funciones.h"//Cabecera con las declaraciones


void setup() {
  pinMode(pinIntModo,INPUT_PULLUP); //configura el pin como entrada con R pull-up
  pinMode(pinIntVentilador,INPUT_PULLUP);
  pinMode(pinSelec_AD590,INPUT_PULLUP);
  pinMode(pinSelec_Termopar,INPUT_PULLUP);
  pinMode(pinSelec_PT100,INPUT_PULLUP);
  pinMode(pinSelec_NTC,INPUT_PULLUP);
  pinMode(pinPWMVentilador,OUTPUT); //configura el pin como salida
  pinMode(pinPWM,OUTPUT);
  initFiltros(); //inicia el filtrado de temperaturas
  init_lcd(); //inicia el display LCD
  imprimir_saludo(); //imprime mensaje en pantalla
  imprimir_titulos(); //imprime los nombres en pantalla
  configTimerPWM(); //Configura el temporizador 5 para generar la señal PWM
  configTimer4Interrupt(); //Configura el temporizador 4 para realizar interrupciones
}


void loop() {
  //Actualiza los datos en pantalla
  imprimir_temp();
  imprimir_consig();
  imprimir_error();
  imprimir_modo();
  imprimir_Kp();
  imprimir_Ki();
  imprimir_Kd();
  imprimir_DC();
  if(Tmuestreo==1){ //Flag interrupción
    //Detiene el contador
    TCCR4B                                                    =
(0<<ICNC4)|(0<<ICES4)|(0<<5)|(0<<WGM43)|(1<<WGM42)|(0<<CS42)|(0<<CS41)|(0<<CS
40);
```

```
  TIMSK4 = (0<<ICIE4)|(0<<OCIE4B)|(0<<OCIE4A)|(0<<TOIE4);//Deshabilita la interrupción
  if(Temperatura > TempMax){ //La temperatura supera el límite
   AlertaEnfriamiento(); //Enfriamiento forzado
  }
  Calc_Temp_pt100(pinPT100); //Calcula la temperatura del sensor
  FiltradoPT(); //Filtrado de la temperatura
  Calc_Temp_ad590(pinAD590);
  FiltradoAD();
  Calc_Temp_ntc(pinNTC);
  FiltradoNTC();
  Calc_Temp_termopar(pinTermopar);
  FiltradoK();
  Modo=digitalRead(pinIntModo); //Lectura del interruptor de Modo
  if(Modo==1){//Modo Manual
   if(inicio_PC==1){ //Estaba en modo PC
    inicio_PC=0; //Limpia el flag
    Serial.end(); //Cierra la comunicación serial
   }
   Lect_Entradas_Manual(); //Lectura de las entradas del panel frontal
  }else{//Modo PC
   if(inicio_PC==0){ //Estaba en modo Manual
    Serial.begin(115200); //Inicia comunicación serial
    inicio_PC=1; //Activa el flag del modo PC
   }
   if (Serial.available() >= 6) { //Se han recibido todos los datos
    Lect_Entradas_PC(); //Lectura del buffer de entrada
   }
  }
  Control_Ventilador(); //Controla el ventilador
  Control_PID();//Calcula el control PID
  Resistencia_PWM(); //Aplica el control PID
  if(Modo==0){ //Modo PC
   Enviar_Datos_PC(); //Envía los datos al PC
  }
```

```
  Tmuestreo=0; //Resetea flag interrupción
  TCNT4 = 0; //Resetea el contador
  //Activa el contador
  TCCR4B = (0<<ICNC4)|(0<<ICES4)|(0<<5)|(0<<WGM43)|(1<<WGM42)|(0<<CS42)|(1<<CS41)|(1<<CS40);
  TIMSK4 = (0<<ICIE4)|(0<<OCIE4B)|(1<<OCIE4A)|(0<<TOIE4);//Habilita la interrupción
 }
}
```

ANEXO IV: CÓDIGO MATLAB

```
classdef Maqueta_Control_Temperatura < matlab.apps.AppBase


    % Properties that correspond to app components
    properties (Access = public)
        UIFigure              matlab.ui.Figure
        UIAxes                matlab.ui.control.UIAxes
        Tabla                 matlab.ui.control.Table
        KpKnobLabel           matlab.ui.control.Label
        kp                    matlab.ui.control.Knob
        KpIndicador           matlab.ui.control.NumericEditField
        KiIndicador           matlab.ui.control.NumericEditField
        KiKnobLabel           matlab.ui.control.Label
        ki                    matlab.ui.control.Knob
        KdKnobLabel           matlab.ui.control.Label
        kd                    matlab.ui.control.Knob
        KdIndicador           matlab.ui.control.NumericEditField
        VentiladorSwitchLabel  matlab.ui.control.Label
        ventilador            matlab.ui.control.ToggleSwitch
        LogoEHU               matlab.ui.control.Image
        Iniciar               matlab.ui.control.StateButton
        Exportar              matlab.ui.control.Button
        DutyCycleGaugeLabel    matlab.ui.control.Label
        DutyCycleGauge        matlab.ui.control.Gauge
        PuertoSeriePanel      matlab.ui.container.Panel
        PuertoSerie           matlab.ui.control.EditField
        ImagenPuerto          matlab.ui.control.Image
        ArchivoDatosPanel     matlab.ui.container.Panel
        ExcelDatos            matlab.ui.control.EditField
        SensorPanel           matlab.ui.container.Panel
        Sensor                matlab.ui.control.DropDown
        ConsignaPanel         matlab.ui.container.Panel
        consigna              matlab.ui.control.Spinner
        MostrarTabla          matlab.ui.control.StateButton
        ImagenAviso           matlab.ui.control.Image
    end



    properties (Access = public) %Propiedades públicas de la aplicación
        s
        datos
        Consigna
        sensorPID
        Kp
        Ki
        Kd
        Ventilador
        muestra
        contador
        AD590
        TERMOPAR
        PT100
```

```
        NTC
        grafAD590
        grafTERMOPAR
        grafPT100
        grafNTC
        error
        xLimInf=0
        xLimSup=240
        pos
    end

    methods (Access = private)

        function BytesAvailableFcn(app,~,~)
            %Lectura y almacenamiento de los bytes enviados por Arduino
            app.AD590(app.contador)=read(app.s,1,"uint8");
            app.TERMOPAR(app.contador)=read(app.s,1,"uint8");
            app.PT100(app.contador)=read(app.s,1,"uint8");
            app.NTC(app.contador)=read(app.s,1,"uint8");
            app.DutyCycleGauge.Value=read(app.s,1,"uint8");
            %Selección valores a mostrar
            app.grafAD590(app.pos)=app.AD590(app.contador);
            app.grafTERMOPAR(app.pos)=app.TERMOPAR(app.contador);
            app.grafPT100(app.pos)=app.PT100(app.contador);
            app.grafNTC(app.pos)=app.NTC(app.contador);
            switch app.sensorPID %Muestra datos del sensor seleccionado en la gráfica
                case 1
                    app.error(app.contador)=app.Consigna-app.AD590(app.contador); %Calcula el
error
                    %Ajuste del eje vertical. Muestra datos en el centro
                    yLimInf=app.AD590(app.contador)-10;
                    yLimSup=app.AD590(app.contador)+10;
                    app.UIAxes.YLim=[yLimInf yLimSup];
                    app.UIAxes.YTick=yLimInf:yLimSup;
                    app.UIAxes.YTickLabel=yLimInf:yLimSup;
                    plot(app.UIAxes,app.grafAD590,'m'); %Traza línea magenta en la gráfica
                case 2
                    app.error(app.contador)=app.Consigna-app.TERMOPAR(app.contador); %Calcula
el error
                    %Ajuste del eje vertical. Muestra datos en el centro
                    yLimInf=app.TERMOPAR(app.contador)-10;
                    yLimSup=app.TERMOPAR(app.contador)+10;
                    app.UIAxes.YLim=[yLimInf yLimSup];
                    app.UIAxes.YTick=yLimInf:yLimSup;
                    app.UIAxes.YTickLabel=yLimInf:yLimSup;
                    plot(app.UIAxes,app.grafTERMOPAR,'b'); %Traza línea azul en la gráfica
                case 3
                    app.error(app.contador)=app.Consigna-app.PT100(app.contador); %Calcula el
error
                    %Ajuste del eje vertical. Muestra datos en el centro
                    yLimInf=app.PT100(app.contador)-10;
                    yLimSup=app.PT100(app.contador)+10;
                    app.UIAxes.YLim=[yLimInf yLimSup];
```

```
                app.UIAxes.YTick=yLimInf:yLimSup;
                app.UIAxes.YTickLabel=yLimInf:yLimSup;
                plot(app.UIAxes,app.grafPT100,'r'); %Traza línea roja en la gráfica
            case 4
                app.error(app.contador)=app.Consigna-app.NTC(app.contador); %Calcula el error
                %Ajuste del eje vertical. Muestra datos en el centro
                yLimInf=app.NTC(app.contador)-10;
                yLimSup=app.NTC(app.contador)+10;
                app.UIAxes.YLim=[yLimInf yLimSup];
                app.UIAxes.YTick=yLimInf:yLimSup;
                app.UIAxes.YTickLabel=yLimInf:yLimSup;
                plot(app.UIAxes,app.grafNTC,'k'); %Traza línea negra en la gráfica
            otherwise
                app.error(app.contador)=app.Consigna-app.AD590(app.contador); %Calcula el
error
                %Ajuste del eje vertical. Muestra datos en el centro
                yLimInf=app.AD590(app.contador)-10;
                yLimSup=app.AD590(app.contador)+10;
                app.UIAxes.YLim=[yLimInf yLimSup];
                app.UIAxes.YTick=yLimInf:yLimSup;
                app.UIAxes.YTickLabel=yLimInf:yLimSup;
                plot(app.UIAxes,app.grafAD590,'m'); %Traza línea magenta en la gráfica
        end
        app.pos=app.pos+1; %Actualiza posición
        if app.pos>240 %Supera el rango del eje X
            app.pos=1; %Resetea posición
            %Vacía arrays de temperaturas para gráfica
            app.grafAD590=[];
            app.grafTERMOPAR=[];
            app.grafPT100=[];
            app.grafNTC=[];
        end
        app.muestra(app.contador)=app.contador; %Guarda número de muestra
        %Genera matriz Datos
        app.datos=[app.muestra;app.AD590;app.TERMOPAR;app.PT100;app.NTC;app.error];
        app.Tabla.Data=app.datos'; %Guarda en la tabla la traspuesta de Datos
        app.contador=app.contador+1; %Actualiza valor del contador
        %Envio de la información de las entradas a Arduino
        write(app.s,app.sensorPID,"uint8");
        write(app.s,app.Consigna,"uint8");
        write(app.s,app.Kp,"uint8");
        write(app.s,app.Ki,"uint8");
        write(app.s,app.Kd,"uint8");
        write(app.s,app.Ventilador,"uint8");
    end
end



% Callbacks that handle component events
methods (Access = private)
```

```
% Code that executes after component creation
function startupFcn(app)
    app.PuertoSerie.Visible='on'; %Muestra campo escritura puerto serie
    app.ImagenPuerto.Visible='off'; %Oculta imagen puerto serie conectado
    app.ImagenAviso.Visible='off'; %Oculta imagen aviso equipo desconectado
end


% Value changed function: kp
function kpValueChanged(app, event)
    app.Kp = app.kp.Value; %Almacena el valor
    app.KpIndicador.Value=app.Kp; %Muestra el valor en el indicador
end


% Value changed function: ki
function kiValueChanged(app, event)
    app.Ki = app.ki.Value; %Almacena el valor
    app.KiIndicador.Value=app.Ki; %Muestra el valor en el indicador
end


% Value changed function: kd
function kdValueChanged(app, event)
    app.Kd =app.kd.Value; %Almacena el valor
    app.KdIndicador.Value=app.Kd; %Muestra el valor en el indicador
end


% Value changed function: ventilador
function ventiladorValueChanged(app, event)
    if strcmp(app.ventilador.Value,'On')
        app.Ventilador=0; %Encender ventilador
    else
        app.Ventilador=1; %Apagar ventilador
    end
end


% Value changed function: Sensor
function SensorValueChanged(app, event)
    switch app.Sensor.Value %Asigna el número correspondiente al sensor
        case 'AD590'
            app.sensorPID=1;
        case 'TERMOPAR'
            app.sensorPID=2;
        case 'PT100'
            app.sensorPID=3;
        case 'NTC'
            app.sensorPID=4;
        otherwise
            app.sensorPID=1;
     end
```

```
end


% Value changed function: consigna
function consignaValueChanged(app, event)
    app.Consigna = app.consigna.Value; %Almacena el valor de consigna
end


% Value changed function: Iniciar
function IniciarValueChanged(app, event)
    if app.Iniciar.Value==0 %Apagar
        app.Iniciar.Icon = 'ON3.png'; %Icono botón verde
        app.ImagenPuerto.Visible='off'; %Oculta icono puerto COM conectado
        app.PuertoSerie.Visible='on'; %Muestra área de escritura puerto COM
        app.ExcelDatos.Visible='on'; %Muestra área de escritura nombre fichero
        app.Exportar.Visible='on'; %Muestra botón exportar
        app.MostrarTabla.Visible='on'; %Muestra botón tabla
        write(app.s,1,"uint8"); %SensorPID=1
        write(app.s,0,"uint8"); %Consigna=0
        write(app.s,0,"uint8"); %Kp=0
        write(app.s,0,"uint8"); %Ki=0
        write(app.s,0,"uint8"); %Kd=0
        write(app.s,1,"uint8"); %Ventilador=1
        app.s= []; %cierra la comunicación
        app.ImagenAviso.Visible='on';%Muestra aviso equipo desconectado
    else
        app.Iniciar.Icon = 'OFF3.png'; %Icono botón rojo
        Puerto=app.PuertoSerie.Value; %Guarda el nombre del puerto
        if strcmp(Puerto,'') %No se ha indicado ningún nombre
          mensaje=sprintf(['Puerto COM no definido. ' ...
          '\n\nIntroduzca el nombre del puerto al que se desea conectar' ...
          ' (COM4,COM5...).']); %Mensaje de error
          uialert(app.UIFigure,mensaje,'Error'); %Muestra el mensaje de error
          app.Iniciar.Value=0; %Vuelve al estado apagado
          app.Iniciar.Icon = 'ON3.png'; %Icono botón verde
        else
          app.PuertoSerie.Visible='off'; %Oculta área de escritura puerto COM
          app.ImagenPuerto.Visible='on'; %Muestra icono puerto COM conectado
          app.ExcelDatos.Visible='off'; %Oculta área de escritura nombre fichero
          app.Exportar.Visible='off'; %Oculta botón exportar
          app.MostrarTabla.Value=0; %Desactiva botón tabla
          app.MostrarTabla.Visible='off'; %Oculta botón tabla
          app.Tabla.Visible='off'; %Oculta tabla
          app.s = serialport(char(Puerto),115200,"Timeout",5); %Inicia el puerto serial
          app.ImagenAviso.Visible='off'; %Oculta aviso equipo desconectado
          %Configura callback cada 5 bytes recibidos
          configureCallback(app.s,"byte",5,@app.BytesAvailableFcn);
          flush(app.s);%Limpia buffer entrada y salida
          %Inicializa propiedades y restaura indicadores
          app.muestra=[];
          app.contador=1;
          app.AD590=[];
```

```
            app.TERMOPAR=[];
            app.PT100=[];
            app.NTC=[];
            app.grafAD590=[];
            app.grafTERMOPAR=[];
            app.grafPT100=[];
            app.grafNTC=[];
            app.pos=1;
            app.error=[];
            app.datos=[];
            app.Tabla.Data=app.datos';
            plot(app.UIAxes,0);
            app.Kp=0;
            app.kp.Value=0;
            app.KpIndicador.Value=0;
            app.Ki=0;
            app.ki.Value=0;
            app.KiIndicador.Value=0;
            app.Kd=0;
            app.kd.Value=0;
            app.KdIndicador.Value=0;
            app.Ventilador=1;
            app.ventilador.Value='Off';
            app.sensorPID=1;
            app.Sensor.Value='AD590';
            app.Consigna=0;
            app.consigna.Value=0;
            app.DutyCycleGauge.Value=0;
            %Define ejes iniciales para la gráfica
            app.UIAxes.YLim=[0 150]; %Límites eje Y
            app.UIAxes.YTick=0:10:150; %Líneas eje Y
            app.UIAxes.YTickLabel=0:10:150; %Números eje Y
            app.UIAxes.XLim=[app.xLimInf app.xLimSup]; %Límites eje X
        end


    end
end


% Button pushed function: Exportar
function ExportarButtonPushed(app, event)
    Nombre=app.ExcelDatos.Value;
    if strcmp(Nombre,'') %Campo vacío
        uialert(app.UIFigure,'Nombre del archivo no definido.','Error'); %Mensaje error
    else
        NombreXLS=strcat(Nombre,'.xls'); % Concatena extensión de archivo
        writematrix(app.datos',NombreXLS); % Escribe los datos en el fichero
        app.ExcelDatos.Value=''; %Vacía campo Nombre
        %Mensaje operación completada
        uialert(app.UIFigure,'Datos correctamente exportados.','Info','Icon','success');
    end
end
```

```
        % Value changed function: MostrarTabla
        function MostrarTablaValueChanged(app, event)
            if app.MostrarTabla.Value==1 %Mostrar tabla
                app.Tabla.Visible='on';
            else %Ocultar tabla
                app.Tabla.Visible='off';
            end
        end
    end


    % Component initialization
    methods (Access = private)


        % Create UIFigure and components
        function createComponents(app)


            % Create UIFigure and hide until all components are created
            app.UIFigure = uifigure('Visible', 'off');
            app.UIFigure.Position = [100 100 1150 580];
            app.UIFigure.Name = 'UI Figure';


            % Create UIAxes
            app.UIAxes = uiaxes(app.UIFigure);
            title(app.UIAxes, 'Gráfica de temperaturas')
            xlabel(app.UIAxes, '')
            ylabel(app.UIAxes, 'Temperatura (ºC)')
            app.UIAxes.PlotBoxAspectRatio = [1.5080971659919 1 1];
            app.UIAxes.FontName = 'Times New Roman';
            app.UIAxes.FontUnits = 'points';
            app.UIAxes.FontSize = 16;
            app.UIAxes.XLim = [0 240];
            app.UIAxes.YLim = [0 150];
            app.UIAxes.ZLim = [0 1];
            app.UIAxes.CLim = [0 1];
            app.UIAxes.ALim = [0 1];
            app.UIAxes.TickDir = 'in';
            app.UIAxes.GridColor = [0.8 0.8 0.8];
            app.UIAxes.GridAlpha = 1;
            app.UIAxes.MinorGridColor = [0.1 0.1 0.1];
            app.UIAxes.MinorGridAlpha = 0.25;
            app.UIAxes.XColor = [0.15 0.15 0.15];
            app.UIAxes.XTick = [];
            app.UIAxes.XTickLabel = '';
            app.UIAxes.YColor = [0.15 0.15 0.15];
            app.UIAxes.YTick = [0 10 20 30 40 50 60 70 80 90 100 110 120 130 140 150];
            app.UIAxes.YTickLabel = {'0'; '10'; '20'; '30'; '40'; '50'; '60'; '70'; '80'; '90'; '100'; '110';
'120'; '130'; '140'; '150'};
```

```
app.UIAxes.ZColor = [0.15 0.15 0.15];
app.UIAxes.ZTick = [0 0.5 1];
app.UIAxes.ZTickLabel = '';
app.UIAxes.YGrid = 'on';
app.UIAxes.LabelFontSizeMultiplier = 1;
app.UIAxes.TitleFontSizeMultiplier = 1.3;
app.UIAxes.BackgroundColor = [0.9412 0.9412 0.9412];
app.UIAxes.Position = [33 37 570 403];


% Create Tabla
app.Tabla = uitable(app.UIFigure);
app.Tabla.ColumnName = {'Muestra'; 'AD590'; 'Termopar'; 'PT100'; 'NTC'; 'Error'};
app.Tabla.RowName = {};
app.Tabla.Visible = 'off';
app.Tabla.FontName = 'Times New Roman';
app.Tabla.FontSize = 16;
app.Tabla.Position = [659 419 451 131];


% Create KpKnobLabel
app.KpKnobLabel = uilabel(app.UIFigure);
app.KpKnobLabel.BusyAction = 'cancel';
app.KpKnobLabel.Interruptible = 'off';
app.KpKnobLabel.HorizontalAlignment = 'center';
app.KpKnobLabel.FontName = 'Times New Roman';
app.KpKnobLabel.FontSize = 16;
app.KpKnobLabel.FontWeight = 'bold';
app.KpKnobLabel.Position = [693 58 27 22];
app.KpKnobLabel.Text = {'Kp'; ''};


% Create kp
app.kp = uiknob(app.UIFigure, 'continuous');
app.kp.MajorTicks = [0 10 20 30 40 50 60 70 80 90 100];
app.kp.ValueChangedFcn = createCallbackFcn(app, @kpValueChanged, true);
app.kp.MinorTicks = [5 15 25 35 45 55 65 75 85 95];
app.kp.Interruptible = 'off';
app.kp.FontName = 'Times New Roman';
app.kp.FontSize = 16;
app.kp.Position = [690 105 80 80];


% Create KpIndicador
app.KpIndicador = uieditfield(app.UIFigure, 'numeric');
app.KpIndicador.RoundFractionalValues = 'on';
app.KpIndicador.ValueDisplayFormat = '%.0f';
app.KpIndicador.BusyAction = 'cancel';
app.KpIndicador.Interruptible = 'off';
app.KpIndicador.Editable = 'off';
app.KpIndicador.FontName = 'Times New Roman';
app.KpIndicador.FontSize = 16;
app.KpIndicador.Position = [718 58 52 22];
```

```
% Create KiIndicador
app.KiIndicador = uieditfield(app.UIFigure, 'numeric');
app.KiIndicador.RoundFractionalValues = 'on';
app.KiIndicador.ValueDisplayFormat = '%.0f';
app.KiIndicador.Editable = 'off';
app.KiIndicador.FontName = 'Times New Roman';
app.KiIndicador.FontSize = 16;
app.KiIndicador.Position = [869 58 52 22];


% Create KiKnobLabel
app.KiKnobLabel = uilabel(app.UIFigure);
app.KiKnobLabel.HorizontalAlignment = 'center';
app.KiKnobLabel.FontName = 'Times New Roman';
app.KiKnobLabel.FontSize = 16;
app.KiKnobLabel.FontWeight = 'bold';
app.KiKnobLabel.Position = [845 58 25 22];
app.KiKnobLabel.Text = {'Ki'; ''};


% Create ki
app.ki = uiknob(app.UIFigure, 'continuous');
app.ki.MajorTicks = [0 10 20 30 40 50 60 70 80 90 100];
app.ki.ValueChangedFcn = createCallbackFcn(app, @kiValueChanged, true);
app.ki.MinorTicks = [5 15 25 35 45 55 65 75 85 95];
app.ki.Interruptible = 'off';
app.ki.FontName = 'Times New Roman';
app.ki.FontSize = 16;
app.ki.Position = [840 106 81 81];


% Create KdKnobLabel
app.KdKnobLabel = uilabel(app.UIFigure);
app.KdKnobLabel.Interruptible = 'off';
app.KdKnobLabel.HorizontalAlignment = 'center';
app.KdKnobLabel.FontName = 'Times New Roman';
app.KdKnobLabel.FontSize = 16;
app.KdKnobLabel.FontWeight = 'bold';
app.KdKnobLabel.Position = [1004 58 27 22];
app.KdKnobLabel.Text = {'Kd'; ''};


% Create kd
app.kd = uiknob(app.UIFigure, 'continuous');
app.kd.Limits = [0 50];
app.kd.MajorTicks = [0 10 20 30 40 50];
app.kd.ValueChangedFcn = createCallbackFcn(app, @kdValueChanged, true);
app.kd.MinorTicks = [5 15 25 35 45];
app.kd.Interruptible = 'off';
app.kd.FontName = 'Times New Roman';
app.kd.FontSize = 16;
```

```
        app.kd.Position = [999 106 82 82];


        % Create KdIndicador
        app.KdIndicador = uieditfield(app.UIFigure, 'numeric');
        app.KdIndicador.RoundFractionalValues = 'on';
        app.KdIndicador.ValueDisplayFormat = '%.0f';
        app.KdIndicador.Interruptible = 'off';
        app.KdIndicador.Editable = 'off';
        app.KdIndicador.FontName = 'Times New Roman';
        app.KdIndicador.FontSize = 16;
        app.KdIndicador.Position = [1029 58 52 22];


        % Create VentiladorSwitchLabel
        app.VentiladorSwitchLabel = uilabel(app.UIFigure);
        app.VentiladorSwitchLabel.HorizontalAlignment = 'center';
        app.VentiladorSwitchLabel.FontName = 'Times New Roman';
        app.VentiladorSwitchLabel.FontSize = 16;
        app.VentiladorSwitchLabel.FontWeight = 'bold';
        app.VentiladorSwitchLabel.Position = [1038 260 78 22];
        app.VentiladorSwitchLabel.Text = 'Ventilador';


        % Create ventilador
        app.ventilador = uiswitch(app.UIFigure, 'toggle');
        app.ventilador.ValueChangedFcn = createCallbackFcn(app, @ventiladorValueChanged,
true);
        app.ventilador.Interruptible = 'off';
        app.ventilador.FontName = 'Times New Roman';
        app.ventilador.FontSize = 16;
        app.ventilador.FontWeight = 'bold';
        app.ventilador.Position = [1063 309 27 61];


        % Create LogoEHU
        app.LogoEHU = uiimage(app.UIFigure);
        app.LogoEHU.Position = [1 469 318 101];
        app.LogoEHU.ImageSource = 'UPV-EHU-logo.png';


        % Create Iniciar
        app.Iniciar = uibutton(app.UIFigure, 'state');
        app.Iniciar.ValueChangedFcn = createCallbackFcn(app, @IniciarValueChanged, true);
        app.Iniciar.Interruptible = 'off';
        app.Iniciar.Icon = 'ON3.png';
        app.Iniciar.IconAlignment = 'center';
        app.Iniciar.Text = '';
        app.Iniciar.BackgroundColor = [0.9412 0.9412 0.9412];
        app.Iniciar.Position = [372 498 50 43];


        % Create Exportar
```

```
        app.Exportar = uibutton(app.UIFigure, 'push');
        app.Exportar.ButtonPushedFcn = createCallbackFcn(app, @ExportarButtonPushed,
true);
        app.Exportar.Interruptible = 'off';
        app.Exportar.Icon = 'EXCEL.png';
        app.Exportar.Position = [436 498 50 43];
        app.Exportar.Text = '';


        % Create DutyCycleGaugeLabel
        app.DutyCycleGaugeLabel = uilabel(app.UIFigure);
        app.DutyCycleGaugeLabel.HorizontalAlignment = 'center';
        app.DutyCycleGaugeLabel.FontName = 'Times New Roman';
        app.DutyCycleGaugeLabel.FontSize = 16;
        app.DutyCycleGaugeLabel.FontWeight = 'bold';
        app.DutyCycleGaugeLabel.Position = [933 254 81 22];
        app.DutyCycleGaugeLabel.Text = 'Duty Cycle';


        % Create DutyCycleGauge
        app.DutyCycleGauge = uigauge(app.UIFigure, 'circular');
        app.DutyCycleGauge.MajorTicks = [0 10 20 30 40 50 60 70 80 90 100];
            app.DutyCycleGauge.MinorTicks = [1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
         20 21 22 23 24 25 26 27          28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46
         47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74
         75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100];
        app.DutyCycleGauge.Interruptible = 'off';
        app.DutyCycleGauge.FontName = 'Times New Roman';
        app.DutyCycleGauge.FontSize = 14;
        app.DutyCycleGauge.Position = [913 276 121 121];


        % Create PuertoSeriePanel
        app.PuertoSeriePanel = uipanel(app.UIFigure);
        app.PuertoSeriePanel.AutoResizeChildren = 'off';
        app.PuertoSeriePanel.Title = 'Puerto Serie';
        app.PuertoSeriePanel.FontName = 'Times New Roman';
        app.PuertoSeriePanel.FontWeight = 'bold';
        app.PuertoSeriePanel.FontSize = 16;
        app.PuertoSeriePanel.Position = [659 248 114 76];


        % Create PuertoSerie
        app.PuertoSerie = uieditfield(app.PuertoSeriePanel, 'text');
        app.PuertoSerie.Interruptible = 'off';
        app.PuertoSerie.FontName = 'Times New Roman';
        app.PuertoSerie.FontSize = 26;
        app.PuertoSerie.Position = [10 6 94 40];


        % Create ImagenPuerto
        app.ImagenPuerto = uiimage(app.PuertoSeriePanel);
        app.ImagenPuerto.Visible = 'off';
```

```
app.ImagenPuerto.Position = [11 8 35 37];
app.ImagenPuerto.ImageSource = 'USB2.png';


% Create ArchivoDatosPanel
app.ArchivoDatosPanel = uipanel(app.UIFigure);
app.ArchivoDatosPanel.AutoResizeChildren = 'off';
app.ArchivoDatosPanel.Title = 'Archivo Datos';
app.ArchivoDatosPanel.FontName = 'Times New Roman';
app.ArchivoDatosPanel.FontWeight = 'bold';
app.ArchivoDatosPanel.FontSize = 16;
app.ArchivoDatosPanel.Position = [659 322 114 76];


% Create ExcelDatos
app.ExcelDatos = uieditfield(app.ArchivoDatosPanel, 'text');
app.ExcelDatos.Interruptible = 'off';
app.ExcelDatos.FontName = 'Times New Roman';
app.ExcelDatos.FontSize = 16;
app.ExcelDatos.Position = [10 6 94 40];


% Create SensorPanel
app.SensorPanel = uipanel(app.UIFigure);
app.SensorPanel.AutoResizeChildren = 'off';
app.SensorPanel.Title = 'Sensor';
app.SensorPanel.FontName = 'Times New Roman';
app.SensorPanel.FontWeight = 'bold';
app.SensorPanel.FontSize = 16;
app.SensorPanel.Position = [772 322 114 76];


% Create Sensor
app.Sensor = uidropdown(app.SensorPanel);
app.Sensor.Items = {'AD590', 'TERMOPAR', 'PT100', 'NTC'};
app.Sensor.ValueChangedFcn = createCallbackFcn(app, @SensorValueChanged, true);
app.Sensor.Interruptible = 'off';
app.Sensor.FontName = 'Times New Roman';
app.Sensor.FontSize = 16;
app.Sensor.BackgroundColor = [1 1 1];
app.Sensor.Position = [9 7 94 35];
app.Sensor.Value = 'AD590';


% Create ConsignaPanel
app.ConsignaPanel = uipanel(app.UIFigure);
app.ConsignaPanel.AutoResizeChildren = 'off';
app.ConsignaPanel.Title = 'Consigna';
app.ConsignaPanel.FontName = 'Times New Roman';
app.ConsignaPanel.FontWeight = 'bold';
app.ConsignaPanel.FontSize = 16;
app.ConsignaPanel.Position = [772 248 114 76];
```

```
        % Create consigna
        app.consigna = uispinner(app.ConsignaPanel);
        app.consigna.ValueChangedFcn = createCallbackFcn(app, @consignaValueChanged,
true);
        app.consigna.Interruptible = 'off';
        app.consigna.FontName = 'Times New Roman';
        app.consigna.FontSize = 16;
        app.consigna.Position = [11 8 96 36];


        % Create MostrarTabla
        app.MostrarTabla = uibutton(app.UIFigure, 'state');
        app.MostrarTabla.ValueChangedFcn = createCallbackFcn(app,
@MostrarTablaValueChanged, true);
        app.MostrarTabla.Interruptible = 'off';
        app.MostrarTabla.Icon = 'Tabla.png';
        app.MostrarTabla.IconAlignment = 'center';
        app.MostrarTabla.Text = '';
        app.MostrarTabla.BackgroundColor = [0.9412 0.9412 0.9412];
        app.MostrarTabla.Position = [500 498 50 43];


        % Create ImagenAviso
        app.ImagenAviso = uiimage(app.UIFigure);
        app.ImagenAviso.Position = [290 427 342 69];
        app.ImagenAviso.ImageSource = 'Aviso.png';


        % Show the figure after all components are created
        app.UIFigure.Visible = 'on';
    end
end


% App creation and deletion
methods (Access = public)


    % Construct app
    function app = Maqueta_Control_Temperatura


        % Create UIFigure and components
        createComponents(app)


        % Register the app with App Designer
        registerApp(app, app.UIFigure)


        % Execute the startup function
        runStartupFcn(app, @startupFcn)
```

```
        if nargout == 0
            clear app
        end
    end

    % Code that executes before app deletion
    function delete(app)

        % Delete UIFigure when app is deleted
        delete(app.UIFigure)
    end
  end
end
```

ANEXO V: MANUAL DE USUARIO

Universidad del País Vasco Euskal Herriko Unibertsitatea
BILBOKO INGENIARITZA ESKOLA
ESCUELA DE INGENIERÍA DE BILBAO

GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

TRABAJO FIN DE GRADO

DISEÑO Y FABRICACIÓN DE UN SISTEMA DE CONTROL Y MONITORIZACIÓN DE TEMPERATURA

DOCUMENTO IV- MANUAL DE USUARIO

Alumna: España Hernández, Sara

Director: Sainz de Murieta Mangado, Joseba Andoni

Curso: 2020-2021

Fecha: Bilbao, 30 de junio de 2021

ÍNDICE

ÍNDICE DE FIGURAS

1. INTRODUCCIÓN

Este documento tiene como finalidad orientar a quien desee hacer uso de la maqueta de control y regulación de temperaturas.

Esta cuenta con dos modos de funcionamiento principales, denominados modo manual y modo PC. Dado que su utilización depende en gran medida del modo seleccionado, en este documento se trata cada uno de ellos de manera independiente.

2. MODO MANUAL

En el modo manual, la interacción entre la persona usuaria y la maqueta se realiza a través de los componentes físicos situados en el panel frontal de la maqueta. Los resultados obtenidos se muestran en la pantalla que se encuentra junto a estos.

2.1 REQUISITOS TÉCNICOS

- Maqueta de control y regulación de temperaturas.
- Cable de alimentación conector hembra C7 a conector macho.

2.2 ELEMENTOS DE LA MAQUETA

Figura 1 Elementos de la maqueta.

2.3 GUÍA DE USO

Primeramente, es necesario conectar el cable de alimentación del equipo.

Una vez alimentado, se procede a encender el equipo a través del interruptor On/Off.

La pantalla mostrará un mensaje de bienvenida, y una vez esté listo para funcionar aparecerán los datos.

En este momento, es importante comprobar que el modo seleccionado es el modo manual. Para ello, verificar que en la esquina inferior izquierda de la pantalla aparece escrito "/manual/".

De lo contrario, se debe utilizar el interruptor Manual/PC para seleccionar el modo manual.

Para realizar un ensayo, en primer lugar, se selecciona el sensor a utilizar mediante el selector situado a la derecha del panel frontal.

A continuación, se definen las constantes proporcional, integral y derivativa a emplear por el control PID. Estas se modifican a través de los mandos situados bajo la pantalla.

Por último, se introduce el valor de la consigna de temperatura a alcanzar.

Para controlar el ventilador, se tiene el interruptor de encendido/apagado y el mando que permite regular su velocidad.

Finalmente, se muestra una imagen de la pantalla durante el funcionamiento de la maqueta en el modo manual.

En la primera columna se presentan los datos relacionados con la temperatura en grados centígrados. Se puede observar el sensor seleccionado junto con su temperatura. También el valor de consigna indicado, así como el error. El error es el resultado de la diferencia entre la temperatura del sensor y la de consigna. Por último, el modo de funcionamiento.

En la segunda columna se muestran todos los datos relacionados con el control PID. Estos son las constantes antes introducidas, y el ciclo de trabajo obtenido como resultado del control. Este último dato viene expresado en porcentaje.

Figura 2 Pantalla del panel frontal.

3. MODO PC

En el modo PC, la interacción entre la persona usuaria y la maqueta se realiza a través de la aplicación Maqueta_Control_Temperatura.exe. Desde esta se puede controlar y visualizar la maqueta de forma similar al modo manual, si bien se tienen algunas opciones extra que se explicarán más adelante. En cuanto al panel frontal de la maqueta, en este caso los mandos de control no permiten modificar las entradas del sistema. En la pantalla se muestran los datos de la misma manera que en el modo manual.

3.1 REQUISITOS TÉCNICOS

- Cable USB macho a USB hembra.
- Cable de alimentación conector hembra C7 a conector macho.
- Aplicación Maqueta_Control_Temperatura.exe.
- Versión 9.7 (R2019b) de MATLAB Runtime.
- Aplicación MyAppInstaller_web.exe.

3.2 INSTALACIÓN Y CONFIGURACIÓN DEL SOFTWARE

En primer lugar, es necesario instalar la versión 9.7 (R2019b) de MATLAB Runtime.

Para ello, se debe acceder a la siguiente página desde donde se podrá descargar el archivo:

https://www.mathworks.com/products/compiler/mcr/index.html

Una vez instalado, es posible ejecutar la aplicación Maqueta_Control_Temperatura.exe. En este momento, el programa estará listo para ser utilizado.

Los pasos anteriores pueden ser realizados automáticamente ejecutando la aplicación MyAppInstaller_web.exe, la cual se encarga de realizar tanto la descarga como la instalación.

3.3 ELEMENTOS DE LA APLICACIÓN

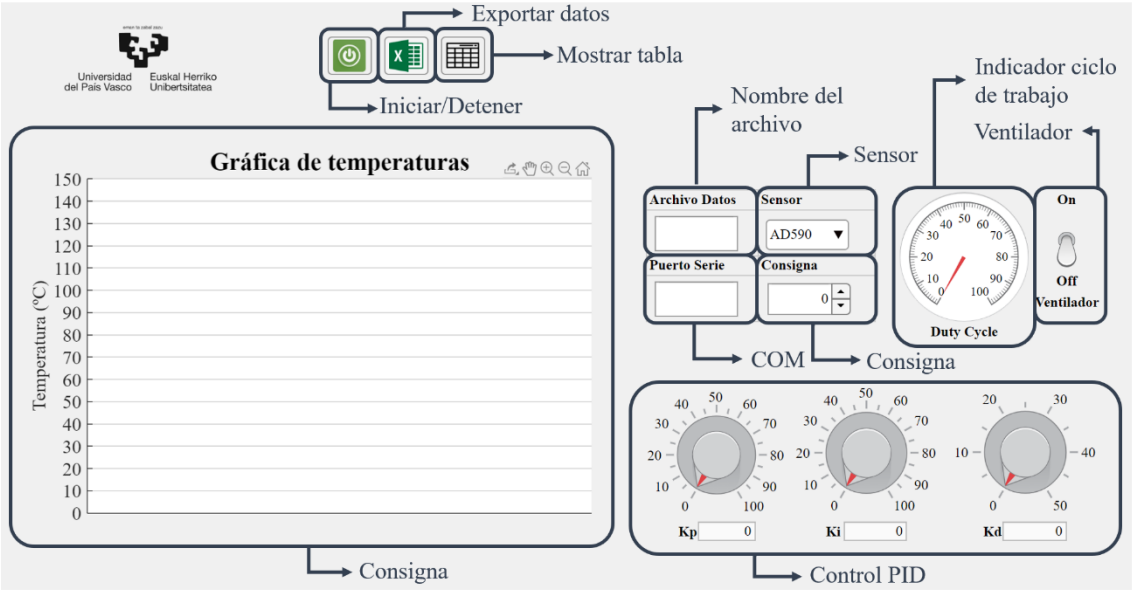


Figura 3 Elementos de la interfaz gráfica.

3.4 GUÍA DE USO

Antes de comenzar, se deben conectar los cables de alimentación y USB. A continuación. se procede al encendido, usando para ello el interruptor balancín rojo del panel frontal de la maqueta. La pantalla mostrará un mensaje de bienvenida, y una vez esté listo para funcionar aparecerán los datos. Antes de seleccionar el modo PC, es conveniente iniciar la aplicación. Para ello, se debe introducir el nombre del puerto COM del ordenador al que se encuentra conectado la maqueta.

El nombre del puerto viene indicado en el Administrador de dispositivos, en el apartado Puertos COM y LPT. En la siguiente figura se muestra un ejemplo, según el cual, el nombre a escribir en el campo Puerto Serie sería COM4.

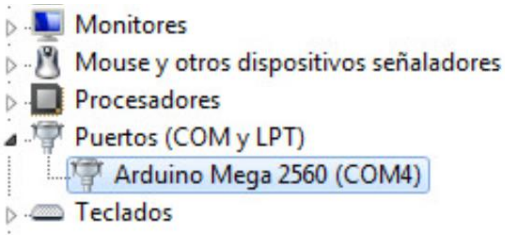


Figura 4 Puerto COM.

Una vez indicado, será posible iniciar la aplicación. Para ello, es necesario pulsar el primero de los tres botones situados en la parte superior de la interfaz gráfica.

La aplicación está lista para recibir los datos enviados por la maqueta. Para que la maqueta empiece a enviar datos, es necesario cambiar del modo manual al modo PC en el panel frontal de la maqueta.

Una vez conectados, se puede observar cómo en la gráfica de temperaturas comienzan a representarse los datos recibidos. A continuación, se procede a realizar el ensayo.

En primer lugar, se selecciona el sensor a utilizar mediante el menú deslizante. Este, por defecto selecciona el sensor AD590 al iniciar la aplicación.

A continuación, se definen las constantes proporcional, integral y derivativa a emplear por el control PID. Estas se modifican a través de los mandos de control.

Por último, se introduce el valor de la consigna de temperatura a alcanzar. Este valor puede ser introducido a través del teclado o bien desde los cursores del componente.

Para controlar el ventilador, se tiene el interruptor de encendido/apagado.

Una vez finalizado el ensayo, se detiene la conexión pulsando para ello el botón empleado para iniciarla. En este momento aún permanecen almacenados los datos del ensayo realizado. Para mostrar dichos datos, se debe pulsar el tercer botón de la parte superior, cuyo icono muestra una tabla. Si se desean exportar los datos a un archivo Excel para su posterior análisis, se debe introducir el nombre del archivo a generar en el campo Archivo datos y posteriormente pulsar el segundo botón de la parte superior, con el icono de un archivo Excel.

Para realizar el siguiente ensayo basta con iniciar de nuevo la aplicación, lo que restaura la interfaz gráfica y elimina los datos del ensayo anterior.

ANEXO VI: HOJAS DE CARACTERÍSTICAS

2-Terminal IC Temperature Transducer

Data Sheet

AD590

FEATURES

Linear current output: 1 µA/K
Wide temperature range: −55°C to +150°C
Probe-compatible ceramic sensor package
2-terminal device: voltage in/current out
Laser trimmed to ±0.5°C calibration accuracy (AD590M)
Excellent linearity: ±0.3°C over full range (AD590M)
Wide power supply range: 4 V to 30 V
Sensor isolation from case
Available in 2-lead FLATPACK, 4-lead LFCSP, 3-pin TO-52, 8-lead SOIC, and die form

GENERAL DESCRIPTION

The AD590 is a 2-terminal integrated circuit temperature transducer that produces an output current proportional to absolute temperature. For supply voltages between 4 V and 30 V, the device acts as a high impedance, constant current regulator passing 1 µA/K. Laser trimming of the chip's thin-film resistors is used to calibrate the device to 298.2 µA output at 298.2 K (25°C).

The AD590 should be used in any temperature-sensing application below 150°C in which conventional electrical temperature sensors are currently employed. The inherent low cost of a monolithic integrated circuit combined with the elimination of support circuitry makes the AD590 an attractive alternative for many temperature measurement situations. Linearization circuitry, precision voltage amplifiers, resistance measuring circuitry, and cold junction compensation are not needed in applying the AD590.

In addition to temperature measurement, applications include temperature compensation or correction of discrete components, biasing proportional to absolute temperature, flow rate measurement, level detection of fluids and anemometry. The AD590 is available in die form, making it suitable for hybrid circuits and fast temperature measurements in protected environments.

The AD590 is particularly useful in remote sensing applications. The device is insensitive to voltage drops over long lines due to its high impedance current output. Any well-insulated twisted pair is sufficient for operation at hundreds of feet from the receiving circuitry. The output characteristics also make the AD590 easy to multiplex: the current can be switched by a CMOS multiplexer, or the supply voltage can be switched by a logic gate output.

PIN CONFIGURATIONS

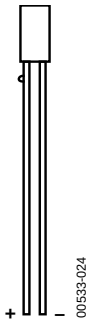


Figure 1. 2-Lead FLATPACK

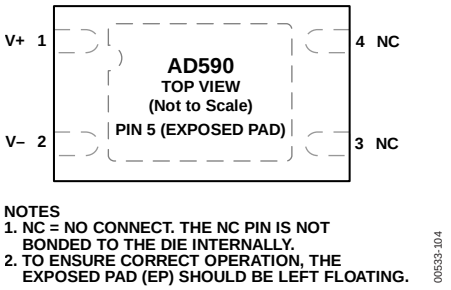


Figure 2. 4-Lead LFCSP

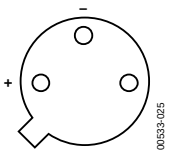


Figure 3. 3-Pin TO-52

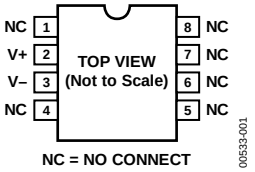


Figure 4. 8-Lead SOIC

PRODUCT HIGHLIGHTS

1. The AD590 is a calibrated, 2-terminal temperature sensor requiring only a dc voltage supply (4 V to 30 V). Costly transmitters, filters, lead wire compensation, and linearization circuits are all unnecessary in applying the device.
2. State-of-the-art laser trimming at the wafer level in conjunction with extensive final testing ensures that AD590 units are easily interchangeable.
3. Superior interface rejection occurs because the output is a current rather than a voltage. In addition, power requirements are low (1.5 mW @ 5 V @ 25°C). These features make the AD590 easy to apply as a remote sensor.
4. The high output impedance (>10 MΩ) provides excellent rejection of supply voltage drift. For instance, changing the power supply from 5 V to 10 V results in only a 1 µA maximum current change, or 1°C equivalent error.
5. The AD590 is electrically durable: it withstands a forward voltage of up to 44 V and a reverse voltage of 20 V. Therefore, supply irregularities or pin reversal does not damage the device.

Rev. G Document Feedback

One Technology Way, P.O. Box 9106, Norwood, MA 02062-9106, U.S.A.
Tel: 781.329.4700

Technical Support **www.analog.com**

SPECIFICATIONS

AD590J AND AD590K SPECIFICATIONS

25°C and V_S = 5 V, unless otherwise noted.[1]

Table 1.

	AD590J[2]			AD590K			
Parameter	**Min**	**Typ**	**Max**	**Min**	**Typ**	**Max**	**Unit**
POWER SUPPLY							
Operating Voltage Range	**4**		30	**4**		30	V
OUTPUT							
Nominal Current Output @ 25°C (298.2 K)		298.2			298.2		µA
Nominal Temperature Coefficient		1			1		µA/K
Calibration Error @ 25°C			**±5.0**			**±2.5**	°C
Absolute Error (Over Rated Performance Temperature Range)							
Without External Calibration Adjustment			**±10**			**±5.5**	°C
With 25°C Calibration Error Set to Zero			**±3.0**			**±2.0**	°C
Nonlinearity							
For TO-52 and FLATPACK Packages			**±1.5**			**±0.8**	°C
For 8-Lead SOIC Package			±1.5			±1.0	°C
For 4-Lead LFCSP Package			±1.5				°C
Repeatability[3]			±0.1			±0.1	°C
Long-Term Drift[4]			±0.1			±0.1	°C
Current Noise		40			40		pA/√Hz
Power Supply Rejection							
4 V ≤ V_S ≤ 5 V		0.5			0.5		µA/V
5 V ≤ V_S ≤ 15 V		0.2			0.2		µV/V
15 V ≤ V_S ≤ 30 V		0.1			0.1		µA/V
Case Isolation to Either Lead		10^{10}			10^{10}		Ω
Effective Shunt Capacitance		100			100		pF
Electrical Turn-On Time		20			20		µs
Reverse Bias Leakage Current (Reverse Voltage = 10 V)[5]		10			10		pA

[1] Specifications shown in **boldface** are tested on all production units at final electrical test. Results from those tests are used to calculate outgoing quality levels. All minimum and maximum specifications are guaranteed, although only those shown in **boldface** are tested on all production units.
[2] The LFCSP package has a reduced operating temperature range of −40°C to +125°C.
[3] Maximum deviation between +25°C readings after temperature cycling between −55°C and +150°C; guaranteed, not tested.
[4] Conditions: constant 5 V, constant 125°C; guaranteed, not tested.
[5] Leakage current doubles every 10°C.

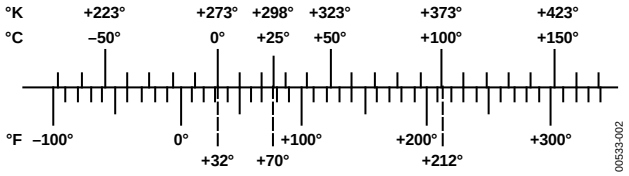
AD590L AND AD590M SPECIFICATIONS

25°C and V_S = 5 V, unless otherwise noted.[1]

Table 2.

	AD590L			AD590M			
Parameter	**Min**	**Typ**	**Max**	**Min**	**Typ**	**Max**	**Unit**
POWER SUPPLY							
Operating Voltage Range	**4**		30	**4**		30	V
OUTPUT							
Nominal Current Output @ 25°C (298.2 K)		298.2			298.2		µA
Nominal Temperature Coefficient		1			1		µA/K
Calibration Error @ 25°C			**±1.0**			**±0.5**	°C
Absolute Error (Over Rated Performance Temperature Range)							°C
Without External Calibration Adjustment			**±3.0**			**±1.7**	°C
With ± 25°C Calibration Error Set to Zero			**±1.6**			**±1.0**	°C
Nonlinearity			**±0.4**			**±0.3**	°C
Repeatability[2]			±0.1			±0.1	°C
Long-Term Drift[3]			±0.1			±0.1	°C
Current Noise		40			40		pA/√Hz
Power Supply Rejection							
4 V ≤ V_S ≤ 5 V		0.5			0.5		µA/V
5 V ≤ V_S ≤ 15 V		0.2			0.2		µA/V
15 V ≤ V_S ≤ 30 V		0.1			0.1		µA/V
Case Isolation to Either Lead		10^{10}			10^{10}		Ω
Effective Shunt Capacitance		100			100		pF
Electrical Turn-On Time		20			20		µs
Reverse Bias Leakage Current (Reverse Voltage = 10 V)[4]		10			10		pA

[1] Specifications shown in **boldface** are tested on all production units at final electrical test. Results from those tests are used to calculate outgoing quality levels. All minimum and maximum specifications are guaranteed, although only those shown in **boldface** are tested on all production units.
[2] Maximum deviation between +25°C readings after temperature cycling between −55°C and +150°C; guaranteed, not tested.
[3] Conditions: constant 5 V, constant 125°C; guaranteed, not tested.
[4] Leakage current doubles every 10°C.



$$^{\circ}C = \frac{5}{9}\left(^{\circ}F - 32\right) K = {}^{\circ}C + 273.15$$

$$^{\circ}F = \left(\frac{9}{5}\,{}^{\circ}C + 32\right)\,{}^{\circ}R = {}^{\circ}F + 459.7$$

Figure 5. Temperature Scale Conversion Equations

PRODUCT DESCRIPTION

The AD590 is a 2-terminal temperature-to-voltage transducer. It is available in a variety of accuracy grades and packages. When using the AD590 in die form, the chip substrate must be kept electrically isolated (floating) for correct circuit operation.

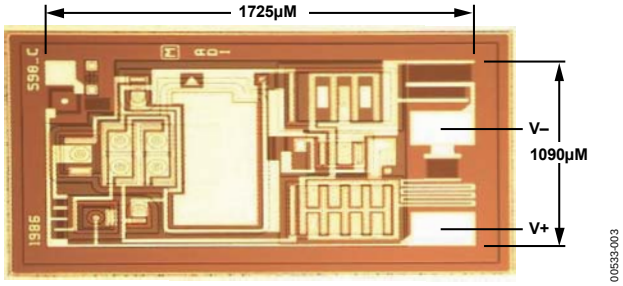


Figure 6. Metallization Diagram

The AD590 uses a fundamental property of the silicon transistors from which it is made to realize its temperature proportional characteristic: if two identical transistors are operated at a constant ratio of collector current densities, r, then the difference in their base-emitter voltage is (kT/q)(In r). Because both k (Boltzman's constant) and q (the charge of an electron) are constant, the resulting voltage is directly proportional to absolute temperature (PTAT). (For a more detailed description, see M.P. Timko, "A Two-Terminal IC Temperature Transducer," IEEE J. Solid State Circuits, Vol. SC-11, p. 784-788, Dec. 1976. Understanding the Specifications–AD590.)

In the AD590, this PTAT voltage is converted to a PTAT current by low temperature coefficient thin-film resistors. The total current of the device is then forced to be a multiple of this PTAT current. Figure 7 is the schematic diagram of the AD590. In this figure, Q8 and Q11 are the transistors that produce the PTAT voltage. R5 and R6 convert the voltage to current. Q10, whose collector current tracks the collector currents in Q9 and Q11, supplies all the bias and substrate leakage current for the rest of the circuit, forcing the total current to be PTAT. R5 and R6 are laser-trimmed on the wafer to calibrate the device at 25°C.

Figure 8 shows the typical V–I characteristic of the circuit at 25°C and the temperature extremes.

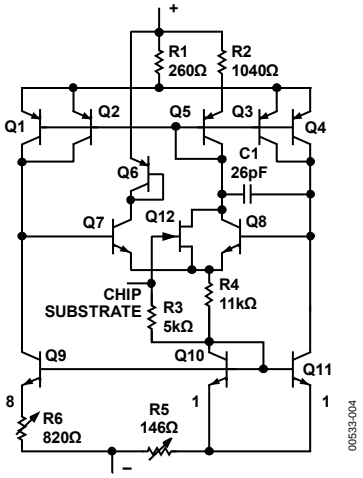


Figure 7. Schematic Diagram

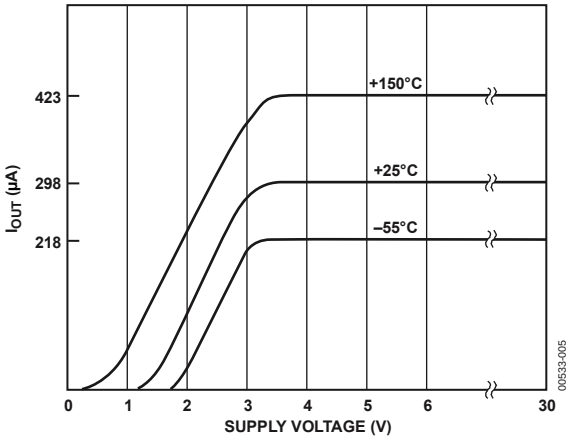


Figure 8. V–I Plot

EXPLANATION OF TEMPERATURE SENSOR SPECIFICATIONS

The way in which the AD590 is specified makes it easy to apply it in a wide variety of applications. It is important to understand the meaning of the various specifications and the effects of the supply voltage and thermal environment on accuracy.

The AD590 is a PTAT current regulator. (Note that T (°C) = T (K) − 273.2. Zero on the Kelvin scale is absolute zero; there is no lower temperature.) That is, the output current is equal to a scale factor times the temperature of the sensor in degrees Kelvin. This scale factor is trimmed to 1 μA/K at the factory, by adjusting the indicated temperature (that is, the output current) to agree with the actual temperature. This is done with 5 V across the device at a temperature within a few degrees of 25°C (298.2 K). The device is then packaged and tested for accuracy over temperature.

CALIBRATION ERROR

At final factory test, the difference between the indicated temperature and the actual temperature is called the calibration error. Since this is a scale factory error, its contribution to the total error of the device is PTAT. For example, the effect of the 1°C specified maximum error of the AD590L varies from 0.73°C at −55°C to 1.42°C at 150°C. Figure 9 shows how an exaggerated calibration error would vary from the ideal over temperature.

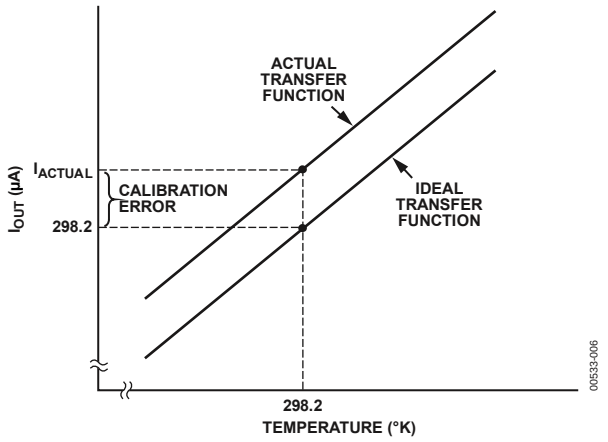


Figure 9. Calibration Error vs. Temperature

The calibration error is a primary contributor to the maximum total error in all AD590 grades. However, because it is a scale factor error, it is particularly easy to trim. Figure 10 shows the most elementary way of accomplishing this.

To trim this circuit, the temperature of the AD590 is measured by a reference temperature sensor and R is trimmed so that V_T = 1 mV/K at that temperature. Note that when this error is trimmed out at one temperature, its effect is zero over the entire temperature range. In most applications, there is a current-to-voltage conversion resistor (or, as with a current input ADC, a reference) that can be trimmed for scale factor adjustment.

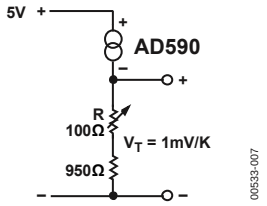


Figure 10. One Temperature Trim

ERROR VS. TEMPERATURE: CALIBRATION ERROR TRIMMED OUT

Each AD590 is tested for error over the temperature range with the calibration error trimmed out. This specification could also be called the variance from PTAT, because it is the maximum difference between the actual current over temperature and a PTAT multiplication of the actual current at 25°C. This error consists of a slope error and some curvature, mostly at the temperature extremes. Figure 11 shows a typical AD590K temperature curve before and after calibration error trimming.

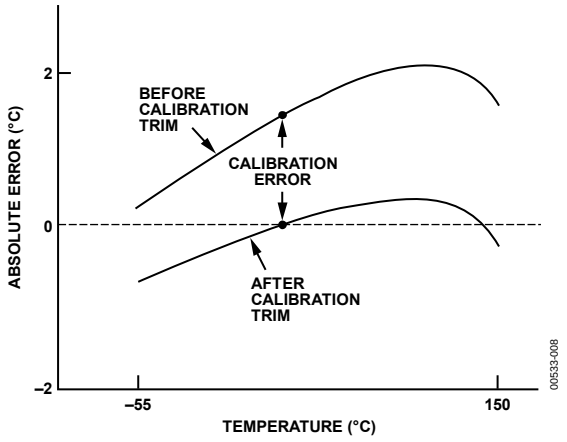


Figure 11. Effect to Scale Factor Trim on Accuracy

ERROR VS. TEMPERATURE: NO USER TRIMS

Using the AD590 by simply measuring the current, the total error is the variance from PTAT, described above, plus the effect of the calibration error over temperature. For example, the AD590L maximum total error varies from 2.33°C at −55°C to 3.02°C at 150°C. For simplicity, only the large figure is shown on the specification page.

NONLINEARITY

Nonlinearity as it applies to the AD590 is the maximum deviation of current over temperature from a best-fit straight line. The nonlinearity of the AD590 over the −55°C to +150°C range is superior to all conventional electrical temperature sensors such as thermocouples, RTDs, and thermistors. Figure 12 shows the nonlinearity of the typical AD590K from Figure 11.

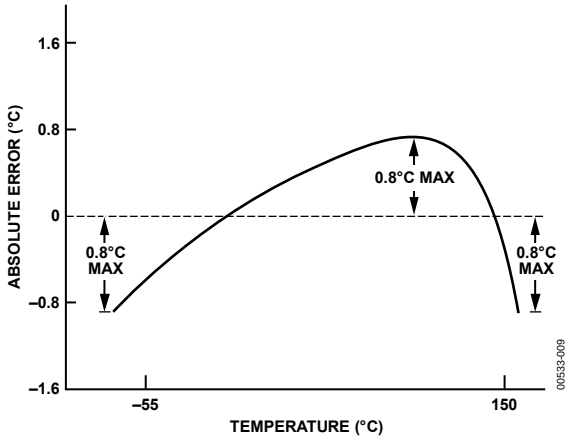


Figure 12. Nonlinearity

Figure 13 shows a circuit in which the nonlinearity is the major contributor to error over temperature. The circuit is trimmed by adjusting R1 for a 0 V output with the AD590 at 0°C. R2 is then adjusted for 10 V output with the sensor at 100°C. Other pairs of temperatures can be used with this procedure as long as they are measured accurately by a reference sensor. Note that for 15 V output (150°C), the V+ of the op amp must be greater than 17 V. Also, note that V− should be at least −4 V; if V− is ground, there is no voltage applied across the device.

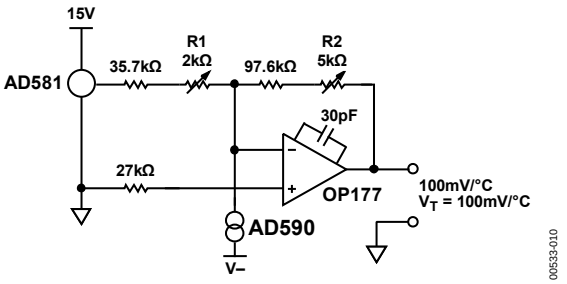


Figure 13. 2-Temperature Trim

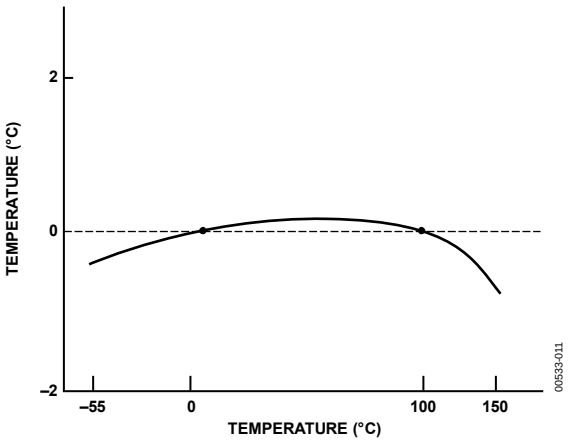


Figure 14. Typical 2-Trim Accuracy

VOLTAGE AND THERMAL ENVIRONMENT EFFECTS

The power supply rejection specifications show the maximum expected change in output current vs. input voltage changes. The insensitivity of the output to input voltage allows the use of unregulated supplies. It also means that hundreds of ohms of resistance (such as a CMOS multiplexer) can be tolerated in series with the device.

It is important to note that using a supply voltage other than 5 V does not change the PTAT nature of the AD590. In other words, this change is equivalent to a calibration error and can be removed by the scale factor trim (see Figure 11).

The AD590 specifications are guaranteed for use in a low thermal resistance environment with 5 V across the sensor. Large changes in the thermal resistance of the sensor's environment change the amount of self-heating and result in changes in the output, which are predictable but not necessarily desirable.

The thermal environment in which the AD590 is used determines two important characteristics: the effect of self-heating and the response of the sensor with time. Figure 15 is a model of the AD590 that demonstrates these characteristics.

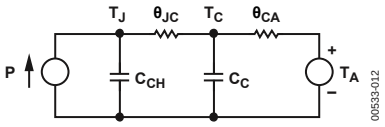


Figure 15. Thermal Circuit Model

As an example, for the TO-52 package, θ_{JC} is the thermal resistance between the chip and the case, about 26°C/W. θ_{CA} is the thermal resistance between the case and the surroundings and is determined by the characteristics of the thermal connection. Power source P represents the power dissipated on the chip. The rise of the junction temperature, T_J, above the ambient temperature, T_A, is

$$T_J - T_A = P(\theta_{JC} + \theta_{CA}) \quad (1)$$

Table 4 gives the sum of θ_{JC} and θ_{CA} for several common thermal media for both the H and F packages. The heat sink used was a common clip-on. Using Equation 1, the temperature rise of an AD590 H package in a stirred bath at 25°C, when driven with a 5 V supply, is 0.06°C. However, for the same conditions in still air, the temperature rise is 0.72°C. For a given supply voltage, the temperature rise varies with the current and is PTAT. Therefore, if an application circuit is trimmed with the sensor in the same thermal environment in which it is used, the scale factor trim compensates for this effect over the entire temperature range.

Table 4. Thermal Resistance

	$\theta_{JC} + \theta_{CA}$ (°C/Watt)		τ (sec)[1]	
Medium	**H**	**F**	**H**	**F**
Aluminum Block	30	10	0.6	0.1
Stirred Oil[2]	42	60	1.4	0.6
Moving Air[3]				
With Heat Sink	45	–	5.0	–
Without Heat Sink	115	190	13.5	10.0
Still Air				
With Heat Sink	191	–	108	–
Without Heat Sink	480	650	60	30

[1] τ is dependent upon velocity of oil; average of several velocities listed above.
[2] Air velocity @ 9 ft/sec.
[3] The time constant is defined as the time required to reach 63.2% of an instantaneous temperature change.

The time response of the AD590 to a step change in temperature is determined by the thermal resistances and the thermal capacities of the chip, C_{CH}, and the case, C_C. C_{CH} is about 0.04 Ws/°C for the AD590. C_C varies with the measured medium, because it includes anything that is in direct thermal contact with the case. The single time constant exponential curve of Figure 16 is usually sufficient to describe the time response, T (t). Table 4 shows the effective time constant, τ, for several media.

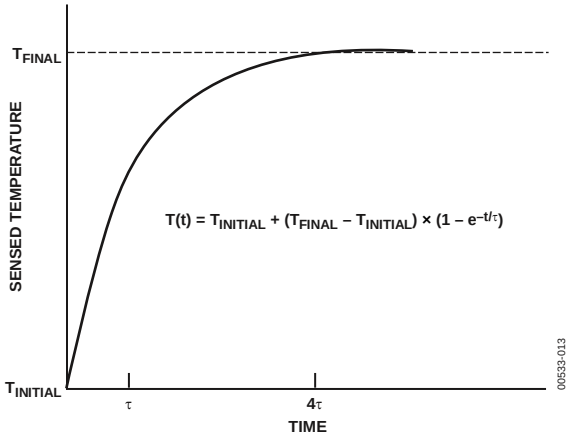


Figure 16. Time Response Curve

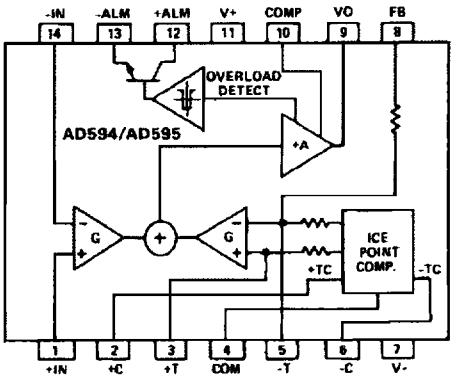
Monolithic Thermocouple Amplifiers with Cold Junction Compensation

AD594*/AD595*

FEATURES

- **Pretrimmed for Type J (AD594) or Type K (AD595) Thermocouples**
- **Can Be Used with Type T Thermocouple Inputs**
- **Low Impedance Voltage Output: 10mV/°C**
- **Built-in Ice Point Compensation**
- **Wide Power Supply Range: +5V to ±15V**
- **Low Power: <1mW typical**
- **Thermocouple Failure Alarm**
- **Laser Wafer Trimmed to 1°C Calibration Accuracy**
- **Set-Point Mode Operation**
- **Self-Contained Celsius Thermometer Operation**
- **High Impedance Differential Input**
- **Side-Brazed DIP or Low Cost Cerdip**

FUNCTIONAL BLOCK DIAGRAM



PRODUCT DESCRIPTION

The AD594/AD595 is a complete instrumentation amplifier and thermocouple cold junction compensator on a monolithic chip. It combines an ice point reference with a precalibrated amplifier to produce a high level (10mV/°C) output directly from a thermocouple signal. Pin-strapping options allow it to be used as a linear amplifier-compensator or as a switched output set-point controller using either fixed or remote set-point control. It can be used to amplify its compensation voltage directly, thereby converting it to a stand-alone Celsius transducer with a low-impedance voltage output.

The AD594/AD595 includes a thermocouple failure alarm that indicates if one or both thermocouple leads become open. The alarm output has a flexible format which includes TTL drive capability.

The AD594/AD595 can be powered from a single ended supply (including +5V) and by including a negative supply, temperatures below 0°C can be measured. To minimize self-heating, an unloaded AD594/AD595 will typically operate with a total supply current of 160μA, but is also capable of delivering in excess of ±5mA to a load.

The AD594 is precalibrated by laser wafer trimming to match the characteristic of type J (iron-constantan) thermocouples and the AD595 is laser trimmed for type K (chromel-alumel) inputs. The temperature transducer voltages and gain control resistors are available at the package pins so that the circuit can be recalibrated for other thermocouple types by the addition of two or three resistors. These terminals also allow more precise calibration for both thermocouple and thermometer applications.

*Protected by U.S. Patent No. 4,029,974.

The AD594/AD595 is available in two performance grades. The C and the A versions have calibration accuracies of ±1°C and ±3°C, respectively. Both are designed to be used from 0 to +50°C, and are available in 14-pin, hermetically sealed, side-brazed ceramic DIPs as well as low cost cerdip packages.

PRODUCT HIGHLIGHTS

1. The AD594/AD595 provides cold junction compensation, amplification, and an output buffer in a single IC package.
2. Compensation, zero, and scale factor are all precalibrated by laser wafer trimming (LWT) of each IC chip.
3. Flexible pin-out provides for operation as a set-point controller or a stand-alone temperature transducer calibrated in degrees Celsius.
4. Operation at remote application sites is facilitated by low quiescent current and a wide supply voltage range of +5V to dual supplies spanning 30V.
5. Differential input rejects common-mode noise voltage on the thermocouple leads.

AD594/AD595—SPECIFICATIONS (@ +25°C and V_S = 5 V, Type J (AD594), Type K (AD595) Thermocouple, unless otherwise noted)

Model	AD594A			AD594C			AD595A			AD595C			Units
	Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
ABSOLUTE MAXIMUM RATINGS													
$+V_S$ to $-V_S$			36			36			36			36	Volts
Common-Mode Input Voltage	$-V_S$ −0.15		$+V_S$	$-V_S$ −0.15		$+V_S$	$-V_S$ −0.15		$+V_S$	$-V_S$ −0.15		$+V_S$	Volts
Differential Input Voltage	$-V_S$		$+V_S$	$-V_S$		$+V_S$	$-V_S$		$+V_S$	$-V_S$		$+V_S$	Volts
Alarm Voltages													
+ALM	$-V_S$		$-V_S$ +36	$-V_S$		$-V_S$ +36	$-V_S$		$-V_S$ +36	$-V_S$		$-V_S$ +36	Volts
−ALM	$-V_S$		$+V_S$	$-V_S$		$+V_S$	$-V_S$		$+V_S$	$-V_S$		$+V_S$	Volts
Operating Temperature Range	−55		+125	−55		+125	−55		+125	−55		+125	°C
Output Short Circuit to Common	Indefinite			Indefinite			Indefinite			Indefinite			
TEMPERATURE MEASUREMENT (Specified Temperature Range 0 to +50°C)													
Calibration Error at +25°C[1]			±3			±1			±3			±1	°C
Stability vs. Temperature[2]			±0.05			±0.025			±0.05			±0.025	°C/°C
Gain Error			±1.5			±0.75			±1.5			±0.75	%
Nominal Transfer Function			10			10			10			10	mV/°C
AMPLIFIER CHARACTERISTICS													
Closed Loop Gain[3]		193.4			193.4			247.3			247.3		
Input Offset Voltage	(Temperature in °C) × 51.70 μV/°C			(Temperature in °C) × 51.70 μV/°C			(Temperature in °C) × 40.44 μV/°C			(Temperature in °C) × 40.44 μV/°C			μV
Input Bias Current		0.1			0.1			0.1			0.1		μA
Differential Input Range	−10		+50				−10		+50	−10		+50	mV
Common-Mode Range	$-V_S$ −0.15		$+V_S$ −4	$-V_S$ −0.15		$+V_S$ −4	$-V_S$ −0.15		$+V_S$ −4	$-V_S$ −0.15		$+V_S$ −4	Volts
Common-Mode Sensitivity – RTO			10			10			10			10	mV/V
Power Supply Sensitivity – RTO			10			10			10			10	mV/V
Output Voltage Range													
Dual Supplies	$-V_S$ +2.5		$+V_S$ −2	$-V_S$ +2.5		$+V_S$ −2	$-V_S$ +2.5		$+V_S$ −2	$-V_S$ +2.5		$+V_S$ −2	Volts
Single Supply	0		$+V_S$ −2	0		$+V_S$ −2	0		$+V_S$ +2	0		$+V_S$ −2	Volts
Usable Output Current[4]		±5			±5			±5			±5		mA
3 dB Bandwidth		15			15			15			15		kHz
ALARM CHARACTERISTICS													
$V_{CE(SAT)}$ at 2 mA		0.3			0.3			0.3			0.3		Volts
Leakage Current			±1			±1			±1			±1	μA max
Operating Voltage at – ALM			$+V_S$ −4			$+V_S$ −4			$+V_S$ −4			$+V_S$ −4	Volts
Short Circuit Current		20			20			20			20		mA
POWER REQUIREMENTS													
Specified Performance	$+V_S$ = 5, $-V_S$ = 0			$+V_S$ = 5, $-V_S$ = 0			$+V_S$ = 5, $-V_S$ = 0			$+V_S$ = 5, $-V_S$ = 0			Volts
Operating[5]	$+V_S$ to $-V_S$ ≤ 30			$+V_S$ to $-V_S$ ≤ 30			$+V_S$ to $-V_S$ ≤ 30			$+V_S$ to $-V_S$ ≤ 30			Volts
Quiescent Current (No Load)													
$+V_S$		160	300		160	300		160	300		160	300	μA
$-V_S$		100			100			100			100		μA
PACKAGE OPTION[6]													
TO-116 (D-14)	AD594AD			AD594CD			AD595AD			AD595CD			
Cerdip (Q-14A)	AD594AQ			AD594CQ			AD595AQ			AD595CQ			

NOTES

[1]Calibrated for minimum error at +25°C using a thermocouple sensitivity of 51.7 μV/°C. Since a J type thermocouple deviates from this straight line approximation, the AD594 will normally read 3.1 mV when the measuring junction is at 0°C. The AD595 will similarly read 2.7 mV at 0°C.

[2]Defined as the slope of the line connecting the AD594/AD595 errors measured at 0°C and 50°C ambient temperature.

[3]Pin 8 shorted to Pin 9.

[4]Current Sink Capability in single supply configuration is limited to current drawn to ground through a 50 kΩ resistor at output voltages below 2.5 V.

[5]$-V_S$ must not exceed −16.5 V.

[6]For outline information see Package Information section.

Specifications subject to change without notice.

Specifications shown in **boldface** are tested on all production units at final electrical test. Results from those tests are used to calculate outgoing quality levels.
All min and max specifications are guaranteed, although only those shown in boldface are tested on all production units.

Thermocouple Temperature °C	Type J Voltage mV	AD594 Output mV	Type K Voltage mV	AD595 Output mV	Thermocouple Temperature °C	Type J Voltage mV	AD594 Output mV	Type K Voltage mV	AD595 Output mV
-200	-7.890	-1523	-5.891	-1454	500	27.388	5300	20.640	5107
-180	-7.402	-1428	-5.550	-1370	520	28.511	5517	21.493	5318
-160	-6.821	-1316	-5.141	-1269	540	29.642	5736	22.346	5529
-140	-6.159	-1188	-4.669	-1152	560	30.782	5956	23.198	5740
-120	-5.426	-1046	-4.138	-1021	580	31.933	6179	24.050	5950
-100	-4.632	-893	-3.553	-876	600	33.096	6404	24.902	6161
-80	-3.785	-729	-2.920	-719	620	34.273	6632	25.751	6371
-60	-2.892	-556	-2.243	-552	640	35.464	6862	26.599	6581
-40	-1.960	-376	-1.527	-375	660	36.671	7095	27.445	6790
-20	-.995	-189	-.777	-189	680	37.893	7332	28.288	6998
-10	-.501	-94	-.392	-94	700	39.130	7571	28.128	7206
0	0	3.1	0	2.7	720	40.382	7813	29.965	7413
10	.507	101	.397	101	740	41.647	8058	30.799	7619
20	1.019	200	.798	200	750	42.283	8181	31.214	7722
25	1.277	250	1.000	250	760	-	-	31.629	7825
30	1.536	300	1.203	300	780	-	-	32.455	8029
40	2.058	401	1.611	401	800	-	-	33.277	8232
50	2.585	503	2.022	503	820	-	-	34.095	8434
60	3.115	606	2.436	605	840	-	-	34.909	8636
80	4.186	813	3.266	810	860	-	-	35.718	8836
100	5.268	1022	4.095	1015	880	-	-	36.524	9035
120	6.359	1233	4.919	1219	900	-	-	37.325	9233
140	7.457	1445	5.733	1420	920	-	-	38.122	9430
160	8.560	1659	6.539	1620	940	-	-	38.915	9626
180	9.667	1873	7.338	1817	960	-	-	39.703	9821
200	10.777	2087	8.137	2015	980	-	-	40.488	10015
220	11.887	2302	8.938	2213	1000	-	-	41.269	10209
240	12.998	2517	9.745	2413	1020	-	-	42.045	10400
260	14.108	2732	10.560	2614	1040	-	-	42.817	10591
280	15.217	2946	11.381	2817	1060	-	-	43.585	10781
300	16.325	3160	12.207	3022	1080	-	-	44.349	10970
320	17.432	3374	13.039	3327	1100	-	-	45.108	11158
340	18.537	3588	13.874	3434	1120	-	-	45.863	11345
360	19.640	3801	14.712	3641	1140	-	-	46.612	11530
380	20.743	4015	15.552	3849	1160	-	-	47.356	11714
400	21.846	4228	16.395	4057	1180	-	-	48.095	11897
420	22.949	4441	17.241	4266	1200	-	-	48.828	12078
440	24.054	4655	18.088	4476	1220	-	-	49.555	12258
460	25.161	4869	18.938	4686	1240	-	-	50.276	12436
480	26.272	5084	19.788	4896	1250	-	-	50.633	12524

Table I. Output Voltage vs. Thermocouple Temperature (Ambient +25°C, V_S = −5V, +15V)

INTERPRETING AD594/AD595 OUTPUT VOLTAGES

To achieve a temperature proportional output of 10mV/°C and accurately compensate for the reference junction over the rated operating range of the circuit, the AD594/AD595 is gain trimmed to match the transfer characteristic of J and K type thermocouples at 25°C. For a type J output in this temperature range the TC is 51.70μV/°C, while for a type K it is 40.44μV/°C. The resulting gain for the AD594 is 193.4 (10mV/°C divided by 51.7μV/°C) and for the AD595 is 247.3 (10mV/°C divided by 40.44μV/°C). In addition, an absolute accuracy trim induces an input offset to the output amplifier characteristic of 16μV for the AD594 and 11μV for the AD595. This offset arises because the AD594/AD595 is trimmed for a 250mV output while applying a 25°C thermocouple input.

Because a thermocouple output voltage is nonlinear with respect to temperature, and the AD594/AD595 linearly amplifies the compensated signal, the following transfer functions should be used to determine the actual output voltages:

AD594 output = (Type J Voltage + 16μV) × 193.4

AD595 output = (Type K Voltage + 11μV) × 247.3

or conversely:

Type J voltage = (AD594 output / 193.4) − 16μV

Type K voltage = (AD595 output / 247.3) − 11μV

Table I above lists the ideal AD594/AD595 output voltages as a function of Celsius temperature for type J and K ANSI standard thermocouples, with the package and reference junction at 25°C. As is normally the case, these outputs are subject to calibration, gain and temperature sensitivity errors. Output values for intermediate temperatures can be interpolated, or calculated using the output equations and ANSI thermocouple voltage tables referred to zero degrees Celsius. Due to a slight variation in alloy content between ANSI type J and DIN FE–CUNI thermocouples Table I should not be used in conjunction with European standard thermocouples. Instead the transfer function given previously and a DIN thermocouple table should be used. ANSI type K and DIN NICR–NI thermocouples are composed of identical alloys and exhibit similar behavior. The upper temperature limits in Table I are those recommended for type J and type K thermocouples by the majority of vendors.

SINGLE AND DUAL SUPPLY CONNECTIONS

The AD594/AD595 is a completely self-contained thermocouple conditioner. Using a single +5V supply the interconnections shown in Figure 1 will provide a direct output from a type J thermocouple (AD594) or type K thermocouple (AD595) measuring from 0 to +300°C.

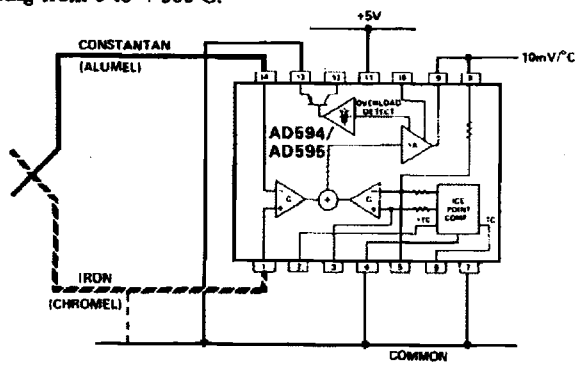


Figure 1. Basic Connection, Single Supply Operation

Any convenient supply voltage from +5V to +30V may be used, with self-heating errors being minimized at lower supply levels. In the single supply configuration the +5V supply connects to pin 11 with the V− connection at pin 7 strapped to power and signal common at pin 4. The thermocouple wire inputs connect to pins 1 and 14 either directly from the measuring point or through intervening connections of similar thermocouple wire type. When the alarm output at pin 13 is not used it should be connected to common or −V. The precalibrated feedback network at pin 8 is tied to the output at pin 9 to provide a 10mV/°C nominal temperature transfer characteristic.

By using a wider ranging dual supply, as shown in Figure 2, the AD594/AD595 can be interfaced to thermocouples measuring both negative and extended positive temperatures.

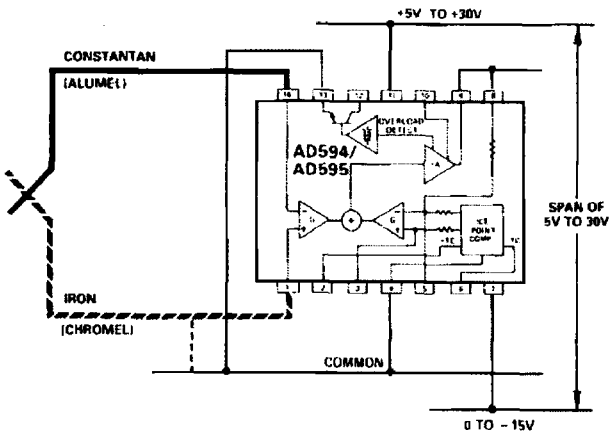


Figure 2. Dual Supply Operation

With a negative supply the output can indicate negative temperatures and drive grounded loads or loads returned to positive voltages. Increasing the positive supply from 5V to 15V extends the output voltage range well beyond the 750°C temperature limit recommended for type J thermocouples (AD594) and the 1250°C for type K thermocouples (AD595).

Common-mode voltages on the thermocouple inputs must remain within the common-mode range of the AD594/AD595, with a return path provided for the bias currents. If the thermocouple is not remotely grounded, then the dotted line connections in Figures 1 and 2 are recommended. A resistor may be needed in this connection to assure that common mode voltages induced in the thermocouple loop are not converted to normal mode.

THERMOCOUPLE CONNECTIONS

The isothermal terminating connections of a pair of themocouple wires forms an effective reference junction. This junction must be kept at the same temperature as the AD594/AD595 for the internal cold junction compensation to be effective.

A method that provides for thermal equilibrium is the printed circuit board connection layout illustrated in Figure 3.

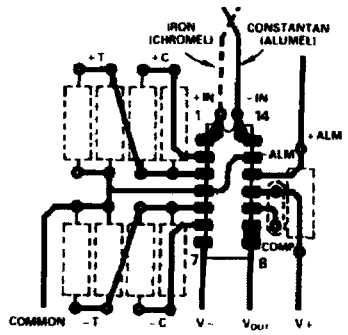


Figure 3. PCB Connections

Here the AD594/AD595 package temperature and circuit board are thermally contacted in the copper printed circuit board tracks under pins 1 and 14. The reference junction is now composed of a copper-constantan (or copper-alumel) connection and copper-iron (or copper-chromel) connection, both of which are at the same temperature as the AD594/AD595.

The printed circuit board layout shown also provides for placement of optional alarm load resistors, recalibration resistors and a compensation capacitor to limit bandwidth.

To ensure secure bonding the thermocouple wire should be cleaned to remove oxidation prior to soldering. Noncorrosive rosin flux is effective with iron, constantan, chromel and alumel and the following solders: 95% tin-5% antimony, 95% tin-5% silver or 90% tin-10% lead.

FUNCTIONAL DESCRIPTION

The AD594 behaves like two differential amplifiers. The outputs are summed and used to control a high-gain amplifier, as shown in Figure 4.

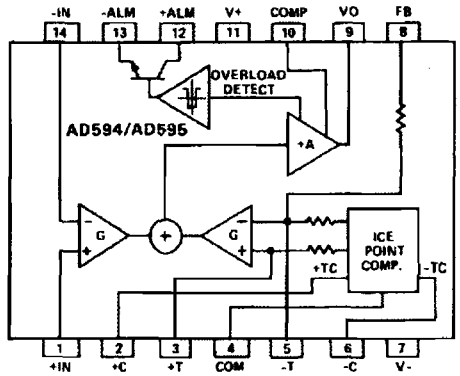


Figure 4. AD594/AD595 Block Diagram

In normal operation the main amplifier output, at pin 9, is connected to the feedback network, at pin 8. Thermocouple signals applied to the floating input stage, at pins 1 and 14, are

amplified by gain G of the differential amplifier and are then further amplified by gain A in the main amplifier. The output of the main amplifier is fed back to a second differential stage in an inverting connection. The feedback signal is amplified by this stage and is also applied to the main amplifier input through a summing circuit. Because of the inversion, the amplifier causes the feedback to be driven to reduce this difference signal to a small value. The two differential amplifiers are made to match and have identical gains, G. As a result, the feedback signal that must be applied to the right-hand differential amplifier will precisely match the thermocouple input signal when the difference signal has been reduced to zero. The feedback network is trimmed so that the effective gain to the output, at pins 8 and 9, results in a voltage of 10mV/°C of thermocouple excitation.

In addition to the feedback signal, a cold junction compensation voltage is applied to the right-hand differential amplifier. The compensation is a differential voltage proportional to the Celsius temperature of the AD594/AD595. This signal disturbs the differential input so that the amplifier output must adjust to restore the input to equal the applied thermocouple voltage.

The compensation is applied through the gain scaling resistors so that its effect on the main output is also 10mV/°C. As a result, the compensation voltage adds to the effect of the thermocouple voltage a signal directly proportional to the difference between 0°C and the AD594/AD595 temperature. If the thermocouple reference junction is maintained at the AD594/AD595 temperature, the output of the AD594/AD595 will correspond to the reading that would have been obtained from amplification of a signal from a thermocouple referenced to an ice bath.

The AD594/AD595 also includes an input open circuit detector that switches on an alarm transistor. This transistor is actually a current-limited output buffer, but can be used up to the limit as a switch transistor for either pull-up or pull-down operation of external alarms.

The ice point compensation network has voltages available with positive and negative temperature coefficients. These voltages may be used with external resistors to modify the ice point compensation and recalibrate the AD594/AD595 as described in the next column.

The feedback resistor is separately pinned out so that its value can be padded with a series resistor, or replaced with an external resistor between pins 5 and 9. External availability of the feedback resistor allows gain to be adjusted, and also permits the AD594/AD595 to operate in a switching mode for set-point operation.

CAUTIONS:

The temperature compensation terminals (+C and −C) at pins 2 and 6 are provided to supply small calibration currents only. The AD594/AD595 may be permanently damaged if they are grounded or connected to a low impedance.

The AD594/AD595 is internally frequency compensated for feedback ratios (corresponding to normal signal gain) of 75 or more. If a lower gain is desired, additional frequency compensation should be added in the form of a 300pF capacitor from pin 10 to the output at pin 9. As shown in Figure 5 an additional 0.01μF capacitor between pins 10 and 11 is recommended.

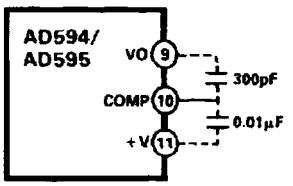


Figure 5. Low Gain Frequency Compensation

RECALIBRATION PRINCIPLES AND LIMITATIONS

The ice point compensation network of the AD594/AD595 produces a differential signal which is zero at 0°C and corresponds to the output of an ice referenced thermocouple at the temperature of the chip. The positive TC output of the circuit is proportional to Kelvin temperature and appears as a voltage at +T. It is possible to decrease this signal by loading it with a resistor from +T to COM, or increase it with a pull-up resistor from +T to the larger positive TC voltage at +C. Note that adjustments to +T should be made by measuring the voltage which tracks it at −T. To avoid destabilizing the feedback amplifier the measuring instrument should be isolated by a few thousand ohms in series with the lead connected to −T.

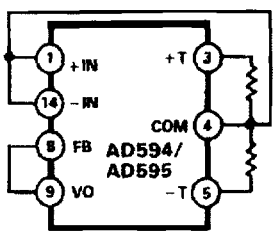


Figure 6. Decreased Sensitivity Adjustment

Changing the positive TC half of the differential output of the compensation scheme shifts the zero point away from 0°C. The zero can be restored by adjusting the current flow into the negative input of the feedback amplifier, the −T pin. A current into this terminal can be produced with a resistor between −C and −T to balance an increase in +T, or a resistor from −T to COM to offset a decrease in +T.

If the compensation is adjusted substantially to accommodate a different thermocouple type, its effect on the final output voltage will increase or decrease in proportion. To restore the nominal output to 10mV/°C the gain may be adjusted to match the new compensation and thermocouple input characteristics. When reducing the compensation the resistance between −T and COM automatically increases the gain to within 0.5% of the correct value. If a smaller gain is required, however, the nominal 47kΩ internal feedback resistor can be paralleled or replaced with an external resistor.

Fine calibration adjustments will require temperature response measurements of individual devices to assure accuracy. Major reconfigurations for other thermocouple types can be achieved without seriously compromising initial calibration accuracy, so long as the procedure is done at a fixed temperature using the factory calibration as a reference. It should be noted that intermediate recalibration conditions may require the use of a negative supply. An example using a type E thermocouple and an AD594 is given on the next page.

EXAMPLE: TYPE E RECALIBRATION – AD594/AD595

Both the AD594 and AD595 can be configured to condition the output of a type E (chromel-constantan) thermocouple. Temperature characteristics of type E thermocouples differ less from type J, than from type K, therefore the AD594 is preferred for recalibration.

While maintaining the device at a constant temperature follow the recalibration steps given here. First, measure the device temperature by tying both inputs to common (or a selected common mode potential) and connecting FB to V_O. The AD594 is now in the stand alone Celsius thermometer mode. For this example assume the ambient is 24°C and the initial output V_O is 240mV. Check the output at V_O to verify that it corresponds to the temperature of the device.

Next, measure the voltage $-T$ at pin 5 with a high impedance DVM (capacitance should be isolated by a few thousand ohms of resistance at the measured terminals). At 24°C the $-T$ voltage will be about 8.3mV. To adjust the compensation of an AD594 to a type E thermocouple a resistor, R1, should be connected between $+T$ and $+C$, pins 2 and 3, to raise the voltage at $-T$ by the ratio of thermocouple sensitivities. The ratio for converting a type J device to a type E characteristic is:

$$r\,(AD594) = (60.9\mu V/°C)/(51.7\mu V/°C) = 1.18$$

Thus, multiply the initial voltage measured at $-T$ by r and experimentally determine the R1 value required to raise $-T$ to that level. For the example the new $-T$ voltage should be about 9.8mV. The resistance value should be approximately 1.8kΩ.

The zero differential point must now be shifted back to 0°C. This is accomplished by multiplying the original output voltage V_O by r and adjusting the measured output voltage to this value by experimentally adding a resistor, R2, between $-C$ and $-T$, pins 5 and 6. The target output value in this case should be about 283mV. The resistance value of R2 should be approximately 240kΩ.

Finally, the gain must be recalibrated such that the output V_O indicates the device's temperature once again. Do this by adding a third resistor, R3, between FB and $-T$, pins 8 and 5. V_O should now be back to the initial 240mV reading. The resistance value of R3 should be approximately 280kΩ. The final connection diagram is shown in Figure 7. An approximate verification of the effectiveness of recalibration is to measure the differential gain to the output. For type E it should be 164.2.

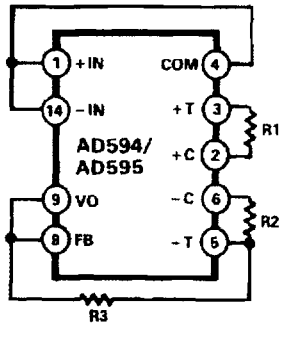


Figure 7. Type E Recalibration

When implementing a similar recalibration procedure for the AD595 the values for R1, R2, R3 and r will be approximately 650Ω, 84kΩ, 93kΩ and 1.51, respectively. Power consumption will increase by about 50% when using the AD595 with type E inputs.

Note that during this procedure it is crucial to maintain the AD594/AD595 at a stable temperature because it is used as the temperature reference. Contact with fingers or any tools not at ambient temperature will quickly produce errors. Radiational heating from a change in lighting or approach of a soldering iron must also be guarded against.

USING TYPE T THERMOCOUPLES WITH THE AD595

Because of the similarity of thermal EMFs in the 0 to 50°C range between type K and type T thermocouples, the AD595 can be directly used with both types of inputs. Within this ambient temperature range the AD595 should exhibit no more than an additional 0.2°C output calibration error when used with type T inputs. The error arises because the ice point compensator is trimmed to type K characteristics at 25°C. To calculate the AD595 output values over the recommended -200 to 350°C range for type T thermocouples, simply use the ANSI thermocouple voltages referred to 0°C and the output equation given on page 3 for the AD595. Because of the relatively large nonlinearities associated with type T thermocouples the output will deviate widely from the nominal 10mV/°C. However, cold junction compensation over the rated 0 to 50°C ambient will remain accurate.

STABILITY OVER TEMPERATURE

Each AD594/AD595 is tested for error over temperature with the measuring thermocouple at 0°C. The combined effects of cold junction compensation error, amplifier offset drift and gain error determine the stability of the AD594/AD595 output over the rated ambient temperature range. Figure 8 shows an AD594/AD595 drift error envelope. The slope of this figure has units of °C/°C.

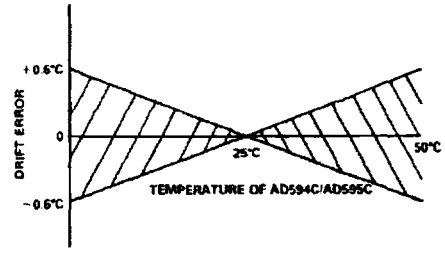


Figure 8. Drift Error vs. Temperature

THERMAL ENVIRONMENT EFFECTS

The inherent low power dissipation of the AD594/AD595 and the low thermal resistance of the package make self-heating errors almost negligible. For example, in still air the chip to ambient thermal resistance is about 80°C/watt (for the D package). At the nominal dissipation of 800μW the self-heating in free air is less than 0.065°C. Submerged in fluorinert liquid (unstirred) the thermal resistance is about 40°C/watt, resulting in a self-heating error of about 0.032°C.

SET-POINT CONTROLLER

The AD594/AD595 can readily be connected as a set-point controller as shown in Figure 9.

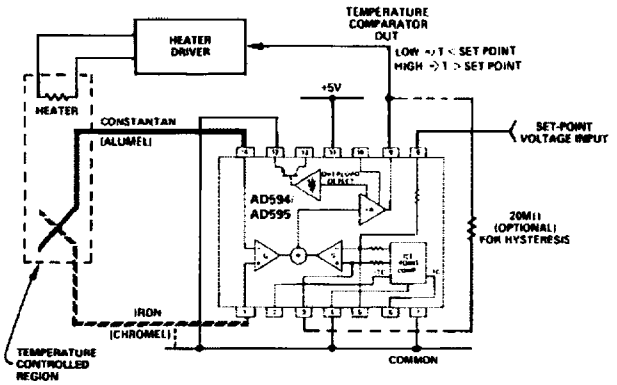


Figure 9. Set-Point Controller

The thermocouple is used to sense the unknown temperature and provide a thermal EMF to the input of the AD594/AD595. The signal is cold junction compensated, amplified to 10mV/°C and compared to an external set-point voltage applied by the user to the feedback at pin 8. Table I lists the correspondence between set-point voltage and temperature, accounting for the nonlinearity of the measurement thermocouple. If the set-point temperature range is within the operating range (−55°C to +125°C) of the AD594/AD595, the chip can be used as the transducer for the circuit by shorting the inputs together and utilizing the nominal calibration of 10mV/°C. This is the centigrade thermometer configuration as shown in Figure 13.

In operation if the set-point voltage is above the voltage corresponding to the temperature being measured the output swings low to approximately zero volts. Conversely, when the temperature rises above the set-point voltage the output switches to the positive limit of about 4 volts with a +5V supply. Figure 9 shows the set-point comparator configuration complete with a heater element driver circuit being controlled by the AD594/AD595 toggled output. Hysteresis can be introduced by injecting a current into the positive input of the feedback amplifier when the output is toggled high. With an AD594 about 200nA into the +T terminal provides 1°C of hysteresis. When using a single 5V supply with an AD594, a 20MΩ resistor from V_O to +T will supply the 200nA of current when the output is forced high (about 4V). To widen the hysteresis band decrease the resistance connected from V_O to +T.

ALARM CIRCUIT

In all applications of the AD594/AD595 the −ALM connection, pin 13, should be constrained so that it is not more positive than (V+) − 4V. This can be most easily achieved by connecting pin 13 to either common at pin 4 or V− at pin 7. For most applications that use the alarm signal, pin 13 will be grounded and the signal will be taken from +ALM on pin 12. A typical application is shown in Figure 10.

In this configuration the alarm transistor will be off in normal operation and the 20k pull up will cause the +ALM output on pin 12 to go high. If one or both of the thermocouple leads are interrupted, the +ALM pin will be driven low. As shown in Figure 10 this signal is compatible with the input of a TTL gate which can be used as a buffer and/or inverter.

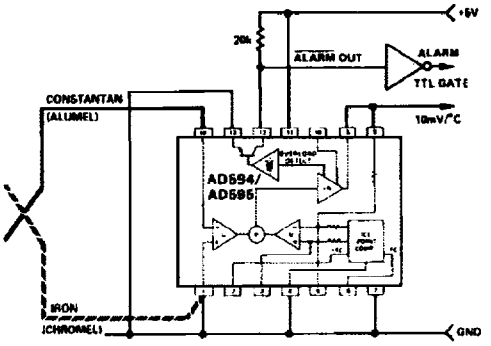


Figure 10. Using the Alarm to Drive a TTL Gate ("Grounded" Emitter Configuration)

Since the alarm is a high level output it may be used to directly drive an LED or other indicator as shown in Figure 11.

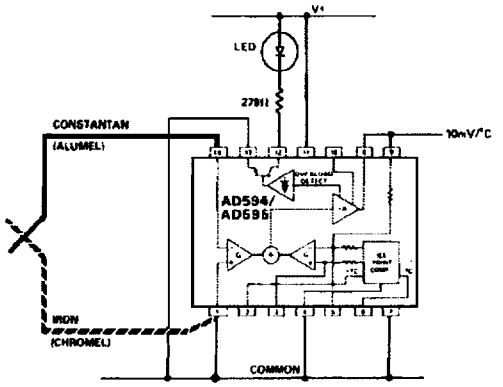


Figure 11. Alarm Directly Drives LED

A 270Ω series resistor will limit current in the LED to 10mA, but may be omitted since the alarm output transistor is current limited at about 20mA. The transistor, however, will operate in a high dissipation mode and the temperature of the circuit will rise well above ambient. Note that the cold junction compensation will be affected whenever the alarm circuit is activated. The time required for the chip to return to ambient temperature will depend on the power dissipation of the alarm circuit, the nature of the thermal path to the environment and the alarm duration.

The alarm can be used with both single and dual supplies. It can be operated above or below ground. The collector and emitter of the output transistor can be used in any normal switch configuration. As an example a negative referenced load can be driven from −ALM as shown in Figure 12.

The collector (+ALM) should not be allowed to become more positive than (V−) +36V, however, it may be permitted to be more positive than V+. The emitter voltage (−ALM) should be constrained so that it does not become more positive than 4 volts below the V+ applied to the circuit.

Additionally, the AD594/AD595 can be configured to produce an extreme upscale or downscale output in applications where an extra signal line for an alarm is inappropriate. By tying either of the thermocouple inputs to common most runaway control conditions can be automatically avoided. A +IN to common connection creates a downscale output if the thermocouple opens, while connecting −IN to common provides an upscale output.

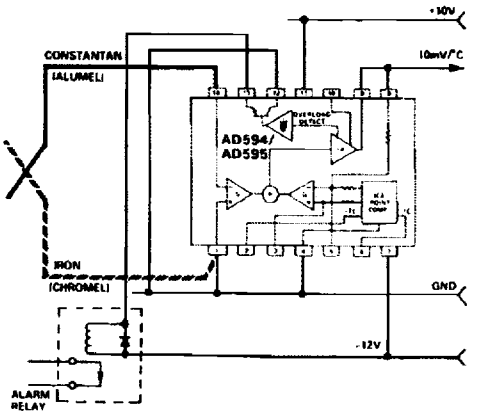


Figure 12. −ALM Driving A Negative Referenced Load

CELSIUS THERMOMETER

The AD594/AD595 may be configured as a stand-alone celsius thermometer as shown in Figure 13.

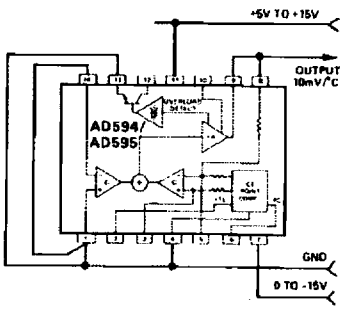


Figure 13. AD594/AD595 as a Stand-Alone Celsius Thermometer

Simply omit the thermocouple and connect the inputs (pins 1 and 14) to common. The output now will reflect the compensation voltage and hence will indicate the AD594/AD595 temperature with a scale factor of 10mV/°C. In this three terminal, voltage output, temperature sensing mode, the AD594/AD595 will operate over the full military −55°C to +125°C temperature range.

THERMOCOUPLE BASICS

Thermocouples are economical and rugged; they have reasonably good long-term stability. Because of their small size, they respond quickly and are good choices where fast response is important. They function over temperature ranges from cryogenics to jet-engine exhaust and have reasonable linearity and accuracy.

Because the number of free electrons in a piece of metal depends on both temperature and composition of the metal, two pieces of dissimilar metal in isothermal contact will exhibit a potential difference that is a repeatable function of temperature, as shown in Figure 14. The resulting voltage depends on the temperatures, T1 and T2, in a repeatable way.

Since the thermocouple is basically a differential rather than absolute measuring device, a known reference temperature is required for one of the junctions if the temperature of the other is to be inferred from the output voltage. Thermocouples made of specially selected materials have been exhaustively characterized in terms of voltage versus temperature compared to primary temperature standards. Most notably the water-ice point of 0°C is used for tables of standard thermocouple performance

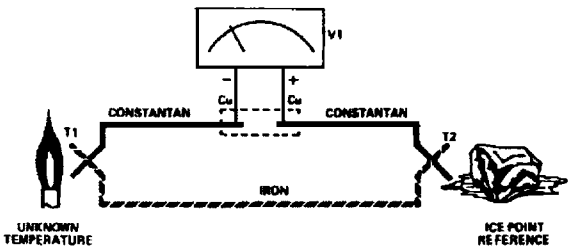


Figure 14. Thermocouple Voltage with 0°C Reference

An alternative measurement technique, illustrated in Figure 15, is used in most practical applications where accuracy requirements do not warrant maintenance of primary standards. The reference junction temperature is allowed to change with the environment

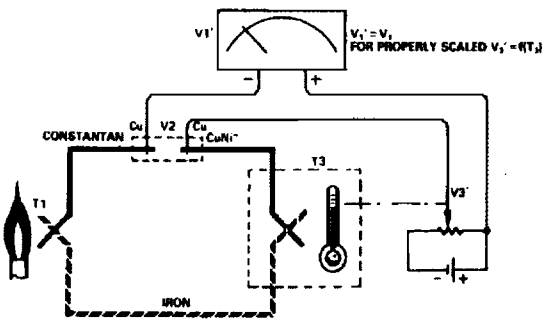


Figure 15. Substitution of Measured Reference Temperature for Ice Point Reference

of the measurement system, but it is carefully measured by some type of absolute thermometer. A measurement of the thermocouple voltage combined with a knowledge of the reference temperature can be used to calculate the measurement junction temperature. Usual practice, however, is to use a convenient thermoelectric method to measure the reference temperature and to arrange its output voltage so that it corresponds to a thermocouple referred to 0°C. This voltage is simply added to the thermocouple voltage and the sum then corresponds to the standard voltage tabulated for an ice-point referenced thermocouple.

The temperature sensitivity of silicon integrated circuit transistors is quite predictable and repeatable. This sensitivity is exploited in the AD594/AD595 to produce a temperature related voltage to compensate the reference or "cold" junction of a thermocouple as shown in Figure 16.

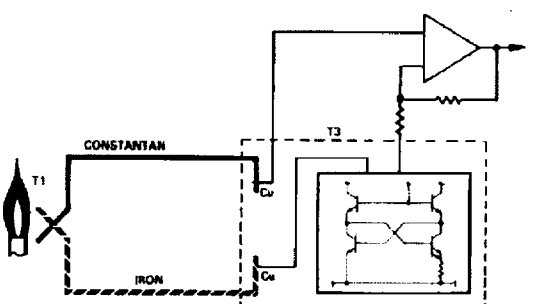


Figure 16. Connecting Isothermal Junctions

Since the compensation is at the reference junction temperature, it is often convenient to form the reference "junction" by connecting directly to the circuit wiring. So long as these connections and the compensation are at the same temperature no error will result

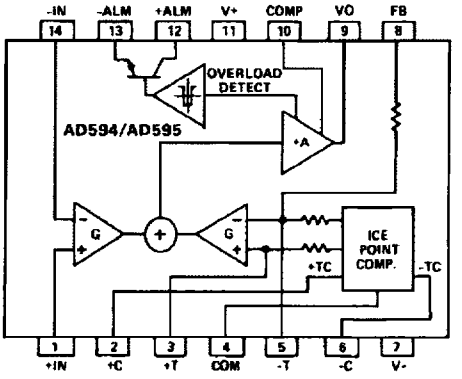
Monolithic Thermocouple Amplifiers with Cold Junction Compensation

AD594*/AD595*

FEATURES

Pretrimmed for Type J (AD594) or Type K (AD595) Thermocouples
Can Be Used with Type T Thermocouple Inputs
Low Impedance Voltage Output: 10mV/°C
Built-in Ice Point Compensation
Wide Power Supply Range: +5V to ±15V
Low Power: <1mW typical
Thermocouple Failure Alarm
Laser Wafer Trimmed to 1°C Calibration Accuracy
Set-Point Mode Operation
Self-Contained Celsius Thermometer Operation
High Impedance Differential Input
Side-Brazed DIP or Low Cost Cerdip

FUNCTIONAL BLOCK DIAGRAM



PRODUCT DESCRIPTION

The AD594/AD595 is a complete instrumentation amplifier and thermocouple cold junction compensator on a monolithic chip. It combines an ice point reference with a precalibrated amplifier to produce a high level (10mV/°C) output directly from a thermocouple signal. Pin-strapping options allow it to be used as a linear amplifier-compensator or as a switched output set-point controller using either fixed or remote set-point control. It can be used to amplify its compensation voltage directly, thereby converting it to a stand-alone Celsius transducer with a low-impedance voltage output.

The AD594/AD595 includes a thermocouple failure alarm that indicates if one or both thermocouple leads become open. The alarm output has a flexible format which includes TTL drive capability.

The AD594/AD595 can be powered from a single ended supply (including +5V) and by including a negative supply, temperatures below 0°C can be measured. To minimize self-heating, an unloaded AD594/AD595 will typically operate with a total supply current of 160μA, but is also capable of delivering in excess of ±5mA to a load.

The AD594 is precalibrated by laser wafer trimming to match the characteristic of type J (iron-constantan) thermocouples and the AD595 is laser trimmed for type K (chromel-alumel) inputs. The temperature transducer voltages and gain control resistors are available at the package pins so that the circuit can be recalibrated for other thermocouple types by the addition of two or three resistors. These terminals also allow more precise calibration for both thermocouple and thermometer applications.

*Protected by U.S. Patent No. 4,029,974.

The AD594/AD595 is available in two performance grades. The C and the A versions have calibration accuracies of ±1°C and ±3°C, respectively. Both are designed to be used from 0 to +50°C, and are available in 14-pin, hermetically sealed, side-brazed ceramic DIPs as well as low cost cerdip packages.

PRODUCT HIGHLIGHTS

1. The AD594/AD595 provides cold junction compensation, amplification, and an output buffer in a single IC package.
2. Compensation, zero, and scale factor are all precalibrated by laser wafer trimming (LWT) of each IC chip.
3. Flexible pin-out provides for operation as a set-point controller or a stand-alone temperature transducer calibrated in degrees Celsius.
4. Operation at remote application sites is facilitated by low quiescent current and a wide supply voltage range of +5V to dual supplies spanning 30V.
5. Differential input rejects common-mode noise voltage on the thermocouple leads.

AD594/AD595—SPECIFICATIONS (@ +25°C and V_S = 5 V, Type J (AD594), Type K (AD595) Thermocouple, unless otherwise noted)

Model	AD594A			AD594C			AD595A			AD595C			Units
	Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
ABSOLUTE MAXIMUM RATINGS													
$+V_S$ to $-V_S$			36			36			36			36	Volts
Common-Mode Input Voltage	$-V_S$ −0.15		$+V_S$	$-V_S$ −0.15		$+V_S$	$-V_S$ −0.15		$+V_S$	$-V_S$ −0.15		$+V_S$	Volts
Differential Input Voltage	$-V_S$		$+V_S$	$-V_S$		$+V_S$	$-V_S$		$+V_S$	$-V_S$		$+V_S$	Volts
Alarm Voltages													
+ALM	$-V_S$		$-V_S$ +36	$-V_S$		$-V_S$ +36	$-V_S$		$-V_S$ +36	$-V_S$		$-V_S$ +36	Volts
−ALM	$-V_S$		$+V_S$	$-V_S$		$+V_S$	$-V_S$		$+V_S$	$-V_S$		$+V_S$	Volts
Operating Temperature Range	−55		+125	−55		+125	−55		+125	−55		+125	°C
Output Short Circuit to Common	Indefinite			Indefinite			Indefinite			Indefinite			
TEMPERATURE MEASUREMENT													
(Specified Temperature Range 0 to +50°C)													
Calibration Error at +25°C[1]			**±3**			**±1**			**±3**			**±1**	°C
Stability vs. Temperature[2]			**±0.05**			**±0.025**			**±0.05**			**±0.025**	°C/°C
Gain Error			**±1.5**			**±0.75**			**±1.5**			**±0.75**	%
Nominal Transfer Function			10			10			10			10	mV/°C
AMPLIFIER CHARACTERISTICS													
Closed Loop Gain[3]		**193.4**				**193.4**			**247.3**			**247.3**	
Input Offset Voltage	(Temperature in °C)×51.70 µV/°C			(Temperature in °C)×51.70 µV/°C			(Temperature in °C)×40.44 µV/°C			(Temperature in °C)×40.44 µV/°C			µV
Input Bias Current		0.1			0.1			0.1			0.1		µA
Differential Input Range	−10		+50				−10		+50	−10		+50	mV
Common-Mode Range	$-V_S$ −0.15		$+V_S$ −4	$-V_S$ −0.15		$+V_S$ −4	$-V_S$ −0.15		$+V_S$ −4	$-V_S$ −0.15		$+V_S$ −4	Volts
Common-Mode Sensitivity – RTO			**10**			10			**10**			10	mV/V
Power Supply Sensitivity – RTO			**10**			10			**10**			10	mV/V
Output Voltage Range													
Dual Supplies	$-V_S$ +2.5		$+V_S$ −2	$-V_S$ +2.5		$+V_S$ −2	$-V_S$ +2.5		$+V_S$ −2	$-V_S$ +2.5		$+V_S$ −2	Volts
Single Supply	0		$+V_S$ −2	0		$+V_S$ −2	0		$+V_S$ +2	0		$+V_S$ −2	Volts
Usable Output Current[4]		±5			±5			±5			±5		mA
3 dB Bandwidth		15			15			15			15		kHz
ALARM CHARACTERISTICS													
$V_{CE(SAT)}$ at 2 mA		0.3			0.3			0.3			0.3		Volts
Leakage Current			**±1**			**±1**			**±1**			**±1**	µA max
Operating Voltage at – ALM			$+V_S$ −4			$+V_S$ −4			$+V_S$ −4			$+V_S$ −4	Volts
Short Circuit Current		20			20			20			20		mA
POWER REQUIREMENTS													
Specified Performance	$+V_S$ = 5, $-V_S$ = 0			$+V_S$ = 5, $-V_S$ = 0			$+V_S$ = 5, $-V_S$ = 0			$+V_S$ = 5, $-V_S$ = 0			Volts
Operating[5]	$+V_S$ to $-V_S$ ≤30			$+V_S$ to $-V_S$ ≤30			$+V_S$ to $-V_S$ ≤30			$+V_S$ to $-V_S$ ≤30			Volts
Quiescent Current (No Load)													
$+V_S$		160	300		160	300		160	300		160	300	µA
$-V_S$		100			100			100			100		µA
PACKAGE OPTION[6]													
TO-116 (D-14)	AD594AD			AD594CD			AD595AD			AD595CD			
Cerdip (Q-14A)	AD594AQ			AD594CQ			AD595AQ			AD595CQ			

NOTES

[1]Calibrated for minimum error at +25°C using a thermocouple sensitivity of 51.7 µV/°C. Since a J type thermocouple deviates from this straight line approximation, the AD594 will normally read 3.1 mV when the measuring junction is at 0°C. The AD595 will similarly read 2.7 mV at 0°C.

[2]Defined as the slope of the line connecting the AD594/AD595 errors measured at 0°C and 50°C ambient temperature.

[3]Pin 8 shorted to Pin 9.

[4]Current Sink Capability in single supply configuration is limited to current drawn to ground through a 50 kΩ resistor at output voltages below 2.5 V.

[5]$-V_S$ must not exceed −16.5 V.

[6]For outline information see Package Information section.

Specifications subject to change without notice.

Specifications shown in **boldface** are tested on all production units at final electrical test. Results from those tests are used to calculate outgoing quality levels.
All min and max specifications are guaranteed, although only those shown in boldface are tested on all production units.

Thermocouple Temperature °C	Type J Voltage mV	AD594 Output mV	Type K Voltage mV	AD595 Output mV	Thermocouple Temperature °C	Type J Voltage mV	AD594 Output mV	Type K Voltage mV	AD595 Output mV
-200	-7.890	-1523	-5.891	-1454	500	27.388	5300	20.640	5107
-180	-7.402	-1428	-5.550	-1370	520	28.511	5517	21.493	5318
-160	-6.821	-1316	-5.141	-1269	540	29.642	5736	22.346	5529
-140	-6.159	-1188	-4.669	-1152	560	30.782	5956	23.198	5740
-120	-5.426	-1046	-4.138	-1021	580	31.933	6179	24.050	5950
-100	-4.632	-893	-3.553	-876	600	33.096	6404	24.902	6161
-80	-3.785	-729	-2.920	-719	620	34.273	6632	25.751	6371
-60	-2.892	-556	-2.243	-552	640	35.464	6862	26.599	6581
-40	-1.960	-376	-1.527	-375	660	36.671	7095	27.445	6790
-20	-.995	-189	-.777	-189	680	37.893	7332	28.288	6998
-10	-.501	-94	-.392	-94	700	39.130	7571	28.128	7206
0	0	3.1	0	2.7	720	40.382	7813	29.965	7413
10	.507	101	.397	101	740	41.647	8058	30.799	7619
20	1.019	200	.798	200	750	42.283	8181	31.214	7722
25	1.277	250	1.000	250	760	-	-	31.629	7825
30	1.536	300	1.203	300	780	-	-	32.455	8029
40	2.058	401	1.611	401	800	-	-	33.277	8232
50	2.585	503	2.022	503	820	-	-	34.095	8434
60	3.115	606	2.436	605	840	-	-	34.909	8636
80	4.186	813	3.266	810	860	-	-	35.718	8836
100	5.268	1022	4.095	1015	880	-	-	36.524	9035
120	6.359	1233	4.919	1219	900	-	-	37.325	9233
140	7.457	1445	5.733	1420	920	-	-	38.122	9430
160	8.560	1659	6.539	1620	940	-	-	38.915	9626
180	9.667	1873	7.338	1817	960	-	-	39.703	9821
200	10.777	2087	8.137	2015	980	-	-	40.488	10015
220	11.887	2302	8.938	2213	1000	-	-	41.269	10209
240	12.998	2517	9.745	2413	1020	-	-	42.045	10400
260	14.108	2732	10.560	2614	1040	-	-	42.817	10591
280	15.217	2946	11.381	2817	1060	-	-	43.585	10781
300	16.325	3160	12.207	3022	1080	-	-	44.349	10970
320	17.432	3374	13.039	3327	1100	-	-	45.108	11158
340	18.537	3588	13.874	3434	1120	-	-	45.863	11345
360	19.640	3801	14.712	3641	1140	-	-	46.612	11530
380	20.743	4015	15.552	3849	1160	-	-	47.356	11714
400	21.846	4228	16.395	4057	1180	-	-	48.095	11897
420	22.949	4441	17.241	4266	1200	-	-	48.828	12078
440	24.054	4655	18.088	4476	1220	-	-	49.555	12258
460	25.161	4869	18.938	4686	1240	-	-	50.276	12436
480	26.272	5084	19.788	4896	1250	-	-	50.633	12524

Table I. Output Voltage vs. Thermocouple Temperature (Ambient +25°C, V_S = −5V, +15V)

INTERPRETING AD594/AD595 OUTPUT VOLTAGES

To achieve a temperature proportional output of 10mV/°C and accurately compensate for the reference junction over the rated operating range of the circuit, the AD594/AD595 is gain trimmed to match the transfer characteristic of J and K type thermocouples at 25°C. For a type J output in this temperature range the TC is 51.70μV/°C, while for a type K it is 40.44μV/°C. The resulting gain for the AD594 is 193.4 (10mV/°C divided by 51.7μV/°C) and for the AD595 is 247.3 (10mV/°C divided by 40.44μV/°C). In addition, an absolute accuracy trim induces an input offset to the output amplifier characteristic of 16μV for the AD594 and 11μV for the AD595. This offset arises because the AD594/AD595 is trimmed for a 250mV output while applying a 25°C thermocouple input.

Because a thermocouple output voltage is nonlinear with respect to temperature, and the AD594/AD595 linearly amplifies the compensated signal, the following transfer functions should be used to determine the actual output voltages:

AD594 output = (Type J Voltage + 16μV) × 193.4

AD595 output = (Type K Voltage + 11μV) × 247.3

or conversely:

Type J voltage = (AD594 output / 193.4) − 16μV

Type K voltage = (AD595 output / 247.3) − 11μV

Table I above lists the ideal AD594/AD595 output voltages as a function of Celsius temperature for type J and K ANSI standard thermocouples, with the package and reference junction at 25°C. As is normally the case, these outputs are subject to calibration, gain and temperature sensitivity errors. Output values for intermediate temperatures can be interpolated, or calculated using the output equations and ANSI thermocouple voltage tables referred to zero degrees Celsius. Due to a slight variation in alloy content between ANSI type J and DIN Fe–CuNi thermocouples Table I should not be used in conjunction with European standard thermocouples. Instead the transfer function given previously and a DIN thermocouple table should be used. ANSI type K and DIN NiCr–Ni thermocouples are composed of identical alloys and exhibit similar behavior. The upper temperature limits in Table I are those recommended for type J and type K thermocouples by the majority of vendors.

SINGLE AND DUAL SUPPLY CONNECTIONS

The AD594/AD595 is a completely self-contained thermocouple conditioner. Using a single +5V supply the interconnections shown in Figure 1 will provide a direct output from a type J thermocouple (AD594) or type K thermocouple (AD595) measuring from 0 to +300°C.

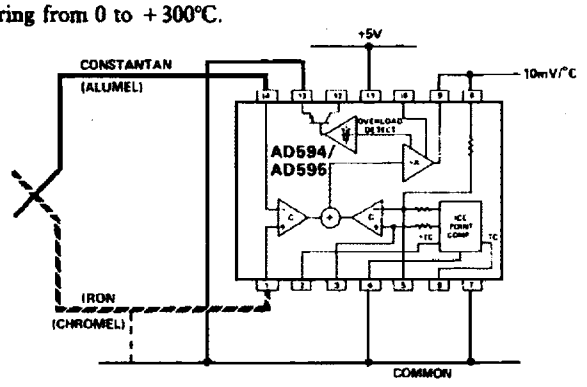


Figure 1. Basic Connection, Single Supply Operation

Any convenient supply voltage from +5V to +30V may be used, with self-heating errors being minimized at lower supply levels. In the single supply configuration the +5V supply connects to pin 11 with the V− connection at pin 7 strapped to power and signal common at pin 4. The thermocouple wire inputs connect to pins 1 and 14 either directly from the measuring point or through intervening connections of similar thermocouple wire type. When the alarm output at pin 13 is not used it should be connected to common or −V. The precalibrated feedback network at pin 8 is tied to the output at pin 9 to provide a 10mV/°C nominal temperature transfer characteristic.

By using a wider ranging dual supply, as shown in Figure 2, the AD594/AD595 can be interfaced to thermocouples measuring both negative and extended positive temperatures.

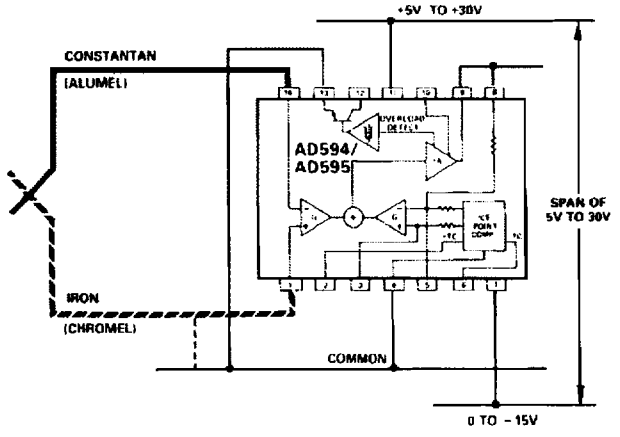


Figure 2. Dual Supply Operation

With a negative supply the output can indicate negative temperatures and drive grounded loads or loads returned to positive voltages. Increasing the positive supply from 5V to 15V extends the output voltage range well beyond the 750°C temperature limit recommended for type J thermocouples (AD594) and the 1250°C for type K thermocouples (AD595).

Common-mode voltages on the thermocouple inputs must remain within the common-mode range of the AD594/AD595, with a return path provided for the bias currents. If the thermocouple is not remotely grounded, then the dotted line connections in Figures 1 and 2 are recommended. A resistor may be needed in this connection to assure that common mode voltages induced in the thermocouple loop are not converted to normal mode.

THERMOCOUPLE CONNECTIONS

The isothermal terminating connections of a pair of themocouple wires forms an effective reference junction. This junction must be kept at the same temperature as the AD594/AD595 for the internal cold junction compensation to be effective.

A method that provides for thermal equilibrium is the printed circuit board connection layout illustrated in Figure 3.

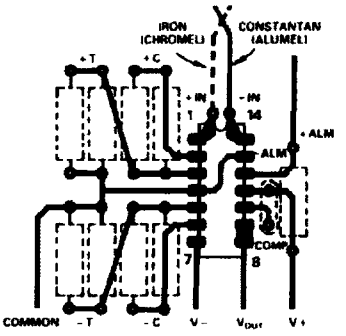


Figure 3. PCB Connections

Here the AD594/AD595 package temperature and circuit board are thermally contacted in the copper printed circuit board tracks under pins 1 and 14. The reference junction is now composed of a copper-constantan (or copper-alumel) connection and copper-iron (or copper-chromel) connection, both of which are at the same temperature as the AD594/AD595.

The printed circuit board layout shown also provides for placement of optional alarm load resistors, recalibration resistors and a compensation capacitor to limit bandwidth.

To ensure secure bonding the thermocouple wire should be cleaned to remove oxidation prior to soldering. Noncorrosive rosin flux is effective with iron, constantan, chromel and alumel and the following solders: 95% tin-5% antimony, 95% tin-5% silver or 90% tin-10% lead.

FUNCTIONAL DESCRIPTION

The AD594 behaves like two differential amplifiers. The outputs are summed and used to control a high-gain amplifier, as shown in Figure 4.

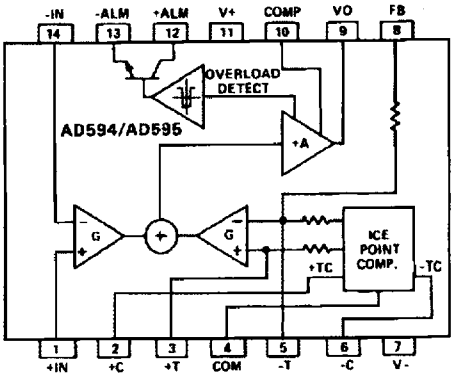


Figure 4. AD594/AD595 Block Diagram

In normal operation the main amplifier output, at pin 9, is connected to the feedback network, at pin 8. Thermocouple signals applied to the floating input stage, at pins 1 and 14, are

amplified by gain G of the differential amplifier and are then further amplified by gain A in the main amplifier. The output of the main amplifier is fed back to a second differential stage in an inverting connection. The feedback signal is amplified by this stage and is also applied to the main amplifier input through a summing circuit. Because of the inversion, the amplifier causes the feedback to be driven to reduce this difference signal to a small value. The two differential amplifiers are made to match and have identical gains, G. As a result, the feedback signal that must be applied to the right-hand differential amplifier will precisely match the thermocouple input signal when the difference signal has been reduced to zero. The feedback network is trimmed so that the effective gain to the output, at pins 8 and 9, results in a voltage of 10mV/°C of thermocouple excitation.

In addition to the feedback signal, a cold junction compensation voltage is applied to the right-hand differential amplifier. The compensation is a differential voltage proportional to the Celsius temperature of the AD594/AD595. This signal disturbs the differential input so that the amplifier output must adjust to restore the input to equal the applied thermocouple voltage.

The compensation is applied through the gain scaling resistors so that its effect on the main output is also 10mV/°C. As a result, the compensation voltage adds to the effect of the thermocouple voltage a signal directly proportional to the difference between 0°C and the AD594/AD595 temperature. If the thermocouple reference junction is maintained at the AD594/AD595 temperature, the output of the AD594/AD595 will correspond to the reading that would have been obtained from amplification of a signal from a thermocouple referenced to an ice bath.

The AD594/AD595 also includes an input open circuit detector that switches on an alarm transistor. This transistor is actually a current-limited output buffer, but can be used up to the limit as a switch transistor for either pull-up or pull-down operation of external alarms.

The ice point compensation network has voltages available with positive and negative temperature coefficients. These voltages may be used with external resistors to modify the ice point compensation and recalibrate the AD594/AD595 as described in the next column.

The feedback resistor is separately pinned out so that its value can be padded with a series resistor, or replaced with an external resistor between pins 5 and 9. External availability of the feedback resistor allows gain to be adjusted, and also permits the AD594/AD595 to operate in a switching mode for set-point operation.

CAUTIONS:

The temperature compensation terminals (+C and −C) at pins 2 and 6 are provided to supply small calibration currents only. The AD594/AD595 may be permanently damaged if they are grounded or connected to a low impedance.

The AD594/AD595 is internally frequency compensated for feedback ratios (corresponding to normal signal gain) of 75 or more. If a lower gain is desired, additional frequency compensation should be added in the form of a 300pF capacitor from pin 10 to the output at pin 9. As shown in Figure 5 an additional 0.01μF capacitor between pins 10 and 11 is recommended.

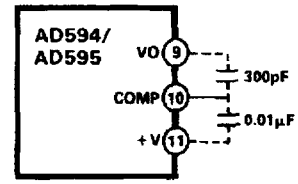


Figure 5. Low Gain Frequency Compensation

RECALIBRATION PRINCIPLES AND LIMITATIONS

The ice point compensation network of the AD594/AD595 produces a differential signal which is zero at 0°C and corresponds to the output of an ice referenced thermocouple at the temperature of the chip. The positive TC output of the circuit is proportional to Kelvin temperature and appears as a voltage at +T. It is possible to decrease this signal by loading it with a resistor from +T to COM, or increase it with a pull-up resistor from +T to the larger positive TC voltage at +C. Note that adjustments to +T should be made by measuring the voltage which tracks it at −T. To avoid destabilizing the feedback amplifier the measuring instrument should be isolated by a few thousand ohms in series with the lead connected to −T.

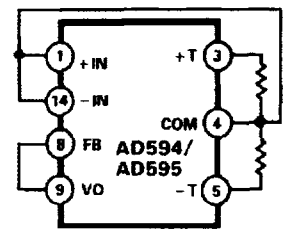


Figure 6. Decreased Sensitivity Adjustment

Changing the positive TC half of the differential output of the compensation scheme shifts the zero point away from 0°C. The zero can be restored by adjusting the current flow into the negative input of the feedback amplifier, the −T pin. A current into this terminal can be produced with a resistor between −C and −T to balance an increase in +T, or a resistor from −T to COM to offset a decrease in +T.

If the compensation is adjusted substantially to accommodate a different thermocouple type, its effect on the final output voltage will increase or decrease in proportion. To restore the nominal output to 10mV/°C the gain may be adjusted to match the new compensation and thermocouple input characteristics. When reducing the compensation the resistance between −T and COM automatically increases the gain to within 0.5% of the correct value. If a smaller gain is required, however, the nominal 47kΩ internal feedback resistor can be paralleled or replaced with an external resistor.

Fine calibration adjustments will require temperature response measurements of individual devices to assure accuracy. Major reconfigurations for other thermocouple types can be achieved without seriously compromising initial calibration accuracy, so long as the procedure is done at a fixed temperature using the factory calibration as a reference. It should be noted that intermediate recalibration conditions may require the use of a negative supply. An example using a type E thermocouple and an AD594 is given on the next page.

EXAMPLE: TYPE E RECALIBRATION – AD594/AD595
Both the AD594 and AD595 can be configured to condition the output of a type E (chromel-constantan) thermocouple. Temperature characteristics of type E thermocouples differ less from type J, than from type K, therefore the AD594 is preferred for recalibration.

While maintaining the device at a constant temperature follow the recalibration steps given here. First, measure the device temperature by tying both inputs to common (or a selected common mode potential) and connecting FB to V_O. The AD594 is now in the stand alone Celsius thermometer mode. For this example assume the ambient is 24°C and the initial output V_O is 240mV. Check the output at V_O to verify that it corresponds to the temperature of the device.

Next, measure the voltage $-T$ at pin 5 with a high impedance DVM (capacitance should be isolated by a few thousand ohms of resistance at the measured terminals). At 24°C the $-T$ voltage will be about 8.3mV. To adjust the compensation of an AD594 to a type E thermocouple a resistor, R1, should be connected between $+T$ and $+C$, pins 2 and 3, to raise the voltage at $-T$ by the ratio of thermocouple sensitivities. The ratio for converting a type J device to a type E characteristic is:

$$r\,(AD594) = (60.9\mu V/°C)/(51.7\mu V/°C) = 1.18$$

Thus, multiply the initial voltage measured at $-T$ by r and experimentally determine the R1 value required to raise $-T$ to that level. For the example the new $-T$ voltage should be about 9.8mV. The resistance value should be approximately 1.8kΩ.

The zero differential point must now be shifted back to 0°C. This is accomplished by multiplying the original output voltage V_O by r and adjusting the measured output voltage to this value by experimentally adding a resistor, R2, between $-C$ and $-T$, pins 5 and 6. The target output value in this case should be about 283mV. The resistance value of R2 should be approximately 240kΩ.

Finally, the gain must be recalibrated such that the output V_O indicates the device's temperature once again. Do this by adding a third resistor, R3, between FB and $-T$, pins 8 and 5. V_O should now be back to the initial 240mV reading. The resistance value of R3 should be approximately 280kΩ. The final connection diagram is shown in Figure 7. An approximate verification of the effectiveness of recalibration is to measure the differential gain to the output. For type E it should be 164.2.

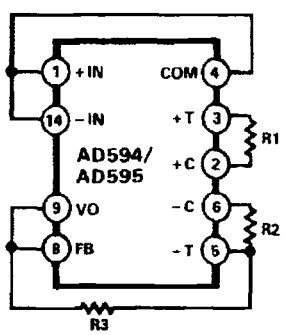


Figure 7. Type E Recalibration

When implementing a similar recalibration procedure for the AD595 the values for R1, R2, R3 and r will be approximately 650Ω, 84kΩ, 93kΩ and 1.51, respectively. Power consumption will increase by about 50% when using the AD595 with type E inputs.

Note that during this procedure it is crucial to maintain the AD594/AD595 at a stable temperature because it is used as the temperature reference. Contact with fingers or any tools not at ambient temperature will quickly produce errors. Radiational heating from a change in lighting or approach of a soldering iron must also be guarded against.

USING TYPE T THERMOCOUPLES WITH THE AD595
Because of the similarity of thermal EMFs in the 0 to 50°C range between type K and type T thermocouples, the AD595 can be directly used with both types of inputs. Within this ambient temperature range the AD595 should exhibit no more than an additional 0.2°C output calibration error when used with type T inputs. The error arises because the ice point compensator is trimmed to type K characteristics at 25°C. To calculate the AD595 output values over the recommended -200 to 350°C range for type T thermocouples, simply use the ANSI thermocouple voltages referred to 0°C and the output equation given on page 3 for the AD595. Because of the relatively large nonlinearities associated with type T thermocouples the output will deviate widely from the nominal 10mV/°C. However, cold junction compensation over the rated 0 to 50°C ambient will remain accurate.

STABILITY OVER TEMPERATURE
Each AD594/AD595 is tested for error over temperature with the measuring thermocouple at 0°C. The combined effects of cold junction compensation error, amplifier offset drift and gain error determine the stability of the AD594/AD595 output over the rated ambient temperature range. Figure 8 shows an AD594/AD595 drift error envelope. The slope of this figure has units of °C/°C.

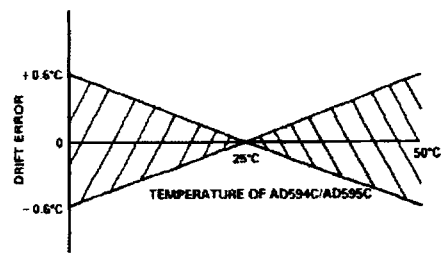


Figure 8. Drift Error vs. Temperature

THERMAL ENVIRONMENT EFFECTS
The inherent low power dissipation of the AD594/AD595 and the low thermal resistance of the package make self-heating errors almost negligible. For example, in still air the chip to ambient thermal resistance is about 80°C/watt (for the D package). At the nominal dissipation of 800μW the self-heating in free air is less than 0.065°C. Submerged in fluorinert liquid (unstirred) the thermal resistance is about 40°C/watt, resulting in a self-heating error of about 0.032°C.

SET-POINT CONTROLLER

The AD594/AD595 can readily be connected as a set-point controller as shown in Figure 9.

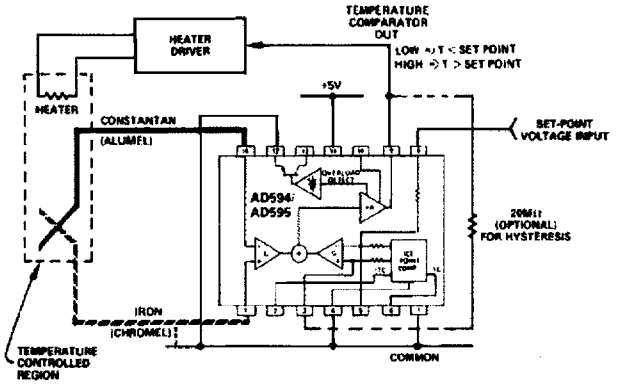


Figure 9. Set-Point Controller

The thermocouple is used to sense the unknown temperature and provide a thermal EMF to the input of the AD594/AD595. The signal is cold junction compensated, amplified to 10mV/°C and compared to an external set-point voltage applied by the user to the feedback at pin 8. Table I lists the correspondence between set-point voltage and temperature, accounting for the nonlinearity of the measurement thermocouple. If the set-point temperature range is within the operating range (−55°C to +125°C) of the AD594/AD595, the chip can be used as the transducer for the circuit by shorting the inputs together and utilizing the nominal calibration of 10mV/°C. This is the centigrade thermometer configuration as shown in Figure 13.

In operation if the set-point voltage is above the voltage corresponding to the temperature being measured the output swings low to approximately zero volts. Conversely, when the temperature rises above the set-point voltage the output switches to the positive limit of about 4 volts with a +5V supply. Figure 9 shows the set-point comparator configuration complete with a heater element driver circuit being controlled by the AD594/AD595 toggled output. Hysteresis can be introduced by injecting a current into the positive input of the feedback amplifier when the output is toggled high. With an AD594 about 200nA into the +T terminal provides 1°C of hysteresis. When using a single 5V supply with an AD594, a 20MΩ resistor from V_O to +T will supply the 200nA of current when the output is forced high (about 4V). To widen the hysteresis band decrease the resistance connected from V_O to +T.

ALARM CIRCUIT

In all applications of the AD594/AD595 the −ALM connection, pin 13, should be constrained so that it is not more positive than (V+) − 4V. This can be most easily achieved by connecting pin 13 to either common at pin 4 or V− at pin 7. For most applications that use the alarm signal, pin 13 will be grounded and the signal will be taken from +ALM on pin 12. A typical application is shown in Figure 10.

In this configuration the alarm transistor will be off in normal operation and the 20k pull up will cause the +ALM output on pin 12 to go high. If one or both of the thermocouple leads are interrupted, the +ALM pin will be driven low. As shown in Figure 10 this signal is compatible with the input of a TTL gate which can be used as a buffer and/or inverter.

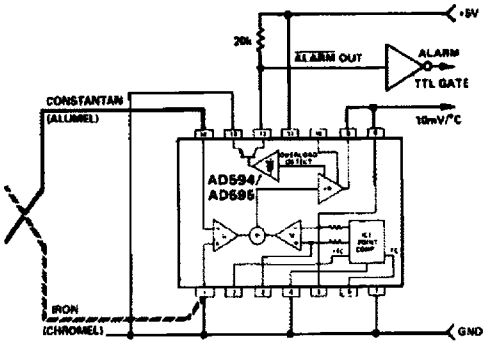


Figure 10. Using the Alarm to Drive a TTL Gate ("Grounded" Emitter Configuration)

Since the alarm is a high level output it may be used to directly drive an LED or other indicator as shown in Figure 11.

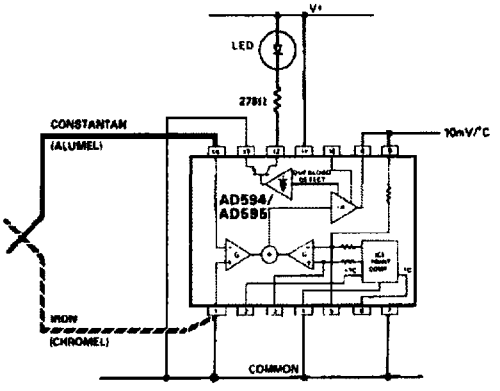


Figure 11. Alarm Directly Drives LED

A 270Ω series resistor will limit current in the LED to 10mA, but may be omitted since the alarm output transistor is current limited at about 20mA. The transistor, however, will operate in a high dissipation mode and the temperature of the circuit will rise well above ambient. Note that the cold junction compensation will be affected whenever the alarm circuit is activated. The time required for the chip to return to ambient temperature will depend on the power dissipation of the alarm circuit, the nature of the thermal path to the environment and the alarm duration.

The alarm can be used with both single and dual supplies. It can be operated above or below ground. The collector and emitter of the output transistor can be used in any normal switch configuration. As an example a negative referenced load can be driven from −ALM as shown in Figure 12.

The collector (+ALM) should not be allowed to become more positive than (V−) +36V, however, it may be permitted to be more positive than V+. The emitter voltage (−ALM) should be constrained so that it does not become more positive than 4 volts below the V+ applied to the circuit.

Additionally, the AD594/AD595 can be configured to produce an extreme upscale or downscale output in applications where an extra signal line for an alarm is inappropriate. By tying either of the thermocouple inputs to common most runaway control conditions can be automatically avoided. A +IN to common connection creates a downscale output if the thermocouple opens, while connecting −IN to common provides an upscale output.

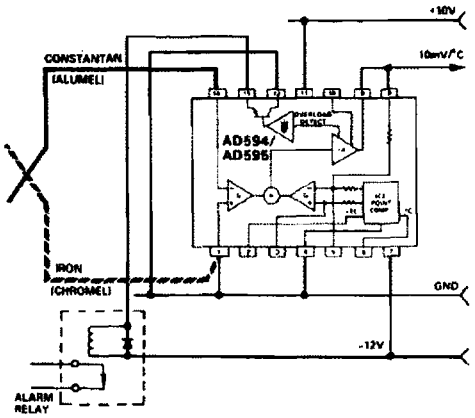


Figure 12. –ALM Driving A Negative Referenced Load

CELSIUS THERMOMETER

The AD594/AD595 may be configured as a stand-alone celsius thermometer as shown in Figure 13.

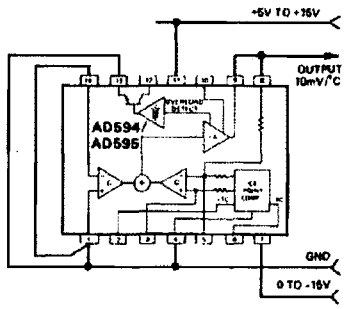


Figure 13. AD594/AD595 as a Stand-Alone Celsius Thermometer

Simply omit the thermocouple and connect the inputs (pins 1 and 14) to common. The output now will reflect the compensation voltage and hence will indicate the AD594/AD595 temperature with a scale factor of 10mV/°C. In this three terminal, voltage output, temperature sensing mode, the AD594/AD595 will operate over the full military −55°C to +125°C temperature range.

THERMOCOUPLE BASICS

Thermocouples are economical and rugged; they have reasonably good long-term stability. Because of their small size, they respond quickly and are good choices where fast response is important. They function over temperature ranges from cryogenics to jet-engine exhaust and have reasonable linearity and accuracy.

Because the number of free electrons in a piece of metal depends on both temperature and composition of the metal, two pieces of dissimilar metal in isothermal contact will exhibit a potential difference that is a repeatable function of temperature, as shown in Figure 14. The resulting voltage depends on the temperatures, T1 and T2, in a repeatable way.

Since the thermocouple is basically a differential rather than absolute measuring device, a known reference temperature is required for one of the junctions if the temperature of the other is to be inferred from the output voltage. Thermocouples made of specially selected materials have been exhaustively characterized in terms of voltage versus temperature compared to primary temperature standards. Most notably the water-ice point of 0°C is used for tables of standard thermocouple performance

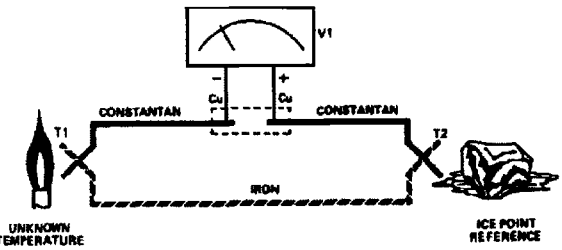


Figure 14. Thermocouple Voltage with 0°C Reference

An alternative measurement technique, illustrated in Figure 15, is used in most practical applications where accuracy requirements do not warrant maintenance of primary standards. The reference junction temperature is allowed to change with the environment

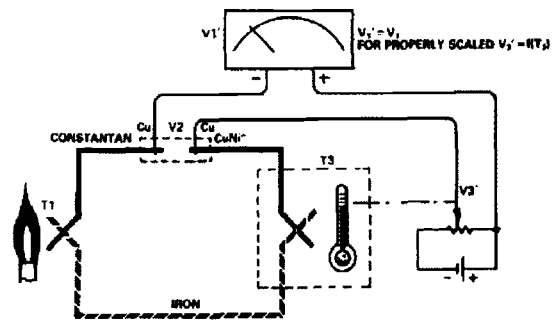


Figure 15. Substitution of Measured Reference Temperature for Ice Point Reference

of the measurement system, but it is carefully measured by some type of absolute thermometer. A measurement of the thermocouple voltage combined with a knowledge of the reference temperature can be used to calculate the measurement junction temperature. Usual practice, however, is to use a convenient thermoelectric method to measure the reference temperature and to arrange its output voltage so that it corresponds to a thermocouple referred to 0°C. This voltage is simply added to the thermocouple voltage and the sum then corresponds to the standard voltage tabulated for an ice-point referenced thermocouple.

The temperature sensitivity of silicon integrated circuit transistors is quite predictable and repeatable. This sensitivity is exploited in the AD594/AD595 to produce a temperature related voltage to compensate the reference or "cold" junction of a thermocouple as shown in Figure 16.

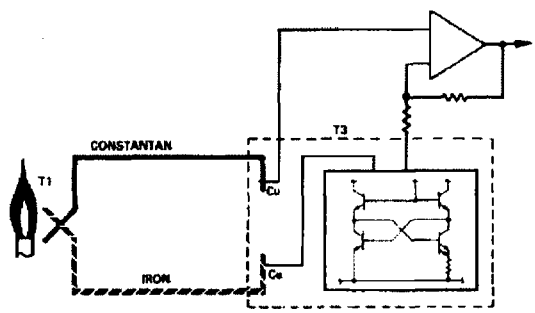


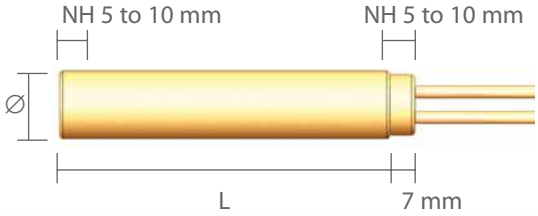
Figure 16. Connecting Isothermal Junctions

Since the compensation is at the reference junction temperature, it is often convenient to form the reference "junction" by connecting directly to the circuit wiring. So long as these connections and the compensation are at the same temperature no error will

STANDARD CARTRIDGE HEATERS

- Whether they are kept in stock or not, standard cartridge heaters are manufactured according to high density technology.
- This technology requires a H7 bore fit (see p.8).
- Stocked cartridge heaters are available within 48 hours, within the limit of our stock.
- Maximum watt density over the surface of the cartridge can reach 50 W/cm^2, under specific conditions.
- Rated Wattage : from 75 to 2500 W.
- Rated Voltage : 230 V single phase.
- Stainless steel sheath.
- Ceramic end cap protecting the connection leads.
- Standard connection leads ; flexible multi strands, nickel core wire with fiberglass silk silicone impregnated insulation.
- Tolerance over the length : L < 100 mm : ± 2 mm.
  L ≥ 100 mm : ± 2 %.
- Manufactured in accordance with EN 60335-1 standards
  Tolerance over the wattage : +5% -10%
  Leakage current < 0.5 mA/kW

- Standard cartridge heater's dimensions :



NC : non heating

Nominal Ø- mm (equiv. inches)	Tolerance over Ø – mm	Min Length - mm (equiv. inches)	Max Length. - mm (equiv. inches)
6.35 - (1/4")	-0.03 / -0.05	31.75 (1"1/4)	152.4 (6")
6.5	-0.03 / -0.05	32	160
8	-0.04 / -0.06	32	160
9.52 - (3/8")	-0.04 / -0.07	31.75 (1"1/4)	203.2 (8")
10	-0.04 / -0.07	32	200
12.5	-0.05 / -0.08	40	300
12.7 - (1/2")	-0.05 / -0.08	38.1 (1"1/2)	304.8 (12")
15.87 - (5/8")	-0.05 / -0.08	50.8 (2")	304.8 (12")
16	-0.05 / -0.08	40	300
19.05 - (3/4")	-0.06 / -0.1	63.5 (2"1/2)	304.8 (12")
20	-0.06 / -0.1	40	300
25.4 - (1")	-0.06 / -0.16	63.5 (2"1/2)	Nous consulter

- Some standard cartridge heaters can be fitted with built in "J" thermocouples (shown in the table as P+tcj), see the layout sketches and the assembling recommendations p.8 & 16. In this case, the standard length of connection leads and thermocouple wires is 1000 mm.
- Our full range of cartridge, from the table below, is UL recognized by Underwriters Laboratories for the USA and Canada. c UL us
- Standard cartridge heaters can be fitted with several types of connection (see p.9) and/or accessories (see p.12)
- Special manufacturing : odd sizes, variable power zone, built in thermocouple .... (see p.16)

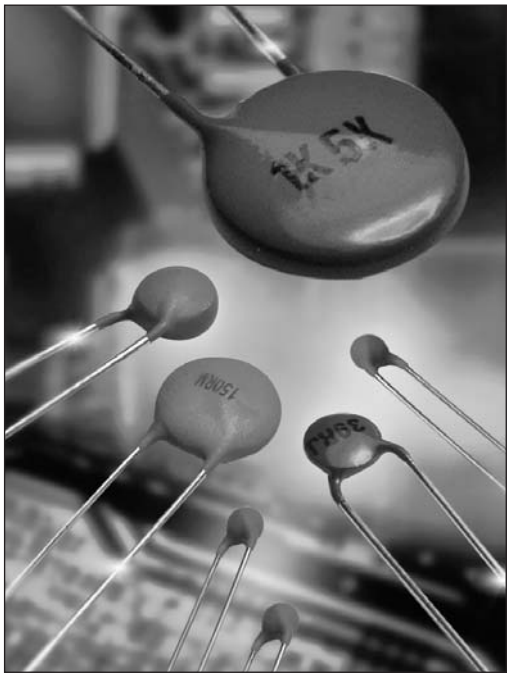
Diam. Ø(mm)	Length L(mm)	Watt. P (W)	Leads (mm)	W.dens. (W/cm^2)	Stocked	Non stocked
6.35 (1/4")	31.7 (1"1/4)	75	250	17	-	H1/4X1.2X75
		100	250	22	H1/4X1.2X100	-
		150	250	34	-	H1/4X1.2X150
		175	250	39	H1/4X1.2X175	-
		175+tcj	1000	39	HJ1/4X1.2X175	-
	38.1 (1"1/2)	75	250	15	H1/4X1.5X75	-
		100	250	20	-	H1/4X1.5X100
		100+tcj	1000	20	HJ1/4X1.5X100	-
		125	250	25	H1/4X1.5X125	-
		150	250	29	H1/4X1.5X150	-
		150+tcj	1000	29	HJ1/4X1.5X150	-
		175	250	34	H1/4X1.5X175	-
		200	250	39	H1/4X1.5X200	-
		200+tcj	1000	39	HJ1/4X1.5X200	-
		250	250	49	H1/4X1.5X250	-
	50.8 (2")	100	250	13	H1/4X2X100	-
		100+tcj	1000	13	HJ1/4X2X100	-
		125	250	16	H1/4X2X125	-
		150	250	20	-	H1/4X2X150
		175	250	23	H1/4X2X175	-
		175	500	23	H1/4X2X175A	-
		200	250	26	H1/4X2X200	-
		200+tcj	1000	26	HJ1/4X2X200	-
		250	250	33	H1/4X2X250	-
		300	250	39	H1/4X2X300	-
	63.5 (2"1/2)	100	250	10	-	H1/4X2.5X100
		125	250	12	-	H1/4X2.5X125

Diam. Ø(mm)	Length L(mm)	Watt. P (W)	Leads (mm)	W.dens. (W/cm^2)	Stocked	Non stocked
6.35 (1/4")	63.5 (2"1/2)	150	250	15	-	H1/4X2.5X150
		175	250	17	-	H1/4X2.5X175
		200	250	20	H1/4X2.5X200	-
		200+tcj	1000	20	HJ1/4X2.5X200	-
		250	250	25	H1/4X2.5X250	-
		250+tcj	1000	25	HJ1/4X2.5X250	-
	76.2 (3")	100	250	8	-	H1/4X3X100
		150	250	12	H1/4X3X150	-
		175	250	14	-	H1/4X3X175
		200	250	16	H1/4X3X200	-
		200+tcj	1000	16	HJ1/4X3X200	-
		250	250	20	H1/4X3X250	-
		250+tcj	1000	20	HJ1/4X3X250	-
		300	250	24	H1/4X3X300	-
		300+tcj	1000	24	HJ1/4X3X300	-
		400	250	31	H1/4X3X400	-
	88.9 (3"1/2)	150	250	10	-	H1/4X3.5X150
		200	250	13	H1/4X3.5X200	-
		250	250	16	-	H1/4X3.5X250
		300	250	20	H1/4X3.5X300	-
	101.6 (4")	125	250	7	-	H1/4X4X125
		150	250	8	-	H1/4X4X150
		175	250	10	-	H1/4X4X175
		200	250	11	H1/4X4X200	-
		250	250	14	H1/4X4X250	-
		300	250	17	H1/4X4X300	-
		300+tcj	1000	17	HJ1/4X4X300	-

NTC Disc Thermistors
ND 03/06/09 • NE 03/06/09 • NV 06/09

APPLICATIONS

- Commodity Product: 2 families
  ND or NE : general purpose
  NV : professional
- Alarm and temperature measurement application
- Temperature regulation application
- Level detection application
- Compensation application



TECHNOLOGY

- ND: epoxy-phenolic resin coating
  NE: epoxy resin coating (recommended for severe mounting conditions)
  NV: epoxy varnish coating
- Leads: Radial copper wire tinned
- Marking: on package only for ND03 & NE03
  ND/NE 06/09: Nominal resistance and tolerance for ±5%, ±10%
  NV06/09: Nominal resistance and tolerance
- Delivery Mode: Bulk, reeled or ammopacked

Leaded Discs

N.03

N.06

N.09

PERFORMANCE CHARACTERISTICS

Types	General purpose			Professional	
	ND03 or NE03	ND06 or NE06	ND09 or NE09	NV06	NV09
Climatic category				55/125/56-434	55/125/56-434
Operating Temperature	–55 to +150°C	–55 to +150°C	–55 to +150°C	–55 to +150°C	–55 to +150°C
Tolerance on Rn (25°C)	330Ω to 1MΩ : ± 5, 10, 20% 1500Ω to 150 kΩ : ± 3%	±5%, ±10%, ±20%	±5%, ±10%, ±20%	±2%, ±5%, ±10%	±2%, ±5%, ±10%
Maximum dissipation at 25°C	0.25 W	0.71 W	0.9 W	0.69 W	0.85 W
Thermal dissipation factor	5 mW/°C	7.1 mW/°C	9 mW/°C	6.9 mW/°C	8.5 mW/°C
Thermal time constant	10 s	22 s	30 s	18 s	30 s
Response time	< 3s				

STANDARDIZATION

NV range : approved by NFC 93271
Type: TN115 A for NV06
TN116 for NV09
List: GAM-T1
List: LNZ

OPTIONS

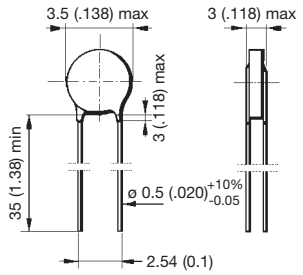
Consult factory for availability of options:

- other nominal resistance values
- other tolerances
- alternative lead materials or lengths
- controlled dimensions

TABLE OF VALUES

ND03/NE03 TYPE

ND03/NE03



Part Number	Rn at 25°C (Ω)	Material Code	B (K) (ΔB/B (1) ± 5% (2) ± 3%)	α at 25°C (%/°C)
N_03I00331 N_03I00471	330 470	I	3250 (1)	– 3.7
N_03J00681 N_03J00102	680 1,000	J	3480 (2)	– 3.9
N_03K00152 N_03K00222	1,500 2,200	K	3630 (2)	– 4.0
N_03L00272 N_03L00332	2,700 3,300	L	3790 (2)	– 4.2
N_03M00472 N_03M00682	4,700 6,800	M	3950 (2)	– 4.4
N_03N00103 N_03N00153	10,000 15,000	N	4080 (2)	– 4.6
N_03P00223 N_03P00333	22,000 33,000	P	4220 (2)	– 4.7
N_03Q00473 N_03Q00683	47,000 68,000	Q	4300 (2)	– 4.7
N_03R00104 N_03R00154	100,000 150,000	R	4400 (2)	– 4.8
N_03S00224	220,000	S	4520 (2)	– 5.0
N_03T00334 N_03T00474	330,000 470,000	T	4630 (2)	– 5.1
N_03U00105	1,000,000	U	4840 (2)	– 5.3

cosmo

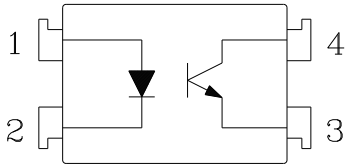
K1010 Series

4PIN PHOTOTRANSISTOR PHOTOCOUPLER

Description

The K1010 series consist of an infrared emitting diode, optically coupled to a phototransistor detector. They are packaged in a 4-pin DIP package and available in wide-lead spacing and SMD option.

Schematic



1. Anode
2. Cathode
3. Emitter
4. Collector

Features

1. Current transfer ratio
   (CTR：Min. 50% at I_F=5mA V_{CE}=5V)
2. High isolation voltage between input and output
   (Viso：5000Vrms)
3. Pb free and RoHS compliant
4. Agency Approvals
   - UL1577 / CUL C22.2 No.1 & NTC No.5, File No. E169586
   - VDE EN60747, File No.101347
   - FIMKO EN60065, File No. NCS/FI23149 A2
   - FIMKO EN60950, File No. NCS/FI24584 A1
   - SEMKO EN60065, File No. FI016484
   - SEMKO EN60950, File No. FI016433
   - CQC GB4943/GB8898-2011, File No.CQC10001049555 / CQC08001023986

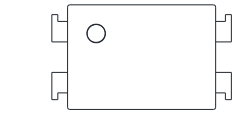
Applications

- System appliances
- Measuring instruments
- Computer terminals
- Programmable controllers
- Medical instruments
- Physical and chemical equipment
- Signal transmission between circuits of different potentials and impedances

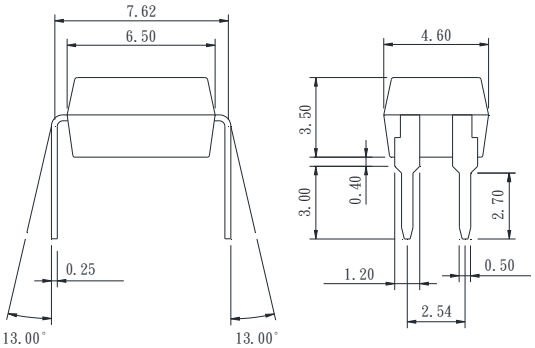
Outside Dimension

Unit : mm

1.Dual-in-line type.

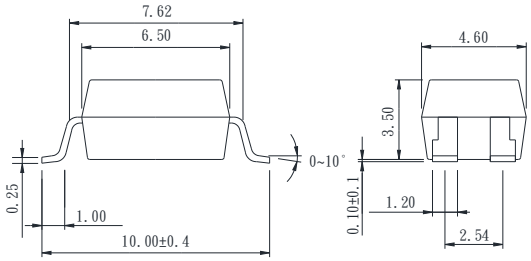


K10101X



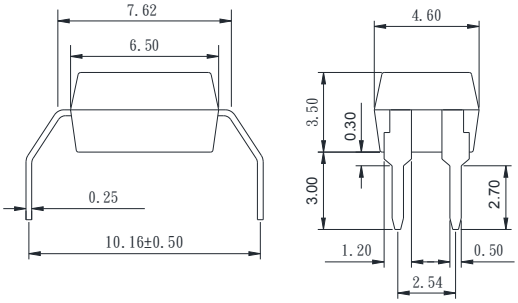
2.Surface mount type.

K10104X



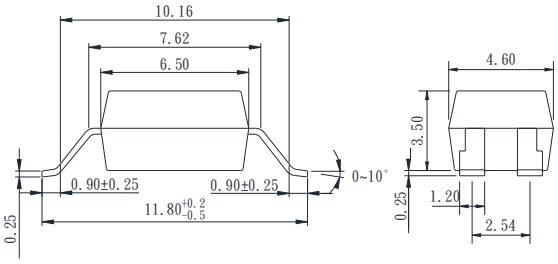
3.Long creepage distance type

K10103X



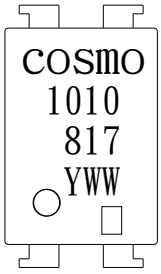
4.Long creepage distance for surface mount type.

K10106X



TOLERANCE：±0.2mm

Device Marking



Notes:

cosmo
1010
817
YWW　Y: Year code / WW: Week code
□　□: CTR rank

Absolute Maximum Ratings

(Ta=25℃)

Parameter		Symbol	Rating	Unit
Input	Forward current	I_F	50	mA
	Peak forward current	I_{FM}	1	A
	Reverse voltage	V_R	6	V
	Power dissipation	P_D	70	mW
Output	Collector-emitter voltage	V_{CEO}	80	V
	Emitter-collector voltage	V_{ECO}	6	V
	Collector current	I_C	50	mA
	Collector power dissipation	P_C	150	mW
Total power dissipation		P_{tot}	200	mW
Isolation voltage 1 minute		V_{iso}	5000	Vrms
Operating temperature		T_{opr}	-55 to +115	℃
Storage temperature		T_{stg}	-55 to +125	℃
Soldering temperature 10 seconds		T_{sol}	260	℃

Electro-optical Characteristics

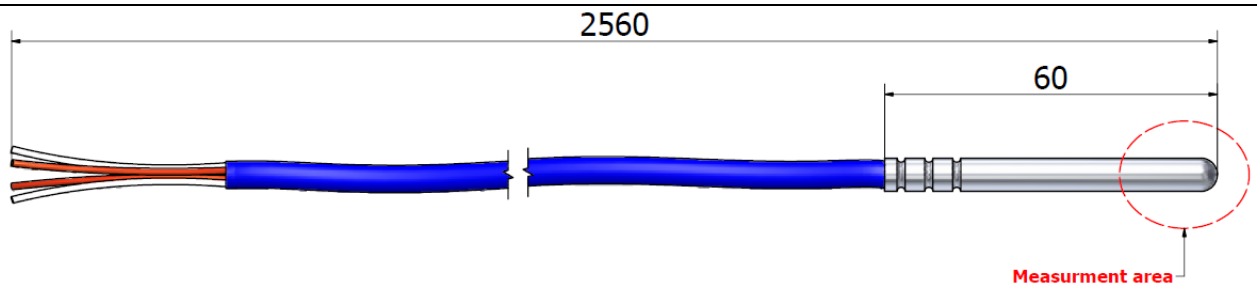
(Ta=25℃)

Parameter		Symbol	Conditions	Min.	Typ.	Max.	Unit
Input	Forward voltage	V_F	I_F=20mA	-	1.2	1.4	V
	Peak forward voltage	V_{FM}	I_{FM}=0.5A	-	-	3.0	V
	Reverse current	I_R	V_R=4V	-	-	10	μA
	Terminal capacitance	C_t	V=0, f=1KHz	-	30	-	pF
Output	Collector dark current	I_{CEO}	V_{CE}=20V, I_F=0	-	-	0.1	μA
Transfer charac-teristics	Current transfer ratio	CTR	I_F=5mA, V_{CE}=5V	50	-	600	%
			I_F=1mA, V_{CE}=5V	15	-	-	
	Collector-emitter saturation	$V_{CE(sat)}$	I_F=20mA, I_C=1mA	-	0.1	0.2	V
	Isolation resistance	R_{iso}	DC500V	$5x10^{10}$	10^{11}	-	Ω
	Floating capacitance	C_f	V=0, f=1MHz	-	0.6	1.0	pF
	Cut-off frequency	f_C	V_{CC}=5V, I_C=2mA, R_L=100Ω	-	80	-	KHz
	Response time (Rise)	t_r	V_{CE}=2V, I_C=2mA, R_L=100Ω	-	4	18	μs
	Response time (Fall)	t_f		-	3	18	μs

Data sheet
PS

RS Stock No. 457-3704

ENGLISH



PT100 THERMIC ELEMENT WITH CABLE AND TIP Ø 6

RANGE :

- -20°C / +200°C

USE :

- Universal, long time dumping

KEY POINT :

- Flexible

SPECIFICATIONS :

- PT100Ω thermic element A Class – 1x3 wires
- Tip in SS316L
- Cable 3 conductors, section 0.22mm²
  Isolated Silicon/Copper braid/Silicon
- Leak tightness by 3 ring finger crimping

DIMENSIONS :

- Tip length = 60mm
- Tip diameter = 6mm
- Total length (tip + cable) = 2560mm

METROLOGICAL DATA :

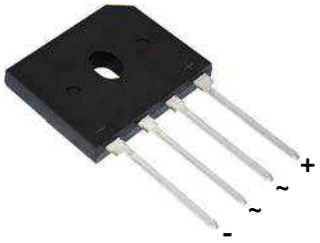
- As per IEC 751
- *Standard tolerance PT100 A class :* ±0.15 + 0.002.[t°c]

Other type:

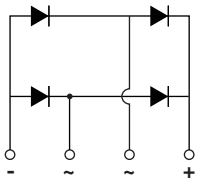
Length / Wiring	2560mm
Mounting 1x3wires	457-3704
Mounting 1x4wires	407-1387

GBU4A, GBU4B, GBU4D, GBU4G, GBU4J, GBU4K, GBU4M

Glass Passivated Single-Phase Bridge Rectifier



Case Style GBU



Case Style GBU

FEATURES

- UL recognition file number E54214
- Ideal for printed circuit boards
- High surge current capability
- High case dielectric strength of 1500 V_{RMS}
- Solder dip 275 °C max. 10 s, per JESD 22-B106
- Material categorization: for definitions of compliance please see www.vishay.com/doc?99912

RoHS COMPLIANT

TYPICAL APPLICATIONS

General purpose use in AC/DC bridge full wave rectification for monitor, TV, printer, switching mode power supply, adapter, audio equipment, and home appliances applications.

MECHANICAL DATA

Case: GBU
Molding compound meets UL 94 V-0 flammability rating
Base P/N-E3 - RoHS-compliant, commercial grade

Terminals: Matte tin plated leads, solderable per J-STD-002 and JESD 22-B102
E3 suffix meets JESD 201 class 1A whisker test

Polarity: As marked on body

Mounting Torque: 10 cm-kg (8.8 inches-lbs) max.

Recommended Torque: 5.7 cm-kg (5 inches-lbs)

PRIMARY CHARACTERISTICS	
Package	GBU
$I_{F(AV)}$	4.0 A
V_{RRM}	50 V, 100 V, 200 V, 400 V, 600 V, 800 V, 1000 V
I_{FSM}	150 A
I_R	5 µA
V_F at I_F = 4.0 A	1.0 V
T_J max.	150 °C
Diode variations	In-line

MAXIMUM RATINGS (T_A = 25 °C unless otherwise noted)

PARAMETER	SYMBOL	GBU4A	GBU4B	GBU4D	GBU4G	GBU4J	GBU4K	GBU4M	UNIT
Maximum repetitive peak reverse voltage	V_{RRM}	50	100	200	400	600	800	1000	V
Maximum RMS voltage	V_{RMS}	35	70	140	280	420	560	700	V
Maximum DC blocking voltage	V_{DC}	50	100	200	400	600	800	1000	V
Maximum average forward rectified output current at T_C = 100 °C (1)	$I_{F(AV)}$	4.0							A
Maximum average forward rectified output current at T_A = 40 °C (2)	$I_{F(AV)}$	3.0							A
Peak forward surge current single sine-wave superimposed on rated load	I_{FSM}	150							A
Rating for fusing (t < 8.3 ms)	I^2t	93							A^2s
Operating junction and storage temperature range	T_J, T_{STG}	-55 to +150							°C

Notes

(1) Unit case mounted on 1.6" x 1.6" x 0.06" thick (4.0 cm x 4.0 cm x 0.15 cm) aluminum plate
(2) Units mounted on PCB with 0.5" x 0.5" (12 mm x 12 mm) copper pads and 0.375" (9.5 mm) lead length

ELECTRICAL CHARACTERISTICS (T_A = 25 °C unless otherwise noted)										
PARAMETER	**TEST CONDITIONS**	**SYMBOL**	**GBU4A**	**GBU4B**	**GBU4D**	**GBU4G**	**GBU4J**	**GBU4K**	**GBU4M**	**UNIT**
Maximum instantaneous forward voltage drop per diode	4.0 A	V_F	1.0							V
Maximum DC reverse current at rated DC blocking voltage per diode	T_A = 25 °C	I_R	5.0							μA
	T_A = 125 °C		500							
Typical junction capacitance per diode	4 V, 1 MHz	C_J	57							pF

THERMAL CHARACTERISTICS (T_A = 25 °C unless otherwise noted)									
PARAMETER	**SYMBOL**	**GBU4A**	**GBU4B**	**GBU4D**	**GBU4G**	**GBU4J**	**GBU4K**	**GBU4M**	**UNIT**
Typical thermal resistance	$R_{\theta JA}$ (2)	22							°C/W
	$R_{\theta JC}$ (1)	4.2							

Notes

(1) Units case mounted on aluminum plate heatsink

(2) Units mounted in free air, no heatsink on PCB, 0.5" x 0.5" (12 mm x 12 mm) copper pads, 0.375" (9.5 mm) lead length

ORDERING INFORMATION				
PREFERRED P/N	**UNIT WEIGHT (g)**	**PREFERRED PACKAGE CODE**	**BASE QUANTITY**	**DELIVERY MODE**
GBU4J-E3/45	3.857	45	20	Tube
GBU4J-E3/51	3.857	51	250	Paper tray

RATINGS AND CHARACTERISTICS CURVES (T_A = 25 °C unless otherwise noted)

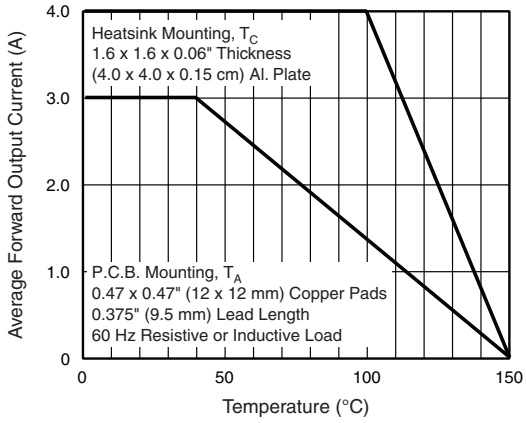


Fig. 1 - Derating Curve Output Rectified Current

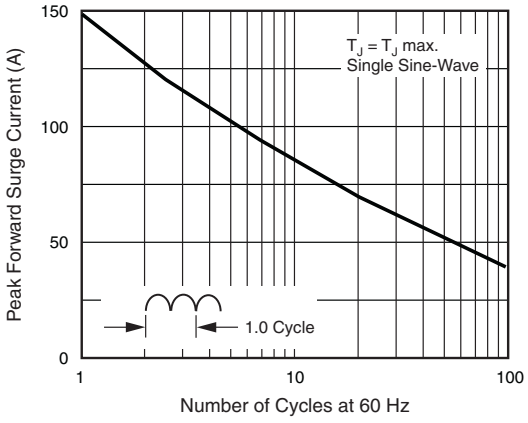


Fig. 2 - Maximum Non-Repetitive Peak Forward Surge Current Per Diode

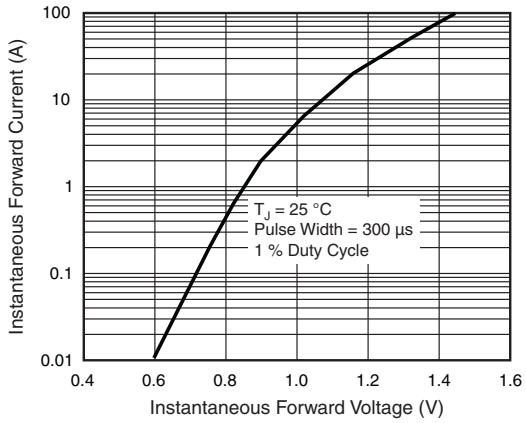


Fig. 3 - Typical Forward Characteristics Per Diode

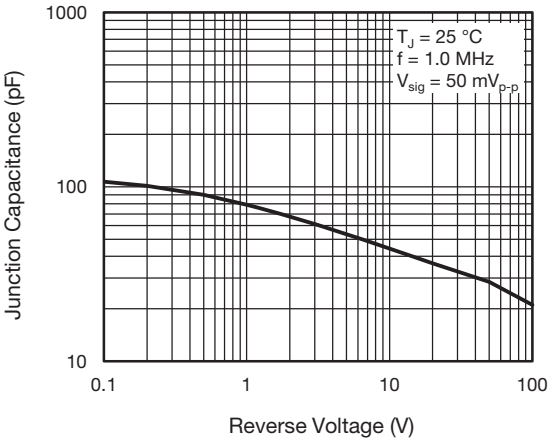


Fig. 5 - Typical Junction Capacitance Per Diode

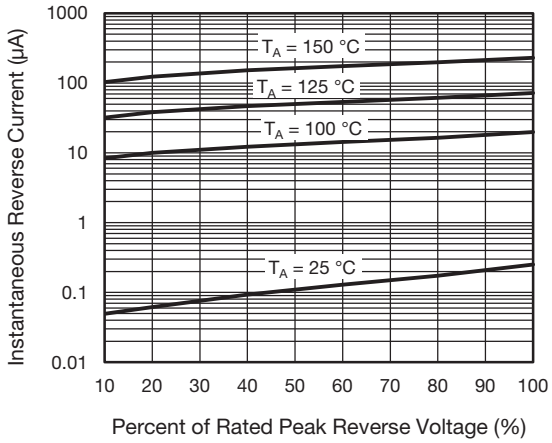


Fig. 4 - Typical Reverse Leakage Characteristics Per Diode

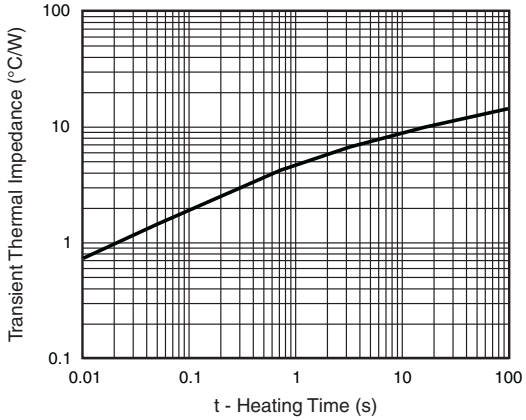
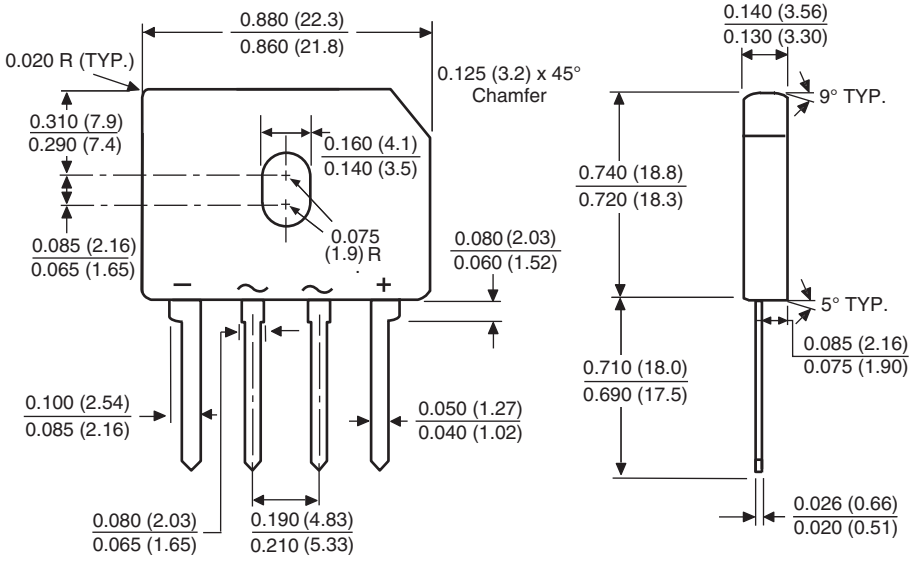


Fig. 6 - Typical Transient Thermal Impedance

PACKAGE OUTLINE DIMENSIONS in inches (millimeters)

Case Type GBU



Polarity shown on front side of case, positive lead by beveled corner

LM4040-N/-Q1 Precision Micropower Shunt Voltage Reference

1 Features

- SOT-23 AEC Q-100 Grades 1 and 3 Available
- Small Packages: SOT-23, TO-92, and SC70
- No Output Capacitor Required
- Tolerates Capacitive Loads
- Fixed Reverse Breakdown Voltages of 2.048 V, 2.5 V, 3 V, 4.096 V, 5 V, 8.192 V, and 10 V
- Key Specifications (2.5-V LM4040-N)
  - Output Voltage Tolerance (A Grade, 25°C): ±0.1% (Maximum)
  - Low Output Noise (10 Hz to 10 kHz): 35 μV_{rms} (Typical)
  - Wide Operating Current Range: 60 µA to 15 mA
  - Industrial Temperature Range: −40°C to +85°C
  - Extended Temperature Range: −40°C to +125°C
  - Low Temperature Coefficient: 100 ppm/°C (Maximum)

2 Applications

- Portable, Battery-Powered Equipment
- Data Acquisition Systems
- Instrumentation
- Process Controls
- Energy Management
- Product Testing
- Automotives
- Precision Audio Components

3 Description

Ideal for space-critical applications, the LM4040-N precision voltage reference is available in the sub-miniature SC70 and SOT-23 surface-mount package. The advanced design of the LM4040-N eliminates the need for an external stabilizing capacitor while ensuring stability with any capacitive load, thus making the LM4040-N easy to use. Further reducing design effort is the availability of several fixed reverse breakdown voltages: 2.048 V, 2.5 V, 3 V, 4.096 V, 5 V, 8.192 V, and 10 V. The minimum operating current increases from 60 µA for the 2.5-V LM4040-N to 100 µA for the 10-V LM4040-N. All versions have a maximum operating current of 15 mA.

The LM4040-N uses a fuse and Zener-zap reverse breakdown voltage trim during wafer sort to ensure that the prime parts have an accuracy of better than ±0.1% (A grade) at 25°C. Bandgap reference temperature drift curvature correction and low dynamic impedance ensure stable reverse breakdown voltage accuracy over a wide range of operating temperatures and currents.

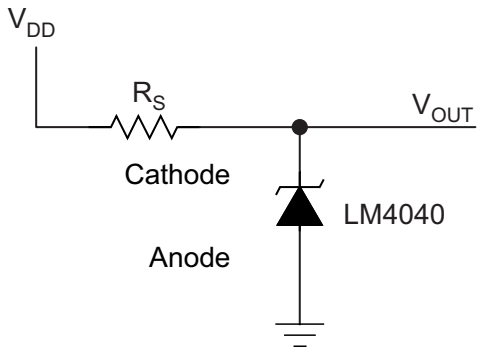
Also available is the LM4041-N with two reverse breakdown voltage versions: adjustable and 1.2 V. See the LM4041-N data sheet (SNOS641).

Device Information[1]

PART NUMBER	PACKAGE	BODY SIZE (NOM)
LM4040-N	TO-92 (3)	4.30 mm × 4.30 mm
	SC70 (5)	2.00 mm × 1.25 mm
	SOT-23 (3)	2.92 mm × 1.30 mm
LM4040-N-Q1	SOT-23 (3)	2.92 mm × 1.30 mm

(1) For all available packages, see the orderable addendum at the end of the data sheet.

Shunt Reference Application Schematic



6 Specifications

6.1 Absolute Maximum Ratings

over operating free-air temperature range (unless otherwise noted)(1)(2)

		MIN	MAX	UNIT
Reverse current			20	mA
Forward current			10	mA
Power dissipation (T_A = 25°C)(3)	SOT-23 (M3) package		306	mW
	TO-92 (Z) package		550	mW
	SC70 (M7) package		241	mW
Soldering temperature(4)	SOT-23 (M3) Package Peak Reflow (30 sec)		260	°C
	TO-92 (Z) Package Soldering (10 sec)		260	°C
	SC70 (M7) Package Peak Reflow (30 sec)		260	°C
Storage temperature		–65	150	°C

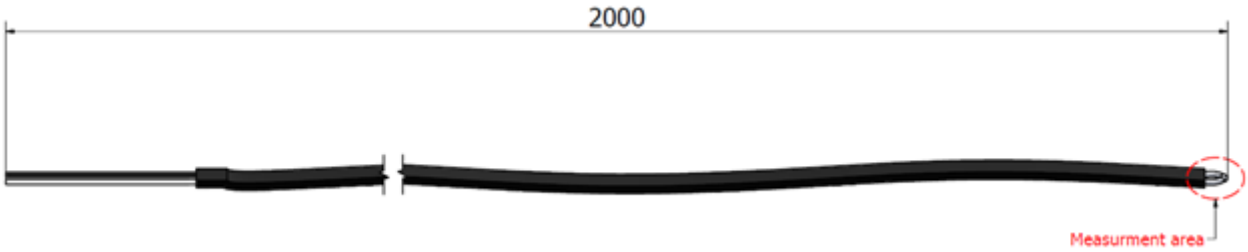
(1) Stresses beyond those listed under *Absolute Maximum Ratings* may cause permanent damage to the device. These are stress ratings only, which do not imply functional operation of the device at these or any other conditions beyond those indicated under *Recommended Operating Conditions*. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.
(2) If Military/Aerospace specified devices are required, please contact the Texas Instruments Sales Office/ Distributors for availability and specifications.
(3) The maximum power dissipation must be derated at elevated temperatures and is dictated by T_{Jmax} (maximum junction temperature), $R_{\theta JA}$ (junction to ambient thermal resistance), and T_A (ambient temperature). The maximum allowable power dissipation at any temperature is $PD_{max} = (T_{Jmax} - T_A)/R_{\theta JA}$ or the number given in the *Absolute Maximum Ratings*, whichever is lower. For the LM4040-N, T_{Jmax} = 125°C, and the typical thermal resistance ($R_{\theta JA}$), when board mounted, is 326°C/W for the SOT-23 package, and 180°C/W with 0.4″ lead length and 170°C/W with 0.125″ lead length for the TO-92 package and 415°C/W for the SC70 Package.
(4) For definitions of Peak Reflow Temperatures for Surface Mount devices, see the TI *Absolute Maximum Ratings for Soldering Application Report* (SNOA549).

6.2 ESD Ratings

			VALUE	UNIT
$V_{(ESD)}$	Electrostatic discharge	Human-body model (HBM), per ANSI/ESDA/JEDEC JS-001(1)	±2000	V
		Charged-device model (CDM), per JEDEC specification JESD22-C101(2)	±200	

(1) JEDEC document JEP155 states that 500-V HBM allows safe manufacturing with a standard ESD control process.
(2) JEDEC document JEP157 states that 250-V CDM allows safe manufacturing with a standard ESD control process.

Datos técnicos Nº de acciones de RS. 621-2170CABLE TERMOPAR TIPO K CON PUNTO CALIENTE



RANGO DE AISLAMIENTO DE LA TEMPERATURA DE FUNCIONAMIENTO NORMAL:

- -50°C/+400°C

EL RANGO DE TEMPERATURA DE TERMOPAR:

- -50°C / +1000°C

UTILIZA :

- Universal, sencilla y económica, ideal para la temperatura de las medidas de prueba o de laboratorio.

PONE DE RELIEVE:

- Tiempo de respuesta corto, fácil implementación y bajo cota.

CARACTERÍSTICAS :

- 2 cables conductores, sección mm² 0,07, aislamiento del vidrio.
- Aparente punto caliente

DIMENSIONES :

- Longitud de Cable = 2000mm
- Plana = 1,7 x 1,1mm
- Peso= 8.33g

DATOS METROLÓGICOS:

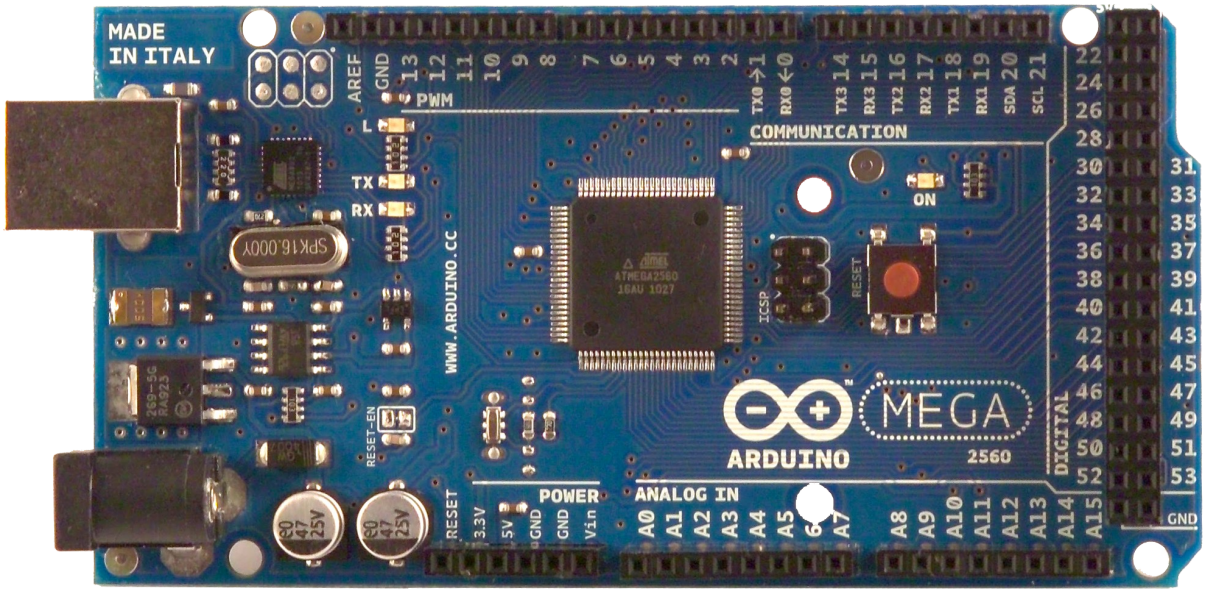
- Cumple con la norma IEC 584 General tolerancia TC 'K' Class1:
  -40°C < t°< +375°C = ± 1,5°C
  375°C < t°< 1000°C= ±0.004
- El tiempo constante de 63% = 0,7 seg.
- EGF (mV) de salida siguiente de la curva de señal del tipo "K" resultado de la norma IEC 60584"

Otros tipos de cable termopar vidrio de seda:

Tipo de termopar	J	N	T
RS en stock no.	621-2186	621-2192	621-2209
T ° de la gama de aislamiento	-50 à 400°C		
Rango de T (termopar)	-50 à 1000°C	-50 à +1200°C	-200 à +350°C
Longitud 2000 mm			

El tipo de termopar es elegido según la temperatura a medir y la señal electrónica que debe ser compatible.

Arduino MEGA 2560



Product Overview

The Arduino Mega 2560 is a microcontroller board based on the ATmega2560 (datasheet). It has 54 digital input/output pins (of which 14 can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started. The Mega is compatible with most shields designed for the Arduino Duemilanove or Diecimila.

Index

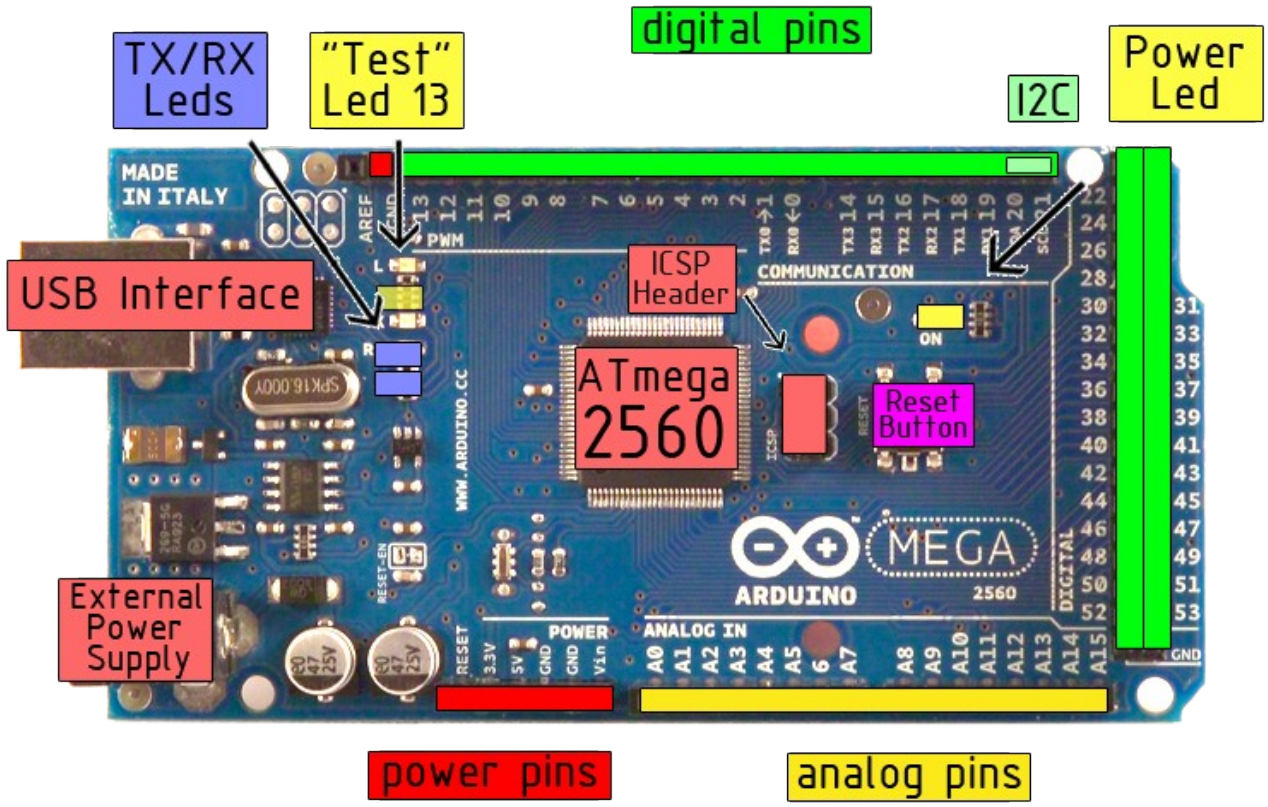
Technical Specification

EAGLE files: arduino-mega2560-reference-design.zip Schematic: arduino-mega2560-schematic.pdf

Summary

Microcontroller	ATmega2560
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	54 (of which 14 provide PWM output)
Analog Input Pins	16
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	256 KB of which 8 KB used by bootloader
SRAM	8 KB
EEPROM	4 KB
Clock Speed	16 MHz

the board



Power

The Arduino Mega2560 can be powered via the USB connection or with an external power supply. The power source is selected automatically. External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the board's power jack. Leads from a battery can be inserted in the Gnd and Vin pin headers of the POWER connector.

The board can operate on an external supply of 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may be unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.

The Mega2560 differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the Atmega8U2 programmed as a USB-to-serial converter.

The power pins are as follows:

- **VIN.** The input voltage to the Arduino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.
- **5V.** The regulated power supply used to power the microcontroller and other components on the board. This can come either from VIN via an on-board regulator, or be supplied by USB or another regulated 5V supply.
- **3V3.** A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50 mA.
- **GND.** Ground pins.

Memory

The ATmega2560 has 256 KB of flash memory for storing code (of which 8 KB is used for the bootloader), 8 KB of SRAM and 4 KB of EEPROM (which can be read and written with the EEPROM library).

Input and Output

Each of the 54 digital pins on the Mega can be used as an input or output, using pinMode(), digitalWrite(), and digitalRead() functions. They operate at 5 volts. Each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor (disconnected by default) of 20-50 kOhms. In addition, some pins have specialized functions:

- **Serial: 0 (RX) and 1 (TX); Serial 1: 19 (RX) and 18 (TX); Serial 2: 17 (RX) and 16 (TX); Serial 3: 15 (RX) and 14 (TX).** Used to receive (RX) and transmit (TX) TTL serial data. Pins 0 and 1 are also connected to the corresponding pins of the ATmega8U2 USB-to-TTL Serial chip .
- **External Interrupts: 2 (interrupt 0), 3 (interrupt 1), 18 (interrupt 5), 19 (interrupt 4), 20 (interrupt 3), and 21 (interrupt 2).** These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. See the attachInterrupt() function for details.
- **PWM: 0 to 13.** Provide 8-bit PWM output with the analogWrite() function.
- **SPI: 50 (MISO), 51 (MOSI), 52 (SCK), 53 (SS).** These pins support SPI communication, which, although provided by the underlying hardware, is not currently included in the Arduino language. The SPI pins are also broken out on the ICSP header, which is physically compatible with the Duemilanove and Diecimila.
- **LED: 13.** There is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.
- **I^2C: 20 (SDA) and 21 (SCL).** Support I^2C (TWI) communication using the Wire library (documentation on the Wiring website). Note that these pins are not in the same location as the I^2C pins on the Duemilanove.

The Mega2560 has 16 analog inputs, each of which provide 10 bits of resolution (i.e. 1024 different values). By default they measure from ground to 5 volts, though is it possible to change the upper end of their range using the AREF pin and analogReference() function.

There are a couple of other pins on the board:

- **AREF.** Reference voltage for the analog inputs. Used with analogReference().
- **Reset.** Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.

Communication

The Arduino Mega2560 has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega2560 provides four hardware UARTs for TTL (5V) serial communication. An ATmega8U2 on the board channels one of these over USB and provides a virtual com port to software on the computer (Windows machines will need a .inf file, but OSX and Linux machines will recognize the board as a COM port automatically. The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the board. The RX and TX LEDs on the board will flash when data is being transmitted via the ATmega8U2 chip and USB connection to the computer (but not for serial communication on pins 0 and 1).

A SoftwareSerial library allows for serial communication on any of the Mega's digital pins.

The ATmega2560 also supports I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus; see the documentation on the Wiring website for details. To use the SPI communication, please see the ATmega2560 datasheet.

Programming

The Arduino Mega2560 can be programmed with the Arduino software (download). For details, see the reference and tutorials.

The Atmega2560 on the Arduino Mega comes preburned with a bootloader that allows you to upload new code to it without the use of an external hardware programmer. It communicates using the original STK500 protocol (reference, C header files).

You can also bypass the bootloader and program the microcontroller through the ICSP (In-Circuit Serial Programming) header; see these instructions for details.