Universidad Nacional Autónoma de México

Facultad de Ciencias

AUTOMATIZACIÓN DE HORNOS PARA TRATAMIENTOS TÉRMICOS CON TECNOLOGÍA ARDUINO

T E S I S

QUE PARA OBTENER EL TÍTULO DE:

FÍSICO

PRESENTA:

Gustavo Meléndez Valentin

DIRECTOR DE TESIS:

Dr. Raúl Arturo Espejel Morales

Ciudad Universitaria, Cd. Mx., 2018

Hoja de Datos del jurado

1. Datos del alumno
Meléndez
Valentin
Gustavo
56673028
Universidad Nacional Autónoma de México
Facultad de Ciencias
Física
308267059

2. Datos del tutor
Dr
Raúl Arturo
Espejel
Morales

3. Datos del sinodal 1
Dr
Alipio Gustavo
Calles
Martínez

4. Datos del sinodal 2
Dr
José Ignacio
Jiménez
Mier y Terán

5. Datos del sinodal 3
Dr
Roberto Ysacc
Sato
Berrú

6. Datos del sinodal 4
Dra
Corina
Solís
Rosales

7. Datos del trabajo escrito
AUTOMATIZACIÓN DEL HORNOS PARA TRATAMIENTOS TÉRMICOS CON TECNOLOGÍA ARDUINO
66 p
2018

AGRADECIMIENTOS

A mis padres; Ricardo y Epifanía, que siempre estuvieron ahí para brindarme su ayuda y consejo, no sólo en este proyecto, sino en toda mi vida

A mis hermanos; Ricardo y Leticia, que de igual manera me apoyaron en todo

Al Dr. Raúl Arturo Espejel Morales, pieza fundamental por su consejo, amistad, guía y apoyo para el comienzo y fin de este proyecto, mil gracias

Al Dr. Enrique Cabrera Bravo, por guiarme en el comienzo de este proyecto

Al Dr. Raúl Espejel Paz, por su gran ayuda en este proyecto

A todos mis amigos que me apoyaron y brindaron su amistad en el transcurso de la carrera

A mis amigos José Ángel, Carlos Maciel, Roxana Mitzayé, Luis Felipe, Eduardo Santiago y Diana Pineda por su gran amistad y apoyo

Índice General

CAPÍTULO 1

INTRODUCCIÓN

INTRODUCCIÓN

1.1. Justificación

Sabemos que el término *tratamiento térmico* describe un proceso en el cual una muestra se somete a temperaturas controladas por intervalos de tiempo definidos. En algunos casos, la muestra puede ser sometida adicionalmente a otras influencias químicas y/o físicas (atmósfera controlada, compresiones, etc.). El objetivo del tratamiento térmico es conferirle a la pieza propiedades requeridas para procesos de transformación posteriores o para su aplicación final [**?**]. Un proceso de tratamiento térmico puede provocar transformaciones de los constituyentes estructurales sin modificar la composición química promedio del material.

Al final del tratamiento térmico, los componentes estructurales pueden estar o no en equilibrio, también puede causar cambios en el tamaño, forma o distribución de los componentes estructurales. Para la realización de los tratamientos térmicos pueden utilizarse distintos tipos de hornos; de llama, gas, eléctricos, etc. e incluso de microondas.

La elección del elemento calefactor (horno, parrilla, etc.) debe hacerse teniendo en cuenta el tamaño y forma de la pieza, además de las temperaturas del tratamiento térmico. Adicionalmente, debe considerarse la velocidad de calentamiento y el medio de enfriamiento que habrán de ser usados. Otros factores importantes son el diseño y tipo de dicho elemento calefactor (cámara, chimenea, tubular, de baño, resistencias, etc.), el medio/atmósfera de tratamiento térmico (aire circulante, gas inerte, vacío, baño de sales, etc.) y el tipo de calentamiento (interno o externo, electrodos, inducción o resistencia, gas, etc.).

Por las razones anteriormente expuestas; por ejemplo, los hornos para los tratamientos térmicos suelen tener un precio elevado y ser de difícil acceso. No obstante, conforme la tecnología ha ido avanzando el precio de estos ha ido decreciendo, pero el acceso a ellos sigue siendo difícil. En conjunto, los hornos y los tratamientos térmicos

tiene cada vez más impacto en su estudio y sus aplicaciones, no solamente en el aspecto científico, sino también en otras áreas, por ejemplo:

- Alimentos [**?**]:

  ▶ Pasteurización

  ▶ Esterilización comercial (Enlatados)

- Salud:

  ▶ Medicina: Esterilizado de acero inoxidable para instrumental quirúrgico e implantes [**?**, **?**].

  ▶ Medicamentos: Tratamiento térmico para reacondicionar medicamentos caducados [**?**].

1.2. Antecedentes

Los tratamientos térmicos así como la construcción de elementos calefactores no es trivial, sino que representa todo un reto. Por ejemplo, los hornos industriales que se comercializan hoy en día suelen ser automatizados de alguna manera; es decir, cuentan con algún controlador programable, no obstante, estos hornos industriales sólo suelen tener una forma de calentar: dar toda la potencia o ir aumentando la potencia de manera proporcional para generar una o pocas rampas de calentamiento lo cual representa una limitación, además de que, en general, estos hornos suelen no tener control sobre el enfriamiento, estos factores pueden representar inconvenientes para realizar un tratamiento térmico específico.

Debido a las grandes aplicaciones que los tratamientos térmicos tienen en la actualidad, éstos se han ido mejorando utilizando *Software* de uso general, por ejemplo,

LabView [?, ?] para el control de temperatura. El control de temperatura de los trabajos anteriores, resultó ser eficiente pero no de fácil reproducción; es decir, para usar su controlador de temperatura, se requiere un gran presupuesto, debido a que el *Sofware LabView* no es libre; el precio de su versión más económica es de $98.55 UDS anuales. Además del gran costo que se tiene en el controlador anterior, la precisión que se logró en los anteriores trabajos es de $\pm$ 10°C.

Algunas otras desventajas (además del gran costo) en los controladores de temperatura basados en *LabView* son: la necesidad de adquirir un conocimiento previo para hacer uso de éste *Software* ya que no es intuitivo y la dependencia de la PC, con lo cual podría limitar su uso.

1.3. Tratamientos Térmicos

Ciertos tratamientos térmicos involucran varios procesos de calentamiento y enfriamento para efectuar cambios en algún material; por ejemplo en cristales, metales y aleaciones [?], en las cuales con el calentamiento se busca modificar sus propiedades mecánicas [?] como: duereza, resistencia, tenacidad, etc., ópticas [?] y eletrónicas sin variar su estructura química[1]. Para ilustrar la importancia de un control riguroso de la temperatura se describirán varias técnicas [?] que requieran de un tratamiento térmico con un control preciso. Entre los mas destacados son: *Temple, Normalizado, Recosido y Revenido*, posteriormente se explicarán un poco más a detalle.

De forma general, los tratamientos térmicos tienen tres etapas [?, ?]:

- **Etapa de calentamiento:**
  La elevación de la temperatura debe seguir una o más rectas concatenadas con las pendientes positivas deseadas.

[1]Los tratamietos termoquímicos son procesos que modifican/alteran la estructura química del material mediante un proceso de difusión [?], éste tipo de tratamiento no conforma parte del presente trabajo

- **Etapa de temperatura constante:**
  En esta etapa la temperatura debe mantenerse constante durante el tiempo requerido.

- **Etapa de enfriamiento:**
  En esta etapa puede requerirse un enfriamiento con uno o más rectas concatenadas con pendientes negativas.

A continuación se muestra en la figura 1.1 las etapas antes mencionadas que conforman un tratamiento térmico.

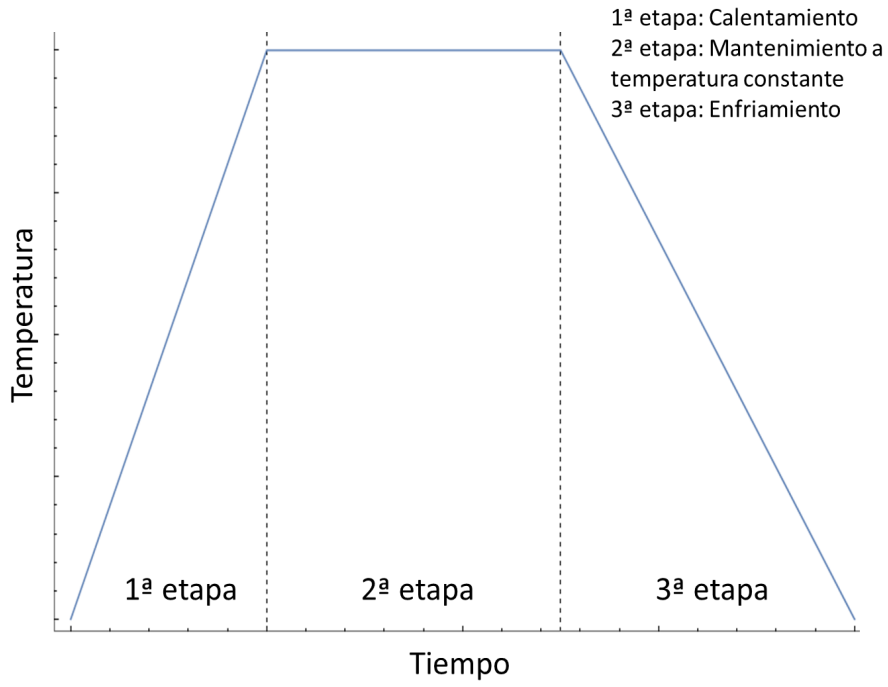

Figura 1.1: Etapas que caracterizan al tratamiento térmico

Existen tres tipos tratamiento térmico, los cuales son [?, ?, ?]:

- **Tratamiento Isotérmicos**

- **Tratamiento con enfriamiento continuo**

- **Tratamientos con cambio y sin cambio en la composición química**

Para caracterizar los tratamientos térmicos isotérmicos o de enfriamiento continuo, anteriormente mencionados, se utilizan Diagramas Tiempo-Temperatura-Transformacion o simplemente Diagramas T.T.T[2]. Los diagramas Tiempo-Temperatura-Transformación (T.T.T.) para los Tratamientos Isotérmicos (TI) y para los tratamientos de Enfriamiento Continuo (TEC) son un complemento para predecir la micro-estructura y dureza de los materiales después del tratamiento térmico o también para especificar el tratamiento térmico que logrará una microestructura y dureza deseadas. Para determinar el tipo de diagrama óptimo para el material, es necesario estar familiarizado con las distintas características, posibilidades y limitaciones de los mismos.

1.3.1. Tratamientos Isotérmicos o Curvas T.T.T.

El tratamiento isotérmico, es un proceso de inmersión que se realiza a temperatura constante se le conoce como curva T.T.T. al diagrama que relaciona al tiempo y la temperatura requeridos para mantener a la temperatura constante. Dichas curvas son gráficas que representan el porcentaje de la transformación y/o cambio de propiedades en función de la temperatura (eje vertical) y el tiempo (eje horizontal, **normalmente en escala logarítmica**); es decir, las curvas T.T.T. son transformaciones de fase como son: **ferrita, austenita, perlita, entre otras**(ver Apéndice 2) que ocurren en las aleaciones y/o metales, con una determinada composición cuando es enfriado a Temperatura constante por debajo de la temperatura de eutectoide (al ser enfriada lentamente llega a esta temperatura de solidificación) durante cierto tiempo.

Los diagramas T.T.T. se construyen utilizando dos métodos, los cuales son [**?**]:

[2]En la literatura, a saber en [**?**], [**?**] y [**?**], se usa el término curva y diagrama indistintamente, pero dentro de este presente trabajo distinguiremos entre uno y otro. Un diagrama es un gráfico que puede ser simple o complejo, con pocos o muchos elementos, pero que sirve para simplificar la información sobre un sistema o fenómeno determinado. Una curva puede ser una línea que representa de manera gráfica la magnitud de un fenómeno de acuerdo a los valores que adoptan sus variables.

- Metalografía:
  En este método se toman pequeñas probetas, las cuales se calientan a una temperatura superior a la crítica, seguida de un temple con baños de sal o plomo liquido a una temperatura fija y constante en la cual se comienza a notar la transformación de fase.

- Dilatometría:
  Esta técnica se realiza como cambio de longitud o volumen respecto a la temperatura.

Estos diagramas tienen un forma característica en forma de **"S"** que es debido a la forma en como están escalados sus ejes.

A continuación se muestra en la figura 1.2 un ejemplo de un diagrama T.T.T.

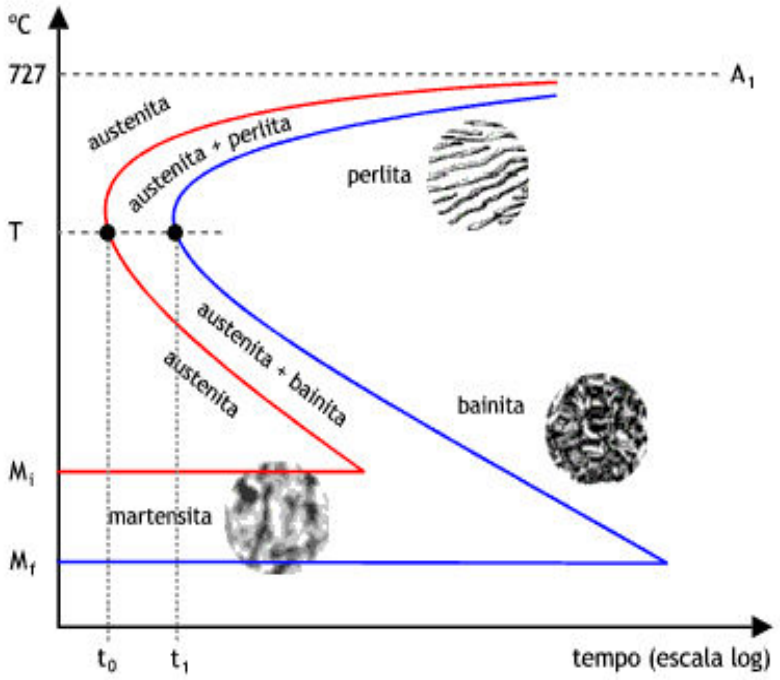

Figura 1.2: Diagrama T.T.T.

1.3.2. Tratamiento de Enfriamiento Continuo

Del diagrama T.T.T. es posible derivar otra curva llamada transformación de enfriamiento continuo. Se le conoce como curva T.E.C. (Tratamiento de Enfriamiento Continuo). Estos tipo de tratamiento comienza como un diagrama de transformación-temperatura-tiempo (T.T.T.) el cual indica el tiempo necesario para que una fase se descomponga en otras fases a diferentes velocidades de enfriamiento. A continuación en la figura 1.3 se presentará una curva que siguen algunas aleaciones.

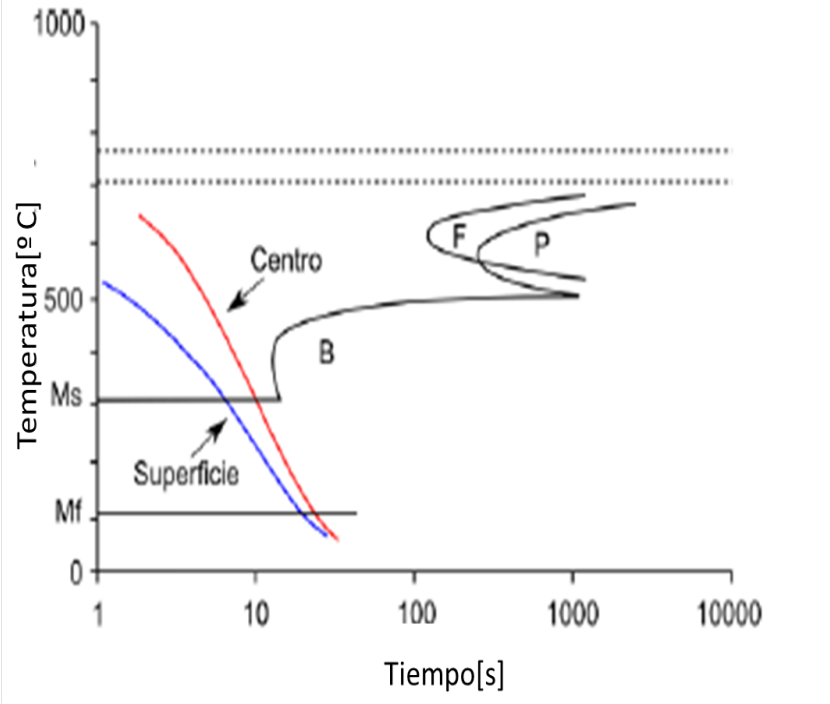

Figura 1.3: Curva de enfriamiento continuo

De acuerdo con lo anterior, consideramos que un dispositivo de control debe de ser capaz de hacer que la temperatura de un horno siga curvas arbitrarias de acuerdo al planteamiento ó trabajo del tratamiento térmico. Debido a las necesidades para llevar a acabo dicho tratamiento, dentro del presente, se propone como dispositivo de control un sistema basado en la plataforma Arduino; ya que dicha plataforma es de Software libre y con componentes de fácil acceso. A continuación se dará una breve explicación de la plataforma Arduino.

1.4. Arduino

Arduino es una plataforma electrónica de código abierto basada en *software* y *hardware* flexibles, baratos, accesibles y fáciles de usar [**?**], además cuenta con la capacidad de envío/recepción de datos mediante entradas analógicas y digitales, con una gran variedad de dispositivos electrónicos y sensores de control como: pantallas LCD, *pantallas OLED, motores* (de DC ó a pasos), sensores de sonido, termopares y tarjetas de memoria SD, por mencionar algunos.

La tarjeta de Arduino está basado en un micro-controlador *AVR* (ver Apéndice 1), y sus programas están basados en el lenguaje de programación *C* y cuenta con un compilador propio, por otro lado, Arduino tiene varias placas electrónicas: ArduinoUNO, ArduinoMEGA, Arduino Ethernet, etc. Notemos que dichas placas Arduino tienen un voltaje de alimentación que va desde 5V hasta 9V[3]. En el presente se usó la tarjeta ArduinoUNO, para primeras pruebas y la placa ArduinoMEGA para el prototipo final.

Los diferentes tipos de placas Arduino tienen distintas caracteristicas, por ejemplo, ArduinoUNO tiene 13 entradas digiales (6 de ellas trabajan con **PWM**[4] *por sus siglas en inglés, modulación por ancho de pulso*) configurables como entradas y/o salida que operan a 5 Volts, cabe mencionar que estas placas poseén salidas de 3.3V o 5V para alimentar sus periféricos, notemos que, aunque el voltaje de alimentación puede variar en un cierto rango las salidas lógicas están siempre referidas a 5V; además también se tiene de 6 entradas analógicas las cuales aceptan niveles entre 0V a 5V. Por otro lado la placa ArduinoMEGA, cuenta con 54 pines digitales (14 de ellos operan con PWM, éstos pines nos interesan mucho dentro del proyecto) y 16 pines analógicos [**?**][**?**]. En términos generales, la estructura de programación de **todas las placas Arduino** consta de dos funciones principales [**?**] `setup()` y `loop()`, las cuales son indispensables para su funcionamiento.

La función `setup()`:

[3]Las placas Arduino tienen un voltaje óptimo para su funcionamiento, pero existe un voltaje límite de 12V(para ArduinoUNO, ArduinoMEGA puede soportar máximo 20V), el cual si se excede puede dañar la placa parcial o totalmente

[4]El término PWM se explicará más adelante

Ésta función se ejecuta una única vez dentro del programa. En ésta función; generalmente se establecen las funciones que llevarán acabo los pines (salida o recepción de datos) de Arduino, en caso de requerirlo, se inicia la comunicación serial (monitor PC), el comando que llama a esta función es `Serial.begin()`, el cual toma como argumento la velocidad de *bit por segundo(bps)* de comunicación con el puerto serie, siendo **9600** el valor más común [**?**]. También dentro de esta función se establecen las configuraciones y métodos que solamente necesitan ejecutarse una única vez, como por ejemplo imprimir un mensaje en el puerto serie, activar el reloj, grabar un encabezado en la memoria SD, etc.

La función `loop()`:

Esta función establece un ciclo de instrucciones que se ejecutarán continuamente durante un tiempo determinado por el usuario o mientras la tarjeta Arduino esté encendida. Dentro de esta función se procede a establecer las instrucciones, operaciones ó medidas que se requieren durante un tiempo de ejecución, por ejemplo, encender y/o apagar un LED cada segundo, grabar en la memoria micro-SD cada segundo las temperaturas registradas por termopares, etcétera. El tiempo en que se ejecuta cada instancia de este ciclo dependerá del tipo de operaciones ó medidas que el usuario deseé realizar; es decir, este tiempo puede modificarse mediante la función `delay()` que recibe como argumento el tiempo en mili-segundos, durante el cual interrumpirá la ejecución del programa a realizar. Todo programa debe contar con estas dos funciones, de lo contrario no será posible ejecutar el mismo. A continuación, se muestra en la tabla 1.1 algunos de los comandos mas utilizados [**?**, **?**].

Comando	Descripción
`pinMode(pin,mode)`	Con esta función se establece el comportamiento de los pines digitales de Arduino. En el parámetro **pin** se escribe el número de pin a configurar, en **mode** se determina si el pin es de entrada "INPUT" o salida "OUTPUT".
`digitalWrite(pin,value)`	Se establece el estado que adoptará un pin determinado. En **pin** se coloca el número del pin con el que se desea trabajar y en **value** el estado, alto (5V) "HIGH" o bajo (referido a tierra, 0V) "LOW". Para poder usar esta función es necesario configurar el **pin** como salida usando el comando anterior.
`Serial.begin(boundRate)`	Se incia la comunicación serial. Hace posible la comunicación entre el micro controlador y la computadora. En **boundRate** se establece la velocidad de transmición de datos, comunmente se utiliza el valor de 9600.
`Serial.print(val)`	Con esta función se imprime el mensaje especificado en **val**. Otra instrucción similar es `Serial.println()`, éste incluye un salto de linea al final del mensaje.
`Serial.read()`	Permite a Arduino leer un valor desde del puerto serial.
`delay(t)`	Interrumpe la ejecución de un programa por un lapso de tiempo t en mili-segundos
`analogWrite(pin,value)`	Esta función permite usar los pines que trabajan con PWM (Pulse Width Modulation), nótese que en las placas de Arduino los pines que trabajan PWM son los digitales. En **pin** se coloca el número del pin de salada y en **value** se coloca un valor dentro del rango 0 a 255, donde 0 es siempre apagado y 255 siempre prendido.
`analogRead(pin)`	En esta función se lee el valor especificado de alguno de los pines analógicos en el intervalo de 0V a 5 V, mapeados en valores enteros de 0 a 1024 respectivamente.

Tabla 1.1: Algunos de los comandos más utilizados en Arduino

1.4.1. PWM

La modulación por ancho de pulso, **PWM** es una técnica que utilizan algunos dispositivos, por ejemplo: motores, atenuadores y Arduino, en la cuál se obtienen señales analógicas por medio de señales digitales. Cabe mencionar que una señal digital se trata de una señal formada por valores discretos con dos estados de funcionamiento: *unos* y *ceros*, que corresponden con estado de alto o bajo nivel de voltaje eléctrico respectivamente. Generalmente podemos caracterizar este tipo de señales digitales como ondas cuadradas con la misma amplitud, en Arduino esta amplitud es de 5V.

Por otro lado, la señal analógica se refiere a valores que varían con respecto al tiempo de manera continua, como ejemplo de esto tenemos las siguientes cantidades físicas: rapidez, posición, **temperatura**, etc. Notemos que este tipo de señales pueden tomar distintos valores posibles dentro de un intervalo.

A continuación se muestra en la figura 1.4 la representación de estos dos tipos de señales

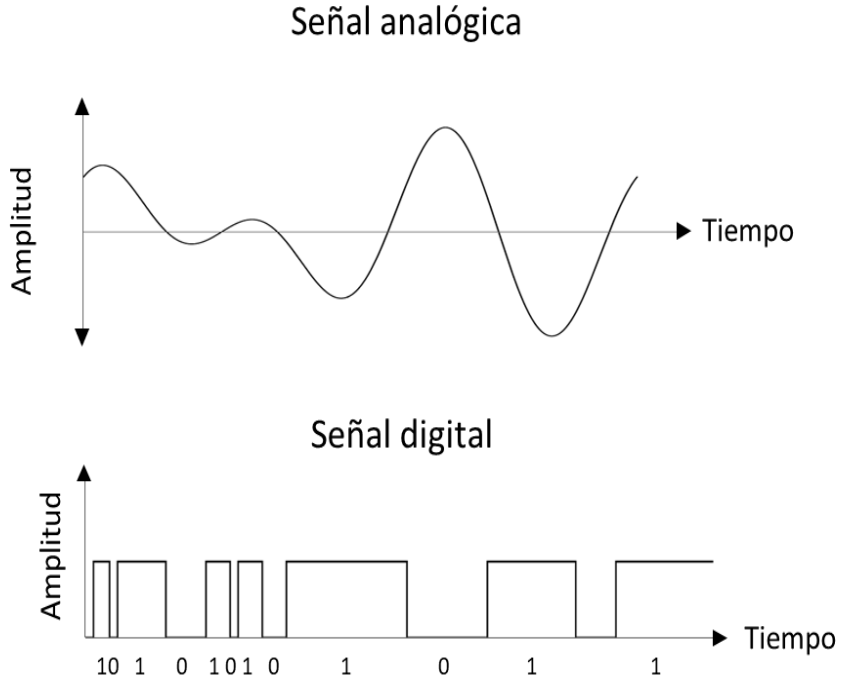

Figura 1.4: Señales Analógicas y Digitales

Como se mencionó anteriormente, Arduino cuenta con entradas analógicas, y la opción para emular una salida analógica por medio de esta modulación. Así Arduino puede simular voltajes entre 0 y 5 volts, en el cuál se puede establecer un intervalo de tiempo (frecuencia/periodo) para cada uno de ellos. Al tiempo de duración de cada estado encendido-apagado se le conoce como ancho de pulso y con la modulación de tiempo, llegamos a lo que se le conoce como PWM.

Regularmente la modulación por ancho de pulso con el que trabaja Arduino tiene una frecuencia de 500 Hz. La función que tiene implementada Arduino que nos permite hacer uso de la PWM, es `analogWrite()` que recibe un argumento entero en el intervalo 0–255, de tal manera que el valor de 255 dicha función genera un ciclo de trabajo del 100 % de trabajo; y valores proporcionales para el resto. A continuación se muestra en la figura 1.5 aún mas este hecho sobre la PWM de Arduino.

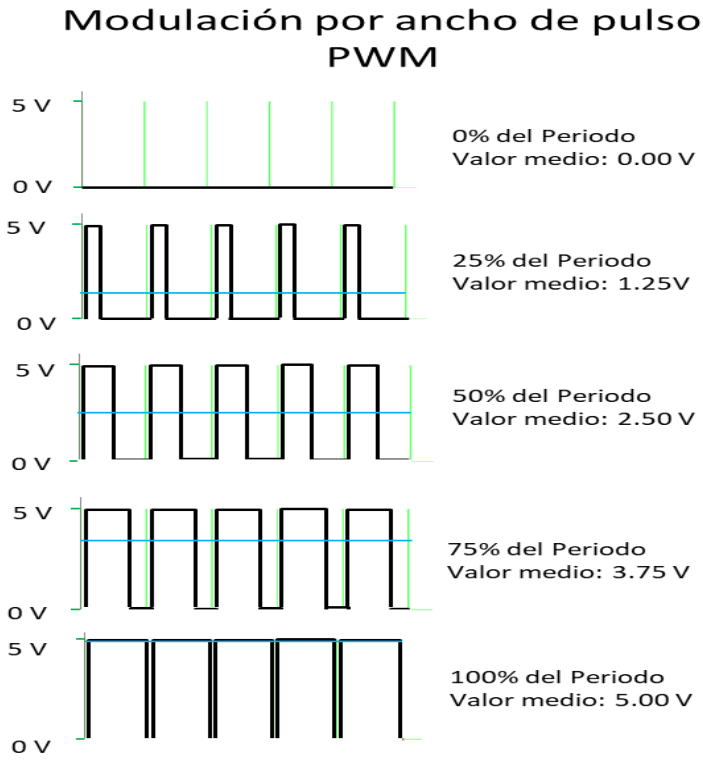

Figura 1.5: Clclo de Trabajo PWM

1.5. Termopares

Un termopar o también llamado termocupla, es un transductor de temperatura; es decir, es un dispositivo que transforma una magnitud física, en este caso la termperatura en una señal eléctrica [**?**]. Un termopar consta de la unión de dos conductores diferentes, unidos en un extremo, el cual constituye el nodo de medición [**?**]. Este nodo de medición, al calentarse produce una diferencia de potencial (*f.e.m.*) proporcional a la diferencia de temperatura entre ellos (*efecto Seebeck*, explicado un poco a detalle en el Apéndice 4). Dicha *f.e.m.* se produce debido a la diferente densidad de electrones de ambos materiales conductores en conjunto a la diferencia de temperaturas entre el punto de medición (T1) y la zona fría (T2), como se muestra en la siguiente figura 1.6.

T1 al ser calentado produce una *f.e.m.* entre las terminales situadas en T2 donde ésta se registra (ya sea con un multímetro ó como en este proyecto, utilizando Arduino) y posteriormente se le asociará un temperatura.

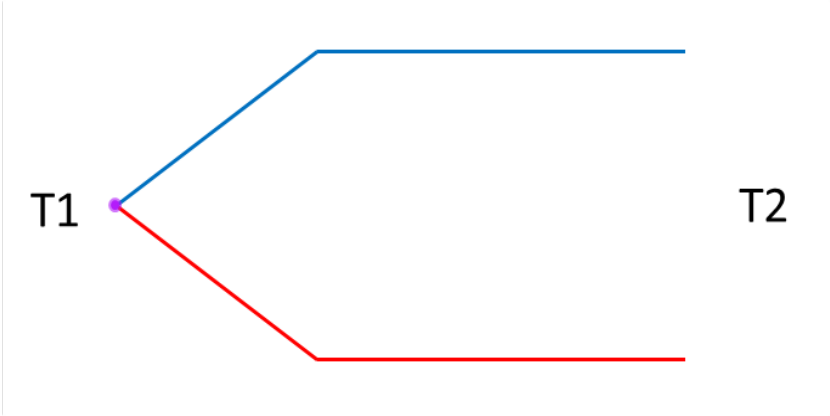

Figura 1.6: Esquema de un Termopar

La construcción de los termopares depende de sus constituyentes (material A y material B), por lo cual existe una gran variedad de estos con distintos rangos de temperaturas.

En la tabla 1.2 se muestran algunos termopares:

Tipo	Descripción
K(cromel/alumel)	Rango de temperatura entre 0°C a 1260°C, también tiene aplicaciones criogénicas en el rango de -200°C a 0°C. Tiene una **sensibilidad** de 41μV/°C. Posee buena resistencia a la oxidación.
E(cromel/constantán)	Rango de temperatura de 95°C a 900°C, también puede ocuparse en aplicaciones criogénicas sin corrosión. Tiene una sensibilidad de 68μV/°C y no es magnético.
J(hierro/constantán)	Su rango de temperatura es de 0°C a 760°C. Por lo general es recomentado ser usado en medio inertes o vacío, debido a su rápida oxidación que sufre el hierro por enmcima de 550°C. La sensibilidad es de $\tilde{5}2\mu$V/°C.
N(Nicrosil(Ni-Cr-Si/Nisil(Ni-Si)))	Es adecuado para temperaturas en el rango de 0°C a 1260°C debido a la estabilidad y resistencia a la oxidación en atmósferas que no contengan azufre. Tiene una sensibilidad de 10μV/°C.
B(Platino(Pt)/Rodio(Rh))	Termopares de temperaturas altas, en el rango de 870°C a 1800°C, resiste atmósferas oxidantes, requiere protección a la contaminación debido a su constitución de metales nobles. Tiene una sensibilidad de 10μV/°C.
T(cobre/constantán)	Resisten atmósferas húmedas y oxidantes, e incluso tienen aplicación en criogénia. Su rango de temperatura esta entre -200°C a 350°C. Tiene una sensibilidad de 43μV/°C.
S(Pt/Rh)	Resistentes en atmósferas oxidantes, fácil de contaminar. Rango de temperatura de 0°C a 1450°C, es utilizado para calibración universal del punto de fusión del oro (1064.43°C). Tiene una sensibilidad de 10μV/°C.

Tabla 1.2: Algunos tipos de Termopar

Debido a sus características y a su costo accesible, dentro del presente trabajo se hará uso de termopares **tipo K**.

1.6. Objetivos

1.6.1. Objetivo General

En el presente trabajo se persigue dar una solución accesible, sencilla y versátil de un sistema controlador de temperatura para la automatización de calefactores eléctricos, para los cuales habrán varias alternativas de calentamiento y control de enfriamiento. Estableceremos los requerimientos más generales de control de calentamiento y enfriamiento mediante rampas con distintas pendientes para estos calefactores.

1.6.2. Objetivos Particulares

Para atracar nuestro objetivos general, se hará mediante los siguientes objetivos particulares dividido en dos bloques.

1: Bloque de Control

Diseñar un controlador de temperatura mediante *Software* y *Hardware* de Arduino junto con la medición de temperatura mediante termopares tipo K. Se contempla que nuestro controlador reciba las temperaturas del tratamiento térmico segundo a segundo mediante un archivo de texto almacenado en una memoria micro-SD.

2: Bloque de Potencia

Diseñar circuito RC el cual se conectará un calefactor de uso general, dicho circuito deberá modular el paso de corriente y voltaje hacia el calefactor.

Ambos bloques, control y potencia, están aislados eléctricamente aunque acoplados ópticamente mediante un LED y una foto-resistencia, de tal manera que todos los elementos cierran el lazo del sistema protegiendo la electrónica de Arduino, así como del usuario. A continuación se presenta en la figura 1.7 lo anteriormente explicado.

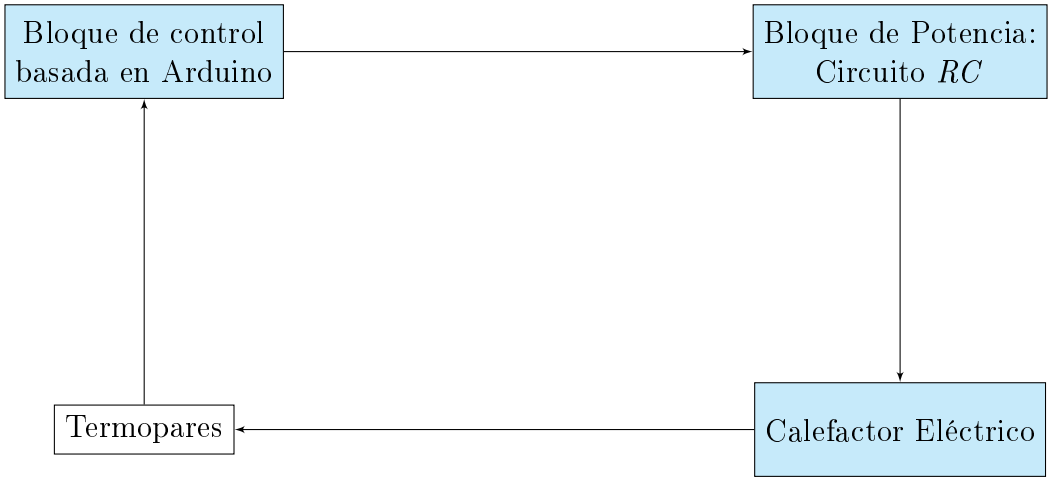

Figura 1.7: Diagrama de bloques del sistema propuesto como controlador de temperatura

1.7. Hipótesis

Es factible la construcción de un sistema controlador de temperatura acoplado a un calefactor que satisfaga los requerimientos mínimos necesarios para la realización de tratamientos térmicos basado en Arduino. Dicho controlador será de bajo costo y sencillo de usar.

CAPÍTULO 2

DESARROLLO EXPERIMENTAL

DESARROLLO EXPERIMENTAL

2.1. CIRCUITOS

2.1.1. ArduinoUNO

Para familiarizarnos con el ambiente y funcionamineto de Arduino se usó la placa ArduinoUNO. Se procedió como sigue:

- *Pantalla LCD*

Se requiere de una interfaz para mostrar datos al usuario, se eligió una pantalla de 16×2 caracteres, la cual es controlable mediante 6 pines digitales para la escritura de mensajes y 2 de alimentación. Adicionalmente se utilizó un potenciometro de 5 kΩ para modular el brillo de la pantalla. Se muestra a continuación (fig. 2.8) el esquema de las conexiones de los pines:

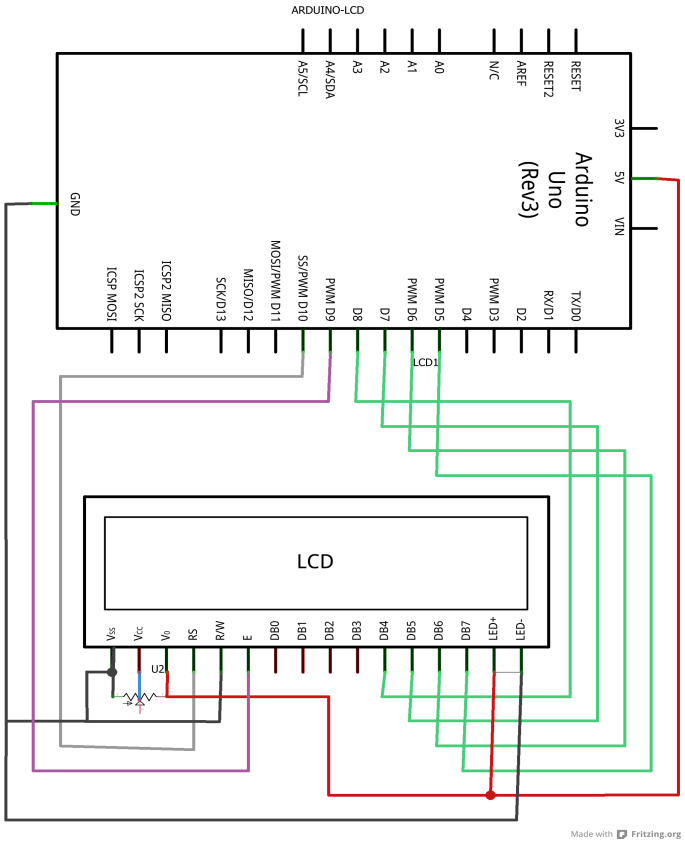

Figura 2.8: Esquema ArduinoUNO y Pantalla LCD

El código que hace funcionar a ArduinoUNO y a todos sus demás elementos es el Código *ArduinoUNO* localizado en la sección de CÓDIGOS.

► *Memoria micro-SD*

Se requiere de un dispositivo de lectura *y/o* almacenamiento de datos para algún experimento a realizar, dicho dispositivo debe contar con la suficiente capacidad de almacenamiento para experimentos prolongados, además debe de ser de fácil acceso. El dispositivo que se usó es un módulo adaptador de memorias micro-SD compatible con Arduino. Se muestra a continuación (fig. 2.9) el diagrama de las conexiones.

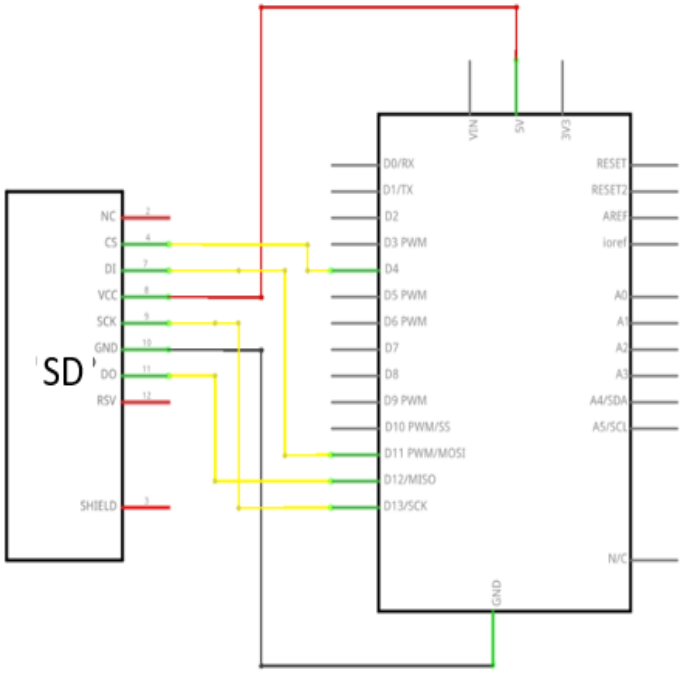

Figura 2.9: Esquema de ArduinoUNO y Adaptador de memoria MicroSD

► *Módulo de Reloj en Tiempo Real (RTC)*

Además de la lectura y almacenamiento datos que definen al tratamiento térmico, también es necesario llevar un registro del tiempo durante el tratamiento térmico, en el cual se muestre la hora exacta en que se están almacenando los datos del

tratamiento térmico en el dispositivo anterior, por lo que se necesita un reloj para llevar esta tarea. Arduino tiene un reloj integrado, el cual inicia un conteo a intervalos regulares en cuanto éste se enciende, además tiene la opción de programar un reloj que lleve la hora exacta pero perderá en cuanto Arduino se desconecte. Para solucionar este problema se propone utilizar un Módulo RTC independiente de la alimentación de Arduino y compatible él para la hora y fecha de realización del tratamiento térmico deseado. A continuación se muestra el diagrama (fig. 2.10) de las conexiones del RTC con ArduinoUNO:

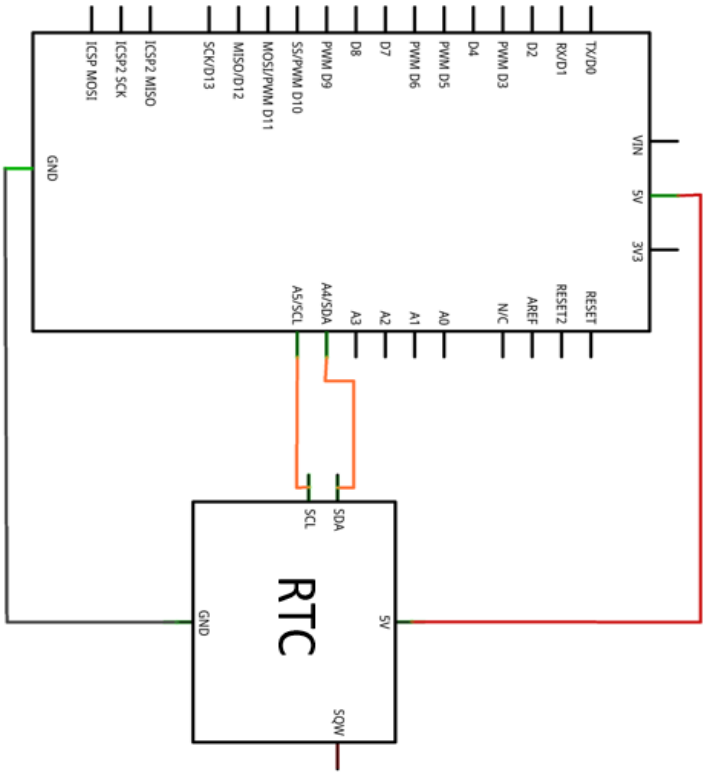

Figura 2.10: Esquema de ArduinoUNO y Módulo RTC

► *Termopares y Módulo Interfaz*

Por último, se requieren dos termopares **tipo K**, uno para el control de temperatura el cual estará acoplado al calefactor y otro para el monitoreo de la temperatura de la muestra.

El módulo interfaz está basado en el *convertidor analógico-digital MAX6675* de 12 bits. Este módulo está diseñado para trabajar con micro-controladores externos, dispositivos inteligentes o aplicaciones de monitoreo. El módulo incluye *hardaware* de acondicionamiento para convertir la señal de los termopares en un voltaje compatible con los *canales de entrada* del convertidor analógico-digital (ADC, por sus siglas en inglés).

Este módulo cuenta con dos entradas específicas para el termopar, llamadas $T+$ y T-; correspondientes a las polaridades positiva y negativa del termopar respectivamente, estas entradas están conectadas en un circuito interno que compensa la temperatura ambiente en las medidas de los termopares debido a la diferencia de temperatura entre la junta fría del termopar; que está a temperatura ambiente, y la *referencia virtual* de 0°C. Posteriormente este *ADC* convierte los voltajes termo-eléctricos en un valor equivalente de temperatura.

Recordemos que la función de un termopar es el censo de la diferencia entre los dos extremos de este. El extremo donde se mide el voltaje es conocido como unión caliente, por consiguiente la unión caliente de un termopar tipo K con esta interfaz tiene el mismo rango mostrado en la tabla 1.2, mientras que la unión fría (extremo a temperatura ambiente) únicamente puede leer en el rango de -20°C a +85°C; mientras la temperatura en el extremo frío fluctúa, la interfaz *MAX6675* continúa con un censo preciso de la temperatura en el extremo opuesto.

Dicho lo anterior, se muestra el diagrama (fig. 2.11) de las conexiones de dos módulos interfaz y sus respectivos termopares.

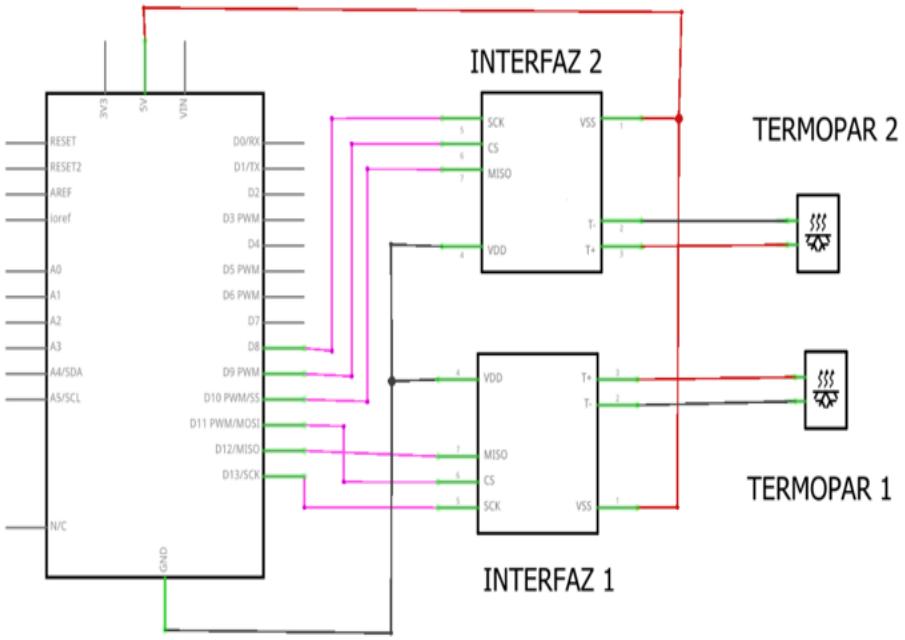

Figura 2.11: Esquema de ArduinoUNO junto con los MAX6675 y los termopares

A continuación, se muestra la figura y esquema (fig. 2.12 y fig. 2.13, respectivamente) de ArduinoUNO con las conexiones con estos cuatro componentes, a saber son: RTC, micro-SD, LCD y 2 interfaz MAX6675 con un termopar cada uno. Cabe mencionar que no se mantuvo el orden y los pines de las conexiones anteriormente expuestas, debido a 2 razones importantes:

- El número de pines de ArduinoUNO es pequeño y algunos elementos tuvieron que ponerse en paralelo, como los módulos MAX6675.
- Algunos componentes usan exclusivamente algunos pines de ArduinoUNO, por lo cual ese pin "queda apartado", como es el ejemplo de la memoria micro-SD que tiene como pin "base" el pin digital 4, dos pines analógicos A4 y A5 del reloj.

Dadas las consideraciones anteriores, el circuito final se muestran a continuación.

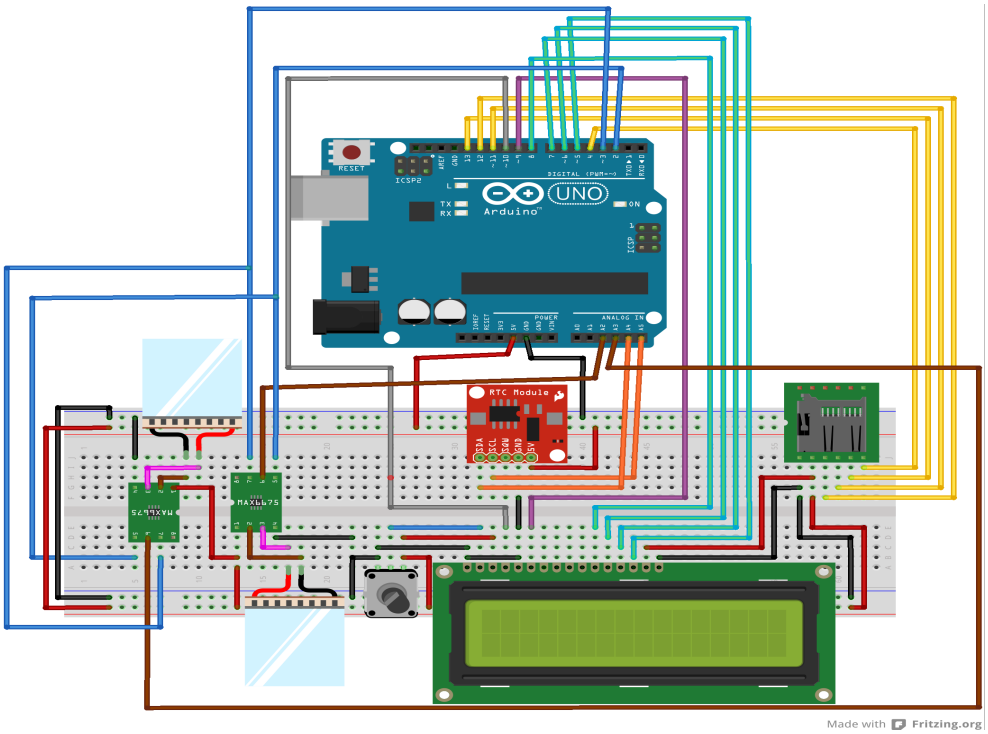

Figura 2.12: Conexiones de ArduinoUNO con RTC, Micro-SD, LCD y 2 Termopares

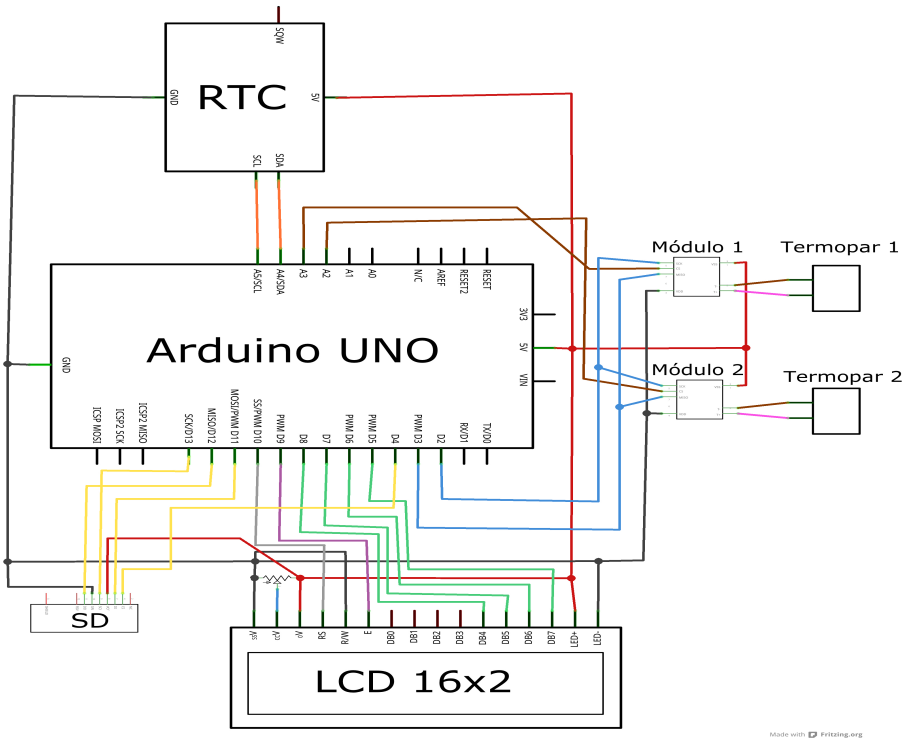

Figura 2.13: Esquema de conexiones de RTC, Micro-SD, 2 termopares y LCD con ArduinoUNO

Cabe resaltar que para llevar esta parte a la práctica los pines de ArduinoUNO son insuficientes, debido a que la propuesta de este proyecto requiere pines que trabajen con PWM. En ArduinoUNO estos pines son: 3, 5, 6, 9, 10 y 11, los cuales, como muestran las figuras y diagramas anteriores están ocupados, debido a estas razones muy importantes se optó por la tarjeta ArduinoMega, lo cual no incrementa significativamente el costo.

2.1.2. ArduinoMega

El traslado de los componentes a la placa ArduinoMega es sencilla, ya que todo el trabajo de la implementación y funcionamiento de cada uno de los componentes a usar se hizo en la placa ArduinoUNO; es decir, sólo se tiene que redefinir los pines usados por cada componente, teniendo cuidado de los pines especiales dedicados a la memoria micro-SD y el reloj, además de evitar usar los pines digitales que trabajan con PWM, ya que su fin es modular la señal al bloque de potencia; cabe mencionar que el trabajo realizado para el *software* en ArduinoUNO es reutilizable para la tarjeta ArduinoMega lo cuál es una característica muy importante y beneficiosa de esta plataforma.

El diagrama y esquema (fig. 2.14 y fig. 2.15, respectivamente) de los componentes y conexiones con la placa de ArduinoMega.

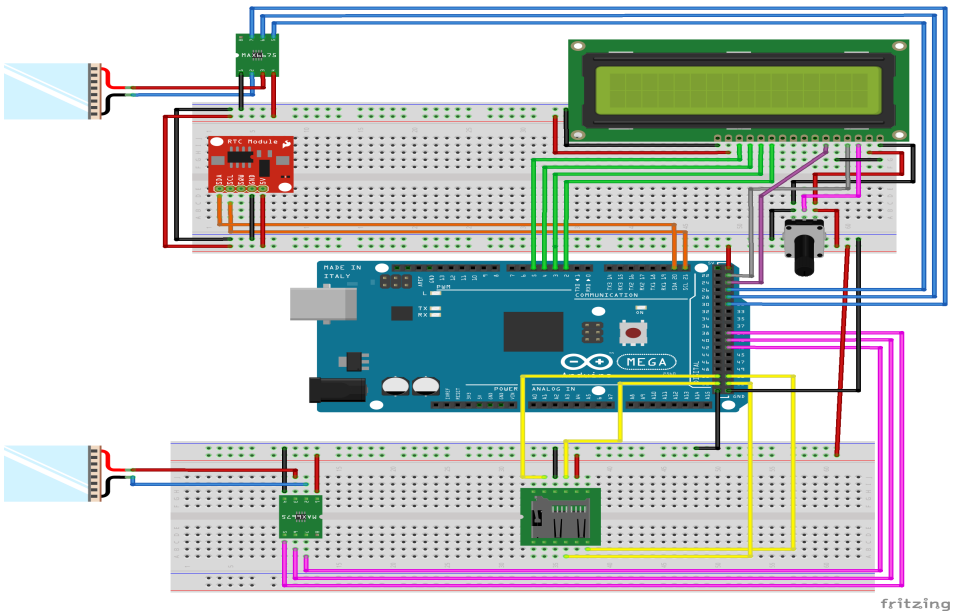

Figura 2.14: Conexiones de todos los elementos a utilizar en la placa ArduinoMega

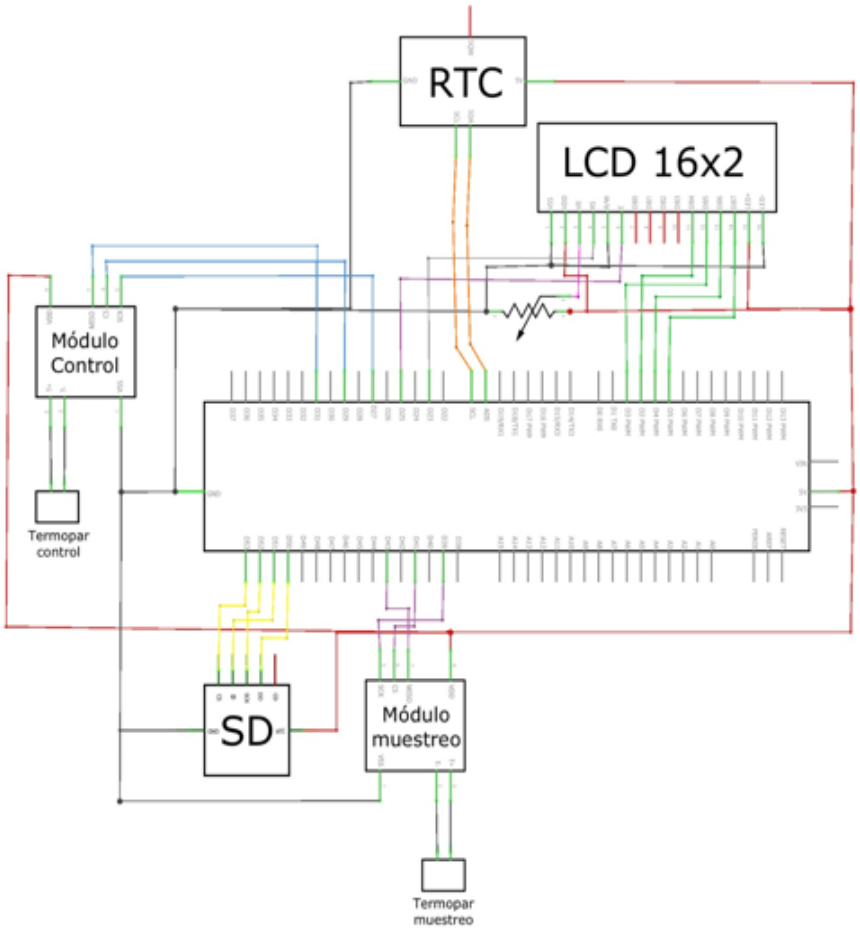

Figura 2.15: Esquema de las conexiones de la figura anterior

2.2. CONTROL DE POTENCIA Y PWM

Como se mencionó anteriormente, debemos lograr que Arduino sea capaz de controlar la potencia entregada al calefactor en función de la temperatura que registre el termopar acoplado al calefactor, ésta potencia se pretende manipular mediante la PWM.

Utilizando la PWM de Arduino; como primera aproximación al control de potencia para el calefactor, se trabajó sobre el control de brillo de un foco de 25 Watts. Para lograrlo se construyó un circuito *RC* como el presentado en la referencia [**?**], el cuál

en términos generales controlará el paso de corriente entre el suministro de electricidad (línea eléctrica) y el foco a manera de atenuador. La importancia de este circuito radica en su control por niveles del paso de corriente alterna (CA) hacia el foco, así como el aislamiento eléctrico entre la toma de corriente y Arduino. A continuación se explica más a detalle dicho circuito.

Dentro del bloque de potencia tenemos un circuito RC formado por la resistencia R y los condensadores C_1 y C_2, formando un *oscilador de relajación* con el DIAC y el GATE del TRIAC, sincronizado con la línea ($\pm$120 V, 60 Hz), de tal manera que en cada cruce por cero se inicia un ciclo del oscilador en ambas polaridades. Al ocurrir la descarga de los condensadores a través del GATE del TRIAC, éste empieza a conducir y permanece en este estado de conducción hasta el siguiente cruce por cero de la línea. De este modo, el oscilador actúa como un elemento de retardo, el cual puede ser desde un tiempo muy pequeño (25 V en una senoide de 160 V, con lo que el TRIAC conducirá durante la mayor parte del semi-ciclo), hasta tiempos mayores o iguales a medio periodo, es decir 1/120 Hz = 83.3 ms, con lo cual el TRIAC no conducirá en ningún momento del ciclo. A continuación se presenta una fotografía de este circuito (fig. 2.16).

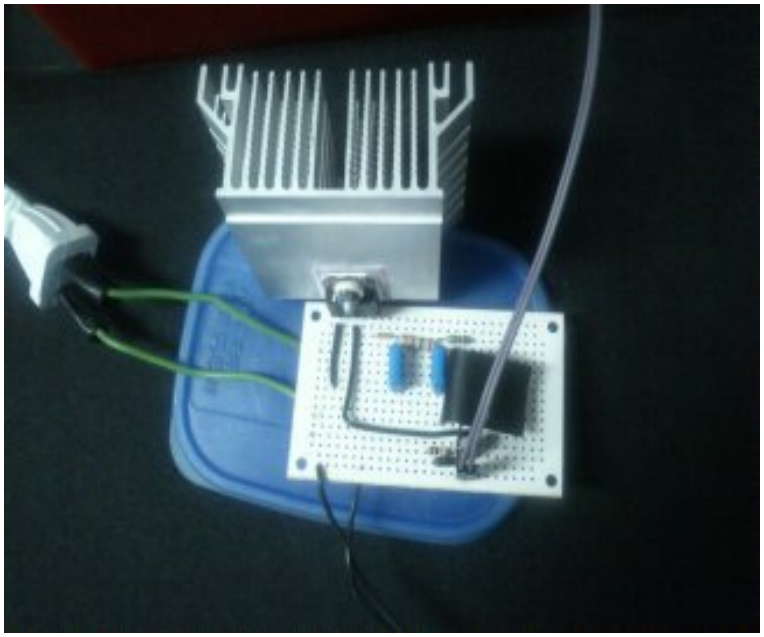

Figura 2.16: Circuito Controlador de Potencia

Cuyo diagrama se presenta a continuación (fig. 2.17).

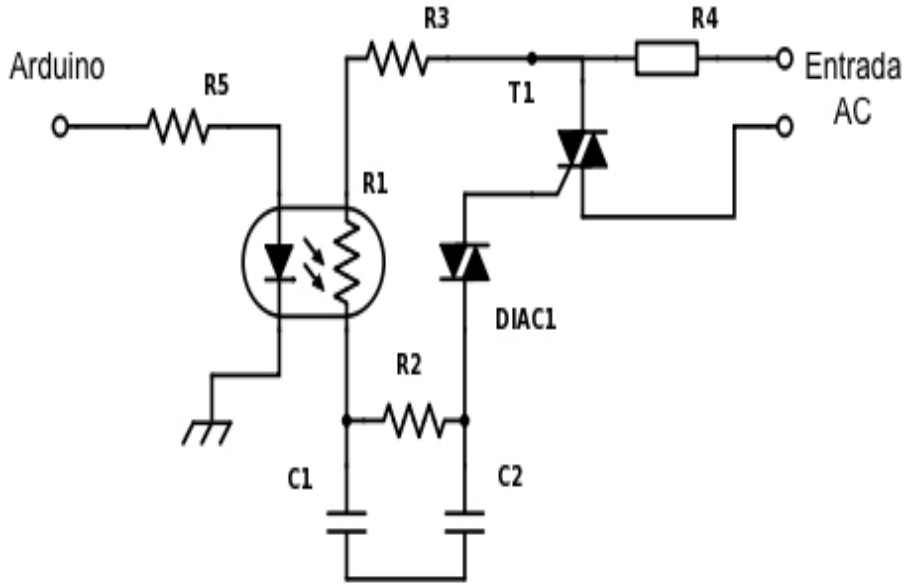

Figura 2.17: Diagrama Circuito Controlador de Potencia

2.2.1. Parrilla Eléctrica y Horno

Después de haber probado el circuito con el foco y corroborando que no había interferencia entre el módulo de potencia y Arduino, se procedió a una siguiente fase, la cual es fundamental para la meta de este proyecto, *el control de una parrilla eléctrica.* Para el control de la parrilla eléctrica, se usó el mismo circuito que la figura anterior, también se adicionó un lata de aluminio con aceite de cocina a modo de carga térmica y emulando la muestra del tratamiento térmico, teniendo cuidado de no rebasar la temperatura de ebullición del aceite, la cual oscila entre 180°C– 260°C.

Para este esquema Arduino-parrilla; se utilizaron dos termopares, uno de control y uno de monitoreo; el termopar de control está colocado en la resistencia de la parrilla, aislado eléctricamente con una mica de silicio resistente a altas temperaturas, aproximadamente 500°C, además a este termopar se le virtió grasa de silicio para mejorar el

acoplamiento térmico entre éste y la parrilla, posteriormente la lata con aceite se colocó encima, de tal manera que el termopar quedará entre la parrilla y la lata; el termopar de monitoreo se sumergió en el aceite dentro de la lata. A continuación en la fig. 2.18 el sistema experimental anteriormente descrito.

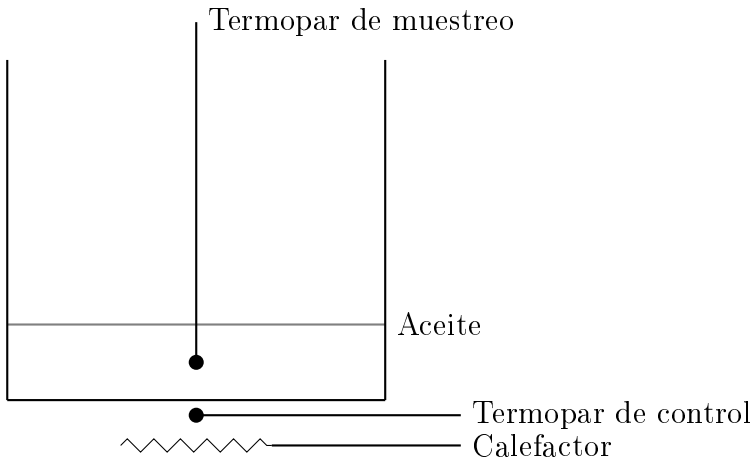

Figura 2.18: Sistema parrilla

Posteriormente el calefactor se conectó al controlador. Para el arreglo experimental anterior se hicieron las pruebas de control, se notó demasiada disipación térmica, pero aún así se obtuvo un control de temperatura aceptable en el aceite dentro de la lata, esta primera experiencia de control de temperatura se hizo subiendo drásticamente la temperatura a partir de la ambiente y manteniendola a 150°C.

Este dispositivo permitió detectar e implementar parámetros para mejorar la estabilidad en el control de temperatura. Alguno de ellos fueron:

- Tiempo de muestreo:

  Cada iteración del ciclo `loop()` toma un cierto tiempo Δt para ejecutarse. Dentro de esta iteración se toma un dato de temperatura y con él se calcula el valor de la salida PWM, además de leer la temperatura de referencia de la memoria y escribir en ella los datos del tratamiento térmico, de modo que el ciclo de control tendrá un periodo mínimo Δt, el cual se mide dentro del programa para ajustar el tiempo de muestreo-control, mediante la función `delay()` a modo de tener el número de muestras por segundo deseadas. Encontramos que el valor

óptimo de t_1 es de 250ms, por lo que se le dio éste como argumento a la función `delay()`, el tiempo de retraso t_2 para dicha función esta dada por:

$$t_2 = (t_1 - \Delta t) \tag{2.1}$$

ó bien

$$t_2 = (250\text{ms} - \Delta t) \tag{2.2}$$

- Distribución de las tareas dentro del ciclo de programación:

  El parámetro t_2 se ajustó para tener 4 ciclos por segundo para optimizar el flujo de datos a y desde la memoria micro-SD, asi como la lectura del termopar de control y el cálculo del valor PWM para el módulo de control, se distribuyeron estas tareas de la siguiente forma:

  1 s:

Iteración 1	Toma de datos de los termopares
Iteración 2	Lectura de datos de temperatura del tratamiento térmico de la memoria micro-SD
Iteración 3	Escritura de datos de hora y temperatura a la memoria micro-SD
Iteración 4	Presentación de temperaturas y tiempo en la pantalla LCD

  Cabe mencionar que en cada una de las 4 iteraciones se realiza el ciclo de control.

- Función de control:

  La potencia que se le entrega al calefactor debe ser función de la diferencia entre la temperatura medida por el termopar de control (en el calefactor) y la temperatura de referencia (temperatura a la que se quiere llegar); ΔT. Esta función debe tomar un valor constante correspondiente al 100 % de la potencia para diferencias grandes de temperatura y decrecer rápidamente para

diferencias pequeñas, por lo que se propuso la siguiente función:

$$P(\Delta T) = 255.0\left(1 - \frac{1}{k\Delta T + 1}\right) \tag{2.3}$$

donde:

$P(\Delta T)$	Es la potencia en función de la diferencia de temperaturas.
255.0	El valor máximo de la función `analogWrite()` con el que puede trabajar la PWM.
k	Es una constante entre 0 y 1, que dependerá del horno a usar.

A continuación se presenta la forma que se espera obtener con la función de potencia (fig.2.19).

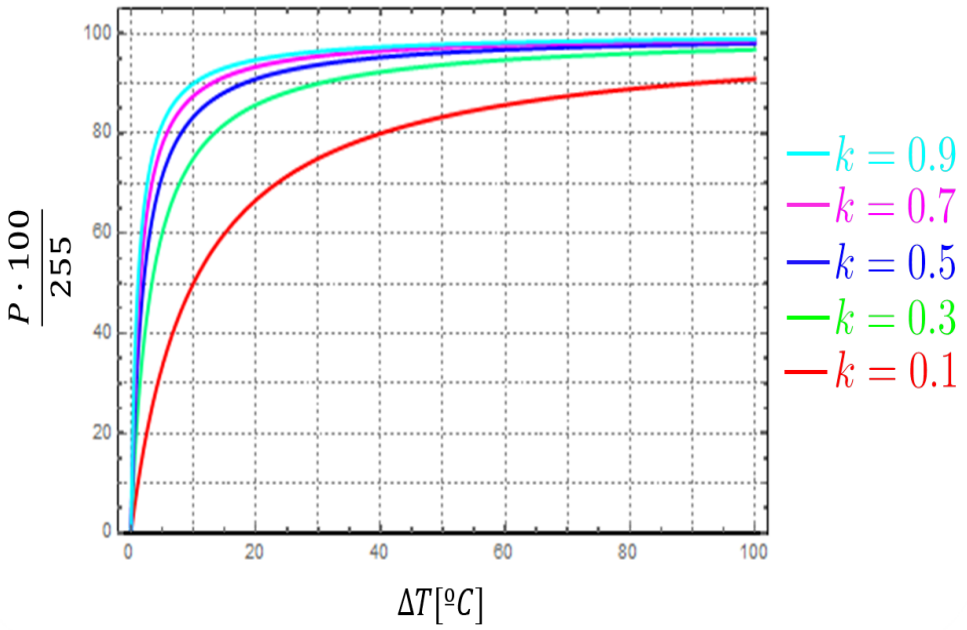

Figura 2.19: Gráficas de la función de control variando k

Analizando las gráficas anteriores y considerando el armado de nuestro sistema controlador de temperatura y la inercia térmica del calefactor de resistencia, se tomó el valor de ***k=0.1***.

2.3. Programa Edit-Term

Como parte del trabajo de Tesis, se elaboró el programa *Edit-Term*[5] el cual facilita enormemente el diseño y edición del tratamiento térmico mediante una interfaz gráfica, como la que se presenta a continuación (fig. 2.20).

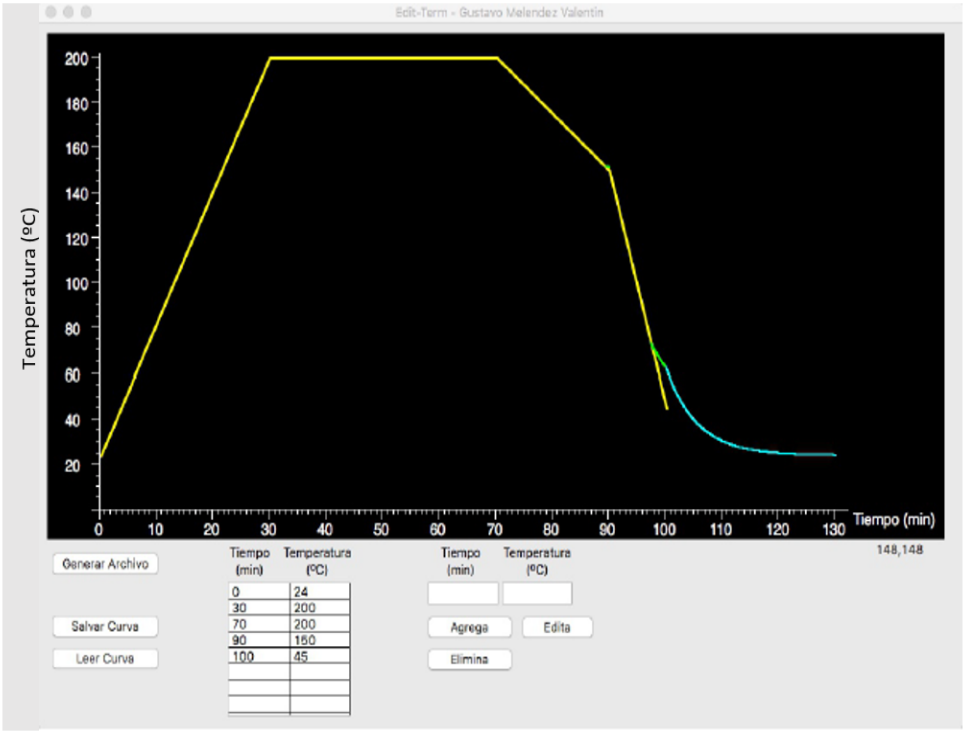

Figura 2.20: Captura de pantalla del programa *Edit-Term.*

El modo de operar este programa es sencillo y se explicará a continuación. El usuario puede introducir, editar o eliminar los puntos que forman la curva de *temperatura a seguir*, ya sea seleccionándolos con el cursor del *mouse*, o capturando los valores en los campos indicados como se muestra en la figura anterior. El programa muestra la gráfica ***temperatura vs tiempo*** que deberá de seguir el bloque de control para el tratamiento térmico; es decir, el gráfico de control está resaltado por el color amarillo, el gráfico en color azul nos indicará la curva de enfriamiento cuando se apaga el calefactor y se deja enfriar (a temperatura ambiente). Cabe mencionar que dentro de la figura de

[5]El programa se implementó en los lenguajes de programación *Python y Xojo*, para asegurar que el programa *Edt-Term* funcionará en de igual manera en los sistemas operativos Windows y Mac

tratamiento térmico existe un fragmento de rampa de color verde, esta parte se refiere a que el control (rampa) de enfriamiento es imposible de seguir, debido a que implicaría enfriar más rápido que dejando al ambiente, entonces el programa mostrará en este color la curva aproximada que seguirá el proceso.

Una vez obtenido el tratamiento térmico deseado, el usuario debe de dar *click* en el botón *Generar Archivo* y seleccionar la ruta de la memoria *microSD*, de este modo se grabará el archivo de tratabjo(tratamiento térmico) llamado por default ***TT01.TXT***, que será reconocido por Arduino dentro del bloque de control. Posteriormente, el usuario puede leer o salvar la lista de puntos del tratamiento térmico en su propio formato *.cvs ó .txt* el cuál es reconocido por diversos editores de hoja de cálculo como Excel.

CAPÍTULO 3

RESULTADOS

Y

CONCLUSIONES

RESULTADOS

3.1. Control de Temperatura

Con el controlador de temperatura (fig. 3.21), se obtuvieron los siguientes resultados:

Figura 3.21: Dispositivo Experimental Arduino-parrilla

Como primera prueba, se hizo un experimento para el termopar de control, iniciando con temperatura ambiente de 22°C y temperatura final de 150°C, con un tiempo de muestreo de 1 segundo durante aproximadamente 90 minutos. A continuación se muestra la gráfica (fig. 3.22) de este primer experimento.

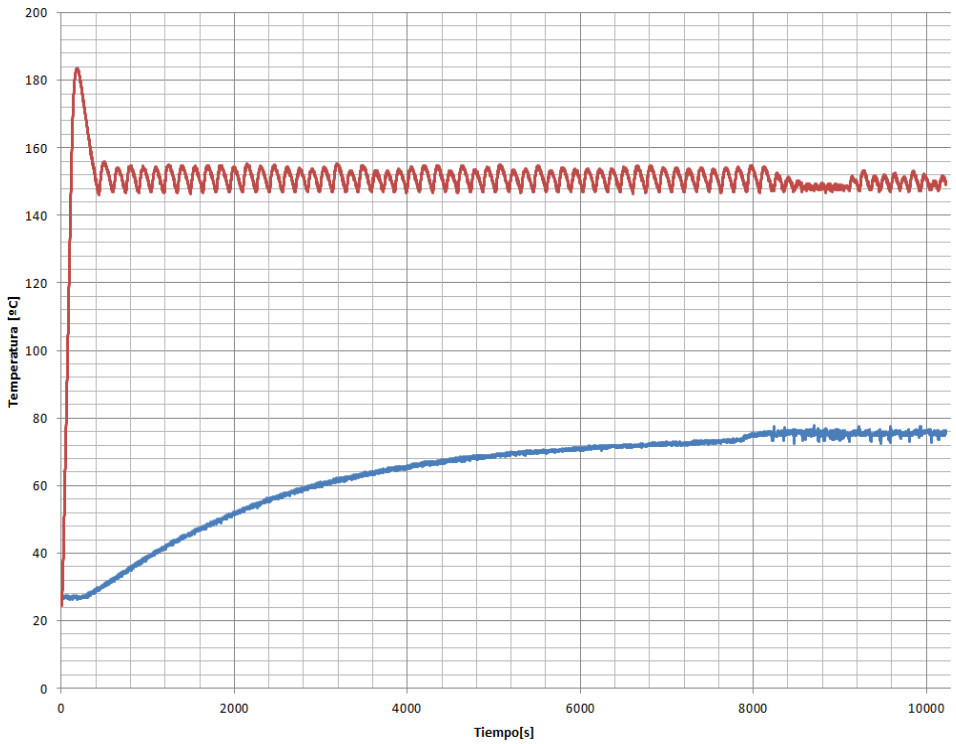

Figura 3.22: Controlador de temperatura a 150°C

En la gráfica anterior, se muestra en rojo la temperatura registrada por el termopar de control y en azul la temperatura registrada por el termopar de muestreo. Notemos que particularmente en la gráfica del termopar de control, sí se obtiene una temperatura estable de 150°C 5 minutos después de haber iniciado el tratamiento térmico con una oscilación de $\pm$ 3°C; notemos que antes de los 5 minutos hay un pico de temperatura de + 30°C debido a la inercia térmica, el cual posteriormente se atenúa como se explicó anteriormente. Por otro lado, en el termopar de muestreo, la temperatura registrada por éste no se ve afectada por las fluctuaciones en la temperatura del calefactor, lo cual se ve claramente en la suavidad de la gráfica. Con estos primeros resultados, se puede notar que la propuesta de nuestro controlador de temperatura es viable, aunque es necesario optimizar algunos parámetros para obtener mejores resultados, como por ejemplo, no tener un pico muy grande al inicio del tratamiento térmico y obtener una rampa de enfriamiento.

Para mejorar los resultados anteriores, se usó nuestra propuesta del programa *Edit-Term*, para diseñar un tratamiento térmico que consta de una rampa lineal a 200°C, con

una pendiente de $\boldsymbol{0.0983}$ °C/$\boldsymbol{s}$, se mantuvo esta temperatura y posteriormente se añadió una rampa de enfriamiento con pendiente de $\boldsymbol{-0.042}$ °C/$\boldsymbol{s}$ (para la cual se tuvieron las consideraciones expuestas en la sección 2.3). Cabe mencionar que la duración del tratamiento térmico nuevamente fue de 90 minutos y con un tiempo de muestreo-control en las medidas y cálculo de PWM de 250 mili-segundos, como se explicó en la sección 2.2.2. A continuación se presentan los resultados (fig. 3.23) obtenidos del experimento.

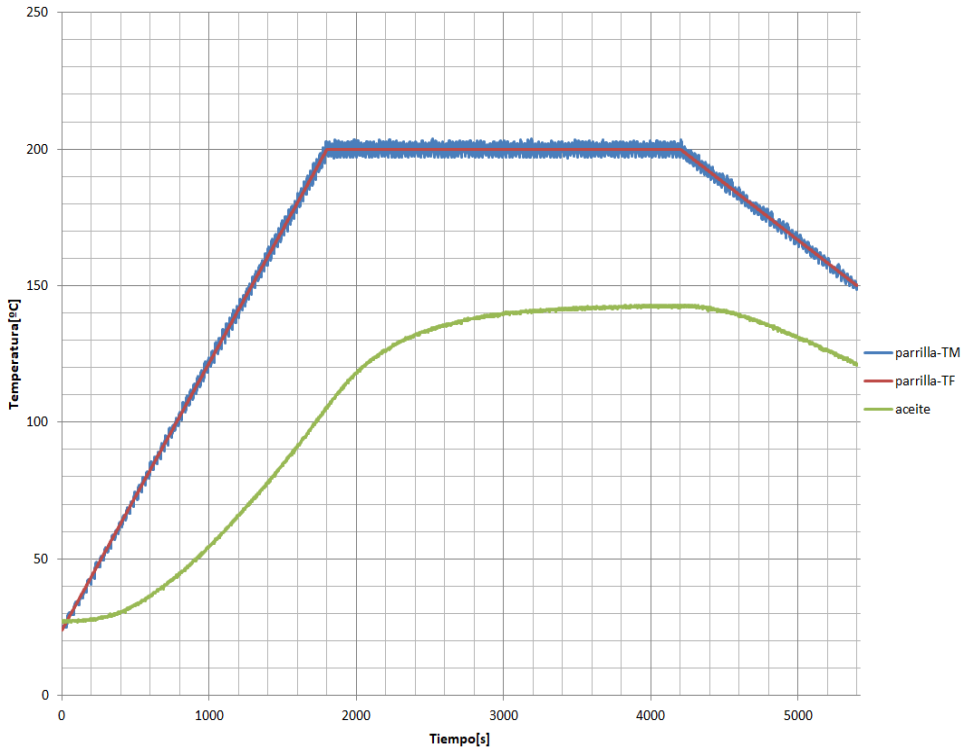

Figura 3.23: Controlador de temperatura a 200°C.

La gráfica roja corresponde a la generada por el programa *Edit-Term*, la gráfica azul a la temperatura registrada por el termopar de control y finalmente la gráfica verde a la temperatura de la muestra.

Notemos que dentro de este segundo experimento los resultados obtenidos son lo que se esperaba, solucionamos el pico de temperatura debido a la inercia térmica del sistema calefactor-muestra reduciendo el tiempo de muestreo a 250 mili-segundos, el cual encontramos fue el óptimo para nuestro arreglo experimental. Respecto al termopar de control se obtuvo una oscilación en la temperatura de $\pm$ 1.5°C. La gráfica de color verde, son las temperatura registradas por el termopar de muestras, nuevámente notamos que las fluctuaciones de temperatura del calefactor no se ven reflejadas en la temperatura de la muestra, en este caso, aceite.

Los resultados de este último experimento son muy buenos, pero había un pequeño detalle, el experimento iniciaba en cuanto se alimentaba a Arduino, lo cual podría entorpecer al tratamiento térmico. Para solucionar este pequeño detalle se añadió un botón y dos LED's, cuya función es que al ser oprimido se inicia el tratamiento térmico. Al terminar el tratamiento térmico uno de los LED's se apaga, pero aún el Arduino seguirá en funcionamiento en cuanto a la escritura de datos de tiempo y temperatura. Para detener la escritura de datos se oprime por segunda vez el botón y con esto se apagará el segundo LED. Con estas pequeñas modificaciones, se tiene más control sobre el inicio y final del tratamiento térmico. A continuación se muestra la gráfica (fig. 3.24) de las mejoras del experimento anterior.

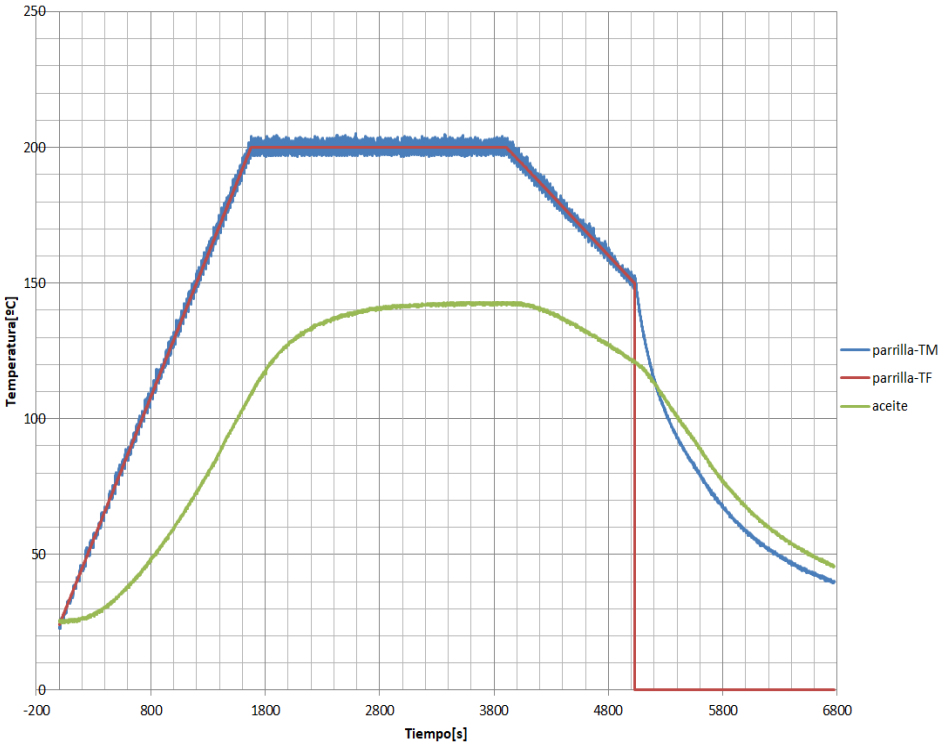

Figura 3.24: Controlador de temperatura a 200°C

Básicamente se obtuvieron los mismos resultados que la gráfica anterior, mostrando además los datos registrados por ambos termopares después de concluir el tratamiento térmico y la escritura de datos, notemos que la forma de la gráfica del termopar de control es muy similar a la mostrada en la interfaz *Edit-Term*.

Notemos que la forma presentada en el programa *Edit-Term* para el tratamiento térmico no es **única**; es decir, con este controlador de temperatura, se podrá reproducir un tratamiento térmico casi con cualquier forma. A continuación se presenta un tratamiento térmico en forma sinusoidal como el de la figura 3.25, en la cual la temperatura sigue la función:

$$T = 150 + 50 \sin\left(\frac{2\pi t}{90}\right) \tag{3.4}$$

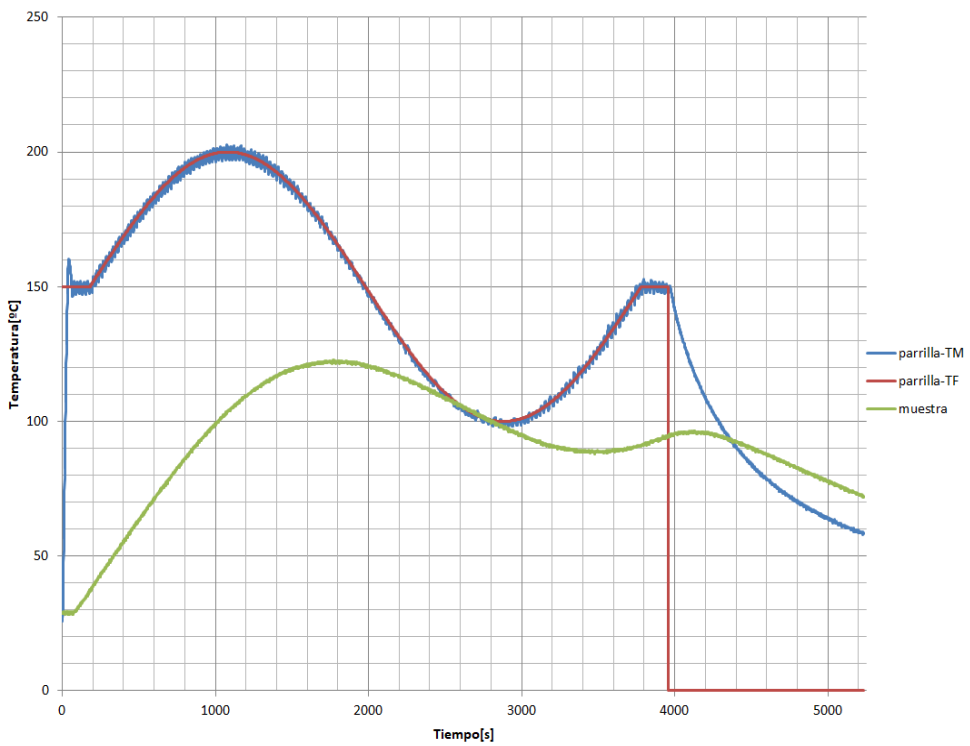

Figura 3.25: Control en forma sinusoidal alrededor de 150°C.

Notemos que el ciclo sinusoidal, se completa en 90 minutos. Los resultado obtenidos por el control de temperatura en este último experimento fueron satisfactorios.

3.2. Conclusiones

Dados los resultados de nuestros experimentos podemos concluir lo siguiente:

- La propuesta para la automatización de hornos dentro de este proyecto fue exitosa ya que se logró un control automático de la temperatura con la tarjeta Arduino.

- El *Software* y *Hardware* de Arduino resultaron ser accesibles, tanto de costo como en flexibilidad de uso para los fines deseados. El costo neto máximo fue menos de $ 48.00 USD.

- Se construyó un sistema de control de temperatura con las siguientes características:
  - Programable mediante *Software* propio para Windows/Mac.
  - Capacidad para controlar calefactores de resistencias, con una potencia de hasta 2 kWatts (de acuerdo a características del triac).
  - Precisión en el control de temperatura de $\pm$ 1.5°C.
  - Capacidad para grabar en la memoria micro-SD los resultados del tratamiento térmico para revisión posterior en PC.
- La implementación del circuito *RC* para la etapa de potencia fue fundamental para regular el paso de corriente y voltaje específico para el calefactor. Además el aislamiento eléctrico con el que fue diseñado permitió la protección de los componentes del bloque de control, así como del usuario.

- El uso del Programa *Edit-Term* en Arduino para la elaboración de tratamientos térmicos en una interfaz gráfica, resultó muy amigable para el usuario.

- Esta propuesta de controlador de temperatura para la automatización de calefactores, tiene la ventaja que es actualizable en los siguientes aspectos:

* Se puede mejorar el programa *Edit-Term* para la creación de los tratamientos térmicos.
* Al ser independiente, el programa *Edit-Term* puede ser actualizado, incorporando mejoras en la interfaz gráfica, así como en funcionamiento.
* El programa interno de Arduino puede actualizarse, adaptándose para una placa, horno ó calefactor específico.
* Pueden agregarse sensores, botones, displays, etc. al tener fácil acceso al *Hardware* para incorporar aplicaciones futuras.

APÉNDICES

APÉNDICES

4.1. Micro-controlador AVR

Los micro-controladores **AVR** son desarrollados por ATMEL desde 1996. Este tipo de micro-controladores están basados en una arquitectura de 8 bits en un solo chip, AVR fue uno de los primero es utilizar el chip de memoria flash para el almacenamiento de programas. Los micro-controladores AVR encuentran como aplicación en sistemas *incrustados*(es un sistema computarizado con funciones especificas dentro de un sistema mecánico o eléctrico más grande, que a menudo tiene limitaciones en tiempo real, este se incrusta como parte de de un dispositivo completo a menudo incluyendo *hardware* y partes mecánicas), como aplicación directa podríamos mencionar a **Arduino**.

Por otro lado la memoria Flash es un tipo de memoria informática basada en semiconductores, no volátil y reescribible. Esto significa que posee muchas de las características de la memoria RAM, excepto que sus datos no se eliminan al apagarse el dispositivo. La memoria Flash almacena porciones de datos en las celdas de memoria, pero esos datos permanecen almacenados aunque se produzca un corte de energía.

Debido a su alta velocidad, durabilidad y bajo consumo de energía, la memoria flash resulta ideal para muchos usos, como por ejemplo en cámaras digitales, teléfonos móviles, impresoras, PDA, ordenadores laptop, Arduino y dispositivos que puedan almacenar y reproducir sonido, como los reproductores de MP3. Además, este tipo de memoria no tiene partes móviles, lo que la hace más resistente a eventuales golpes. A continuación se muestra en la figura 4.26, señalando en 1 el micro-controladores AVR y memoria flash en ArduinoMega.

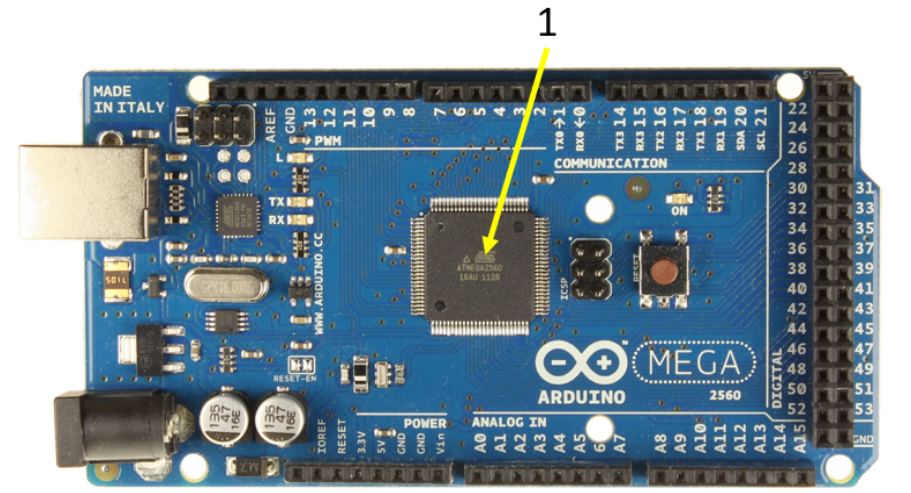

Figura 4.26: En 1 es ejemplo de Micro-controlador AVR y memoria flash para ArduinoMega

Por otro lado cabe mencionar que un micro-controlador, en términos generales, es un circuito integrado que en su interior contiene una unidad central de procesamiento(CPU), unidades de memoria (RAM y ROM), puertos de entrada/salida y periféricos. Estas partes están interconectadas dentro del mismo, y en todo lo anterior en conjunto forman lo que se le conoce como micro-computadora.

4.2. Transformaciones de Fase

Las transformaciones de fase, para los tratamientos térmicos, consisten en sólo transformar la micro-estructura, que la compone. Dichos micro-componentes varían de acuerdo con el tipo de tratamiento a utilizar, material utilizado, procesos de fabricación y algunos otros a los cuales el material haya sido sometido. Para las aleaciones(principalmente de aceros y metales) los principales constituyentes son:

- **Ferrita(α)**: Consiste en átomos de hierro con estructura cristalina *BBO* y átomos de carbono en los sitios huecos que existen entre los átomos de hierro. La cantidad de átomos de carbono en la ferrita es pequeña. La ferrita es una fase muy suave, dúctil y magnética.

- **Austenita(γ)**: Generalmente consiste en átomos de hierro con estructura FCC y átomos de carbono en los sitios huecos. Presenta menor suavidad y ductibilidad que la ferrita. Esta es una fase no magnética.

- **Cementita**: También es llamado carburo de hierro. Es un compuesto íntermetálico (son sólidos que que contienen dos o más elementos metálicos, con opcionalmente uno o más elementos no metálicos, cuya estructura cristalina se diferencia de los otros constituyentes), es además, el constituyente más duro de los aceros, por lo que, no es posible utilizarla para operaciones de laminado debido a su dificultad para ajustarse a las concentraciones de esfuerzos.

- **Perlita**: Es un constituyente compuesto aproximadamente de ferrita y cementita. Su micro-estructura está formada por láminas o capas alternas a las dos fases durante el enfriamiento lento de un acero a temperatura eutectoide (es la temperatura a la cual un sólido se enfría y se transforma en otros dos sólidos distintos entre si.

4.3. Técnicas de Tratamiento Térmico

Existen varios tipos de tratamiento térmicos, pero en general sólo son derivados de tres tipos de tratamiento los cuales presentaremos a continuación en una forma general:

- **Normalizado**: En este tipo de tratamiento térmico se calienta el material hasta la austenita completa y se deja enfriar al aire. Tiene como fin dejar al material con la estructura y propiedades arbitrariamente consideradas como normales. Con este tratamiento se borran cualquier otro tratamiento anterior al que se haya sometido la pieza, elimina tensiones, corrige enfriamientos irregulares y sobre-calentados.

- **Recocido**: Es un tratamiento térmico que normalmente consiste en en calentar un material a temperatura muy elevada durante un largo periodo de tiempo, con el fin de disminuir la densidad de dislocaciones, y de esta manera, se logra una mejor ductibilidad.
  El recocido se realiza normalmente para:

  - Alterar la estructura del material para obtener las propiedades mecánicas deseadas, ablandando el material y mejorando su maquinabilidad.

  - Re-cristalizar las aleaciones(principalmente metales) trabajados en frío.

  - Para aliviar los esfuerzos residuales.

- **Temple**: El temple es un tratamiento térmico que tiene por objetivo aumentar la dureza y resistencia mecánica del material, transformando toda la masa de este en *austenita* con el calentamiento y después, por medio de un enfriamiento brusco (con aceite o agua), se convierte en martensita, que es el constituyente duro de los acero templado por ejemplo. Cabe mencionar que en el temple es muy importante la fase de enfriamiento y la alta velocidad del mismo, además, la temperatura de calentamiento debe ser siempre superior a la crítica para obtener la martensita. Existen varios tipos de temple los cuales se obtiene según la característica deseada

y la templeabilidad (capacidad de penetración del temple) que a su vez depende del diámetro o espesor del material.

- **Revenido**: Es un tratamiento térmico complementario al temple, generalmente prosigue de éste. En general después del temple, en las aleaciones (por ejemplo los aceros) suelen quedar demasiado duros y frágiles para los usos que están destinados, lo anterior se puede corregir mediante el revenido, que disminuye la dureza y la fragilidad excesiva, sin perder tenacidad.

A continuación se muestra un diagrama de estos tipos de tratamientos térmicos.

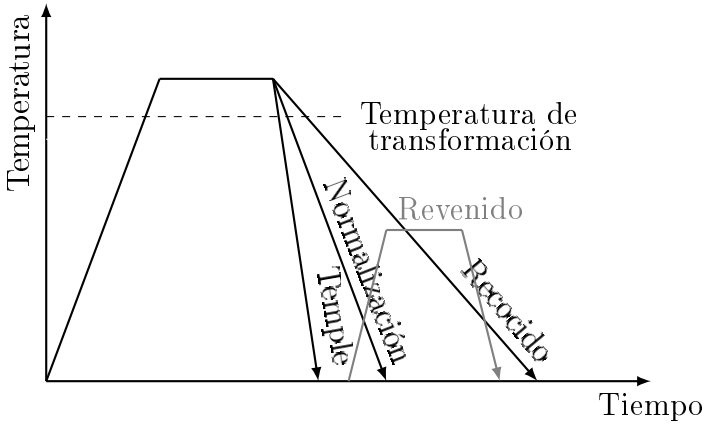

Figura 4.27

4.4. Efecto Seebek

Este efecto fue descubierto por el físico Thomas Johan Seebeck en 1821. Realizó investigaciones intentando establecer conexión entre calor y electricidad, el modo en que descubrió este efecto fue uniendo una lámina de cobre con otra de bismuto, en un circuito cerrado. Notó que al calentar una de las uniones se genera una flujo de voltaje en el circuito en tanto persista la diferencia de temperatura; es decir, el voltaje producido es debido a la diferencia de temperatura que existe entre la unión de dos materiales distintos. Este tipo de objeto compuesto por dos alambres distintos unidos es un extremo es conocido como **termopar**. Se muestra a continuación la figura 4.28 de este arreglo para hacer un análisis más a fondo.

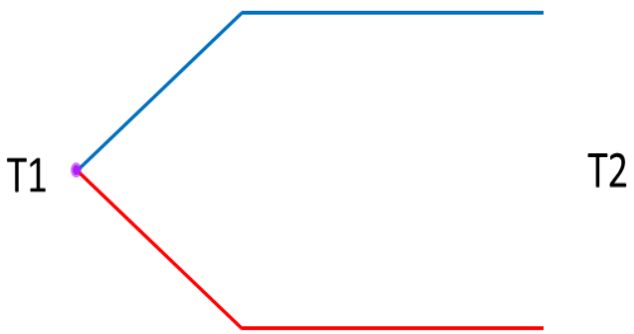

Figura 4.28: Efecto Seebeck y Termopar

Todo lo anteriormente mencionado, se puede expresar dentro de una fórmula matemática, que es la siguiente:

$$\Delta V = -\alpha \Delta T \tag{4.5}$$

donde

ΔV: es el voltaje medido en los extremos libres (voltaje en T2)

ΔT: es la diferencia de temperatura entre la unión (T1) y los extremos libres (T2)

α: es una constante de proporcionalidad. Es conocida como coeficiente *Seebeck*

El coeficiente Seebeck, puede tener signo positivo o negativo, y su valor numérico

varia según los materiales que se estén ocupando, cabe mencionar, que dicho coeficiente es función de la temperatura.

MANUAL DE USO

Para la creación, edición y desarrollo del Tratamiento Térmico siga las instrucciones.

- Instalado en el ordenador el Software Xojo para posteriormente ejecutar y correr el archivo ***Edit-Term.xojo*** para la creación del Tratamiento térmico deseado.
- Dentro del programa *Edit-Term*, figura 2.20, podemos identificar (de derecha a izquierda) los siguientes elementos:
    * Dos campos en blanco para llenar, en el primero se colocará el tiempo (en minutos), en el segundo se colocará la temperatura deseada que corresponderá al tiempo anterior. Estos pares Tiempo-Temperatura se llenarán uno a uno. Abajo de los campos anteriores, se tienen tres botones. El primer botón *agrega* puntos al tratamiento térmico, en el formato anterior. El segundo botón *edita*, en caso de requerirlo, los puntos del tratamiento térmico. Por último, el botón *elimina* aquellos puntos seleccionados que se agregaron por error dentro del tratamiento térmico.
    * Una tabla que lleva por encabezado, *Tiempo (min)* y *Temperatura (°C)*, en ella se visualizan los puntos necesarios para crear rampas de calentamiento y control de enfriamiento, así como las mesetas de temperatura constante. Cabe mencionar que en la parte de control de enfriamiento, la pendiente de las rampas creadas no deben de exceder la velocidad de enfriamiento dada por la ***Ley de Enfriamiento de Newton***. En el caso de que se exceda esta velocidad, el programa indicará el enfriamiento aproximado que se seguirá.
    * Por último, el botón *Genera Archivo* arroja todos los puntos del tratamiento térmico creados con los apartados anteriores en formato **.txt** y lleva por nombre ***TT01.TXT***. Éste se debe almacenar en la memoria micro-SD que se insertará en el controlador. El botón *Salvar Curva* sirve para guardar en

la PC un tratamiento térmico predeterminado que se realice con frecuencia o para editarlo posteriormente. El botón *Leer Curva* sirve para que el programa *Edit-Term* lea punto a punto los datos (en formato *.txt*) de algún tratamiento térmico creado por éste o algún otro medio distinto a este Software como Excel.

Cabe mencionar que dicha creación y edición de rampas del tratamiento térmico se representan instantáneamente en la interfaz gráfica se visualiza la forma que tomará el tratamiento.

- Como se mencionó anteriormente, el archivo generado por el programa *Edit-Term* se colocará en la memoria micro-SD, la cual se introducirá en la ranura del controlador.

- Al prender el controlador se encenderán dos LED (amarillo y verde) y aparecerá en la pantalla LCD el nombre del archivo de entrada del tratamiento y justo abajo de él aparecerá el nombre del archivo de salida de la toma de datos de los termopares y reloj, este archivo estará en formato *.TXT* y llevará por nombre la fecha en que se realizó el tratamiento térmico en forma *DDMMAAAA (día, mes, año)*. El tratamiento térmico comenzará al presionar el botón, con lo cual el LED verde se apagará y un LED verde indicará que el tratamiento térmico está en proceso. Una vez terminado el tratamiento térmico, el LED verde se apagará y presionando nuevamente el botón indicará la **extracción segura** de la memoria micro-SD con el archivo que contiene los datos de tiempo y temperatura de ambos termopares registrados por el controlador. Este archivo puede ser abierto y analizado (en caso de requerirlo) con algún software, ya sea *Excel, Origin, Mathematica, etc.* para corroborar que dicho tratamiento no tuvo algún tipo percance dentro de su elaboración.

CÓDIGOS

CÓDIGO ARDUINOUNO

El código que hace funcionar a la pantalla LCD, reloj y tarjeta de momoria microSD es el siguiente:

Código Arduino UNO

```
   noindent 01 #include <Arduino.h>
02 #include <Wire.h>
03 #include "RTClib.h"
04 #include <LiquidCrystal.h>
05 #include <SPI.h>
06 #include <SD.h>
07 #include "max6675.h"
08 RTC_DS1307 rtc;
09 LiquidCrystal lcd(10, 9, 8, 7, 6, 5);
10 File myFile1,myFile2;
11 String MuestraHora(void);
12 int ktcSO1 = 3;
13 int ktcCS1 = A3;
14 int ktcCLK1 = 2;
15 int ktcSO2 = 3;
16 int ktcCS2 = A2;
17 int ktcCLK2 = 2;
18 MAX6675 ktc1(ktcCLK1, ktcCS1, ktcSO1);
19 MAX6675 ktc2(ktcCLK2, ktcCS2, ktcSO2);
20 //-------------------------//
21 void setup()
22 {
23  lcd.begin(16, 2);
24 rtc.begin();
25 rtc.adjust(DateTime(F(__DATE__),F(__TIME__)));
26 if (!SD.begin(4))
27   {
28   }
29    myFile1 = SD.open("parametr.txt", FILE_READ);
30   if (myFile1)
31   {
32     while (myFile1.available()) {
33       Serial.write(myFile1.read());
34     }
35     myFile1.close();
36   }
```

```
37 else{
38   }
39
40 myFile2 = SD.open("datos.txt", FILE_WRITE);
41 if (myFile2) {
42    myFile2.println("Medida  tTermopar1  tTermopar2  tFecha");
43    myFile2.close();
44   }
45 else{
46   }
47
48 }
49 //----------------------------//
50 void loop()
51 {
52    if (millis()%1000==0){
53    Serial.println(MuestraHora());
54    myFile2 = SD.open("datos.txt", FILE_WRITE);
55    myFile2.print(millis()/1000);
56    myFile2.print("\t t");
57    myFile2.print(ktc1.readCelsius());
58    myFile2.print("\t\t");
59    myFile2.print(ktc2.readCelsius());
60    myFile2.print("\t\t");
61    if (Hora()<10) myFile2.print("0");
62    myFile2.print(Hora()); myFile2.print(":");
63    if (Min()<10) myFile2.print("0");
64    myFile2.print(Min()); myFile2.print(":");
65    if (Seg()<10) myFile2.print("0");
66    myFile2.println(Seg());
67    myFile2.close();
68
69   lcd.setCursor(0,1);
70   lcd.print("T1:");
71   lcd.print(ktc1.readCelsius(),1);
72   lcd.setCursor(8,1);
73   lcd.print("T2:");
74   lcd.print(ktc2.readCelsius(),1);
75    }
76 }
77 //------------------------//
78 String MuestraHora(void)
79 {
80   char cuerda[30];
81     DateTime now;
82     now=rtc.now();
83     sprintf(cuerda,"%02d:%02d:%02d",now.hour(),now.minute(),now.second());
84     lcd.setCursor(0,0);
85     lcd.print(cuerda);
86     return(cuerda);
87 }
88 int Hora(void)
89 {
90     int h;
91     DateTime now;
92     now=rtc.now();
93     h=now.hour();
94     return(h);
95 }
96 int Min(void)
97 {
98   int m;
99   DateTime now;
100   now=rtc.now();
101   m=now.minute();
```

```
102   return(m);
103 }
104 int Seg(void)
105 {
106   int s;
107   DateTime now;
108   now=rtc.now();
109   s=now.second();
110   return(s);
111 }
```

CÓDIGO ARDUINOMEGA

El código que hace funcionar Arduino Mega con los elementos: pantalla LCD, reloj, tres termopares y tarjeta de momoria microSD es el siguiente:

Código Arduino Mega

```
#include <Arduino.h>
#include "max6675.h"
#include <math.h>
#include <Wire.h>
#include "RTClib.h"
#include <SPI.h>
#include <SD.h>
#include <LiquidCrystal.h>
LiquidCrystal lcd(23,25,2,3,4,5);
RTC_DS1307 rtc;
File archivo1,archivo2; //archivo1=lectura de datos y archivo2=escritura de datos
int ktcSO1 = 31;
int ktcCS1 = 29;
int ktcCLK1 = 27;
int ktcSO3 = 43;
int ktcCS3 = 41;
int ktcCLK3 = 39;
int F;
unsigned long N;
const int PINTT =8;
const int PINSD= 9;
int PWM=11;
float pulso,TI,Periodo,TC,TM,DT,TS;//TC=temperatura calefactor, TS=temperatura
a seguir, TM=temperatura muestra
float t=0.0;
int Bot = 7;
String LL;
boolean Continua=true,Graba=true,Inicio=true;
String Nombre;
MAX6675 ktc1(ktcCLK1, ktcCS1, ktcSO1);
MAX6675 ktc3(ktcCLK3, ktcCS3, ktcSO3);
void setup()
{
  pinMode(Bot,INPUT);
  lcd.begin(16,2);
  rtc.begin();
```

```
36   rtc.adjust(DateTime(F(__DATE__),F(__TIME__)));
37   pinMode(PINTT , OUTPUT);//definir pin como salida
38   digitalWrite(PINTT , HIGH); // poner el Pin en HIGH
39   pinMode(PINSD,OUTPUT);
40   pinMode(13,OUTPUT);
41   digitalWrite(PINSD,HIGH);
42
43 /*****************************************************************/
44 //Creación y renombramiento de archivo .txt para no escribir siempre en el mismo
45   if (!SD.begin(53)) {
46     return;
47   }
48   Nombre = MuestraFecha();
49
50    int I=1;
51   while(SD.exists(Nombre)){
52     Nombre=Nombre.substring(0,5);
53     Nombre.concat(I);
54     Nombre.concat(".txt");
55     I++;
56     }
57 /*****************************************************************/
58   archivo1 = SD.open("TT01.TXT");
59   /*****************************************************************/
60   archivo2=SD.open(Nombre, FILE_WRITE);
61   if (archivo2)
62   {
63       archivo2.println(‘TS TC TM Pulso Hora Tiempo‘);
64      }
65   else {
66   }
67 /*************************************************************************/
68  TI=(ktc1.readCelsius() -4.9);
69  Periodo = 250.0;
70  F = int(1000.0/Periodo);
71  N = 0;
72  t = 0.0;
73              lcd.setCursor(0,0); lcd.print("IN: "); lcd.print("TT01");
74              lcd.setCursor(0,2); lcd.print("OUT:"); lcd.print(Nombre);
75      while(Inicio){
76         int val0=digitalRead(Bot);
77           if  (val0==HIGH) Inicio=false;
78           }
79           while(digitalRead(Bot)){}
80
81 lcd.clear();
82  }
83 void loop()
84 {
85   int T1=millis();
86   int val=digitalRead(Bot);
87   if  (val==LOW) Graba=HIGH;
88   if  (val==HIGH && Graba){
89   Graba=LOW;
90   archivo2.close();
91   digitalWrite(PINSD , LOW);
93   }
94    if (N% F==0 && Continua){
95     TS = LeeLinea(archivo1,1);
96     if (TS<0.0){
97       Continua = false;
98       archivo1.close();
99       TS = 0.0;
100       digitalWrite(PINTT , LOW);
102     }
```

```
    }
/*************************************************************************/
  TC=(ktc1.readCelsius()-4.9);
  TM=(ktc3.readCelsius()); //TMP1 = Temperatura Medida Muestra 1
  DT=TS-TC;

  //Funcion de control
  pulso=(DT<=0)? 0.0 : 255.0*(1-(1/(0.1*DT+1)));
  if (pulso>=255.0) pulso = 255.0;
  if (pulso<=15.0) pulso = 15.0;
  analogWrite(PWM,int(pulso));
/*************************************************************************/
  if (N%F==2 && Graba){
 // archivo2.print(N);
// archivo2.print(‘\t ‘);
  archivo2.print(TS);
  archivo2.print("\t");
  archivo2.print(TC);
  archivo2.print("\t");
  archivo2.print(TM);
  archivo2.print("\t");
  archivo2.print(pulso);
  archivo2.print("\t");
  if (Hora()<10) archivo2.print("0");
  archivo2.print(Hora()); archivo2.print(":");
  if (Min()<10) archivo2.print("0");
  archivo2.print(Min()); archivo2.print(":");
  if (Seg()<10) archivo2.print("0");
  archivo2.print(Seg());
  archivo2.print("\t");
  archivo2.println(t);
  //myFile2.close();
  digitalWrite(13 , HIGH);
  }
  if (N%F==1){
  lcd.setCursor(9,0);
  lcd.print("TS:");
  lcd.print(TS,1);
  lcd.setCursor(0,2);
  lcd.print("TM:");
  lcd.print(TM,1);
  lcd.setCursor(8,2);
  lcd.print("TC:");
  lcd.print(TC);
  digitalWrite(13 , LOW);
  }
 N+=1;
 t=N*Periodo/1000.0;
 int T2=millis()-T1;
 delay(Periodo-T2);
}
/*************************************************************/
String MuestraHora(void)
{
  char cuerda[30];
    DateTime now;
    now=rtc.now();
    sprintf(cuerda,"%02d:%02d:%02d",now.hour(),now.minute(),now.second());
    lcd.setCursor(0,0);
    lcd.print(cuerda);
    return(cuerda);
}
int Hora(void)
{
    int h;
```

```
168     DateTime now;
169     now=rtc.now();
170     h=now.hour();
171     return(h);
172 }
173 int Min(void)
174 {
175   int m;
176   DateTime now;
177   now=rtc.now();
178   m=now.minute();
179   return(m);
180 }
181 int Seg(void)
182 {
183   int s;
184   DateTime now;
185   now=rtc.now();
186   s=now.second();
187   return(s);
188 }
189 /*****************************************************************/
190 String MuestraFecha(void)
191 {
192   char cuerda[10];
193   String cuerdaS;
194     DateTime now;
195     now=rtc.now();
196     sprintf(cuerda,"%02d%02d",now.day(),now.month());
197     cuerdaS=String(cuerda);
198     cuerdaS.concat("T.txt");
199     return(cuerdaS);
200 }
201 int Dia(void)
202 {
203     int d;
204     DateTime now;
205     now=rtc.now();
206     d=now.day();
207     return(d);
208 }
209 int Mes(void)
210 {
211   int m;
212   DateTime now;
213   now=rtc.now();
214   m=now.month();
215   return(m);
216 }
217 int Anio(void)
218 {
219   int a;
220   DateTime now;
221   now=rtc.now();
222   a=now.year();
223   return(a);
224 }
225 /*****************************************************************/
226 float LeeLinea(File archivo,int n){
227   String linea,linea_;
228   int t1;
229   char cr = 0;
230     while(cr!='\n'){
231     linea.concat(cr);
232     cr = archivo.read();
```

```
233   }
234
235   linea.concat(',');
236   for (int i=0;i<=n;i++){
237   t1=0; while(linea[t1]!=',') t1++;
238   linea_=linea.substring(0,t1);
239   linea = linea.substring(t1+1,linea.length());
240   }
241   linea_.replace(',',' ');
242   return(linea_.toFloat());
243   }
244 float Token(String linea,int n){
245   int t1;
246   String linea_;
249   linea.concat(',');
250   for (int i=0;i<=n;i++){
251   t1=0; while(linea[t1]!=',') t1++;
252   linea_=linea.substring(0,t1);
253   linea = linea.substring(t1+1,linea.length());
254   }
255   linea_.replace(',',' ');
256   return(linea_.toFloat());
257   }
```

BIBLIOGRAFÍA

[1] Raúl Arturo Espejel Morales. *Estudio de la irradiación del SiO2 y Au y sus efectos fotoluminiscentes y de absorción óptica.* Facultad de Ciencias, U.N.A.M., 1998.

[2] M.E. Perez Reyes and Sosa Morales M.E. Mecanismos de transferencia de calor que ocurren en tratamientos térmicos de alimentos. *Revista Temas Selectos de Ingeniería eb Alimetnos (TSIA)*, 7(1):37–47, 2013.

[3] Lopéz Longas J., J. Canut, R. Ríos, and J.A. Puértolas. Caracterización termomécanica de la aleación Ni-Ti para aplicaciones en medicina. *Biomécanica*, 6(11):73–80, 1998.

[4] Gerardo D. Lopez. Biodeterioro y corroción de implantes y prótesis e implantes. *Revista Medicina(Buenos Aires)*, 53(3):260–274, 1993.

[5] Georgina Fernández Villagómez and Patricia Torres Rivera. *Guía para la disposición sugura de medicamentos caducos acumulados en situaciones de emergencia.* 1ª Edición, Diciembre 2001.

[6] Cristian Michael Olin Pachari. *Diseño e implementación de un sistema para la evaluación del tratamiento térmico de alimentos envasados.* Escuela Profesional de Ingeniería Física, Facultad de Ciencias, Universidad Nacional de Ingeniería, 2015.

[7] Issac Salomón Jiménez Escamilla. *Control de temperatura de un horno eléctrico mediante lógica difusa.* Universidad Tecnológica de la Mixteca, Junio 2012.

[8] Enrique Cabrera, Raúl Espejel, and R. Toca. Sistema Controlador de Temperatura. *Revista Mexicana de Física*, pages 413–420, 1977.

[9] Escuela Comlombiana de Ingeniería "Julio Garavito". *Tratamientos Térmicos Protocolo.* 2008.

[10] http://www.tool dynamics.com/literature/articles/004.php.

[11] M.I. Felipe Díaz del Castillo Rodríguez and M.I. Alberto Reyes Solís. *Aceros, Estructuras y Tratamientos Térmicos.* Departamento Ingeniería, F.E.S. Cuatitlán, U.N.A.M., 2012.

[12] Universidad Nacional de Rosacio: Instituto Politécnico. *Tratamientos Térmicos.*

[13] Marco Antonio Alvarado Meza. *Estudio de Temperatura M_s y microestructura de un Acero Inoxidable Martensítico con 0.12%C mediante Análisis Térmico.* Facultad de Ingeniría Mecánica y Eléctica, División de Estudios de Posgrado, Universidad Autónoma de Nuevo León, Mayo 2010.

[14] Meléndez Gustavo. *Programación con Tecnología Arduino.* 2016.

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

[16] Evans Bryan W. *Arduino Notebook: A Beginner's Reference.* 2007.

[17] Patricia Martínez and Marcelo Azuaga. *Calibración de una Termocupla de Chromel-Alumel.* Laboratorio IV, Depto. Física, UBA, 1997.

[18] Guadalupe Evaristo Cedillo Garza. *Medición de Temperaturas con Termopares.* Facultad de Ingeniría Mecánica y Eléctica, División de Estudios de Posgrado, Universidad Autónoma de Nuevo León, Diciembre 1979.

[19] Francisco Attilio Sarmiento Mendoza. *Espectroscopio para Termoluminiscencia.* Facultad de Ciencias, U.N.A.M., 2015.