Universidad Continental

FACULTAD DE INGENIERÍA

Escuela Académico Profesional de Ingeniería Eléctrica

Tesis

Diseño de un sistema de control para el monitoreo de la temperatura de un motor de bomba de agua de 0.75 kW

Rubén Huaranga Camargo

Para optar el Título Profesional de
Ingeniero Electricista

Huancayo, 2020

ASESOR

MSc. Joel Colonio Llacua

AGRADECIMIENTO

A Dios, que me acompaña en cada instante de mi vida; a mis padres, que se esforzaron por educarme y nutrirme de valores; y a mis maestros.

DEDICATORIA

A mis padres, por la orientación que me dieron, para ir por el camino del bien. También a tu ayuda incondicional pues, a pesar de los momentos muy difíciles que pasamos, estuviste motivándome y ayudándome hasta donde tus fuerzas lo permitieron.

ÍNDICE

LISTA DE TABLAS

LISTA DE FIGURAS

RESUMEN

El presente trabajo de tesis tiene el propósito de diseñar un sistema de control para el monitoreo de la temperatura de un motor de bomba de agua de 0.75kW. Para el desarrollo, se planteó la siguiente metodología: el método de investigación es el científico, el nivel de investigación es el aplicativo y el tipo de investigación el tecnológico. Dentro del alcance, se planteó diseñar e implementar el sistema de control, lo cual llevó a plantear un diseño del tipo experimental; la muestra y la población de este trabajo es el motor de 075 Kw, comúnmente llamado electrobomba. Se llegó a presentar como resultado un diagrama de flujo del sistema de control, se seleccionó un adecuado sensor y actuador y se diseñó un software de interfaz entre el usuario y el sistema de control para lograr implementar dicho sistema y así monitorear la temperatura de un motor de bomba de agua de 0.75kW.

ABSTRACT

This thesis work is designed to design a control system for monitoring the temperature of a 0.75kW water pump motor. For the development the following methodology was proposed, the research method is the scientific method, the level of research is the application and the type of research is of the technological type. Within the scope it was proposed to design and implement the control system leading to the design of an experimental type design, the sample and the population of this work is the 075 Kw motor commonly called an electric pump. Arriving to present as a result a flow chart of the control system, selecting a suitable sensor and actuator and designing an interface software between the user and the control system and then implementing the control system to monitor the temperature of a motor 0.75kW water pump.

INTRODUCCIÓN

En la actualidad el uso de la electrobomba es necesaria para las familias, debido a la escasez del agua que afronta la región. A pesar de que estos tipos de motores de bomba de agua cuentan con un sistema de control para el encendido y apagado basado en los niveles del agua en los tanques y cisternas, el problema se presenta cuando estos sistemas de control no actúan, lo cual origina que los motores se malogren. Ante tal hecho, las familias deben adquirir otro motor. Por lo tanto, se planteó el DISEÑO DE UN SISTEMA DE CONTROL PARA EL MONITOREO DE LA TEMPERATURA DE UN MOTOR DE BOMBA DE AGUA DE 0.75 kW.

El contenido de la presente tesis se encuentra dado por cuatro capítulos. El primer capítulo presenta la descripción, planteamiento, formulación y objetivo del trabajo de investigación. El capítulo dos presenta el marco teórico con el cual se sustenta el diseño. El capítulo tres presenta la metodología de la investigación. El capítulo cuatro presenta los resultados obtenidos en el presente trabajo. Y, por último, se describen las conclusiones y recomendaciones.

El autor

CAPÍTULO I

PLANTEAMIENTO DEL ESTUDIO

1.1. PLANTEAMIENTO Y FORMULACIÓN DEL PROBLEMA

Actualmente en la industria existe muchos tipos de sistemas de control de la temperatura para motores, específicamente para los tipos de motores más comunes como son los de bomba de agua, los cuales son los que se utilizan con mayor frecuencia para realizar la tarea de bombear el agua en las viviendas. Pero estos tipos de motores no cuentan con un sistema de control de la temperatura y presentan el riesgo de averiarse al alcanzar temperaturas para las cuales no han sido diseñadas. A pesar de que estos tipos de motores de bomba de agua cuentan con un sistema de control para el encendido y apagado basado en los niveles del agua en los tanques y cisternas, la pregunta sería qué pasaría si estos sistemas de control no funcionen, provocando que los motores se malogren, como ocurre con muchas familias que deben adquirir otro motor.

Esta situación nos lleva a realizar nuestra investigación con el fin de diseñar un sistema de control para el monitoreo de la temperatura de un motor de bomba de agua de 0.75kW.

1.2. FORMULACIÓN DEL PROBLEMA.

1.2.1. PROBLEMA GENERAL.

¿Cómo diseñar un sistema de control, para el monitoreo de la temperatura de un motor de bomba de agua de 0.75kW?

1.2.2. PROBLEMAS ESPECÍFICOS.

- ¿Cómo desarrollar el controlador adecuado?
- ¿Qué sensor y el actuador elegir?
- ¿Cómo controlar y monitorear la temperatura?
- ¿Cómo verificar el funcionamiento del sistema de control?

1.3. OBJETIVOS.

1.3.1. OBJETIVO GENERAL.

Diseñar un sistema de control para el monitoreo de la temperatura de un motor de bomba de agua de 0.75kW.

1.3.2. OBJETIVOS ESPECÍFICOS.

- Desarrollar el controlador adecuado.
- Seleccionar el sensor y el actuador.
- Diseñar un software de control y monitoreo de la temperatura.
- Implementar el sistema de control.

1.4. JUSTIFICACIÓN E IMPORTANCIA.

1.4.1. JUSTIFICACIÓN TEÓRICA.

El avance de la tecnología permite desarrollar estos tipos de sistemas de control con conceptos teóricos que se pueden poner en práctica y este diseño puede aportar a la verificación de alguna teoría de control.

1.4.2. JUSTIFICACIÓN PRÁCTICA.

La implementación de nuestro sistema de control servirá como modelo para otros diseños orientados a los sistemas de control de motores comunes.

1.4.3. JUSTIFICACIÓN METODOLÓGICA.

Para el diseño del sistema de control se utilizó una metodología que serviría para diseñar otros sistemas de control para diferentes tipos de motores.

1.4.4. JUSTIFICACIÓN SOCIAL.

Actualmente, se tiene mucha demanda de estos motores de bomba de agua. Al presentar esta propuesta de diseño de un sistema de control, se estaría dando una opción de protección a estos motores, lo cual beneficiaría a las familias al ahorrar en la compra del reemplazo del motor.

1.4.5. IMPORTANCIA.

La importancia de la investigación se refleja en el diseño del prototipo para poder ser implementado y mejorar la protección de estos tipos de motores más usados.

1.5. HIPÓTESIS Y DESCRIPCIÓN DE VARÍALES.

1.5.1. HIPÓTESIS GENERAL.

Con el diseño un sistema de control se logrará monitorear la temperatura del motor de bomba de agua de 0.75kW.

1.5.2. HIPÓTESIS ESPECÍFICAS.

- Con el desarrollo, el controlador adecuado se logrará controlar la variable de proceso.
- Con la selección del sensor y el actuador, se logrará diseñar las etapas de sensado y actuación.
- Con el diseño de un software de control y monitoreo de la temperatura, se logrará el interfaz del usuario.
- Con la implementación del sistema de control, se verificará el funcionamiento.

1.6. DESCRIPCIÓN DE VARIABLES.

La Variable Independiente es el diseño de un sistema de control (X).

La Variable Dependiente es el monitoreo de la temperatura (Y).

Podemos definir que:

Y está en función de X

Por lo tanto:

El monitoreo de la temperatura está en función de diseño de un sistema de control

1.7. OPERACIONALIZACION DE VARIABLES.

Tabla 1: Operacionalizacion de variables.

VARIABLES		DEFINICIÓN CONCEPTUAL	DEFINICIÓN OPERATIVA	INDICADOR
Variable Independiente	Diseño de un sistema de control.	Es un esquema complejo que controlará la temperatura de la bobina de un motor.	Conjunto de componentes físicos relacionados, de manera que actúan por si solos.	Prototipo
Variable Dependiente	Monitoreo de la temperatura	Verificación de la temperatura de la bobina del motor	Revisión de la operación de la temperatura de la bobina del motor.	Software

Fuente: Propia

CAPÍTULO II
MARCO TEÓRICO

2.1. ANTECEDENTES DEL PROBLEMA

2.1.1. ANTECEDENTE INTERNACIONAL.

(Jimenez Escamilla, 2012). “Control De Temperatura De Un Horno Eléctrico Mediante Lógica Difusa”. En su resumen concluye lo siguiente:

“En la actualidad, muchos productos requieren algún tratamiento térmico y más cuando se habla de la creación de piezas cerámicas, por lo que resulta de vital importancia el control de la temperatura. En esta tesis se presenta la forma en la cual se diseñó y construyó un sistema para controlar la temperatura de un horno eléctrico para la cocción de piezas cerámicas. Se empleó un controlador difuso de tipo Mamdani, el cual trabaja con las entradas del error (temperatura deseada menos temperatura real), el cambio del error, y provee una salida de voltaje, la cual será la señal de control para un microcontrolador que realizará un control por encendido apagado mediante ciclos completos de la línea de alimentación de voltaje que nos proporciona CFE. El software del sistema y el controlador se desarrollaron en LabVIEW, permitiendo visualizar en un entorno amigable al usuario, la temperatura deseada y real, así como un entorno agradable para el usuario. La digitalización de la temperatura y el envío de la señal de control se realizan con una tarjeta de adquisición de datos. Con la implementación del sistema de control, se mejoró la calidad de las piezas y el tiempo de cocción, además de facilitar el uso del horno eléctrico”.

2.1.2. ANTECEDENTE NACIONAL.

(López de Paz, 2016). “Diseño Del Sistema De Control De Temperatura De Un Invernadero”. En su resumen describe lo siguiente:

“El objetivo de la tesis se centra en el diseño de un sistema de control de temperatura para un invernadero localizado en el departamento de Ancash, provincia de Bolognesi, distrito de Abelardo Pardo Lezameta.

Los cultivos a tratarse son el palto, la chirimoya, la lúcuma y el pino que serán distribuidos en el invernadero ya mencionado.

Con el control de la temperatura de los cultivos se busca óptimo crecimiento (calidad del producto) ya que se construye un microclima adecuado (condiciones de temperatura y humedad). Para ello es necesario el uso de tecnologías aplicadas a la agricultura.”

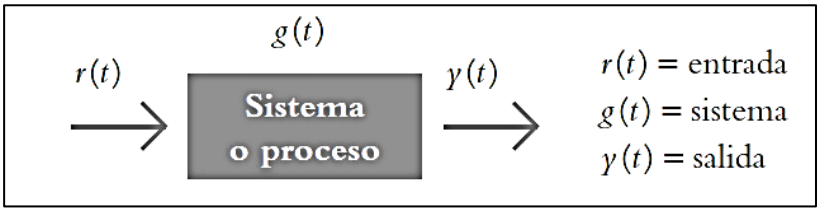
2.2. BASES TEÓRICAS.

2.2.1. SISTEMAS DE CONTROL.

Es importante conocer la teoría de un sistema de control. A continuación podemos definir:

Un sistema de control automático es una interconexión de elementos que forman una configuración denominada sistema, de tal manera que el arreglo resultante es capaz de controlarse por sí mismo. Un sistema o componente del sistema susceptible de ser controlado, al cual se le aplica una señal r(t) a manera de entrada para obtener una respuesta o salida y(t), puede representarse mediante bloques. (Hernandez Gaviño, pág. 97)

Figura 1: Sistema de control.



Fuente: (Hernandez Gaviño)

En términos generales, podemos decir que el objetivo de un sistema de control es controlar la variable salida a través de elementos de un sistema de control.

Los sistemas de control se clasifican en: Sistemas de lazo abierto (no automáticos) y en sistemas de lazo abierto (automáticos).

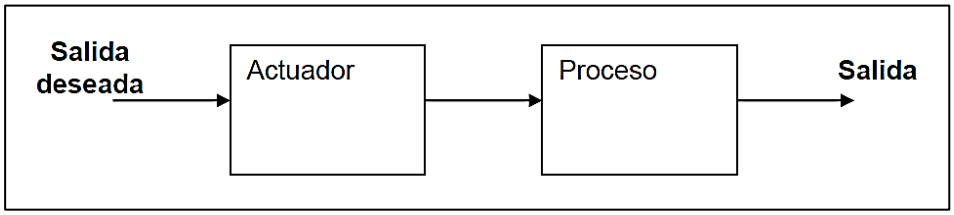
2.2.2. SISTEMA DE CONTROL A LAZO ABIERTO.

Un sistema de control en lazo abierto se caracteriza en cierto modo; por lo siguiente: en cual su acción de control es independiente de la variable de salida.

La exactitud para controlar una acción depende de la calibración de los componentes, y se puede decir que un sistema de control en lazo abierto se regula en base al tiempo.

Por lo tanto, se puede decir que un sistema de lazo abierto o llamado también sistema no realimentado utiliza un dispositivo actuador para controlar el proceso directamente. No existe realimentación de la salida hacia la entrada.

Figura 2: Sistema de control en lazo abierto.



Fuente: Elaboración Propia.

En el sistema de control en lazo abierto, el valor de la salida no tiene ningún efecto de cómo se calcula la señal de control.

Como se muestra en la figura anterior el proceso es simple donde se plantea una salida deseada teniendo como componente el actuador y el proceso.

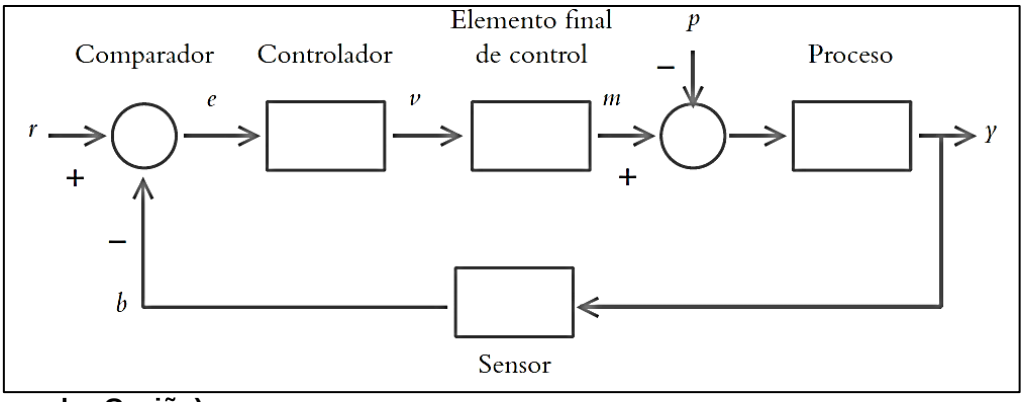
2.2.3. SISTEMA DE CONTROL A LAZO CERRADO.

El sistema de control en lazo cerrado se caracteriza en que la acción de control depende de la salida.

“Dicho sistema utiliza un sensor que detecta la respuesta real para compararla, entonces, con una referencia manera de entrada. Por esta razón, los sistemas de lazo cerrado se denominan sistemas retroalimentados. El término

retroalimentar significa comparar; en este caso, la salida real se compara con respecto al comportamiento deseado, de tal forma que si el sistema lo requiere se aplica una acción correctora sobre el proceso por controlar" (Hernandez Gaviño, pág. 6)

Figura 3: Sistema de control en lazo cerrado.



Fuente: (Hernandez Gaviño)

Por lo tanto, decimos que un sistema de control en lazo cerrado se realimenta la salida y se compara con el valor deseado de la misma entrada de referencia.

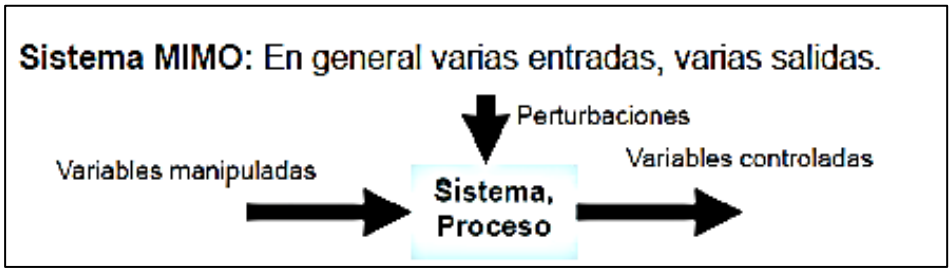
A continuación, se presenta un diagrama en donde se especifica los tipos de sistemas de control en lazo cerrado (SI y MIMO). El sistema de control SI (Tiene una entrada y una salida) y el sistema de control MIMO (Son sistemas que tiene varias entradas y salidas).

Figura 4: Sistema de control de una entrada y salida.

Fuente: Elaboración propia.

Podemos presentar también un sistema de control de una o varias entradas y salidas.

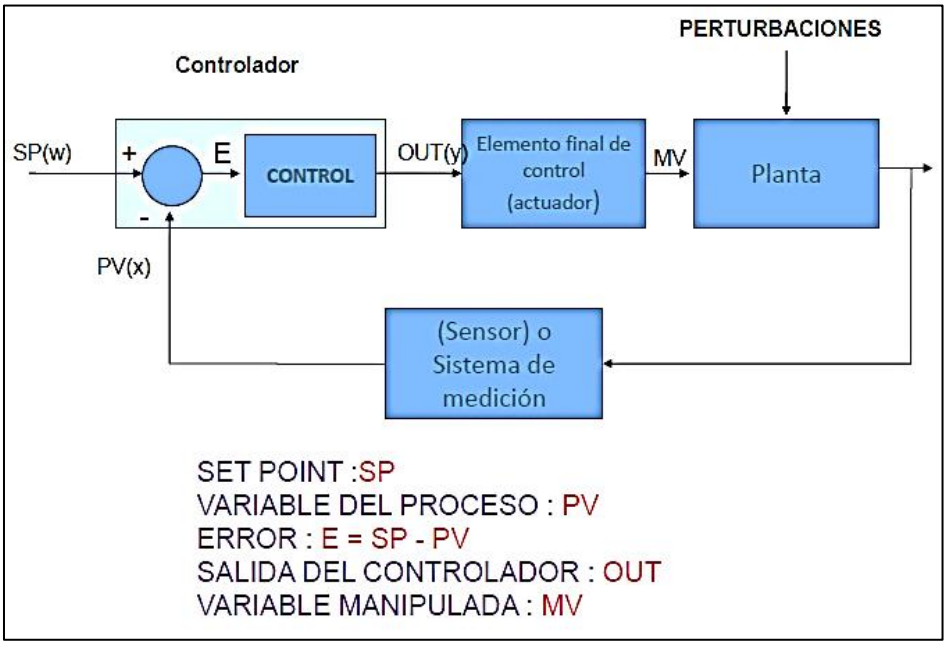
Figura 5: Sistema de control de varias entradas y salidas.



Fuente: Elaboración propia.

Entonces podemos representar mediante un diagrama de bloques un sistema de control automático con realimentación especificando todos sus elementos.

Figura 6: Representación de un sistema de control realimentado.



Fuente: Elaboración propia.

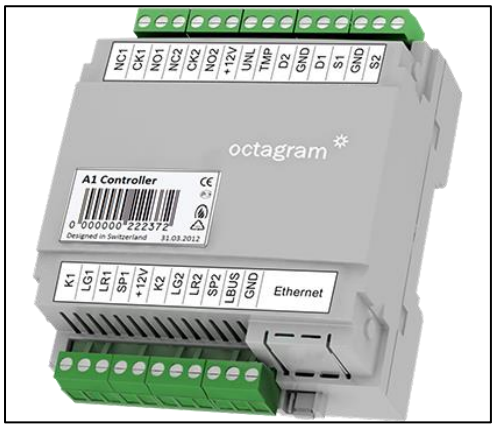
2.2.4. ELEMENTOS PRINCIPALES DE UN SISTEMA DE CONTROL.

2.2.4.1. Controlador.

El controlador es un dispositivo que opera de forma automática para regular la variable en proceso, este controlador recibe la señal del sensor es decir la variable en proceso y la compara con el punto fijo o set point.

La salida del controlador depende del error, si este error se mantiene constante la salida no cambia. Ahora, si el error es positivo, la salida aumenta y si el error es negativo, la salida disminuye.

Figura 7: Controlador de acceso de puerta.



Fuente: https://get.adobe.com/es/reader/

2.2.4.2. EL Actuador.

El actuador es un elemento del sistema de control que permite cambiar la variable manipulada, lo cual afecta directamente a la variable en proceso.

Figura 8: Actuador (Variador de velocidad).

Fuente: Elaboración propia.

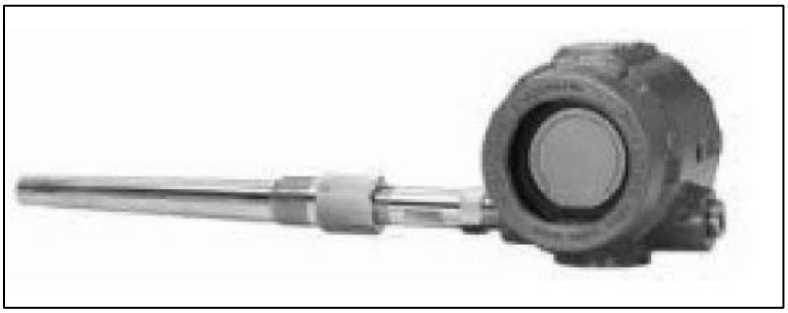
2.2.4.3. El sistema de medición.

EL sistema de medición en un sistema de control permite medir la variable del proceso conformador por lo siguiente:

El sistema de medición se compone del elemento primario de medición y el sensor.

El elemento primario de medición es el que detecta la variable física y nos da la otra variable que está relacionada con la otra variable física. Podemos decir que en nuestra investigación la termocupla detecta la temperatura y entrega una señal de tensión en el orden de los mili voltios.

Figura 9: Transmisor de temperatura.



Fuente: Elaboración propia.

2.2.5. MOTOR BOMBA DE AGUA.

El motor de bomba de agua es un equipo eléctrico que se utiliza para llevar agua de un nivel a otro de mayor altitud. Existen varios tipos de estos motores y se utilizan para diferentes usos; en nuestro caso lo podemos llamar electrobomba.

2.2.5.1. Partes de una electrobomba

a. Carcasa.

La mayoría de las carcasas son fabricadas en fierro fundido para agua potable, pero tienen limitaciones con líquidos agresivos (químicos, aguas residuales, agua de mar). Otro material usado es el bronce. También se usa el acero inoxidable si el líquido es altamente corrosivo.

b. Rodete.

Para el bombeo de agua potable en pequeños, medianos y gran caudal, se usan rodetes centrífugos de álabes radiales y semi axiales. Fabricados en fierro, bronce acero inoxidable, plásticos.

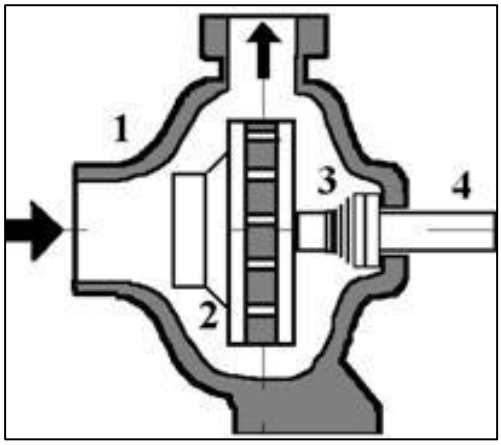
c. Sello mecánico.

Es el cierre mecánico más usado, compuesto por carbón y cerámica. Se lubrica y refrigera con el agua bombeada, por lo que se debe evitar el funcionamiento en seco porque se daña irreparablemente.

d. Eje impulsor.

En pequeñas bombas monoblock, el eje del motor eléctrico se extiende hasta la bomba, descansando sobre los rodamientos del motor. Fabricado en acero inoxidable.

Figura 10: Partes del motor de una electrobomba.



Fuente: Elaboración propia.

2.2.5.2. Partes del motor.

a. Eje rotor.

Es el eje que transmite la potencia mecánica desarrollada por el motor. El centro o núcleo está formado por chapas de acero magnético tratadas para reducir las pérdidas en el hierro. El núcleo del rotor aloja en su interior una bobina o anillo en corto circuito fabricado en aluminio.

b. Ventilador.

Turbina acoplada al eje del rotor, garantiza la refrigeración por aire del motor enfriando las aletas disipadoras de energía calórica que posee el estator. Fabricado en polipropileno

c. Caja de conexión.

Caja donde se alojan los bornes de conexión construidos de bronce y cobre de alta conductividad, que permiten conectar la energía eléctrica al motor. El block aislante es fabricado en plástico de gran resistencia eléctrica y mecánica

d. Rodamientos.

El eje rotor del motor está montado sobre rodamientos en cada extremo, estos son de bolitas o esferas de gran vida útil (20.000 horas de trabajo). Son sellados y lubricados para largos periodos de trabajo.

2.2.5.3. Rendimiento del motor.

El rendimiento de un motor es el cociente entre la potencia útil y la potencia total absorbida de la red y podemos expresarlo en la siguiente ecuación.

$$\eta = \frac{Pu}{Pt} * 100$$

Donde:

$\eta = Rendimiento\ del\ motor$

$P\mu = Potencia\ util\ del\ motor$

$Pt = Potencia\ total\ del\ motor$

2.2.5.4. Potencia eléctrica del motor.

La potencia absorbida por el motor monofásico se puede expresar de la siguiente manera.

$$P = V_L \cdot I_L \cdot \cos\varphi$$

Donde:

$V_L = Voltaje\ de\ linea$

$I_L = Corriente\ de\ linea$

$\eta = Rendimiento\ del\ motor$

Otra fórmula que se tiene que tener en cuenta para determinar el rendimiento del motor es la siguiente:

$$P = \frac{P_{eje}}{\eta}$$

$P = Potencia\ electrica$

$P_{eje} = Potencia\ mecanica$

2.3. DEFINICIÓN DE TÉRMINOS.

Control.

El medir el valor de la variable controlada y aplicar la variable manipulada al sistema para corregir el valor medido hasta el valor deseado.

Sistema Mimo.

Mimo es abreviatura de Multiple Inputs Multiple Outputs. Es un sistema en el que se trabaja más de unas entradas y/o salidas y pueden ser controladas mediante controladores multivariables.

Sistema Siso.

Siso es abreviatura de Single Input Single Output. Es un sistema en el que se trabaja con una entrada y/o salida.

Modelo matemático.

Son ecuaciones diferenciales que definen el comportamiento del sistema que vamos a controlar.

Sistema de control.

Es un conjunto de componentes que se interconectan formando una configuración o sistema que da lugar a una respuesta.

Controlador.

Un controlador puede ser un programa o dispositivo que realiza comparaciones y cálculos matemáticos para calcular la señal que se desea aplicar a la planta y poder obtener la salida deseada.

Referencia.

La referencia en un sistema de control es el valor deseado de la variable controlada o de salida.

Cambio de referencia.

Es el cambio de las condiciones de operación en donde la señal de referencia se cambia y la variable manipulada se ajusta para lograr nueva condición de operación.

Cambio de perturbación.

Es un comportamiento transitorio cuando aparece una perturbación, un sistema de control deberá ser capaz de llevar cada variable controlada a su referencia.

CAPÍTULO III
METODOLOGÍA

3.1. MÉTODOS Y ALCANCE DE LA INVESTIGACIÓN.

3.1.1. MÉTODO DE INVESTIGACIÓN.

El método utilizado es el método científico, porque se realizó el estudio con un conjunto de técnicas, procedimientos para luego ser comprobado mediante el uso de instrumentos de medición.

3.1.2. NIVEL DE INVESTIGACIÓN.

El grado de profundidad de la investigación es aplicativo, porque se diseñó un sistema de control de la temperatura de un motor, para mejorar el proceso de control en el fenómeno de estudio.

3.1.3. TIPO DE INVESTIGACIÓN.

El tipo de investigación que se desarrolló en la presente tesis es una investigación del tipo tecnológica, porque se observó y estudió un fenómeno y se aplicó la tecnología para solucionarlo.

3.1.4. ALCANCE DE LA INVESTIGACIÓN.

La presente tesis de diseño de un sistema de control tiene un alcance aplicativo, porque se llega a implementar un prototipo.

3.2. DISEÑO DE LA INVESTIGACIÓN.

El tipo de diseño de la presente tesis es experimental porque se diseñó el sistema de control y se logró implementar para verificar el funcionamiento.

3.3. POBLACIÓN Y MUESTRA.

3.3.1. POBLACIÓN.

La población en esta investigación es el motor de bomba de agua de 0.75 HP.

3.3.2. MUESTRA.

En este caso la muestra es la misma población que en esta investigación es el motor de bomba de agua de 0.75 HP

3.4. TÉCNICAS E INSTRUMENTACIÓN DE RECOLECCIÓN DE DATOS.

3.4.1. TÉCNICAS DE RECOLECCIÓN DE DATOS.

La técnica que se utilizó es la observación del funcionamiento del sistema de bombeo de agua, también se utilizó información de la tecnología que se encuentra en revistas, informes, investigaciones referentes al tema y especificaciones técnicas.

3.4.2. INSTRUMENTOS DE RECOLECCIÓN DE DATOS.

Para la recolección de los datos en la investigación, el instrumento de recolección de datos fue el registro de fichas de la observación.

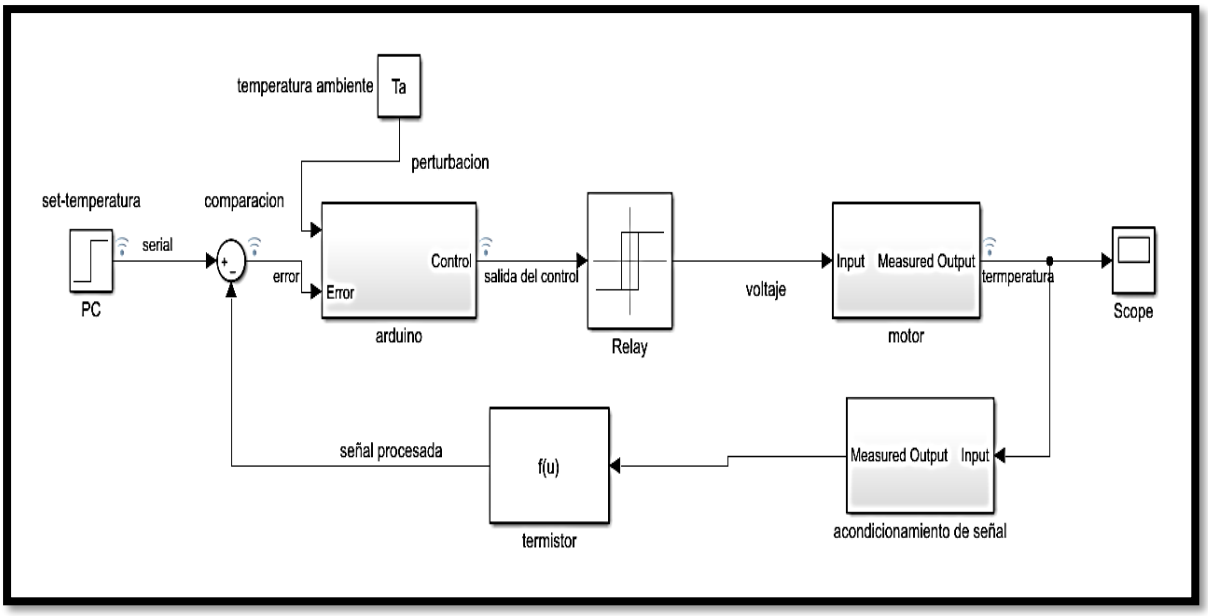
CAPÍTULO IV
RESULTADOS Y DISCUSIÓN

4.1. RESULTADOS.

4.1.1. DISEÑO DE UN SISTEMA DE CONTROL EN LAZO CERRADO.

De acuerdo al tipo de proceso a controlar, se diseñó un sistema de control en lazo cerrado. Este sistema de control permite entender la importancia de cada uno de los elementos que lo conforman y su funcionamiento dentro del sistema de control. A continuación, se presenta un diagrama de bloques y se define cada elemento perteneciente al SISTEMA DE CONTROL PARA EL MONITOREO DE LA TEMPERATURA DE UN MOTOR DE BOMBA DE AGUA DE 0.75 kW

Figura 11: Diagrama de bloque del sistema de control.



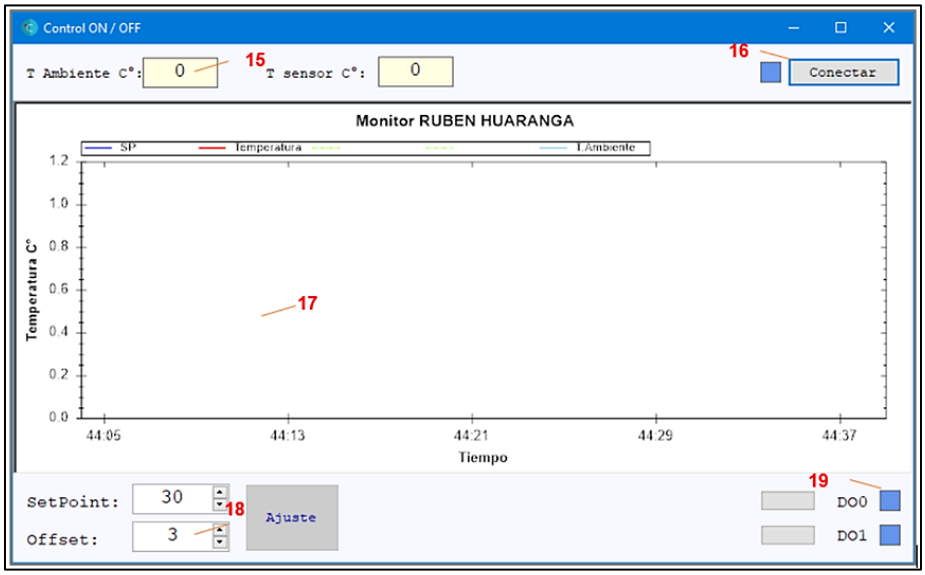
Fuente: Elaboración propia

4.1.1.1. Elementos del diagrama de bloques

a. El controlador.

El controlador en nuestro sistema de control es el software que compara el valor medido y el valor deseado.

Figura 12: Controlador (Software).

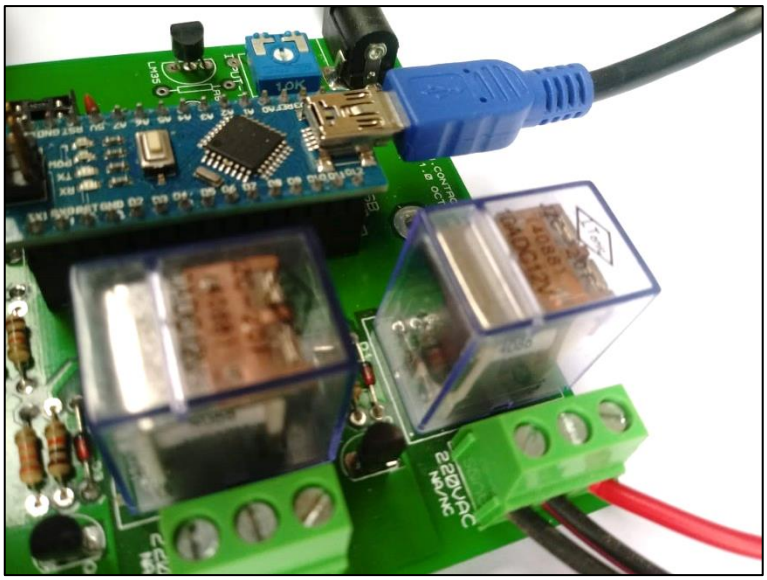


Fuente: Elaboración propia

b. El actuador.

En este caso, el actuador es el relay que permite el accionamiento y parada del sistema.

Figura 13: Relay.



Fuente: Elaboración propia.

c. Planta.

En nuestro diseño la planta de nuestro sistema es el motor.

Figura 14: Motor 0.75kW.

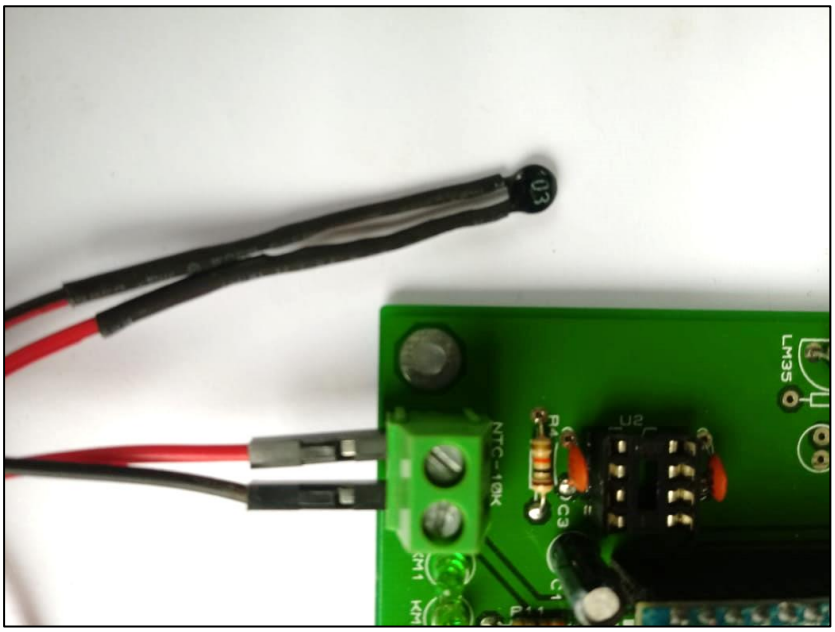
Fuente: Elaboración propia.

d. Sensor.

El sensor en nuestro sistema de control es el termistor; este elemento es de óxido de metal de alta presión, fabricado de cerámica que cambia su resistencia eléctrica con la variación de la temperatura.

Los termistores están disponibles a rangos grandes de tamaño y de valores comunes de resistencia (resistencia a 25 ºC). La capacidad de intercambio es posible a ± 0.05 ºC aunque a ± 1 ºC es más común.

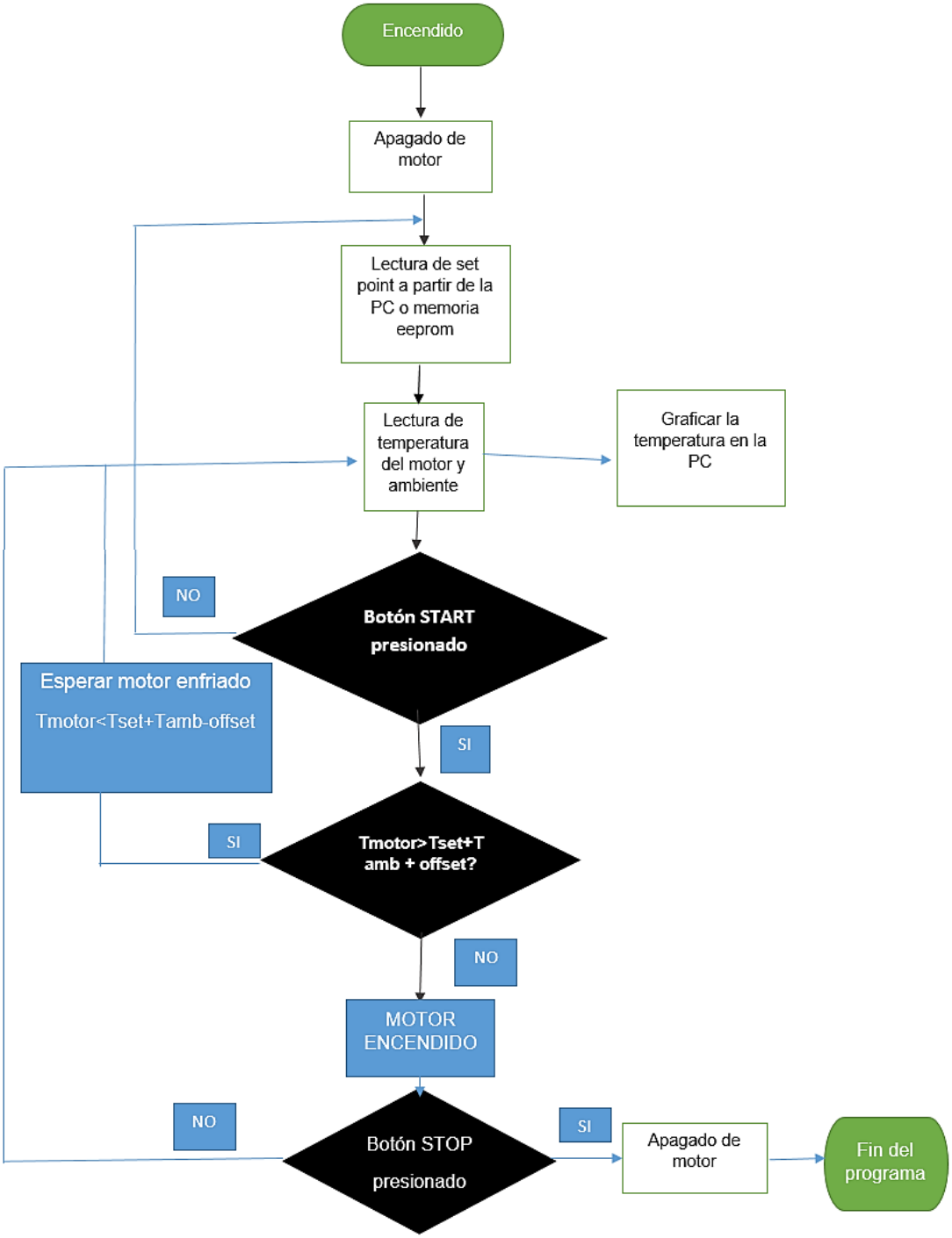
Figura 15: Sensor.



Fuente: Elaboración propia.

4.1.2. DIAGRAMA DE FLUJO.

Figura 16: Diagrama de flujo.



Fuente: Elaboración propia.

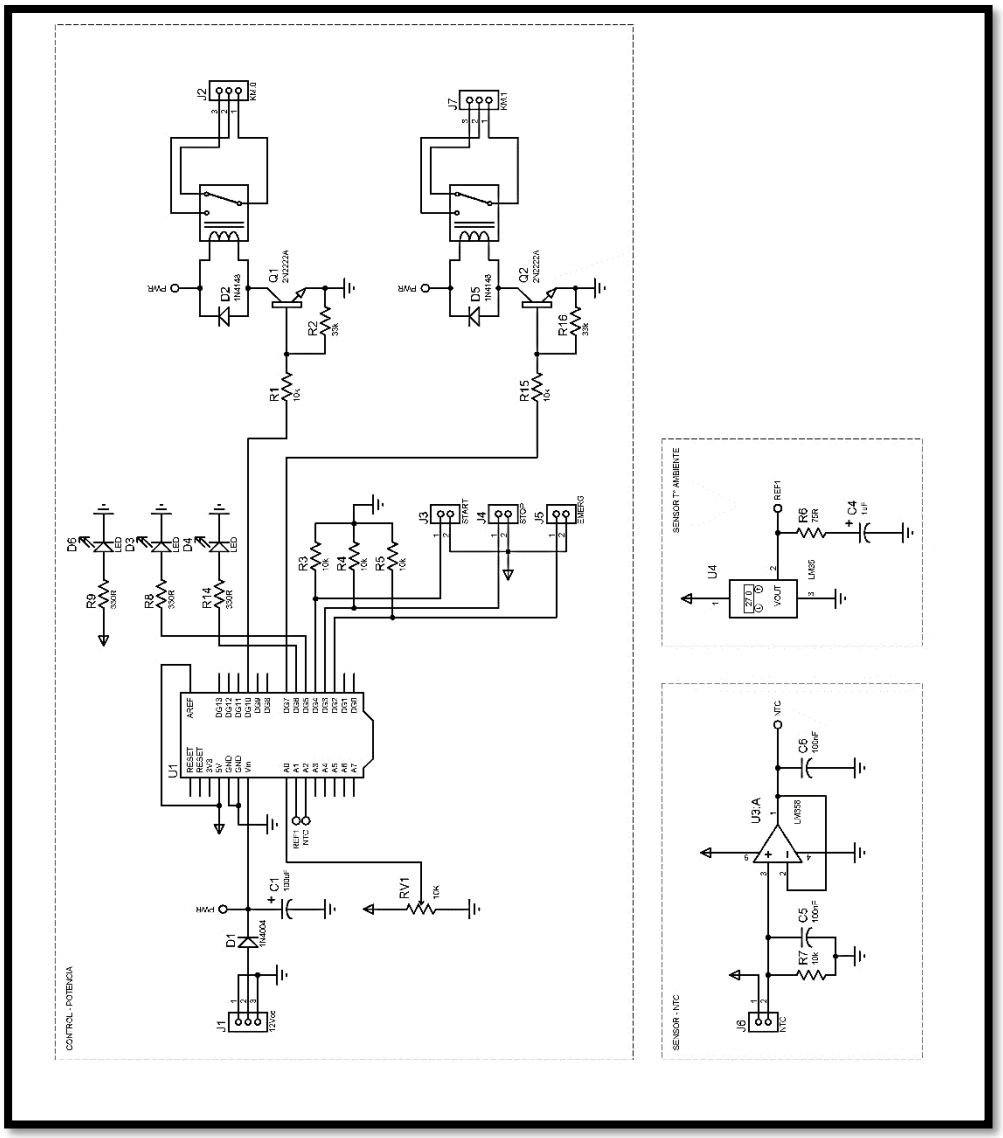
4.1.3. RESULTADOS DEL DISEÑO DE CONTROLADOR.

4.1.3.1. Diseño de hardware del controlador.

4.1.3.1.1. Diseño del circuito de la placa.

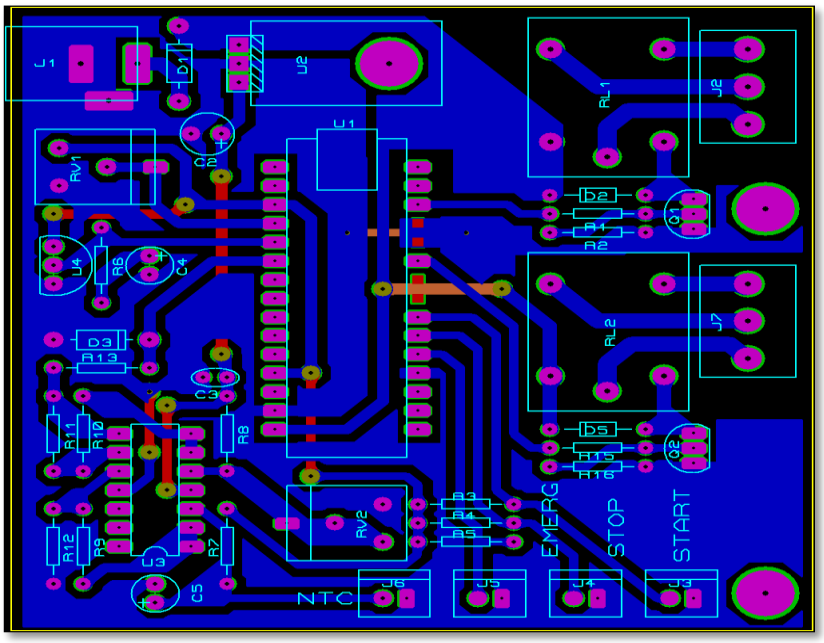
El circuito electrónico fue enteramente diseñado en el software Proteus VSM, dicho programa es un entorno visual tipo CAD para el diseño y simulación de circuitos electrónicos; así como de brindar herramientas para la creación de tarjetas electrónicas (PCB) a partir de un esquema electrónico. Circuito

Figura 17: Diseño del esquema electrónico.



Elaboración propia.

Figura 18: PCB del controlador.

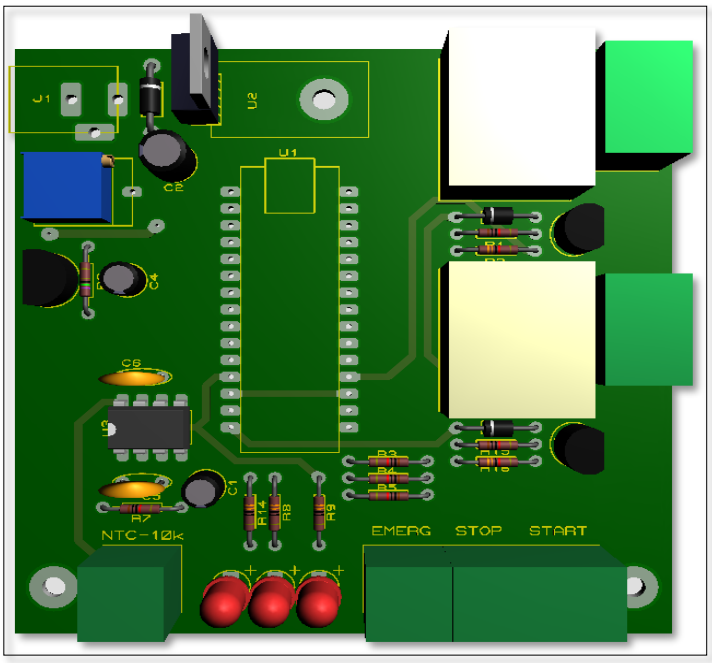


Fuente: Elaboración propia, diseño realizado en software Proteus.

4.1.3.1.2. Diseño de placa en 3D

Es una vista previa de cómo se verá la placa impresa de nuestro circuito en 3D, la cual se presenta en la figura:

Figura 19: Diseño PCB vista en 3D.



Fuente: Elaboración propia.

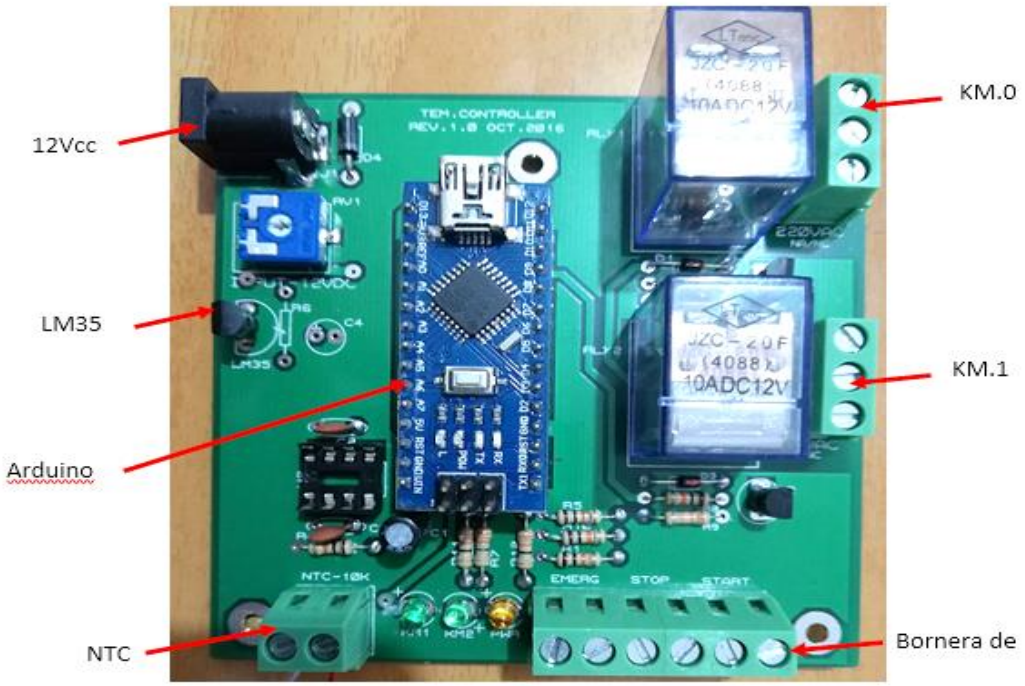
Para la realización física (real) de la tarjeta electrónica se puede optar por dos opciones, siendo:

- Realizar el mecanizado manualmente, que implica imprimir el negativo del modelo PCB para grabar las pistas en placas de cobre, este proceso se considera rudimentario, sin embargo, sigue siendo utilizado.
- Realizar la tarjeta electrónica generando desde el programa Proteus VSM un archivo llamado Gerber, este archivo es enviado a una empresa dedicada a la fabricación de circuitos impresos de nivel profesional, generalmente en el extranjero. Existen varios en su tipo como www.pcbcart.com, entre otros cuyas diferencias son en costo y capacidades de fabricación. Este segundo método permite crear tarjetas PCB con mayor lujo de detalles, ya que la fabricación completa la realizan máquinas industriales dedicadas casi sin intervención de personal.

4.1.3.1.3. Implementación del Hardware.

La implementación del Hardware se presenta en la siguiente figura:

Figura 20: Tarjeta de control construido.



Fuente: Elaboración propia.

4.1.3.1.4. Principales dispositivos que integran la tarjeta de control.

a. El arduino Nano

b. El microcontrolador ATmea328P

c. El circuito integrado FT232RL

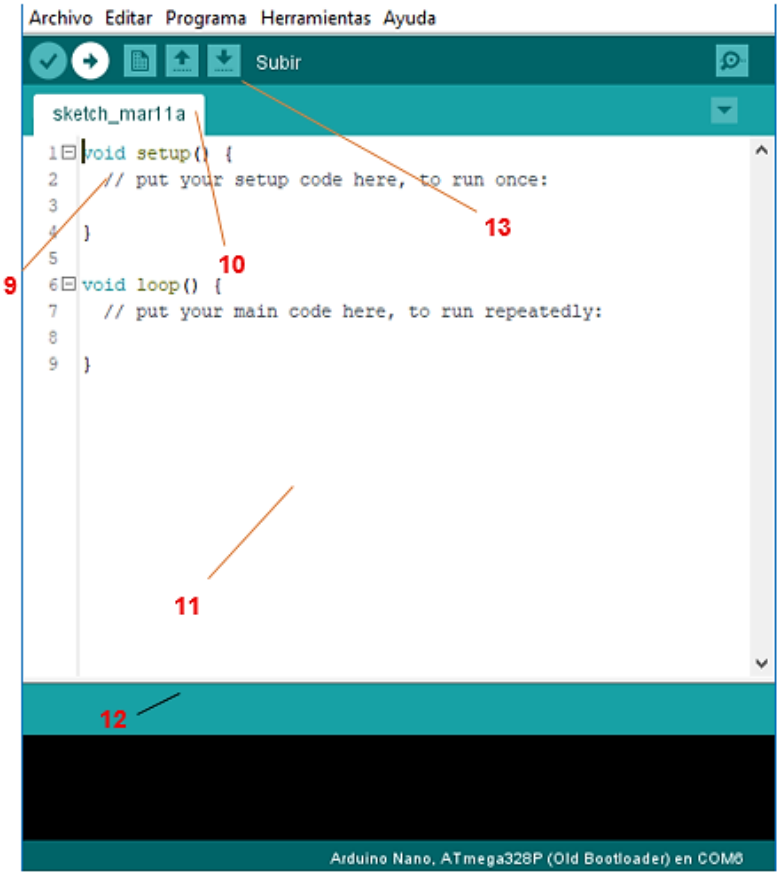
d. El relay.

e. El Transitor.

4.1.3.2. Diseño de Firmware.

Para la implementación del firmware se utilizó el entorno de desarrollo Arduino IDE.

El lenguaje de programación que utiliza Arduino es el C++, además de una serie de librerías (código preestablecido) que facilitan el diseño e implementación de cualquier programa.



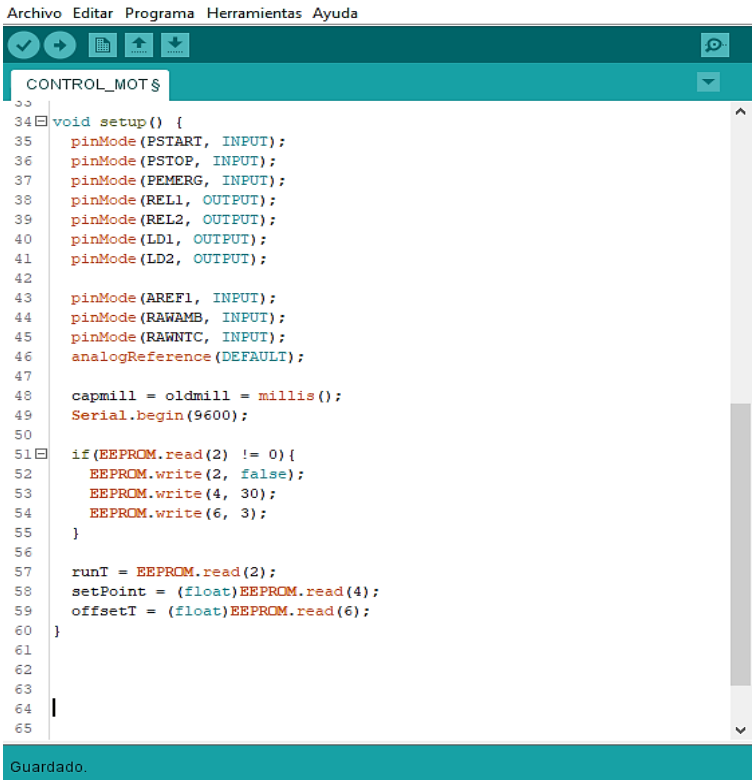
Las opciones más usadas que muestra el entorno del Arduino IDE se resumen en; Compilación del código escrito (9), que revisa la sintaxis o forma de escribir correctamente el código (11), para informar que todo está bien o se ha encontrado algún error (12), opción para subir o grabar el código en el Arduino (10), el cuál compila el código si es que no se realizó y procede a enviar el programa compilado al Arduino nano, de igual forma también informa (12), algún error que pueda suceder en el proceso de grabación.

La selección de la tarjeta Arduino adecuada, el puerto de comunicación y el método de grabación se encuentra en la opción de herramientas (13).

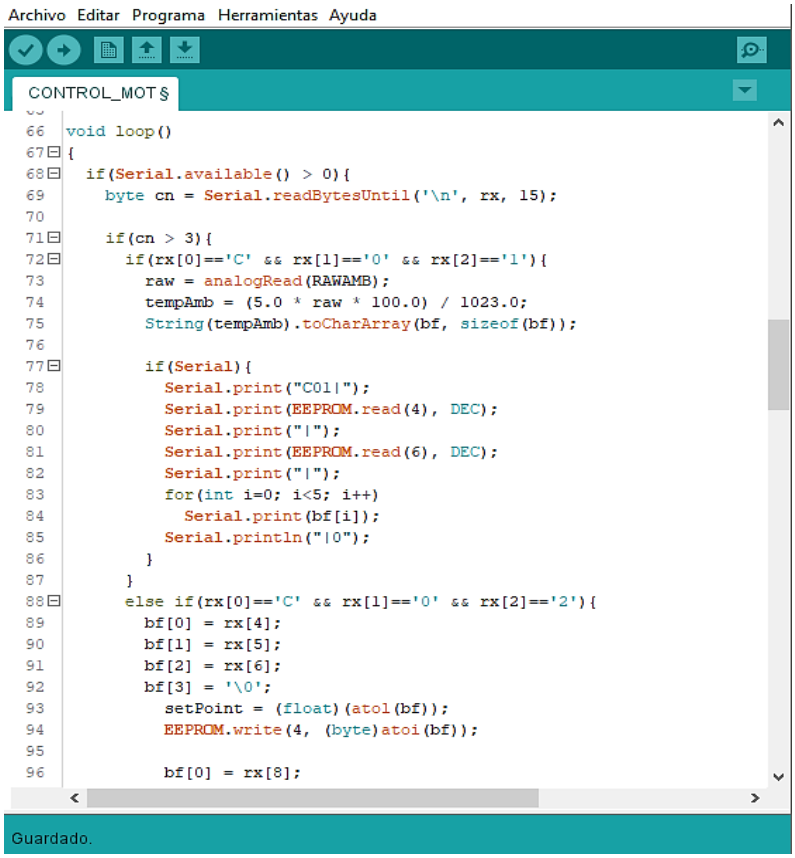
- El lenguaje de programación que utiliza Arduino es el C++, además de una serie de librerías (código preestablecido) que facilitan el diseño e implementación.
- Se escribe el código que incluye las librerías básicas de Arduino “Arduino.h”, “EEPROM.h” usados en este programa.

- Se define (#define) todos aquellos valores numéricos y pines usados para el control y cálculo del programa. Así como la definición de sectores de memoria a modo de variables (long, float, char, bool, etc).
- La ejecución del código inicia en el sector denominado “setup()”, que es donde se configura pines, hardware interno y la asignación de valores a las variables del código para el cálculo.
- La ejecución continua en el sector denominado “loop()”, el cual se repite constantemente mientras el Arduino esté energizado, por lo que el código está compuesto de operaciones lógicas, matemáticas y tomas de decisión que se ejecutan constantemente para cumplir con las tareas de medición, cálculo de temperatura, control de relés y comunicación.
- Incluye bloques de código que se encuentran fuera de los anteriormente descritos, porque estos son usados desde el bloque de código principal loop solo cada vez que sean necesarios.

Bloque de código de declaración.



Bloque de código de configuración.



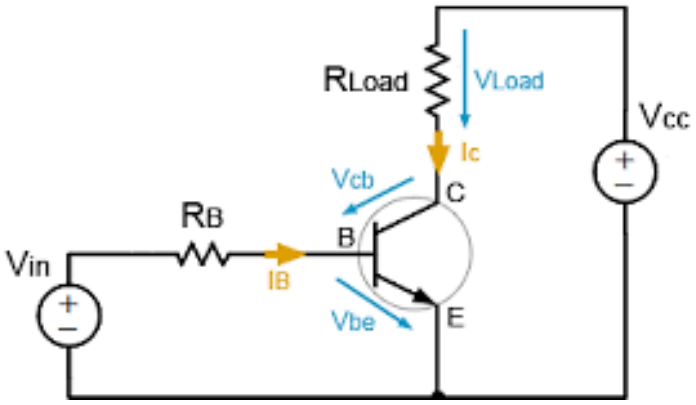
bloque de control y ejecución código del firmware parta el arduino nano (153 líneas de código) ver código en el anexo

4.1.4. RESULTADOS DE LA SELECCIÓN DEL SENSOR Y EL ACTUADOR.

4.1.4.1. El sensor.

Componente electrónico formado por materiales semiconductores, se utilizan en amplios ámbitos de trabajo como amplificadores de señal o como interruptores. De acuerdo al presente trabajo basta ser configurados como interruptores, cuyo esquema electrónico base es:

Figura 21: Transistor bipolar NPN



Fuente: Antonio Hermosa Donate

Donde *Load* es la bobina del relé, *Vin* es la señal digital de 5V producida por el ATmega328P y -*Vcc*- el voltaje con el que funciona el relé 12V.

Es necesario calcular el valor de los componentes para que el transistor pueda conducir la corriente necesaria para que active la carga (relé); para esto es necesario conocer las características eléctricas del transistor seleccionado. El circuito utiliza el transistor 2N2222A por lo que se tiene:

a. **Ganancia del transistor = 100 (β).**

b. **Voltaje Base-Emisor = 0.7V (V_{BE})**

Ecuación de cálculo de la corriente de relé.

$$R_B = \frac{Vcc - V_{BE}}{\frac{I_R}{hFe\ (\beta)}} = \frac{5V - 0.7V}{\frac{0.03A}{50}} = 7.166\ kOhm$$

Siendo un valor comercial de 10Kohm el más cercano al resultado.

4.1.4.2. El actuador.

Es un dispositivo interruptor controlado que consta de 2 partes, 1 circuito electromagnético (electroimán) y un circuito de contactos en el cual se conecta la señal a controlar. Permite asilar mecánicamente dos fuentes de tensión que incluso pueden ser diferentes (AC/DC).

Su funcionamiento se basa en el fenómeno electromagnético. Cuando la corriente atraviesa la bobina, produce un campo magnético que magnetiza un núcleo de hierro dulce (ferrita). Este atrae al inducido que fuerza a los contactos a tocarse. Cuando la corriente se desconecta vuelven a separarse.

Para su accionamiento desde un dispositivo digital, se requiere conocer sus características eléctricas y de ese modo implementar un circuito electrónico de control.

Para el caso se usa relés con las siguientes características eléctricas:

a. Tensión de alimentación: 12Vcc.

b. Impedancia de la bobina: 400 Ohm (Ω).

c. Intensidad máxima en sus contactos: 10 (Amperios).

Para determinar la corriente de consumo de la bobina se procede a usar la ley de Ohm, donde:

Ecuación de consumo de la bobina.

$$I_R = \frac{12V}{400\Omega} = 0.03A\ \approx 30mA\ (consumo\ de\ la\ bobina)$$

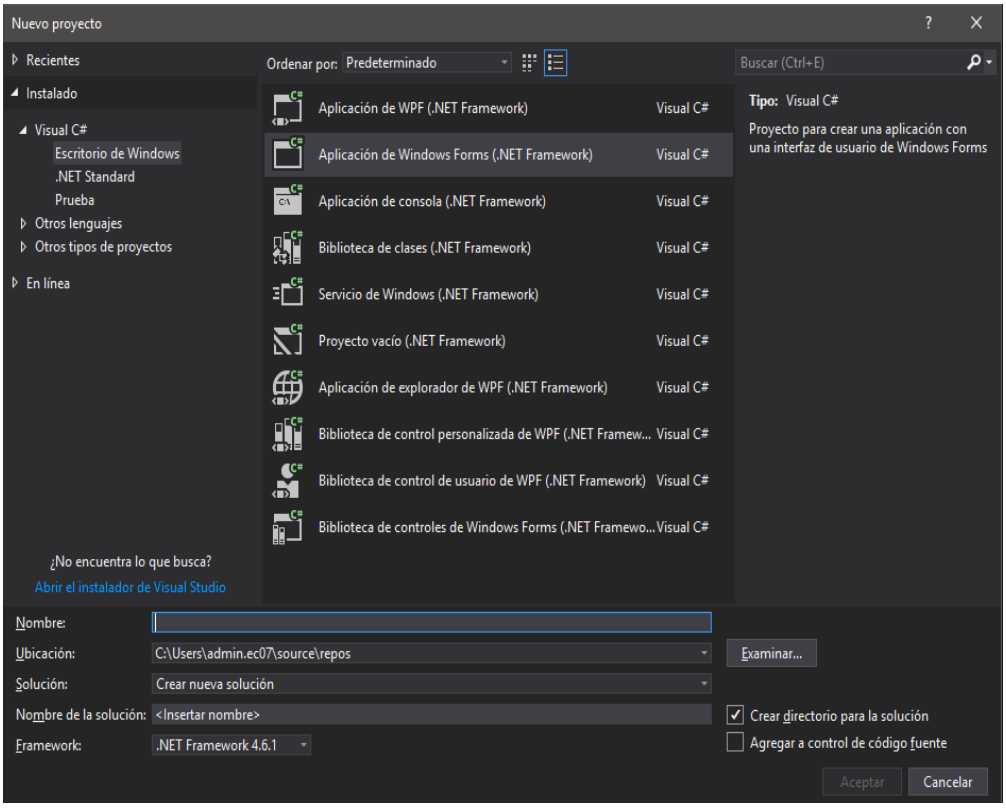
El ATmega328, funciona a 5V y sus pines no pueden entregar más 25mA, por lo que no es posible activar los relés directamente; para esto es necesario un driver constituido por un transistor.

4.1.5. RESULTADOS DEL DISEÑO DEL SOFTWARE DE CONTROL Y MONITOREO DE LA TEMPERATURA.

4.1.5.1. Diseño del Software.

El programa Visual Studio se basa en organizar todo en base a proyectos, es por eso por lo que cada vez que se procede a crear un nuevo programa es necesario iniciar creando el proyecto; un proyecto requiere de un nombre general, la versión del framework a usar y seleccionar el tipo de aplicación.

Figura 22: Creación del proyecto.



Fuente: Visual studio.

Una vez que se carga el nuevo proyecto para el entorno Windows, se carga un espacio de trabajo en el cual se diagrama visualmente el estilo y organización de la ventana de control para el nuevo

programa; este proceso solo es de diseño visual, más no ejecuta ninguna tarea ya que es necesario escribir el código de control.

El código de control se escribe en el leguaje seleccionado al momento de crear el programa; la estructura del leguaje C# es considerablemente más avanzada que la estructura del leguaje usado en Arduino por lo que es necesario tener conocimientos más sólidos del método de programación.

La programación se ejecuta cada vez que hay algún comportamiento desde el entorno visual o externo como al usar los puertos de la computadora. Es decir que el código escrito no es lineal como el de Arduino, ya que este responde cada vez que hay interacción.

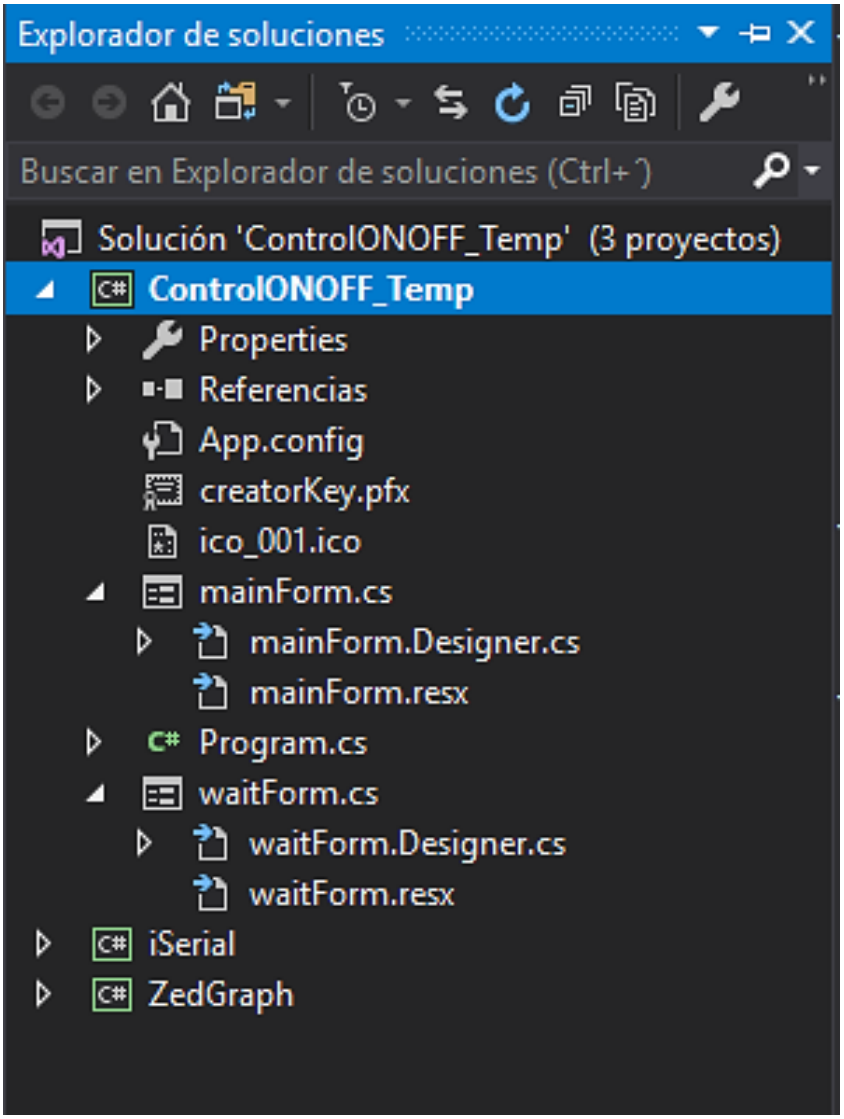
Figura 23: Bloque inicial del programa escrito en C# para computadora. Describe la construcción y preparación del entorno visual del proyecto.

```
private void mainForm_Load(object sender, EventArgs e)
{
    GraphPane gPane = iGraph.GraphPane;
    gPane.Title.Text = "Monitor RUBEN HUARANGA";
    gPane.YAxis.Title.Text = "Temperatura C°";
    gPane.XAxis.Title.Text = "Tiempo";
    gPane.XAxis.Type = AxisType.Date;

    RollingPointPairList list1 = new RollingPointPairList(1200);
    LineItem SPcurve = gPane.AddCurve("SP", list1, Color.Blue, SymbolType.None);
    RollingPointPairList list2 = new RollingPointPairList(1200);
    LineItem Tcurve = gPane.AddCurve("Temperatura", list2, Color.Red, SymbolType.None);
    Tcurve.Line.Width = 2;
    Tcurve.Line.IsOptimizedDraw = true;
    Tcurve.Line.IsSmooth = true;

    RollingPointPairList lst1 = new RollingPointPairList(1200);
    LineItem offsetA = gPane.AddCurve(" ", lst1, Color.GreenYellow, SymbolType.None);
    offsetA.Line.Style = System.Drawing.Drawing2D.DashStyle.DashDotDot;
    RollingPointPairList lst2 = new RollingPointPairList(1200);
    LineItem offsetB = gPane.AddCurve(" ", lst2, Color.GreenYellow, SymbolType.None);
    offsetB.Line.Style = System.Drawing.Drawing2D.DashStyle.DashDotDot;

    RollingPointPairList list3 = new RollingPointPairList(1200);
    LineItem TcurveAmb = gPane.AddCurve("T.Ambiente", list3, Color.SkyBlue, SymbolType.None);

    int numberOfDivisions = 4;
    gPane.XAxis.Scale.MinorStep = (int)displaySeconds / numberOfDivisions;
    gPane.XAxis.Scale.MajorStep = (int)displaySeconds / numberOfDivisions;
    gPane.XAxis.Type = AxisType.Date;
    gPane.XAxis.Scale.Format = "mm:ss";
    gPane.XAxis.Scale.MajorStepAuto = false;
    gPane.XAxis.Scale.MinorStepAuto = false;
    gPane.XAxis.Scale.MajorUnit = DateUnit.Second;
    gPane.XAxis.Scale.MinorUnit = DateUnit.Second;
    iGraph.AxisChange();

    DateTime now = DateTime.Now;
    iGraph.GraphPane.XAxis.Scale.Max = now.ToOADate();
    iGraph.GraphPane.XAxis.Scale.Min = DateTime.Now.AddSeconds(-displaySeconds).ToOADate();
    iGraph.AxisChange();
```

Fuente: Elaboración propia.

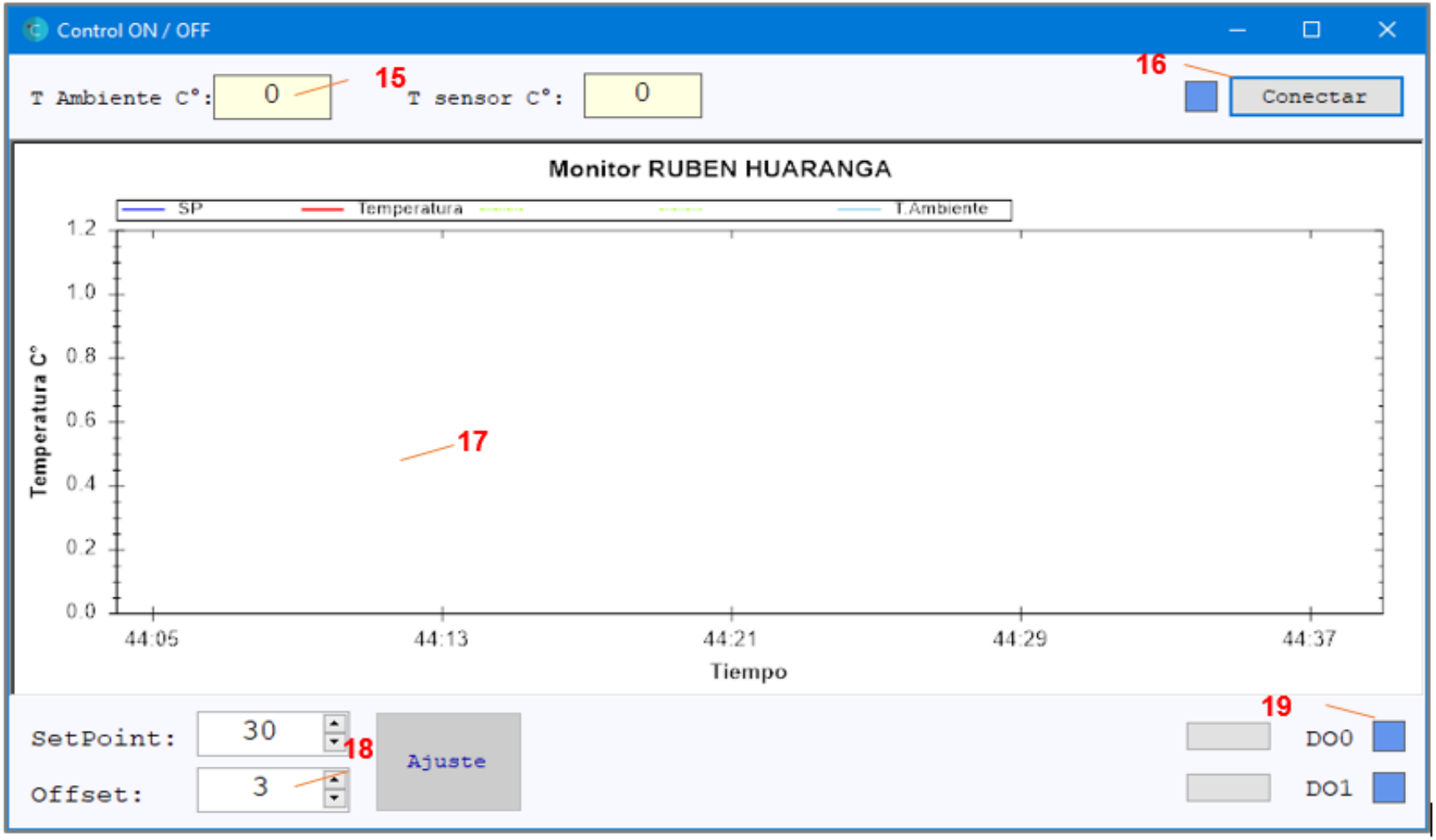
El código escrito se puede organizar ya sea en una sola hoja o múltiples hojas de código y carpetas; eso ya depende de la complejidad del programa. Para el caso no ha sido necesario mayor complejidad ya que el código funcional cabe en una hoja. Además, se utilizan bloques de código a modo de librería que son usados desde el código principal; estos bloques son “iSerial” para comunicación serial usando el puerto USB y el bloque “ZedGraph” que es una librería para crear gráficos. Ambos con su propia organización y complejidad.

Figura 24: Organización de archivos, hojas de código del proyecto.



Fuente: Elaboración propia.

Figura 25: Interfaz Gráfica diseñado en el Visual Studio.



Fuente: Elaboración propia.

- La mayor parte del programa se centra en informar los cambios de temperatura, en una gráfica interactiva entre Temperatura y Tiempo (17), que dibuja las líneas de temperatura de acuerdo a los datos trasmitidos desde el hardware Arduino nano.
- En cuadros de texto indica la temperatura ambiente del sensor LM35 y del sensor NTC (15). Así como de permitir el ajuste del SetPoint y Offset (18) cuyo valor son usados para indicar el punto máximo de temperatura permitido.
- Todo el control y monitoreo es posible siempre y cuando el Software esté conectado al puerto USB, para esto se incluye una opción de conexión/desconexión (16). Cuyo trabajo es buscar en que puerto USB (Puerto Serie Virtual) se encuentra conectado el Arduino nano; en caso de no encontrar el hardware, lanza un aviso.
- Posee la característica de activar o desactivar los relés de la tarjeta electrónica mediante dos botones (19) relacionados con el relé DO0 y DO1 independientemente. El relé DO0 se conmuta ya sea desde la conexión externa o desde el software y el botone respectivo, por lo que posibilita la activación y desactivación local o remota del mismo relé.
- El Puerto Serie Virtual, utiliza la conexión física mediante USB para simular el protocolo RS232 que es nativo en el Arduino nano; además de la facilidad de implementación sin recurrir a aspectos avanzados nativos del puerto USB. Todos los drivers de control se instalan junto a la instalación del Arduino nano.

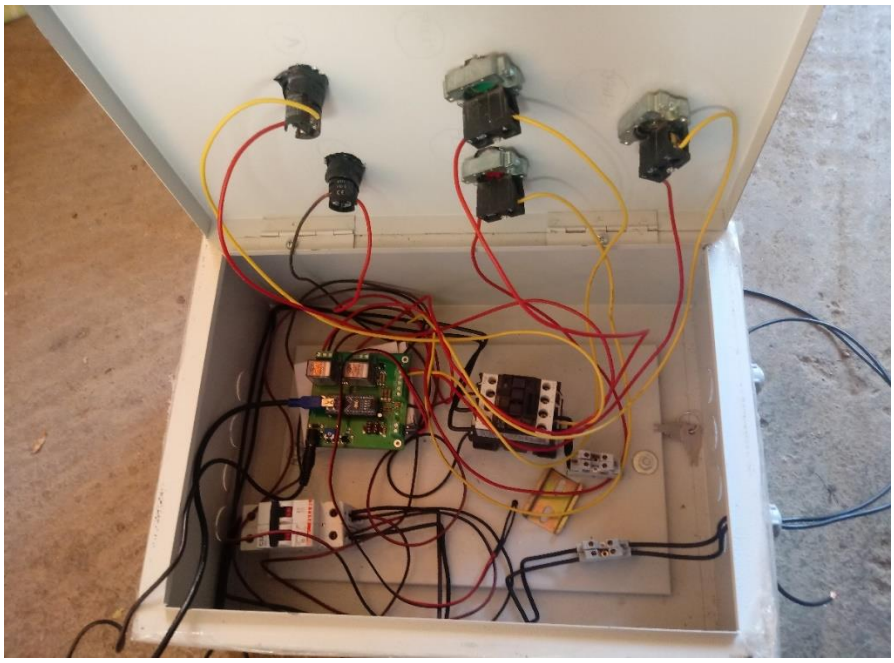
4.1.6. RESULTADOS DE IMPLEMENTACIÓN DEL SISTEMA DE CONTROL.

Como resultado del diseño a continuación se presenta la implementación del sistema de control. Para implementar se utilizaron los siguientes elementos:

- Tablero metálico
- Pulsadores arranque (start), parada (stop) y pare (emergencia).
- Led de parada (rojo) y arranque (verde)
- Prototipo electrónico
- Interruptor termomagnético
- Contactor
- Borneras
- Riel
- Computadora
- Motor de bomba de agua.

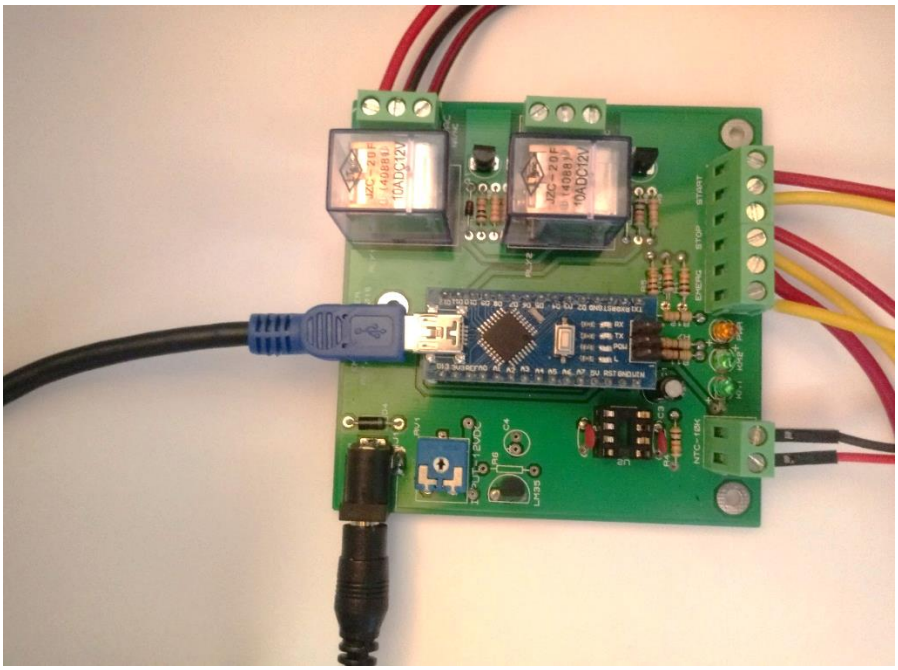
A continuación, se presenta el proceso de implementación y el funcionamiento del prototipo.

Figura 26: Instalando el circuito del tablero.



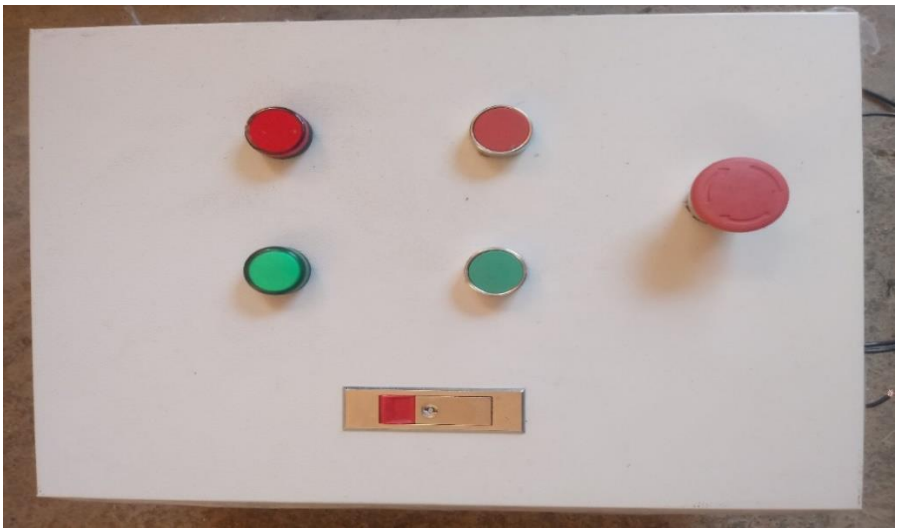
Fuente: Elaboración propia.

Figura 27: Prototipo instalado.



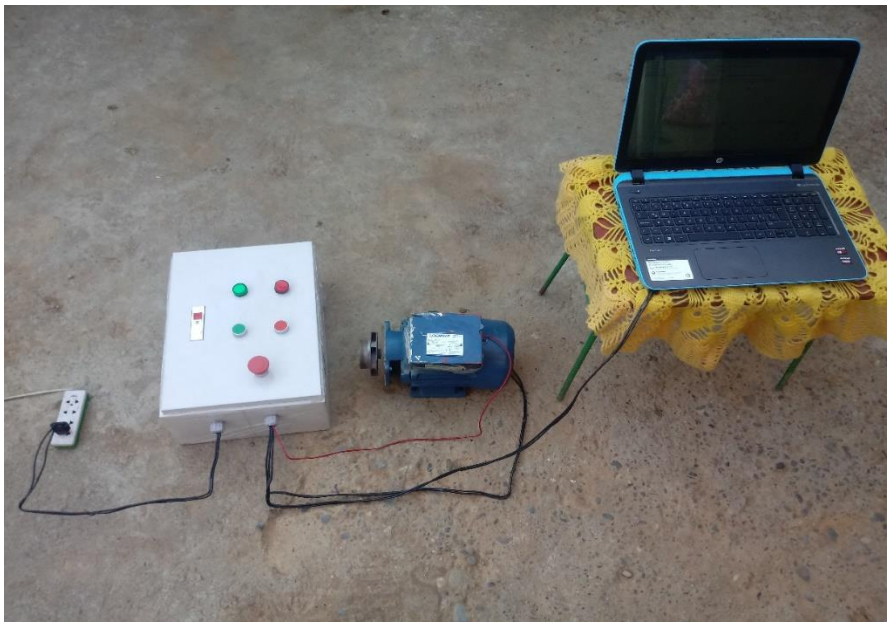
Fuente: Elaboración propia.

Figura 28: Tablero finalizado.



Fuente: Elaboración propia.

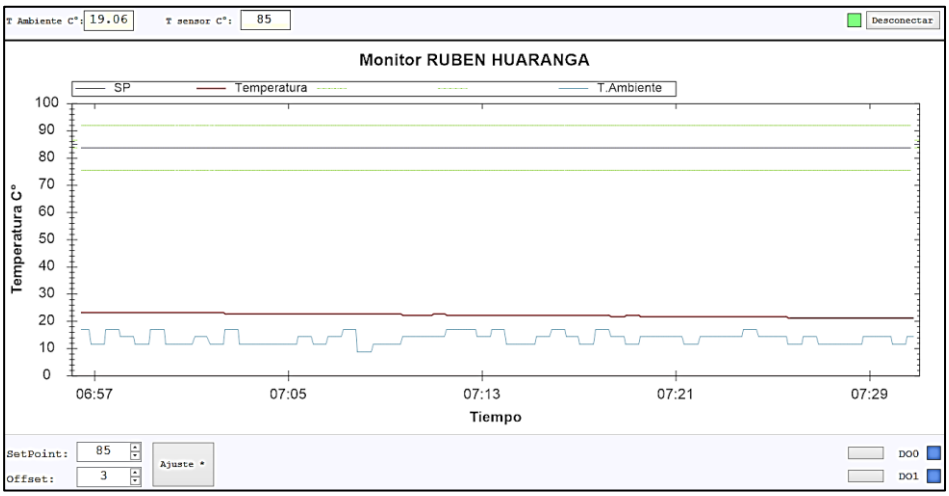
Figura 29: Instalación real del sistema de control del proyecto.



Fuente: Elaboración propia.

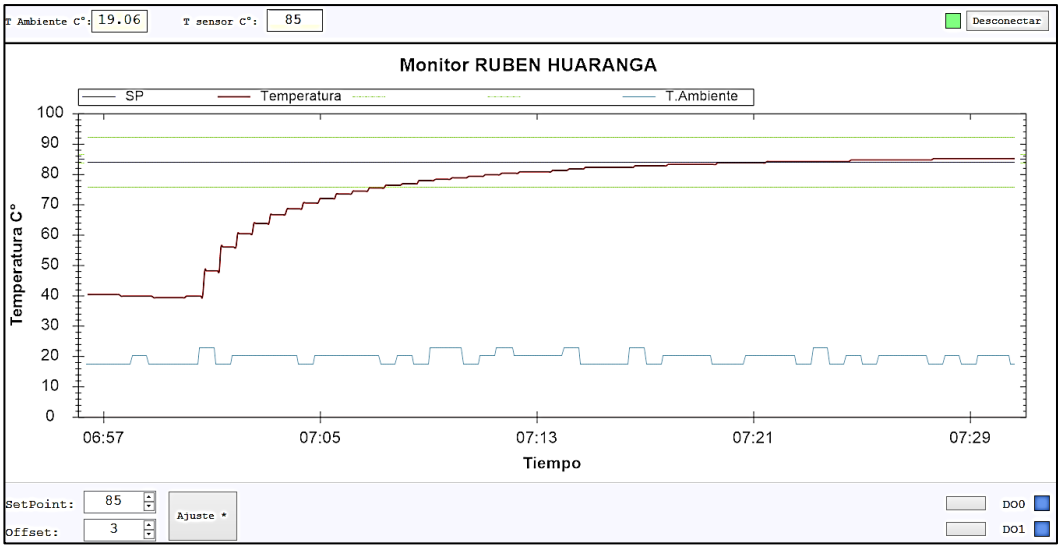
A continuación, se muestra el ensayo que se realizó con el motor.

Figura 30: Motor apagado.



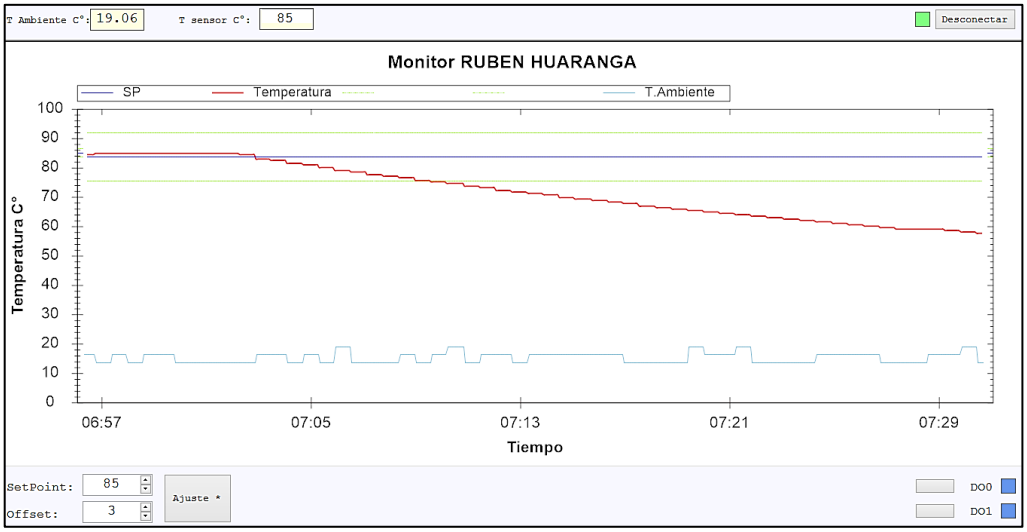
Fuente: Elaboración propia.

Figura 31: Motor con sobre calentamiento.



Fuente: Elaboración propia.

Figura 32: Motor bloqueado para su proceso de enfriamiento.



Fuente: Elaboración propia.

4.2. PRUEBA DE HIPÓTESIS.

Para la prueba de hipótesis de nuestra investigación se plantea las hipótesis nula y alternativa:

Ho: El diseño un sistema de control SI PERMITE monitorear la temperatura del motor de bomba de agua de 0.75kW.

H1: El diseño un sistema de control NO PERMITE monitorear la temperatura del motor de bomba de agua de 0.75kW.

Según los resultados de la investigación se acepta la hipótesis nula.

4.3. DISCUSIÓN DE RESULTADOS.

De los resultados obtenidos en la presente tesis podemos apreciar que se logra implementar el diseño para controlar la temperatura en el motor cuando pasa la temperatura de 90ºC, ayudando con el software de Visual Basic. Ahora de acuerdo a Jimenez Escamilla, Isacc Salomón en su investigación se llega a controlar la temperatura del motor de un horno eléctrico gracias a la implementación de un sistema de control basado en lógica difusa y el software del sistema y el controlador se desarrollaron en LabVIEW.

CONCLUSIONES

1. Se desarrolló un diagrama de flujo de control adecuado para el sistema en función a la temperatura de operación.
2. Se seleccionó el sensor y el actuador del sistema y para ello se implementó en una tarjeta para su comprobación.
3. Se realizó un software de interfaz entre el usuario y el sistema de control el cual e inclusive puede almacenar datos para poder tener un historial.
4. Se logró implementar el sistema de control para monitorear la temperatura de un motor de bomba de agua de 0.75kW.

RECOMENDACIONES

1. Se recomienda mejorar el sistema de control para que se mas interactivo entre usuario y maquina a través de un sistema vía wifi, bluetooth, etc.
2. Si recomienda para la industria implementar un sistema de control más adecuado como podría ser un sistema Scada.

REFERENCIAS BIBLIOGRÁFICAS

Alvarez Urbano, J. E. (2013). “Análisis Técnico Del Calentamiento De Las Máquinas Eléctricas Según Su Prestación De Servicio.

Caycho Huamancondor, J. J. (2004). *Pruebas De Control De Motores Eléctricos En Servicio.*

Hernandez Gaviño, R. (2010). *Introduccion a sistemas de control.* Mexico: Pearson.

Jimenez Escamilla, I. S. (2012). Control de temperatura de un horno eléctrico mediante lógica difusa. HUAJUAPAN DE LEÓN, OAXACA, MEXICO.

López de Paz, R. G. (Abril de 2016). Sistema de control de temperatura de un invernadero. Lima, Peru.

Ogata, K. (1996). *Sistemas de contorl moderna.* Prentice-Hall.

Ogata, K. (1998). *Ingenieria de control moderna.* Prentice-Hall.

Phillips, C. L. (1984). *Digital contorl systemes analysis and design.* Prentice-Hall.

Tapia Ayala, C. H. (2013). Evaluación De La Plataforma Arduino e Implementación De Un Sistema De Control De Posición Horizontal.

ANEXOS

ANEXO 1. PROGRAMA, FIRMWARE DEL ARDUINO NANO:

```
#include <Arduino.h>

unsigned long oldmill, capmill;
unsigned int raw;
unsigned int capTime=6;
unsigned int capCn=0;
float tempAmb;
float tempNTC;
float setPoint = 30;
float offsetT = 3;
char bf[6];
char tx[25];
char rx[15];
int indx = 0;
bool sp = false;
bool runT = false;
bool sendStat = false;

enum OTYPE{TN = -1, TLOW = 0, THIGH = 1};
void updateRelayState(int8_t drel1, int8_t drel2);


void setup() {
        pinMode(PSTART, INPUT);
        pinMode(PSTOP, INPUT);
```

```
	pinMode(PEMERG, INPUT);
	pinMode(REL1, OUTPUT);
	pinMode(REL2, OUTPUT);
	pinMode(LD1, OUTPUT);
	pinMode(LD2, OUTPUT);

	pinMode(AREF1, INPUT);
	pinMode(RAWAMB, INPUT);
	pinMode(RAWNTC, INPUT);
	analogReference(DEFAULT);

	capmill = oldmill = millis();
	Serial.begin(9600);

	if(EEPROM.read(2) != 0){
		EEPROM.write(2, false);
		EEPROM.write(4, 30);
		EEPROM.write(6, 3);
	}

	runT = EEPROM.read(2);
	setPoint = (float)EEPROM.read(4);
	offsetT = (float)EEPROM.read(6);
}

void loop()
{
```

```
if(Serial.available() > 0){
        byte cn = Serial.readBytesUntil('\n', rx, 15);

        if(cn > 3){
                if(rx[0]=='C' && rx[1]=='0' && rx[2]=='1'){
                        raw = analogRead(RAWAMB);
                        tempAmb = (5.0 * raw * 100.0) / 1023.0;
                        String(tempAmb).toCharArray(bf, sizeof(bf));

                        if(Serial){
                                Serial.print("C01|");
                                Serial.print(EEPROM.read(4), DEC);
                                Serial.print("|");
                                Serial.print(EEPROM.read(6), DEC);
                                Serial.print("|");
                                for(int i=0; i<5; i++)
                                        Serial.print(bf[i]);
                                Serial.println("|0");
                                }
                        }
                        else if(rx[0]=='C' && rx[1]=='0' && rx[2]=='2'){
                                bf[0] = rx[4];
                                bf[1] = rx[5];
                                bf[2] = rx[6];
                                bf[3] = '\0';
                        setPoint = (float)(atol(bf));
                        EEPROM.write(4, (byte)atoi(bf));
```

```
                bf[0] = rx[8];
                bf[1] = rx[9];
                bf[2] = '\0';
                offsetT = (float)(atol(bf));
                EEPROM.write(6, (byte)atoi(bf));
                }
                else if(rx[0]=='C' && rx[1]=='0' && rx[2]=='3') {
                        if(rx[4] == '1') {
                                updateRelayState(OTYPE::THIGH, OTYPE::TN);
                                runT = true;
                        } else if(rx[4] == '0') {
                                updateRelayState(OTYPE::TLOW, OTYPE::TN);
                                runT = false;
                        }
                }
                else if(rx[0]=='C' && rx[1]=='0' && rx[2]=='4') {
                if(rx[4] == '1') {
                        updateRelayState(OTYPE::TN, OTYPE::THIGH);
                } else if(rx[4] == '0') {
                        updateRelayState(OTYPE::TN, OTYPE::TLOW);
                }
                }
        }
}

if((millis() - oldmill) > 100) {
```

```
        oldmill = millis();

        if(runT == true){
                capCn += 1;

                if(capCn >= capTime) {
                        capCn = 0;

                        raw = analogRead(AREF1);
                        float adjtemp = (raw*10.0)/1023.0;

                        raw = analogRead(RAWAMB);
                        tempAmb = (5.0 * raw * 100.0) / 1023.0;
                        String(tempAmb).toCharArray(bf, sizeof(bf));

                        tx[0] = 'C';
                        tx[1] = '0';
                        tx[2] = '3';
                        tx[3] = '|';
                        for(int i=0; i<5; i++)
                                tx[i+4] = bf[i];
                        tx[9] = '|';

                        raw = analogRead(RAWNTC);
                        tempNTC = log(((10240000/raw) - 10000));
                        tempNTC = 1 / (0.001129148 + (0.000234125 + (0.0000000876741
* tempNTC * tempNTC ))* tempNTC );
```

```
                tempNTC = tempNTC - 273.15;

                tempNTC += adjtemp;
                String(tempNTC).toCharArray(bf, sizeof(bf));
                for(int i=0; i<5; i++)
                        tx[i+10] = bf[i];
                tx[15] = '|';
                tx[16] = '0';

                if(Serial) {
                        if(Serial) Serial.println(tx);
                }

                delay(50);

                if(tempNTC > setPoint+(offsetT/2)) {
                        updateRelayState(OTYPE::TLOW, OTYPE::TN);
                }else if(tempNTC < setPoint-(offsetT/2)){
                        updateRelayState(OTYPE::THIGH, OTYPE::TN);
                }
                }
        }
}

if(digitalRead(PSTART)) {
        runT = true;
        sendStat = false;
```

```
        }

        if(digitalRead(PSTOP)) {
                runT = false;
                updateRelayState(OTYPE::TLOW, OTYPE::TLOW);
        }

        if(digitalRead(PEMERG)) {
                runT = false;
                updateRelayState(OTYPE::TLOW, OTYPE::TLOW);
        }
}

void updateRelayState(int8_t drel1, int8_t drel2) {

        if(drel1 == 1) {
                digitalWrite(REL1, HIGH);  digitalWrite(LD1, HIGH);
        } else if(drel1 == 0) {
                digitalWrite(REL1, LOW);  digitalWrite(LD1, LOW);
        }

        if(drel2 == 1) {
                digitalWrite(REL2, HIGH);  digitalWrite(LD2, HIGH);
        } else if(drel2 == 0) {
                digitalWrite(REL2, LOW);  digitalWrite(LD2, LOW);
        }
```

```
if(Serial) {
        char ss[10];
        ss[0] = 'C';
        ss[1] = '0';
        ss[2] = '4';
        ss[3] = '|';
        ss[4] = drel1==1?'1':drel1==0?'0':'2';
        ss[5] = '|';
        ss[6] = drel2==1?'1':drel2==0?'0':'2';
        ss[7] = '|';
        ss[8] = '0';

        Serial.println(String(ss));
}
```

ANEXO 2. PROGRAMA, SOFTWARE DE MONITOREO Y CONTROL:

```
namespace ControlONOFF_Temp
{
    public partial class mainForm : Form
    {
        public mainForm()
        {
            InitializeComponent();
            this.Load += mainForm_Load;
            this.FormClosing += mainForm_FormClosing;
        }
        private void mainForm_Load(object sender, EventArgs e)
        {
            GraphPane gPane = iGraph.GraphPane;
            gPane.Title.Text = "Monitor RUBEN HUARANGA";
            gPane.YAxis.Title.Text = "Temperatura C°";
            gPane.XAxis.Title.Text = "Tiempo";
            gPane.XAxis.Type = AxisType.Date;

            RollingPointPairList list1 = new RollingPointPairList(1200);
            LineItem SPcurve = gPane.AddCurve("SP", list1, Color.Blue, SymbolType.None);
            RollingPointPairList list2 = new RollingPointPairList(1200);
            LineItem Tcurve = gPane.AddCurve("Temperatura", list2, Color.Red,
SymbolType.None);
            Tcurve.Line.Width = 2;
            Tcurve.Line.IsOptimizedDraw = true;
```

```
Tcurve.Line.IsSmooth = true;

RollingPointPairList lst1 = new RollingPointPairList(1200);
LineItem offsetA = gPane.AddCurve(" ", lst1, Color.GreenYellow, SymbolType.None);
offsetA.Line.Style = System.Drawing.Drawing2D.DashStyle.DashDotDot;
RollingPointPairList lst2 = new RollingPointPairList(1200);
LineItem offsetB = gPane.AddCurve(" ", lst2, Color.GreenYellow, SymbolType.None);
offsetB.Line.Style = System.Drawing.Drawing2D.DashStyle.DashDotDot;

RollingPointPairList list3 = new RollingPointPairList(1200);
LineItem TcurveAmb = gPane.AddCurve("T.Ambiente", list3, Color.SkyBlue, SymbolType.None);

int numberOfDivisions = 4;
gPane.XAxis.Scale.MinorStep = (int)displaySeconds / numberOfDivisions;
gPane.XAxis.Scale.MajorStep = (int)displaySeconds / numberOfDivisions;
gPane.XAxis.Type = AxisType.Date;
gPane.XAxis.Scale.Format = "mm:ss";
gPane.XAxis.Scale.MajorStepAuto = false;
gPane.XAxis.Scale.MinorStepAuto = false;
gPane.XAxis.Scale.MajorUnit = DateUnit.Second;
gPane.XAxis.Scale.MinorUnit = DateUnit.Second;
iGraph.AxisChange();

DateTime now = DateTime.Now;
iGraph.GraphPane.XAxis.Scale.Max = now.ToOADate();
```

```
            iGraph.GraphPane.XAxis.Scale.Min = DateTime.Now.AddSeconds(-
displaySeconds).ToOADate();
            iGraph.AxisChange();
            iGraph.Invalidate();

            panelDO0.Tag = "0";
            panelDO1.Tag = "0";
            buttonConnect.Click += buttonConnect_Click;
            buttonSend.Click += buttonSend_Click;
            comSerial = new iSerial.iSerial();
            comSerial.comEvent += comSerial_received;

            button1.Click += Button1_Click;
            button2.Click += Button2_Click;

            timer1 = new Timer();
            timer1.Interval = 50;
            timer1.Tick += timer1_Tick;
            timer1.Start();

            numericSetPoint.ValueChanged += numericSetPoint_ValueChanged;
            numericOffset.ValueChanged += numericOffset_ValueChanged;
        }

        private void mainForm_FormClosing(object sender, FormClosingEventArgs e)
        {
            comSerial.unconnect();
```

```
}

private void buttonConnect_Click(object sender, EventArgs e)
{
    if (buttonConnect.Tag.ToString() == "0")
    {
        comSerial.connect();
        panel3.BackColor = Color.FromArgb(255, 192, 128);

        wForm = new waitForm();
        wForm.ShowDialog(this);
    }
    else
    {
        comSerial.unconnect();
        panel3.BackColor = Color.CornflowerBlue;
        buttonConnect.Tag = "0";
        buttonConnect.Text = "Conectar";
        buttonSend.Enabled = false;

        panelDO0.BackColor = Color.CornflowerBlue;
        panelDO0.Tag = "0";
        panelDO1.BackColor = Color.CornflowerBlue;
        panelDO1.Tag = "0";

        LineItem curve1 = iGraph.GraphPane.CurveList[0] as LineItem;
        IPointListEdit list1 = curve1.Points as IPointListEdit;
```

```
            list1.Clear();
            LineItem curve2 = iGraph.GraphPane.CurveList[1] as LineItem;
            IPointListEdit list2 = curve2.Points as IPointListEdit;
            list2.Clear();
            LineItem curve3 = iGraph.GraphPane.CurveList[2] as LineItem;
            IPointListEdit list3 = curve3.Points as IPointListEdit;
            list3.Clear();
            LineItem curve4 = iGraph.GraphPane.CurveList[3] as LineItem;
            IPointListEdit list4 = curve4.Points as IPointListEdit;
            list4.Clear();
            LineItem curve5 = iGraph.GraphPane.CurveList[4] as LineItem;
            IPointListEdit list5 = curve4.Points as IPointListEdit;
            list5.Clear();

            DateTime now = DateTime.Now;
            iGraph.GraphPane.XAxis.Scale.Max = now.ToOADate();
            iGraph.GraphPane.XAxis.Scale.Min = DateTime.Now.AddSeconds(-
displaySeconds).ToOADate();
            iGraph.AxisChange();
            iGraph.Invalidate();
        }
    }

    private void buttonSend_Click(object sender, EventArgs e)
    {
        if (comSerial.isConnect)
        {
```

```
            comSerial.sendData(Convert.ToInt16(numericSetPoint.Value),
Convert.ToInt16(numericOffset.Value));
            buttonSend.Text = "Ajuste";
        }
    }

    private void Button1_Click(object sender, EventArgs e)
    {
        if (comSerial.isConnect)
        {
            if(panelDO0.Tag.ToString() == "0") comSerial.sendData(true, 1);
            else comSerial.sendData(false, 1);
        }
    }
    private void Button2_Click(object sender, EventArgs e)
    {
        if (comSerial.isConnect)
        {
            if (panelDO1.Tag.ToString() == "0") comSerial.sendData(true, 2);
            else comSerial.sendData(false, 2);
        }
    }

    private void comSerial_received(object sender, comParam param)
    {
        this.BeginInvoke(new serialReceivedCallback(serialReceivedTask), new object[] {
param });
```

```
}

private void serialReceivedTask(comParam arg)
{
    switch (arg.RcvMode)
    {
        case msgRCV.CONNECT:
            if (wForm != null) wForm.Close();

            if (comSerial.isConnect == true)
            {
                buttonSend.Enabled = true;
                panel3.BackColor = Color.FromArgb(128, 255, 128);
                buttonConnect.Tag = "1";
                buttonConnect.Text = "Desconectar";

                string[] num = arg.Data.Split(new char[] { ':' });
                if (num.Length > 1)
                {
                    numericSetPoint.Value = Convert.ToDecimal(num[0]);
                    numericOffset.Value = Convert.ToDecimal(num[1]);
                    textAmb.Text = num[2];
                }
            }
            else
            {
                buttonSend.Enabled = false;
```

```
                panel3.BackColor = Color.CornflowerBlue;
                buttonConnect.Tag = "0";
                buttonConnect.Text = "Conectar";

                panelDO0.BackColor = Color.CornflowerBlue;
                panelDO0.Tag = "0";
                panelDO1.BackColor = Color.CornflowerBlue;
                panelDO1.Tag = "0";

                MessageBox.Show("No se encuentra al equipo.\nInstale el driver adecuado y/o verifique el hardware", "Aviso", MessageBoxButtons.OK, MessageBoxIcon.Warning);
            }
            break;

        case msgRCV.RAWTEMP:
            string[] vals = arg.Data.Split(new char[] { ':' });
            if (vals.Length > 1)
            {
                textAmb.Text = vals[0];
                textSens.Text = vals[1];
                Temp = Convert.ToDouble(vals[1]);
                TempAmb = Convert.ToDouble(vals[0]);
            }
            break;

        case msgRCV.RAWRL:
```

```
        string[] stats = arg.Data.Split(new char[] { ':' });
        if (stats.Length > 1)
        {
            if (stats[0] == "0")
            {
                panelDO0.BackColor = Color.CornflowerBlue;
                panelDO0.Tag = "0";
            }
            else if (stats[0] == "1")
            {
                panelDO0.BackColor = Color.FromArgb(128, 255, 128);
                panelDO0.Tag = "1";
            }

            if (stats[1] == "0")
            {
                panelDO1.BackColor = Color.CornflowerBlue;
                panelDO1.Tag = "0";
            }
            else if (stats[1] == "1")
            {
                panelDO1.BackColor = Color.FromArgb(128, 255, 128);
                panelDO1.Tag = "1";
            }
        }
        break;
    }
```

```
        }

        private void timer1_Tick(object sender, EventArgs e)
        {
            if (comSerial.isConnect)
            {
                LineItem curve1 = iGraph.GraphPane.CurveList[0] as LineItem;
                IPointListEdit list1 = curve1.Points as IPointListEdit;
                list1.Add(DateTime.Now.ToOADate(),
Convert.ToDouble(numericSetPoint.Value));

                LineItem curve2 = iGraph.GraphPane.CurveList[1] as LineItem;
                IPointListEdit list2 = curve2.Points as IPointListEdit;
                list2.Add(DateTime.Now.ToOADate(), Temp);

                LineItem lnA = iGraph.GraphPane.CurveList[2] as LineItem;
                IPointListEdit listA = lnA.Points as IPointListEdit;
                listA.Add(DateTime.Now.ToOADate(),
Convert.ToDouble(numericSetPoint.Value) + Convert.ToDouble(numericOffset.Value / 2));
                LineItem lnB = iGraph.GraphPane.CurveList[3] as LineItem;
                IPointListEdit listB = lnB.Points as IPointListEdit;
                listB.Add(DateTime.Now.ToOADate(),
Convert.ToDouble(numericSetPoint.Value) - Convert.ToDouble(numericOffset.Value / 2));

                LineItem curve3 = iGraph.GraphPane.CurveList[4] as LineItem;
                IPointListEdit list3 = curve3.Points as IPointListEdit;
                list3.Add(DateTime.Now.ToOADate(), TempAmb);
```

```
            DateTime now = DateTime.Now;
            iGraph.GraphPane.XAxis.Scale.Max = now.ToOADate();
            iGraph.GraphPane.XAxis.Scale.Min = DateTime.Now.AddSeconds(-
displaySeconds).ToOADate();
            iGraph.AxisChange();
            iGraph.Invalidate();
          }
        }

        private void numericOffset_ValueChanged(object sender, EventArgs e)
        {
          buttonSend.Text = "Ajuste *";
        }

        private void numericSetPoint_ValueChanged(object sender, EventArgs e)
        {
          buttonSend.Text = "Ajuste *";
        }
    }
}
```