PLANTA DIDÁCTICA DE SENSORES ÓPTICOS TIPO FBG PARA LABORATORIO

ANDRÉS FELIPE SUÁREZ MOSCOSO
CAMILO ANDRÉS CORTÉS RODRÍGUEZ

REALIZADO CON LA ASESORÍA DE:

CRISTIAN ANDRÉS TRIANA INFANTE

UNIVERSIDAD EL BOSQUE
FACULTAD DE INGENIERÍA
PROGRAMA DE INGENIERÍA ELECTRÓNICA
MAYO, 2024

UNIVERSIDAD EL BOSQUE
FACULTAD DE INGENIERÍA
PROGRAMA DE INGENIERÍA ELECTRÓNICA

ÁREA DE ÉNFASIS: TELECOMUNICACIONES

PLANTA DIDÁCTICA DE SENSORES ÓPTICOS TIPO FBG PARA LABORATORIO

ANDRÉS FELIPE SUÁREZ MOSCOSO

CAMILO ANDRÉS CORTÉS RODRÍGUEZ

REALIZADO CON LA ASESORÍA DE:

CRISTIAN ANDRÉS TRIANA INFANTE

Página de Aprobación. Inclusión de Acta de grado.

NOTA DE SALVEDAD

Según el artículo 37 del 14 de diciembre de 1989 del acuerdo 017, “La Universidad El Bosque, no se hace responsable de los conceptos emitidos por los investigadores en su trabajo, solo velará por el rigor científico, metodológico y ético del mismo en aras de la búsqueda de la verdad y la justicia”.

DEDICATORIA

Este trabajo es dedicado a nuestros padres, familiares, amigos y compañeros que nos brindaron apoyo, fuerzas, su constante aliento y confianza en nosotros nos han llevado a superar los obstáculos y a alcanzar las metas propuestas tales como este proyecto de ingeniería.

AGRADECIMIENTOS

Queremos expresar nuestra gratitud a nuestro director de proyecto de grado el Profesor Cristian Andrés Triana por su orientación, apoyo, y paciencia a lo largo de todo el proceso de realización del proyecto. Su experiencia y sabiduría fueron fundamentales para la culminación exitosa de este proyecto.

A la Universidad El Bosque, aquellos docentes y personal que fueron de gran ayuda para la realización del proyecto.

RESUMEN

Los sensores de fibra óptica tipo Fiber Bragg Grating (FBG) son generalmente implementados para la medición de variables de temperatura y deformación en estructuras civiles de diferentes materiales, se usan como instrumentos de medida para obtener información que permita conocer el ciclo de vida útil de cada uno de los materiales dentro de una estructura civil [1].

El propósito del proyecto fue diseñar e implementar una planta didáctica de laboratorio que permita caracterizar el sensor óptico para la medición de las variables de temperatura y deformación. El diseño de la planta fue elaborado por medio de planos que tuvieron en cuenta medidas concretas para sus dimensiones, considerando que se compone de dos espacios que permiten establecer y desarrollar el funcionamiento de los sistemas de control. Se usan dos sistemas de control para corroborar y comparar las variaciones en temperatura y deformación medidas por medio de los sensores de fibra óptica usando la misma estructura y funcionamiento.

En este documento está plasmado el proceso de diseño del prototipo para laboratorio en el cual se ven reflejados los objetivos y requerimientos planteados, obteniendo como resultado una herramienta que permite caracterizar sensores de fibra óptica tipo FBG en función de la medición de sus variables directas que son la deformación (ε) y temperatura (T).

Palabras Clave: *Sensores FBG, Temperatura, Deformación, Micro Strain, Python.*

ABSTRACT

Keywords: FBG sensors, Temperature, Deformation, Micro Strain, Python.

Fiber Bragg Grating (FBG) fiber optic sensors are generally implemented for the measurement of temperature and deformation variables in civil structures of different materials, they are used as measuring instruments to obtain information to obtain the life cycle of each of the materials within a civil structure [1].

The purpose of the project was to design and implement a didactic laboratory plant to characterize the optical sensor for the measurement of temperature and deformation variables. The design of the plant was elaborated by using drawings that considered concrete measures for its dimensions, considering that it is composed of two spaces that allow to establish and develop the operation of the control systems. Two control systems are used to corroborate and compare the variations in temperature and deformation measured by fiber optic sensors using the same structure and operation.

In this document is reflected the design process of the laboratory prototype in which the objectives and requirements are reflected, resulting in a tool that allows characterizing fiber optic sensors FBG type based on the measurement of their direct variables which are the deformation and temperature.

GLOSARIO DE TÉRMINOS

Python: Lenguaje de programación de alto nivel utilizado para ciencia de datos, desarrollo de software, machine learning, entre otros.

Optoelectrónica: Es el estudio de los dispositivos electrónicos que interactúan con luz [2].

LISTA DE SIMBOLOS

$\mathcal{E}$: Letra de representación para la deformación mecánica en los materiales, tiene el nombre de Strain.

T: Letra de representación de la temperatura en ecuaciones.

λ_B: Longitud de onda reflejada por la red de difracción de Bragg.

η_{eff}: Índice de refracción de la fibra óptica.

Λ: Espaciamiento entre las rejillas inscritas en la fibra.

ρ_{11}: Constante de relación tensión-óptica del material con valor de 0.113.

ρ_{12}: Constante de relación tensión-óptica del material con valor de 0.252.

v: Constante asociada al material con valor de 0.16 para fibra de silicio.

η_{eff}: Constante asociada al material con valor de 1.482 para fibra de silicio.

ε_z: Deformación ejercida en el eje axial de la fibra óptica con valor en micrones.

α_Λ: Coeficiente de expansión térmico con valor de $0{,}55x10^{-6}$.

α_η: Constante termo-óptica con valor aproximado de $8{,}6x10^{-6}$.

R_T: Resistencia del termistor a una temperatura T.

R_{25}: Resistencia del termistor a una temperatura de 25 °C.

B: Valor B del termistor.

T_{NTC}: Temperatura actual del termistor en grados Kelvin.

T_{25}: Equivale a 298.15 °K.

$V_o\ tensa$: Valor de voltaje que varía dependiendo si se tensa la viga de acero.

$V_o\ no\ tensa$: Valor inicial de voltaje en el cual la viga de aluminio no se tensa.

V_{ex}: Voltaje de entrada de excitación del Puente de Wheatstone.

R_L: Es la resistencia del cable.

ρ: Es la resistividad.

l: Es la longitud del cable.

a es el área transversal del cable.

R_G: Resistencia de la galga extensiométrica.

V_R: Tensión relativa.

GF: Factor Galga.

LISTA DE ABREVIATURAS

FBG: Fiber Bragg Grating en inglés, Red de Difracción de Bragg en español.

PTC: Positive Temperature Coefficient en inglés, Coeficiente de Temperatura Positivo en español.

NTC: Negative Temperature Coefficient en inglés, Coeficiente de Temperatura Negativo en español.

PWM: Pulse Width Modulation en inglés, Modulación por Ancho de Pulso en español.

API: Application Programming Interface en inglés, Interfaz de Programación de Aplicaciones en español.

TABLA DE CONTENIDOS

Pág.

LISTA DE TABLAS

Tabla Pág.

LISTA DE FIGURAS

Figura Pág.

1. INTRODUCCIÓN

En el ámbito de la ingeniería, los sensores de fibra óptica emergen como una tecnología que ha servido como herramienta importante para la ingeniería moderna. A pesar de que su implementación en el campo empezó en la década de 1970, todavía se encuentra en desarrollo y crecimiento, especialmente cuando se requiere un alto nivel de precisión en mediciones [3]. En consecuencia, hay escenarios en los cuales resulta más económico y viable hacer uso de sensores eléctricos para medir variables de temperatura y deformación como alternativa en proyectos de implementación. Sin embargo, los sensores de fibra óptica presentan ventajas significativas frente a los sensores eléctricos. Entre ellas se destaca su notable capacidad para resistir interferencias electromagnéticas, adaptación a condiciones ambientales poco favorables, y una excelente relación fiabilidad - estabilidad a largo plazo [4]. Estas cualidades mencionadas anteriormente son las que han inspirado en el desarrollo del proyecto, buscando aprovechar al máximo las cualidades del sensor de fibra óptica para un proyecto de ingeniería con aplicación en el entorno académico.

Este proyecto propone el diseño y construcción de una planta didáctica con sistemas de control para la medición de deformación y temperatura. Incluye galgas extensiométricas y sensores de temperatura para obtener la medición de cada sistema, así como sus respectivas tarjetas de control, ambas programadas en MicroPython.

Este documento se divide tres fases. La primera fase, que aborda el diseño, implica el boceto de la planta didáctica y los sistemas de control, teniendo en cuenta los requerimientos y limitaciones para el desarrollo de cada sistema. La fase dos, de implementación, abarca la implementación de los sistemas electrónicos, junto con pruebas de funcionamiento que fueron diseñadas previamente en la fase uno. Por último, la fase de obtención de datos de los sistemas, su análisis correspondiente y el procesamiento de las variables de deformación y temperatura que se utilizaron para observar y comparar el funcionamiento de los sensores de fibra óptica de tipo FBG.

2. PLANTEAMIENTO DEL PROBLEMA

2.1 Antecedentes y Estado del Arte

2.1.1 Nivel Internacional

En 2010, en el Instituto de Tecnología de Dublín se presenta un artículo donde los sensores de fibra óptica de tipo FBG hacen parte de un sistema de detección para aplicaciones de corte tele robóticas. En este artículo se incluye el tipo de filtro denominado macrobend que consiste en la utilización de dos sensores FBG para la obtención de las medidas de Strains y para la compensación de la temperatura. Gracias a este filtro se logra pasar por alto la influencia de la temperatura para las medidas de Strains en el sensor FBG. Dichas medidas del sensor FBG fueron comparadas con una galga extensiométrica donde se concluye que el sistema de detección por sensores FBG con el filtro macrobend logra resultados competitivos y precisos para aplicaciones de cortes quirúrgicos [5].

En el Instituto Tecnológico de Tuxtla Gutiérrez México en el año 2020 se realizó una tesis donde se plantea un dispositivo a base de fibra óptica para el monitoreo de deformación en vigas estructurales en grandes claros. Esta tesis para maestría en ciencias en Ingeniería Mecatrónica muestra con mucha importancia como las estructuras civiles requieren de un monitoreo constante de salud estructurales, esto con el fin de analizar su vida útil. Para ello da a entrever que se han desarrollado trabajos importantes para realizar los monitoreos haciendo uso de sensores de fibra óptica y otros tipos de sensores [6]. En el documento se menciona la importancia de los sensores FBG y cómo es su uso para medir y monitorear la vida útil de vigas en construcción en materiales como madera, acero, hierro y concreto. Además, deja ver métodos alternos que compiten para poder medir estas deformaciones con herramientas mucho más económicas como los son las galgas extensiométricas, sensores de tensión y deformación. Identifica como se debe medir la deformación haciendo uso de sensores de fibra, como se debe compensar la medición en caso de pérdidas por temperatura, y los factores que deberá tener el material para ser medido en el modelo experimental, teniendo en cuentas de la fibra que tiene sus ventajas y desventajas, y que

además se debe tener cuidado con las pérdidas que pueden tener debido a radiación, ambiente, reflexión perdidas intrínsecas y extrínsecas, factores bastante controlables [6].

En la Universidad Politécnica de Valencia España, en el año 2023 se desarrolló una tesis en la cual se tiene como objetivo desarrollar un sensor óptico para detección de fugas en tuberías detectando las mismas por medio de vibraciones. Referenciando e investigando previamente sobre sensores de fibra óptica se da cuenta de la cantidad de aplicaciones que puede tener esta herramienta, y centra su desarrollo en estos sensores para detección de fugas. Dentro de la investigación indaga el marco conceptual de las tecnologías relacionadas con redes de dispersión de Bragg y modelos de vibración [7]. El autor establece un sistema modelado entre los sensores y algunas piezas mecánicas para implementar un análisis modal y paramétrico en el software de simulación llamado ANSYS. Detalla como dentro de este modelado fabrica el sensor, el montaje del sistema de medición, y realiza las pruebas experimentales para analizar frecuencias naturales y compara con simulaciones hechas previamente, obteniendo como resultado un sensor óptico capaz de detectar de forma sensible y eficaz fugas en tuberías [7].

2.1.2 Nivel Nacional

En 2010, se presenta la tesis titulada medición de microdeformaciones en losas viales usando sensores de redes de Bragg en fibras ópticas realizada en la Universidad EAFIT. Esta se basa en una investigación que realizan dos estudiantes y un docente, sobre un método importante para medir microdeformaciones en estructura de concreto, es especifico losas viales. Dentro de su campo de investigación ellos plantean el uso en la superficie de las losas viales sensores de fibra óptica, planteando el principio de funcionamiento de las redes de Bragg donde su punto más importarte es útil implementación para medir deformaciones longitudinales con una alta resolución [8]. Plantean la creación de un modelo de losa vial, con 2 sensores de fibra para medir una carga estática de 10kN, con una presión media de 250kPa, y un sensor de fibra de temperatura, que logra medir la temperatura del modelo experimental. Dentro de la aplicación y resultados se logra visualizar unos muy buenos resultados, todos con una pequeña variación respecto a la posición del sensor, esto debido a que hubo un contacto irregular entre la llanta del auto

pasando por la losa vial y el sensor. Destacando al final que se pueden usar los sensores de fibra como alternativa para la identificación de sobrecargas que causan deterioros y daños en las mayas viales [8]. Importante hay que destacar que este estudio reporte es uno de los primeros en realizarse a nivel nacional en Colombia.

En 2012 se presenta un trabajo de investigación en la Universidad Nacional de Colombia donde se realiza la medición y obtención de las medidas de deformación, se considera la intervención de la temperatura cuando afecta directamente la longitud de onda medida por los sensores de fibra óptica tipo FBG. En este trabajo se incluye la utilización de galgas extensiométricas y un extensómetro donde se comparan sus respuestas a deformaciones mecánicas presentes en una lámina metálica para una prueba inicial y en una segunda prueba se analiza el comportamiento en una varilla estructural [9].

En la Universidad Escuela de Ingeniería de Antioquia, en el año 2015 se plantea un esquema para la implementación de medición de deformaciones en edificaciones de hormigón, donde la tesis de grado tiene marcado como objetivo principal proponer un esquema que sea capaz de monitorear deformaciones en edificaciones a porticadas en hormigón [10]. Este estudio se presenta como un análisis de técnicas de instrumentación estructural, centrándose en el uso de galgas extensiométricas, sensores de fibra óptica, y otros dispositivos capaces de medir deformaciones en estructuras de concreto. Lo primero que realiza es el diseño experimental en donde buscan analizar la viga de concreto a deformar y determinar cuáles serán las cargas para utilizar sobre esta misma, definiendo el uso de sensores de fibra y galgas extensiométricas como principales herramientas de medición, siendo las FBG las que tuvieron un peso específico en la calibración para poder tener medidas más exactas. Al finalizar y concluir el experimento se resalta la gran importancia que tienen los sensores de fibra óptica, y su tendencia a usarse en mediciones de concreto reforzado, siendo así aplicados en su mayoría para evaluar los modelos de estructuras civiles [10].

Una de las características de los sensores de fibra óptica tipo FBG es la capacidad de medir temperatura, esta se logra mediante la lectura de la longitud de onda reflejada por la red de

difracción de Bragg (FBG) que luego mediante una ecuación se representa dicho valor de temperatura. Teniendo en cuenta que poseen inmunidad a las interferencias electromagnéticas, no se calientan lo que no genera pérdidas en el sistema y tienen una respuesta lineal a la temperatura, los convierte en sensores ampliamente utilizados para el monitoreo de estructuras civiles y el monitoreo de líneas de transmisión [11]. En 2019 se expone una propuesta de monitoreo de temperatura de un segmento a escala de las líneas de transmisión donde se realiza la implementación de sensores FBG teniendo en cuenta las condiciones de operación de las líneas de transmisión en Colombia, planteando un método de adhesión de los sensores a la línea y a su vez se bosqueja un sistema de medición de energía.

En la Universidad Libre Seccional Cúcuta, se realiza en el año 2022 un análisis a la viabilidad para la implementación de redes de sensores de fibra óptica en el sector industrial de San José de Cúcuta donde la investigación aborda como principal factor la viabilidad tecnológica de implementar redes de sensores de fibra óptica (FBG) en el sector industrial de la ciudad de San José de Cúcuta, se plantea como idea y análisis principal un enfoque cuantitativo donde se busca invitar a las empresas a adoptar el uso de sensores de fibra óptica. Esto debido a que la tecnología en Colombia es nueva, pero tiene una investigación teórica y diagnóstico bastante amplio para poder identificar las necesidades y las ventajas de hacer uso de los sensores de fibra óptica. Dentro del marco de investigación el autor propone actividades de divulgación y apropiación de la tecnología, esto con el fin de generar un gran impacto en la región, así mencionando todas las ventajas de los sensores de fibra óptica, y cómo estas sirven como propuesta para que muchas industrias busquen mejorar sus procesos de producción, y se ayude también a las futuras investigaciones que esta pueda llegar a provocar [12]. Dentro del análisis investigativo que realiza Diego Almeida, es importante e interesante conocer como busca desde un método de enfoque y desarrollo, poder explorar y dar a conocer el uso de los sensores de fibra óptica a nivel nacional e internacional, como estos han sido aplicados en industrias del sector civil, energético, aeronáutico y petrolero, destacando las otras aplicaciones que pueden llegar a tener los sensores y que desarrollo pueden tener en las industrias de la ciudad de Cúcuta, y también a nivel nacional [12].

2.2 Descripción y Formulación del Problema

Los sensores de fibra óptica, con su uso e innovación, abren un amplio espectro de posibilidades para asumir roles protagónicos en investigaciones y aplicaciones. Estos sensores se destacan por su versatilidad y capacidades excepcionales. En el ámbito académico, resulta esencial explorar a fondo estos dispositivos cuando forman parte de los insumos en laboratorios o prácticas de investigación. Esto no solo permite su estudio detallado, sino que también facilita la comprensión de sus capacidades y la caracterización de su funcionamiento, convirtiéndose en el foco central de la presente investigación.

Para el campo de investigación en Colombia, no hay ningún dispositivo y/o herramienta que permita comparar el funcionamiento de los sensores de fibra óptica tipo FBG con sensores de señales eléctricas como lo son las galgas extensiométricas y los sensores de temperatura.

Como solución a esta problemática, se plantea diseñar e implementar una planta didáctica de laboratorio que le permita a los estudiantes, profesores, egresados o a quien le interese saber sobre estos sensores, ya sea conocer su funcionamiento, validar el tipo de características que tiene, poder comparar con sensores análogos que tan fiable y precisa es su medición, con el fin de que se pueda fomentar la investigación y la práctica, y con ello invitar a que más personas se sumen al campo de la sensórica y de la optoelectrónica.

2.3 Justificación

La problemática expuesta presenta la necesidad de la creación de un componente, prototipo y/o dispositivo que aporte una mejor visión de lo que son los sensores de fibra óptica de tipo FBG logrando así ampliar su campo acción en el área de la instrumentación y optoelectrónica, lo que resulta en una comprensión más precisa sobre el principio físico y los fenómenos presentes en dichos sensores que interactúan con luz.

Es por esto por lo que se ha propuesto y se ha desarrollado la idea de este proyecto de ingeniería en el cual mediante una planta didáctica se evidencie el funcionamiento y comprobación de los sensores de fibra óptica tipo FBG utilizando sistemas con sensores eléctricos para deformación y temperatura respectivamente. Mediante la utilización de estos

sistemas se comprobarán las medidas que brindan los sensores FBG verificando las mediciones obtenidas.

2.4 Objetivos

2.4.1 Objetivo General

Desarrollar una planta didáctica de sensores ópticos tipo FBG para laboratorio.

2.4.2 Objetivos Específicos

1. Construir una estructura rígida donde se puedan posicionar los sensores, la cual debe ser apta para el laboratorio.
2. Diseñar e implementar un control de temperatura en la estructura que permita calentar la sección de los sensores en un intervalo de 30° a 50° Celsius.
3. Diseñar e implementar un sistema de control de deformación mecánica utilizando un motor que permita deformar una sección de la estructura.
4. Validar el funcionamiento de los sensores ópticos FBG estableciendo la relación entre longitud de onda reflejada y temperatura/deformación dependiendo el caso.

2.5 Alcance y Limitaciones del Proyecto

Como bien se menciona en el objetivo general, el proyecto está dirigido hacia el diseño y construcción de una planta didáctica o prototipo de laboratorio en el cual los estudiantes, profesores y egresados puedan comprobar experimentalmente el funcionamiento de los sensores ópticos tipo FBG.

Al ser un prototipo de laboratorio el cual estará en uso con personas, tendrá un diseño robusto el cual sea resistente, pero a su vez que permita identificar cada sistema en sus respectivas secciones. Para este prototipo se tendrá en cuenta una investigación sobre sistemas de temperatura y deformación, investigaciones sobre tarjetas de control tales como Arduino, Raspberry Pi, ESP32, Raspberry Pi Pico W y demás tarjetas las cuales nos ayudarán a plantear los sistemas de control para cada sección presente en la planta didáctica.

Estas son las limitaciones presentes:

- Debido a que el prototipo será diseñado para un uso en laboratorio, sus medidas máximas son de 1.5 metros x 1.5 metros.
- El prototipo será diseñado para un laboratorio, no se contemplará su uso expuesto a la intemperie.
- Los sensores ópticos FBG no son de uso exclusivo de este proyecto, ya que son de propiedad de la Universidad El Bosque y de la Universidad Nacional, estos estarán a disposición para otros grupos de investigación y personal que los requiera.

3. MARCO DE REFERENCIA

3.1 Marco Teórico o Conceptual

3.1.1 Sensores ópticos tipo FBG

Estos sensores son estructuras construidas en el interior de una fibra óptica. Estas estructuras están basadas en lo que se conoce como red de difracción de Bragg (FBG); que consiste en una perturbación periódica del índice de refracción a lo largo de la fibra. Esta perturbación es generada por la exposición del núcleo a un patrón de interferencia intensivo [13]. Las perturbaciones generadas tienen una relación periódica de distancia, cuando esta distancia cumple el parámetro de Bragg, el dispositivo adopta un comportamiento de filtro rechaza banda en modo de transmisión y como resultado se ve reflejada una longitud de onda en la fibra óptica. Las variables que se pueden medir directamente con los sensores FBG son las deformaciones mecánicas y la temperatura [14].

El comportamiento de los sensores FBG está descrito por:

$$\lambda_B = 2\eta_{eff}\Lambda \quad (1)$$

En donde λ_B es la longitud de onda reflejada por la red de difracción de Bragg, η_{eff} es el índice de refracción de la fibra óptica y Λ es el espaciamiento entre las rejillas inscritas en la fibra.

La ecuación (2) relaciona cambios de elongación y temperatura a los que se somete la fibra óptica, con longitud de onda reflejada en el sensor.

$$\Delta\lambda_B = 2\left(\Lambda\frac{\partial\eta_{eff}}{\partial l} + \eta_{eff}\frac{\partial\Lambda}{\partial l}\right)\Delta l + 2\left(\Lambda\frac{\partial\eta_{eff}}{\partial l} + \eta_{eff}\frac{\partial\Lambda}{\partial l}\right)\Delta T \quad (2)$$

La ecuación (3) representa la relación entre longitud de onda reflejada y las deformaciones mecánicas. Se necesita utilizar (4) para hallar el valor de (3).

$$\Delta\lambda_B = \lambda_B(1-\rho_e)\varepsilon_z \quad (3)$$

$$\rho_e = \frac{\eta_{eff}^2}{2}[\rho_{12} - v(\rho_{11} + \rho_{12})] \quad (4)$$

En donde ρ_{11}, ρ_{12} son constantes de relación tensión-óptica del material que llegan a tener valores típicos de 0.113 y 0.252 respectivamente, v, η_{eff} son constantes asociadas al material y tienen valores de 0.16 y 1.482 para la fibra de silicio, ε_z siendo el término de deformación ejercida en el eje axial de la fibra óptica y posee valores en micrones.

La ecuación (5) establece la relación presente entre la longitud de onda y el cambio de temperatura a lo largo de la fibra óptica, dando como resultado un valor expresado en pm/°C. Este valor representa el cambio de la longitud de onda reflejada por cada grado Celsius, en otras palabras es la afectación de la temperatura experimentada por el sensor en el cambio de longitud de onda demostrando así una sensibilidad dependiente [14].

$$\Delta\lambda_B = \lambda_B(\alpha_\Lambda - \alpha_\eta)\Delta T \quad (5)$$

En donde α_Λ es el coeficiente de expansión térmico del material, que tiene un valor de 0,55x10^-6, α_η es la constante termo-óptica que posee un valor aproximado de 8,6x10^-6 y λ_B es la longitud de onda reflejada por la red de difracción de Bragg del sensor en cuestión.

3.1.1.1 Interrogador Óptico

El interrogador óptico es un dispositivo electrónico de medición el cual consiste en la emisión de un haz de luz en un rango pequeño de longitudes de onda hacia el y/o los sensores de fibra óptica conectados a este. El interrogador recibe la longitud de onda reflejada por los sensores y transforma los valores recibidos para luego ser visualizados en una interfaz de software.

El Interrogador Óptico SM125 de Micron Optics tiene características como un láser de alta potencia con barridos de longitud de longitud de onda de bajo ruido logrando así un dispositivo altamente preciso, barridos de longitud de onda con bajo ruido haciéndolo un instrumento altamente preciso y más características que se exponen en las tablas 1 y 2 demostrando que es un equipo robusto y fiable en aplicaciones de uso prolongado.

Tabla 1. Propiedades de rendimiento de interrogador óptico SM125 [15].

Propiedades de rendimiento	SM125
Canales ópticos	4
Frecuencia de escaneo	2 Hz
Rango de longitud de onda	1510-1590 nm
Estabilidad de longitud de onda; exactitud	1; 1pm
Repetibilidad de longitud de onda	0.5 pm a 1 Hz, 0.2 pm a 0.1 Hz
Rango dinámico	50 dB
Capacidad típica de sensor FBG	60-120
Medición de espectro completa	Incluido
Sm041 switch compatible	Si
Conectores ópticos	FC/APC

Tabla 2. Interfaces y Software del interrogador óptico SM125 [15].

Interfaces y Software	SM125
Interfaces	Ethernet
Gestión de datos	ENLIGHT Sensing Analysis Software
Software remoto	Análisis del espectro, detección de pico, registro de datos, rastreador de picos y control de instrumento.

Figura 1. Interrogador Óptico Micron Optics SM125 [15].

3.1.2 Temperatura

La temperatura es conocida como la magnitud física que representa el grado de calor y/o frío presente en cuerpos o en el ambiente [16]. Por lo general se mide utilizando dispositivos como sensores resistivos y termómetros.

3.1.2.1 Actuador

Uno de los objetivos específicos consistía en realizar un sistema de control de temperatura que lograra calentar una sección del prototipo en un rango de 30° a 50° Celsius, por lo que se necesitó realizar una búsqueda de aquellos dispositivos o componentes eléctricos que permitieran realizar dicho proceso, entre ellos se encuentran las celdas peltier y los cartuchos calefactores presentes en las impresoras 3D.

Las impresoras 3D poseen un componente denominado Hotend Volcano, este es un conjunto de piezas las cuales son utilizadas para fundir y extruir el filamento utilizado para las impresiones. Trabaja en conjunto con el extrusor que es el encargado de empujar el filamento al bloque donde este es fundido para luego ser empujado por la misma presión a través de una boquilla con el fin de ser puesto en la cámara de construcción [17].

Estos Hotend poseen un bloque tipo Volcano que es bloque donde se funde el material, a su vez poseen un sensor de temperatura que es encargado de medir la temperatura a la cual se está fundiendo el material. Para calentar el bloque, este utiliza un cartucho calefactor que básicamente es una resistencia y su voltaje de alimentación es de 12 a 24V con una potencia de 40W [17].

Figura 2. Hotend volcano, incluye NTC B3950 100K de color blanco, cartucho calefactor y bloque de aluminio [18].

3.1.2.2 Sensor de temperatura

Los sensores de temperatura son componentes eléctricos, por lo general son resistencias variables con la temperatura que permiten medir una señal eléctrica determinada a partir de su variación de resistencia. Existen dos tipos de sensores de temperatura resistivos, los

termistores PTC (Positive Temperature Coefficient) y los termistores NTC (Negative Temperature Coefficient) [14].

El termistor NTC es un termistor que posee un coeficiente de temperatura negativo. Lo que su acrónimo representa es que la resistencia del termistor disminuye cuando la temperatura aumenta, siendo así una relación inversamente proporcional entre temperatura y resistencia [19].

Existen los siguientes tipos de termistores NTC:

- Tipo disco: son utilizados en DC (Corriente Directa) o en AC (Corriente Alterna). Su constante de tiempo es de 30 a 120 segundos.
- Serie 642 NTC: utilizados para compensación de temperatura, poseen una disipación máxima de 0,5 W. Operan entre -25° a 125° C.
- NTC de vidrio miniaturas radiales: poseen una mayor resistencia y capacidad de control de temperatura de hasta 300° C. Se caracteriza por brindar una respuesta rápida [20].

Figura 3. Termistor NTC 100K [21].

Este termistor es altamente sensible, y la gráfica de los valores de resistencia con respecto a la temperatura no es una gráfica lineal, dando así paso a ecuaciones para tener la obtención de su valor de resistencia.

3.1.2.2.1 Principio de funcionamiento de los termistores NTC

La ecuación de los termistores es una representación de la resistencia del termistor a una temperatura t[22]. Esta ecuación es suministrada por los fabricantes de termistores comúnmente encontrada en los datasheets.

La ecuación de los termistores es:

$$R_T = R_{25} e^{\left(\frac{B}{T_{NTC}} - \frac{B}{T_{25}}\right)} \quad (6)$$

En donde R_T es la resistencia del termistor a una temperatura T, R_{25} es la resistencia del termistor a una temperatura de 25°C, B es el valor B del termistor incluido en el datasheet de cada termistor, T_{NTC} es la temperatura actual del termistor en grados Kelvin y T_{25} tiene un valor de 298,15 °K.

Con el uso de esta ecuación, se podrán obtener las variables necesarias para realizar el sistema de control de temperatura despejando en términos de la temperatura. Este procedimiento se explicará en la sección 4.3 de este documento.

3.1.2.3 Controlador PID

Un controlador proporcional, integrador y derivativo es un método utilizado comúnmente en sistemas donde se requiere realizar el control automático de una variable en específico teniendo en cuenta un valor deseado. Consta de 3 factores que son K_p que representa el valor proporcional y consiste en la multiplicación del error con el valor deseado, T_i siendo el factor integral que implica la suma del error actual durante un periodo de tiempo predeterminado y T_d refiriéndose al derivador que involucra la variación del error en el tiempo [23].

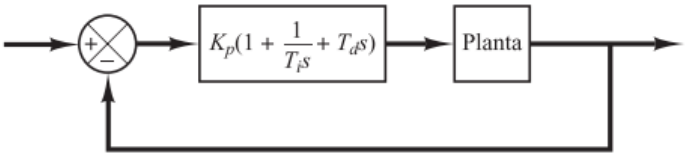

Figura 4. Control PID de una planta [23].

3.1.3 Deformación

Es el cambio de forma que experimenta un material al aplicarle una fuerza. Se mide como el cambio de longitud que produce dicha fuerza [24]. En este proyecto, la deformación se utilizará para determinar cómo cambia un material metálico al aplicar una fuerza de tensión variable con un motor eléctrico.

3.1.3.1 Sensor de deformación

En electrónica la deformación se puede leer como el cambio en resistencia cuando se ejecuta una fuerza sobre una estructura sólida y el sensor ligado a esta logra medir ese cambio de forma rápida. Para ello, los sensores tienen que adaptarse y escogerse según la necesidad del usuario, depende si el área a deformar será pequeña o si al contrario se debe deformar y medir un área bastante amplia, por esa razón para este proyecto el término de deformación será bastante importante [25]. Para el proyecto se optó por el uso de galgas extensiométricas, sensores eléctricos ideales para la medición necesaria del sistema de deformación implementado.

3.1.4 Drivers de motores

Son componentes electrónicos encargados de coger una señal eléctrica generada por un sistema embebido y amplificarla. Esto con el fin de transmitirle al motor una señal de control de corriente que este pueda manejar. También son conocidos bajo el nombre de Puente H. Por lo general, se utiliza el PWM (Pulse Width Modulation) para controlar los motores, controlar una señal que se le suministra a un componente donde puede ser un bombillo o un cartucho calefactor [26]. Algunos de los drivers para motores más conocidos son:

- Driver L298N
- BTS7960
- L293D
- DRV8833

3.1.4.1 Driver L298N

Este módulo es característico en las aplicaciones que se requieran el uso de dos motores. Su voltaje de alimentación va desde 5V hasta 35V, pero este voltaje lo escoge el usuario con el fin de limitar la corriente que llega a entregar el controlador. Posee los pines ENA, ENB, IN1, IN2, IN3 y el IN4. El ENA y el ENB son los encargados de recibir las señales PWM para así controlar el motor. Como es de dos motores, los pines ENA, IN1 y el IN2 son del motor A y los pines ENB, IN3 y el IN4 son del motor B. Puede usarse en sistemas embebidos como lo son las Raspberry, Arduino, básicamente cualquier tarjeta que tenga la capacidad de manejar PWM [27].

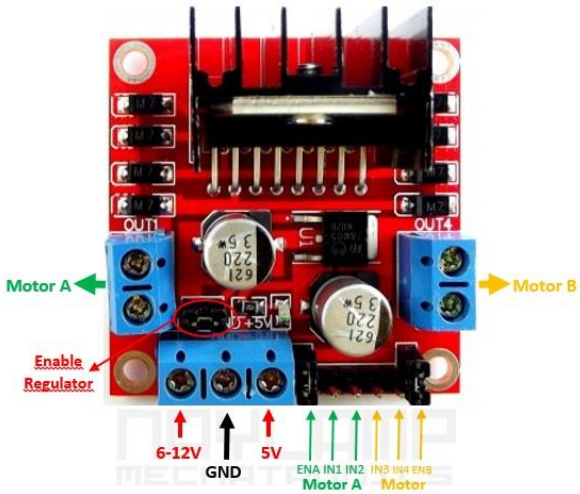

Figura 5. Vista superior del módulo L298N [27].

3.1.4.2 Driver BTS7960

Este driver o controlador es conocido por su gran capacidad de soportar un motor de hasta 43 amperios. Su voltaje de alimentación es desde 5.5V a 27V. A su vez, posee un valor máximo de frecuencia para el PWM de 25KHz. Es utilizado en Arduino, Raspberry, o cualquier sistema embebido capaz de manejar PWM. Se caracteriza por poseer un chip digital integrado que tiene como objetivo la gestión de la comunicación con su tarjeta de control [28].

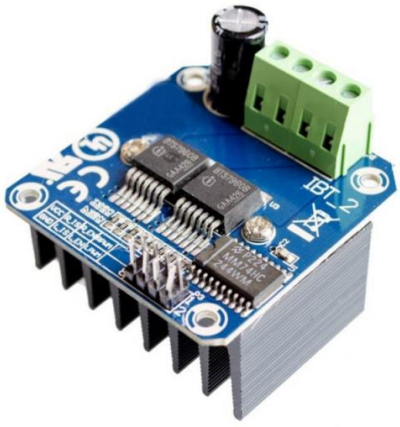

Figura 6. Driver para motor BTS7960 de 43A [25].

3.1.5 Microcontroladores y adquisición de datos

3.1.5.1 Tarjetas de control

Son aquellos dispositivos electrónicos de bajo costo, capaces de ser programados para efectuar tareas específicas, utilizan lenguajes de programación como Python, C++, C, y demás lenguajes de alto nivel. Mediante dichos códigos, se les asignan funciones las cuales son guardadas en las memorias de estos microcomputadores. En conjunto, poseen un microprocesador, memorias RAM y ROM, un grupo de entradas y salidas capaces de programarse dependiendo de la necesidad del programador y pines de alimentación para circuitos comúnmente conocidos como periféricos [30].

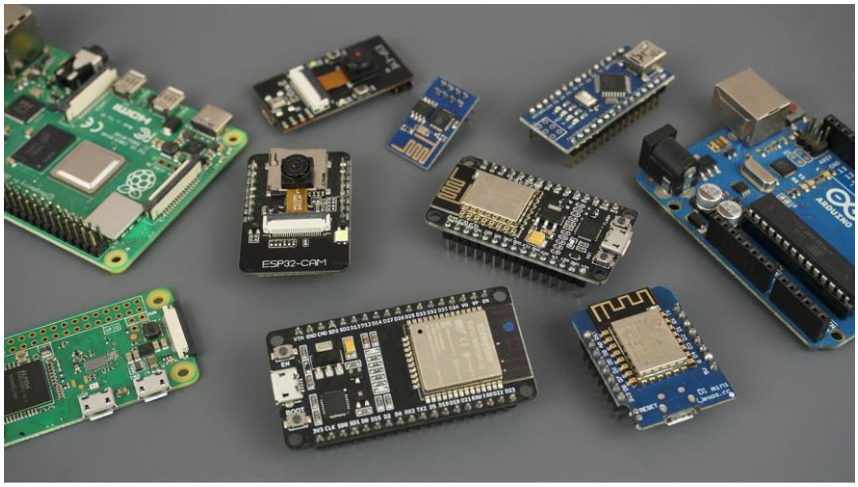

Figura 7. Diferentes microprocesadores. ESP32, Raspberry Pi, Arduino [31].

3.1.5.1.1 Raspberry Pi Pico W

La Raspberry Pi Pico W es una tarjeta embedida de bajo costo, posee Wifi y Bluetooth. Su lenguaje de programación es Python, posee 26 pines GPIO multifuncionales [32]. Es una tarjeta muy funcional ya que a comparación de la Raspberry Pi Pico 4 Model B, esta posee un conversor análogo-digital (Analog-Digital Converter) incorporado. Para usar un ADC externo en la Raspberry Pi 4 toca adquirir un módulo, los más usados son los ADS1015 (Conversor de 12 bits) y el ADS1115 (Conversor de 16 bits).

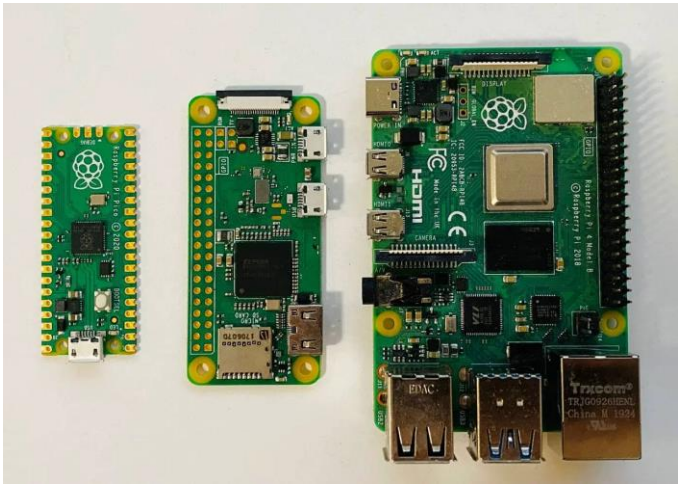

Figura 8. Raspberry Pi Pico, Raspberry Pi Zero, Raspberry Pi 4 Model B [33].

3.1.5.1.2 ESP32

ESP32 es una tarjeta embebida multipropósito que tiene como ventaja ser altamente versátil, además del bajo costo de la mismo, es muy utilizada en el campo de la electrónica, así como también lo es para proyectos de IOT. Esta tarjeta permite amplia gama de funcionalidades donde encontramos un procesador doble núcleo muy capaz de realizar tareas de forma óptima, así como su número significativo de puertos GPIO que funcionan como salidas PWM, entradas ADC, puertos que permiten interacción con la mayoría de los componentes electrónicos, entre ellos sensores, pantalla, actuadores y demás. Tiene también una amplia gama de lenguajes de programación que hacen que su entorno de

desarrollo sea bastante robusto, y que adicional a eso la hace una alternativa muy usada por la comunidad estudiantil.

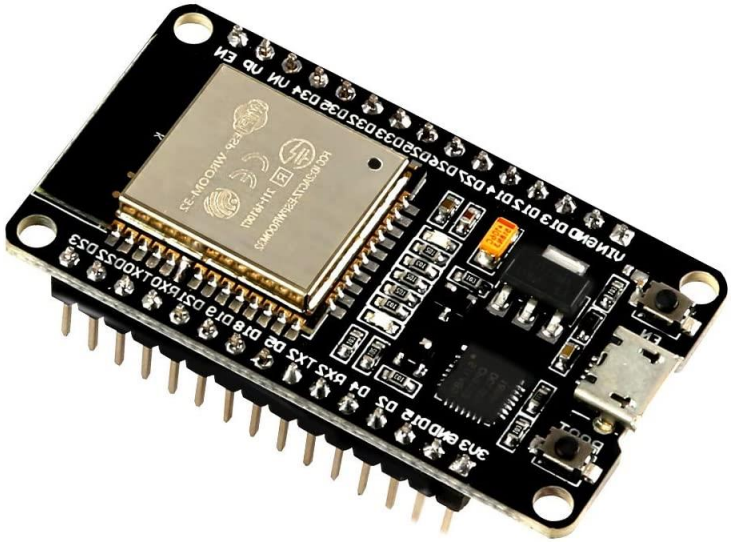

Figura 9. Microcontrolador ESP32 [34].

3.1.5.2 Puente de Wheatstone

El puente de Wheatstone constituye un circuito conformado por cuatro resistencias conectadas a una fuente de alimentación. Es relevante destacar que dentro de este circuito existe una resistencia cuyo valor aún no se conoce. Sin embargo, gracias a la implementación de este circuito, podemos determinar su valor mediante el planteamiento y resolución de ecuaciones. Este proceso brinda una herramienta valiosa para la identificación precisa de la resistencia desconocida en el sistema.

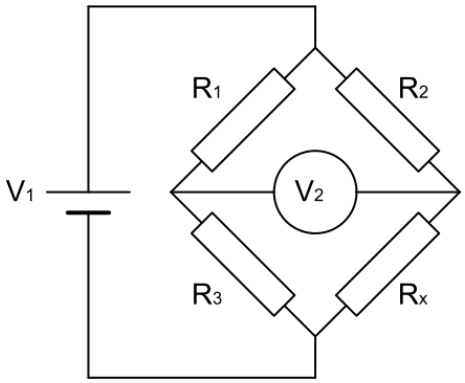

Figura 10. Puente de Wheatstone [35].

El puente de Wheatstone utiliza (7) para hallar R_x basándonos en el circuito de la figura 10.

$$R_x = \frac{R_2 * R_3 + \frac{R_3 * (R_1 + R_2) * V_2}{V_1}}{R_1 - \frac{V_2 * (R_1 + R_2)}{V_1}} \quad (7)$$

3.1.5.3 Conversor análogo digital

3.1.5.3.1 ADS1115

Los conversores análogo-digitales son módulos creados específicamente para convertir señales a señales que puedan ser procesadas por un microcontrolador ya sea una ESP32 o una Raspberry Pi Pico W. Los conversores ADS1115 tienen una resolución de 16 bits haciéndolo un módulo utilizado para proyectos donde se requiera alta precisión. Posee 4 entradas análogas y su protocolo de comunicación con los microcontroladores es mediante I2C [36].

Tabla 3. Características del conversor ADS1115.

Característica	**Valor**
Voltaje de alimentación	2.0V a 5.5V
Consumo de corriente en modo continuo	0.15mA
Tasa programable de datos	8 muestras por segundo hasta 860 muestras por segundo
Resolución	16 bits
Comunicación	I2C

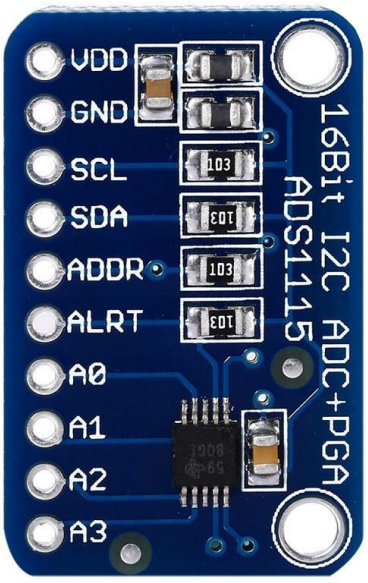

Figura 11. Conversor ADC ADS1115 [34].

4. DESARROLLO DEL PROYECTO DE GRADO

4.1 Requerimientos

4.1.1 Requerimientos Funcionales

- Debe construirse una estructura de laboratorio en aluminio con paredes hechas en acrílico con el fin de tener un buen material conductor de temperatura y a su vez liviano.
- La estructura debe ser rígida y apta para laboratorio.
- La estructura debe contar con un espacio para alojar los circuitos de control y otros componentes.
- Debe diseñarse e implementarse un sistema de control de temperatura que permita calentar una sección en la estructura donde el sensor pueda medir en un rango de 30° a 50° Celsius.
- El sistema de control de temperatura debe ser ajustable por el usuario en el rango definido anteriormente.
- Debe diseñarse e implementarse un sistema de control de deformación mecánica usando un motor que permita deformar una sección de la estructura.
- El motor debe ser controlable y ajustable en al menos tres posiciones de movimiento.
- Debe establecerse la relación entre la longitud de onda reflejada y la temperatura/deformación según corresponda.

4.1.2 Requerimientos No Funcionales

- La estructura debe construirse principalmente en aluminio con paredes de acrílico.
- Los materiales deben ser duraderos y resistentes a las condiciones de un laboratorio.
- La estructura debe ser diseñada de manera que las partes esenciales, como la viga en voladizo y otros componentes, puedan desmontarse y reemplazarse fácilmente.
- El sistema de control de temperatura y deformación debe buscar ser lo más preciso posible.

- Los sensores análogos y ópticos deben ser caracterizados/calibrados para ofrecer una medición estable.
- Los componentes electrónicos y sistemas de adquisición de datos deben ser compatibles con las tarjetas ESP32 y Raspberry Pi Pico.
- Los datos medidos serán enviados por medio de protocolo Wifi, y serán recibidos por un servidor local desplegado en Python.
- Los datos medidos deben registrarse y almacenarse en un archivo CSV de forma que sea posible procesarse la información obtenida.

4.1.3 Requerimientos de restricción

- La planta didáctica será usada solamente en espacios de laboratorio.
- La planta didáctica tendrá como medidas máximas 1.5 metros x 1.5 metros (Alto x Ancho).
- Los sensores ópticos tipo FBG no son de uso exclusivo para la planta didáctica.

4.1.4 Requerimientos de calidad

- Las mediciones de temperatura, deformación y de los sensores ópticos debe ser precisas con un margen de error mínimo.
- La planta debe ser duradera y capaz de soportar un uso prolongado en el laboratorio.
- Los controles y ajustes del sistema de control de temperatura y deformación deben ser fáciles de usar para el usuario final.
- La planta debe ser diseñada de manera que permita un mantenimiento y reemplazo de componentes de forma sencilla y fácil.

4.2 Metodología del Diseño

4.2.1 Diseño de ingeniería conceptual de sistemas de control

Teniendo en cuenta la problemática presente y los requerimientos establecidos anteriormente, se plantea el siguiente diagrama de bloques que constituye el proyecto de forma detallada con sus entradas y salidas.

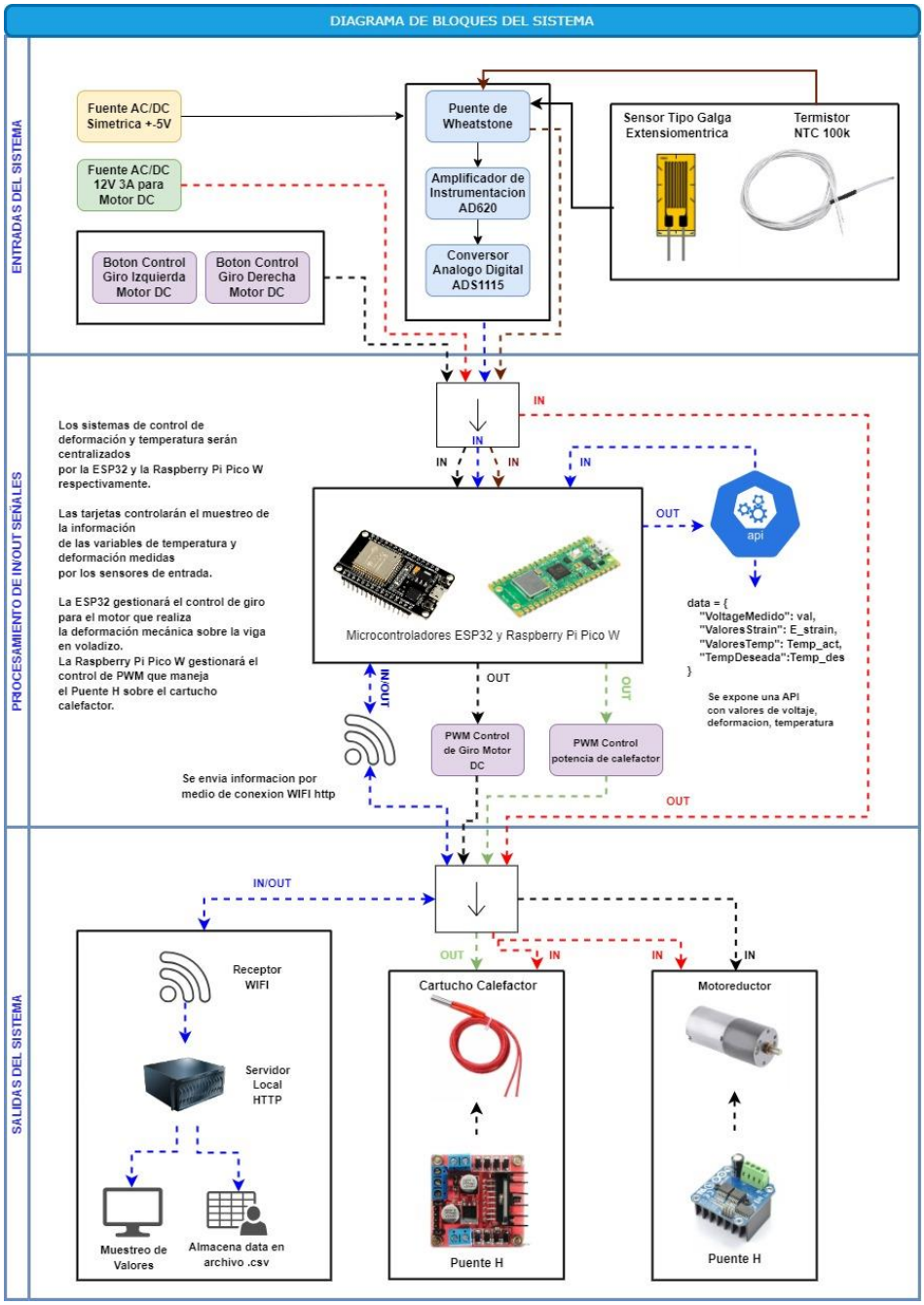

Figura 12. Diagrama de bloques del sistema.

El diagrama de bloques relaciona el flujo completo del funcionamiento de los sistemas de control y una representación gráfica de los componentes electrónicos utilizados en la implementación.

4.2.2 Diseño de ingeniería

El diseño preliminar de la estructura se realizó teniendo en cuenta el diagrama de bloques, las características propuestas en el anteproyecto las cuales buscan que esta sea una construcción sencilla, rígida, y además que sea fácil de transportar y usar en un laboratorio. En las limitaciones físicas se estableció un rango máximo de tamaño de 1.5m x 1.5m (Alto y Ancho) para la construcción de la estructura, al final las medidas seleccionadas fueron 60cm x 40cm (Alto y Ancho).

La estructura se diseñó así por diferentes razones:

1. Se crearon 2 secciones, la sección superior de forma rectangular (Sección 1) que se encargó del sistema de temperatura y la sección inferior de forma cuadrada (Sección 2) que abarcó el sistema de deformación.
2. También se diseñó así ya que su construcción no es compleja, y permite ser modificada en caso de ser necesario, a su vez, permite un fácil acceso al interior de la estructura.

Para la estructura, se optó por usar tubos de aluminio, esto con el fin de que sea liviana su construcción y a su vez rígida. A raíz de esto, se escogió un tubo cuadrado con medidas de 1 pulgada x 1 pulgada con un grosor de 0.9mm haciéndolo de un tamaño ideal para su manipulación. Se utilizaron acoples internos para que en sus esquinas externamente tengan cortes de 45° o 90° y así una presentación pulcra estéticamente.

4.2.2.1 Modelado y planos

El modelado del prototipo fue desarrollado en Autodesk Inventor que es una herramienta que nos ayudó a obtener su representación gráfica para luego proceder con su construcción.

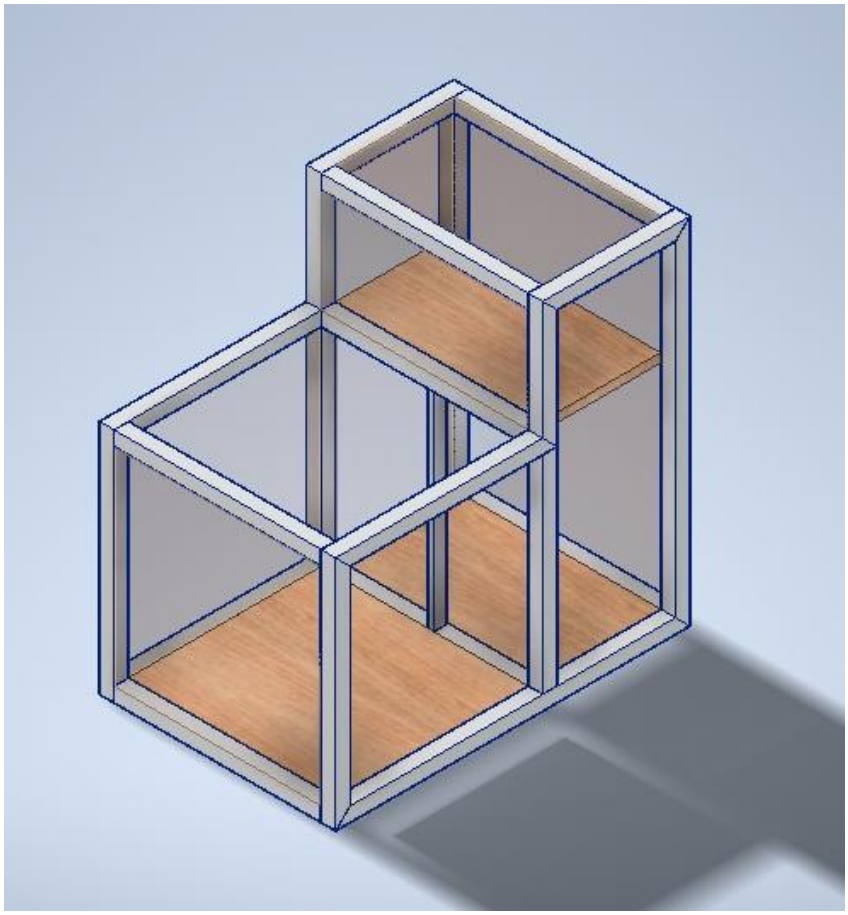

Figura 13. Modelado del prototipo en Autodesk Inventor.

Como se puede ver en la figura 13, la estructura en sus divisiones presenta una superficie en madera. Esta superficie fue escogida con un grosor de 1.8cm con la idea de que, si llega a ser necesario atornillar componentes, se pueda taladrar las tablas y utilizar tornillos con tuercas.

La figura 14 presenta detalladamente los tubos de aluminio utilizados para la construcción de la estructura, esta figura va de la mano con la tabla 4 en la cual se describen los tubos, las láminas protectoras y sus medidas.

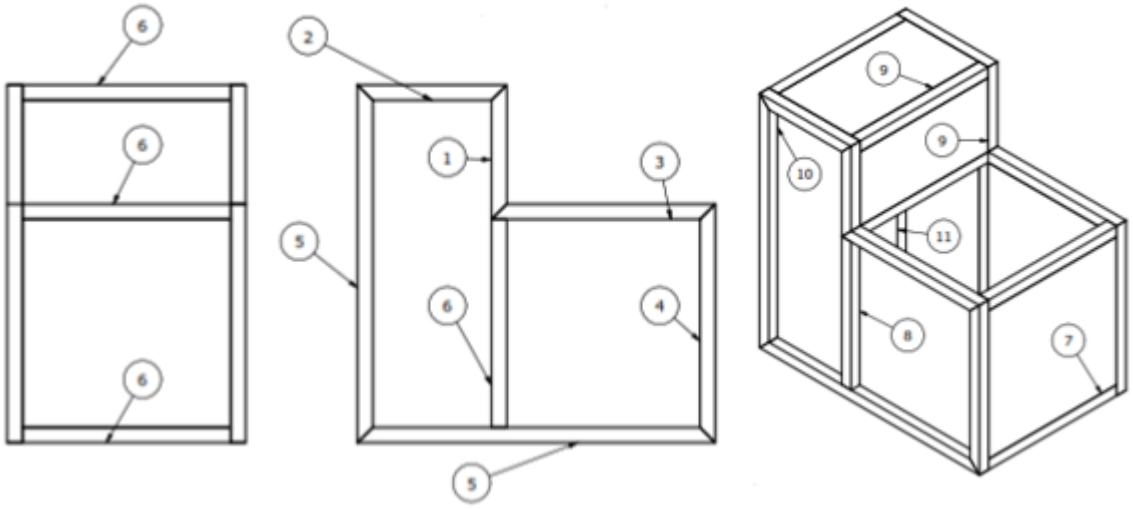

Figura 14. Vistas del prototipo con números indicativos de los tubos y láminas protectoras.

Tabla 4. Lista de componentes utilizados para el modelado de la estructura.

Elemento	**Cantidad**	**No. De pieza**	**Descripción**
1	2	Tubo20	Longitud 20cm
2	2	Tubo25	Longitud 25cm
3	2	Tubo35	Longitud 35cm
4	2	Tubo40	Longitud 40cm
5	4	Tubo60	Longitud 60cm
6	9	Conector40	Longitud 40cm
7	1	Lámina 35x35	Lámina de acrílico de 35cm x 35cm
8	3	Lámina 35x33	Lámina de acrílico de 35cm x 33cm
9	4	Lámina 35x17	Lámina de acrílico de 35cm x 17cm
10	2	Lámina 55x19	Lámina de acrílico de 55cm x 19cm
11	1	Lámina 55x35	Lámina de acrílico de 55cm x 35cm

4.2.2.2 Costos

La cotización del costo total de la construcción de la estructura se encuentra en la tabla 5.

Tabla 5. Costos de construcción de la estructura.

Material	**Precio**
Tubos aluminio cuadrado 1 pulgada (840 cm)	$130.000 pesos colombianos
Acoples de tubos de aluminio (14)	$50.000 pesos colombianos
Superficie de madera (1.5 metros x 1 metro)	$70.000 pesos colombianos
Pliegues para fijación de acrílicos (8)	$60.000 pesos colombianos
Mano de obra	$220.000 pesos colombianos

El valor total de toda la construcción del prototipo fue de $530.000 pesos colombianos.

4.2.2.3 Construcción

En el proceso de construcción de la planta didáctica, se tuvieron en cuenta las medidas y el modelado de dicha planta. Así con el fin de poder obtener dicha estructura como se tiene propuesta.

Figura 15. Estructura en proceso de finalización de su construcción.

Una vez teniendo las bases de la planta didáctica, se fueron agregando los pisos y los acrílicos laterales y superiores. Los acrílicos laterales están ubicados con unos pliegues de

fijación removibles, buscando así en caso de ser necesario retirarlos y poder ingresar a las partes internas de la estructura.

Figura 16. Prototipo de la planta didáctica. En el lado derecho se identifica la tabla que será la base para el segundo piso.

Teniendo ya la estructura, se requirió la utilización de un acople más con el fin de ubicar la viga y/o lámina en voladizo para el sistema de deformación. Se optó por utilizar un tubo cuadrado de aluminio como muestra temporal ya que la viga en voladizo no sería definitiva.

Figura 17. Se evidencia tubo de aluminio cuadrado temporal para sistema de deformación. Se identifica la tabla del piso de sección 1.

4.2.2.4 Ajuste de segundo nivel de sección de temperatura

Como se evidencia en la figura 17, el segundo nivel se encuentra encerrado solamente por acrílicos y sin una división para los circuitos de los sistemas de control, es por esto por lo que se procede a adecuar la sección con icopor en sus alrededores para asegurar de esta forma que se conserve la temperatura en el segundo piso. Esto se hace con el objetivo de que haya menos corrientes de aire que lleguen a afectar las mediciones posteriores de temperatura ya que los sensores ópticos y análogos pueden llegar a ser muy sensibles a cambios repentinos.

Tabla 6. Materiales y costos para segundo nivel de sección de temperatura

Material	**Cantidad**	**Precio**
Lámina de icopor 1.5metros x 1metro (1cm de grosor)	2	$7.000 pesos colombianos c/u
Corta rápido	1	$10.000 pesos colombianos
Lámina de acrílico 30 cm x 30 cm	1	$15.000 pesos colombianos
Barras de silicona	4	$2.000 pesos colombianos c/u

Se realizaron las mediciones internas y se recubrieron las paredes con icopor.

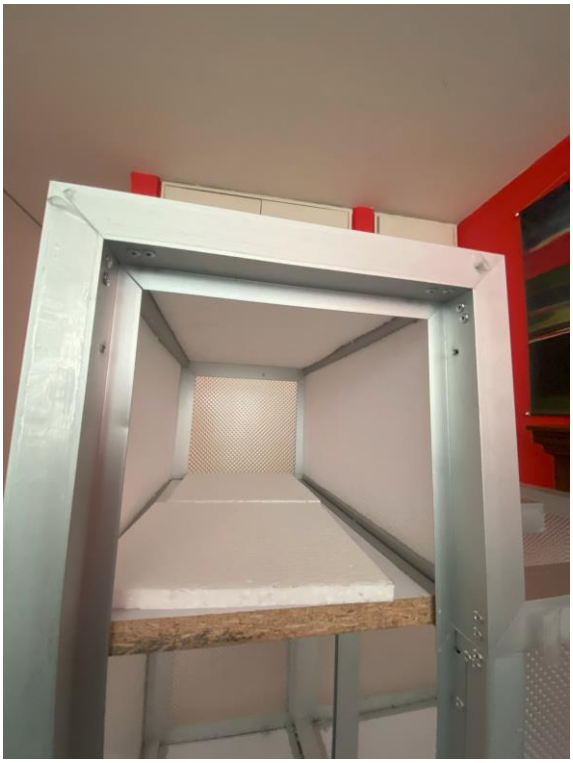

Figura 18. Segundo piso de sección 1 con icopor frontal, superior y trasero.

Una vez protegido el segundo piso, se incluye una división en el segundo piso con el fin de separar los circuitos de los sistemas de control de donde estará el componente que se calentará en el rango de 30° a 50° C. Esto se hace con el objetivo de que haya menos intervención posible en dicho componente y así evitar lecturas erróneas del sensor de temperatura análogo ya que posee alta sensibilidad.

Figura 19. Acrílico de división interna cortado y puesto con icopor para conservar temperatura.

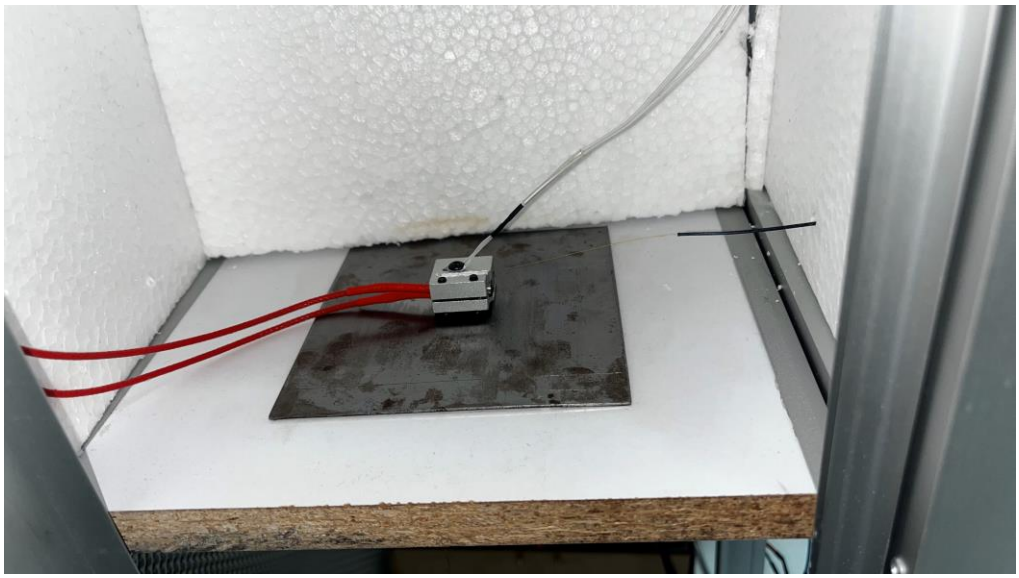

Figura 20. División de acrílico del piso de temperatura. Lado donde está el sensor FBG, el sensor NTC y el bloque calefactor.

Esta división fue elaborada con el objetivo de poder albergar los circuitos de deformación y temperatura al otro lado, buscando así la menor manipulación de estos.

4.2.2.6 Instalación y adaptación Viga en Voladizo y Motor para deformación mecánica

En este espacio el objetivo es organizar e instalar una viga en voladizo de acero con un tamaño definido de 3.5cm de ancho y un largo de 25cm, esta está posicionada en una de las columnas centrales de la estructura, el boceto se presenta en la figura 21.

En la figura 21 se describe la combinación que tiene la lámina de acero, junto a la cadena que conecta la misma con el motor DC, y así se identifica cómo se organiza complemente el esquema de conexión entre los componentes:

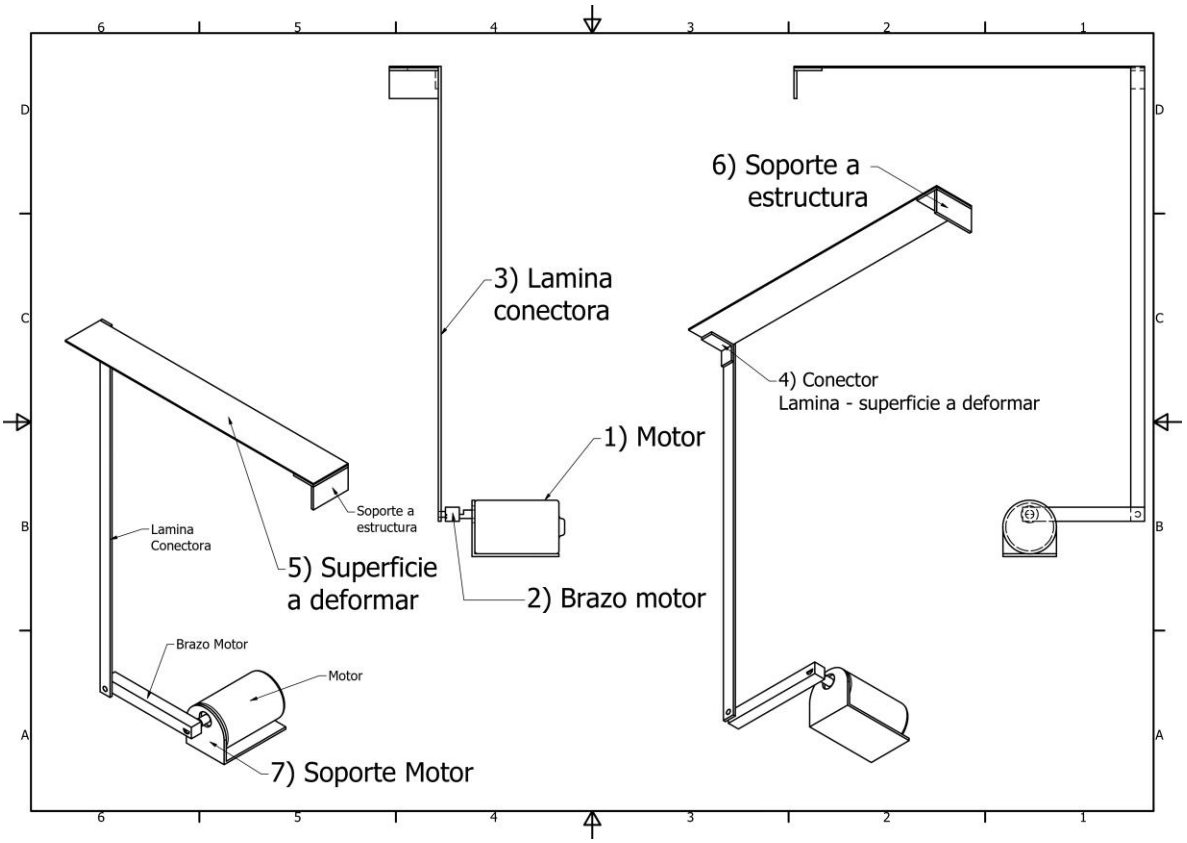

Figura 21. Modelado Viga de Deformación Mecánica.

El boceto de la figura 21 describe cómo se diseñó el modelo de conexión entre componentes para lograr que la viga de acero puede ser tensada por medio del giro de un Motor DC en posición estática, ubicado en la base inferior de la superficie de la estructura. Las partes del modelado son las siguientes:

1) **Motor DC:** Es un motorreductor que tiene capacidad de torque para girar de un sentido a otro con el fin de poder tensar la lámina de acero.
2) **Brazo Motor:** Es un soporte tipo rectangular el cual se ubica en el eje del motor para que este pueda conectarse a la lámina conectora.
3) **Lámina Conectora:** Lámina o cadena que tiene como propósito conectar el brazo del motor con el punto de conexión que tiene la lámina para poder unir el sistema completo.
4) **Conector Lámina:** Conector ubicado en la lámina de acero para poder unir el motor a la viga por medio de la Lámina Conectora.
5) **Viga de Acero**: Es la superficie en acero con el tamaño definido que se usa para posicionar el sensor de deformación y lograr deformarlo sobre esa misma superficie.
6) **Soporte Estructura:** Este soporte está ubicado en la punta que une la viga de acero, junto al pilar central de la estructura y es quien mantiene la viga fija.
7) **Soporte de Motor:** Es una base que sujeta al motor en una posición fija, y permite que este no se mueva en el momento que este tenga que funcionar.

El montaje simulado sobre el modelado completo, juntando cada una de las partes se evidencia en la figura 22.

Figura 22. Modelado completo de viga en voladizo con motor de deformación.

El modelado final junto con todo el prototipo se muestra en la figura 23.

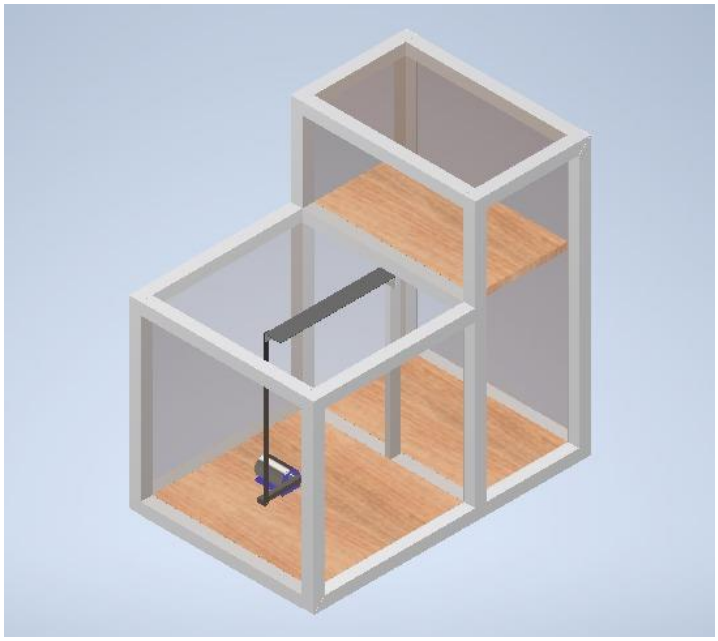

Figura 23. Instalación y posición de la viga de acero en el prototipo.

La versión final de la planta teniendo en cuenta los bocetos presentados, se expone en la figura 24.

Figura 24. Instalación de la viga de acero en la estructura real ya armada.

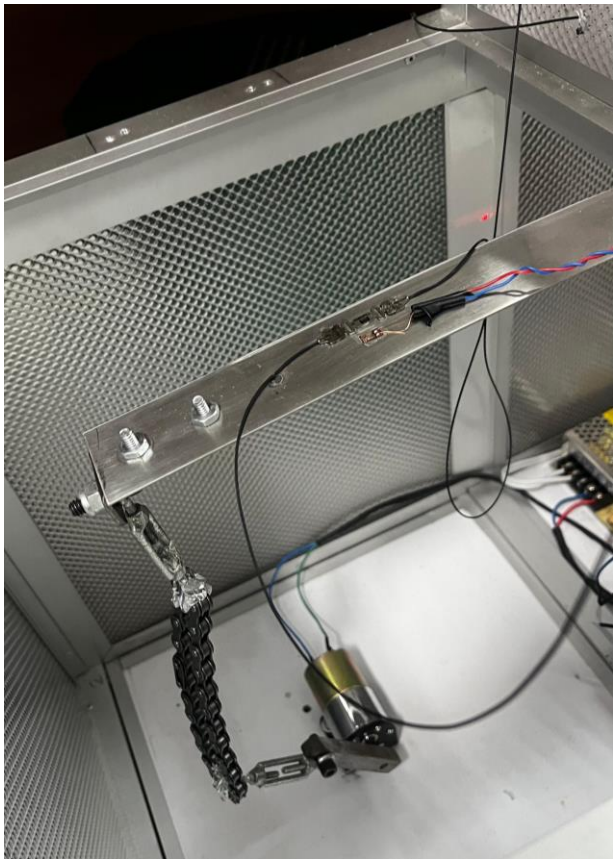

Figura 25. Viga de acero toma superior con sensor FBG y galga.

4.3 Descripción Técnica del Producto

4.3.1 Elección de actuador para control de temperatura

Para el sistema de control de temperatura se requiere calentar un componente entre el rango de 30° a 50° C donde se debe controlar dicho proceso. Para esto, se necesita una unidad o un módulo que permita realizar dicho cambio de temperatura. Existen dos componentes que son utilizados frecuentemente para realizar dichos cambios los cuales son las celdas peltier y los cartuchos calefactores. Las celdas peltier son unidades conformadas por dos semiconductores, un tipo P y un tipo N [37]. Al polarizar y/o alimentar dicha celda, una cara de la celda empezará a tener un ascenso de temperatura y la otra cara tendrá un descenso de temperatura. Estas celdas son utilizadas para proyectos con sistemas de enfriamiento y refrigeración. Por otra parte, los cartuchos calefactores son un pequeño artefacto de forma cilíndrica comúnmente implementados en las impresoras 3D.

Las impresoras 3D utilizan este cartucho calefactor para calentar el material que estén usando para así poder realizar sus impresiones. Por lo general vienen en conjunto con un bloque de aluminio que es el encargado de calentar el extrusor y un sensor de temperatura que es un termistor NTC.

Como este cartucho es de 40W, se requiere de un componente que entregue la suficiente potencia para alimentarlo, es por esto por lo que se usó el controlador L298N como unidad de intervención del cartucho y se controló por PWM.

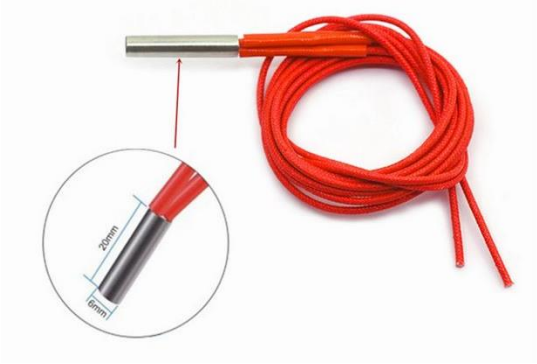

Figura 26. Cartucho calefactor 12/24V.[38]

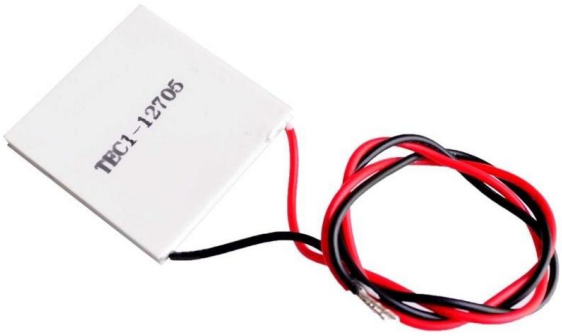

Figura 27. Celda peltier Tec1-12705 12V.[39]

Analizando ambos componentes, se optó por la utilización del cartucho calefactor ya que este es utilizado en sistemas donde solamente se requiere un aumento de temperatura, convirtiéndolo así en una unidad más acorde a los requerimientos del sistema de control de temperatura.

4.3.2 Elección de sensor para sección de temperatura

Para la detección de temperatura, hay varios sensores en el mercado los cuales pueden ayudar a realizar dichas mediciones, aquellos sensores son:

- Sensor DS18B20
- Termistor NTC
- Sensor PT100
- LM35
- DHT22

Estos sensores son usados para medir la temperatura en espacios cerrados, en líquidos, algunos son sumergibles como el DS18B20 y el PT100.

Tabla 7. Sensores de temperatura.

Dispositivos Características	**DSL8B20 SUMERGIBLE**	**TERMISTOR NTC 100K B3950**	**LM35**	**DHT22**	**PT100**
Rango de temperatura	-55°C a 125°C	-55°C a 125°C	-55°C a 150°C	-40°C a 80°C	-200°C a 450°C
Rango de voltaje	3.3v o a 5v	3.3v DC	-0.2v a 35v	3.3V a 5.5V DC	
Corriente	0.5mA		10mA	0.5mA	0.1mA a 2mA
Resolución	Entre 9 a 12 bits		10mV/°C	0.1°C	0.1°C
Sensibilidad	Intermedia	Alta	Intermedia	Intermedia	Buena
Costo	$9.000	$6.000	$10.500	$15.000	$12.500

Al analizar las características de cada sensor, se sabe que el sensor con más sensibilidad es el Termistor NTC B3950 100K haciéndolo un sensor preciso a cambios leves de temperatura siendo el sensor más acorde a los requerimientos establecidos.

El termistor posee un rango de medición de temperatura bastante amplio (-55° a 125° C) [40]. Su característica es que a medida que aumenta la temperatura, su magnitud de resistencia disminuye teniendo una relación inversamente proporcional. Para obtener su valor de resistencia, primero se debe realizar la implementación de un Puente de Wheatstone con el objetivo de identificar el voltaje que varía en el punto de medición.

Teniendo el voltaje del puente, se calcula el valor de resistencia obtenido mediante el despeje de ecuaciones y con este, se obtiene la medida de temperatura que se despeja de (6).

Figura 28. Hotend volcano, incluye NTC B3950 100K de color blanco, cartucho calefactor y bloque de aluminio.[18]

Adicional a esto, como el hotend incluye un bloque aluminio para transmitir la temperatura, se utilizó este mismo bloque como elemento de ubicación para el sensor de fibra óptica como se evidencia en la figura 20.

4.3.3 Elección de sensor para sección de deformación

Para el caso de medición de deformación de la viga de acero que se utilizó, como herramienta para deformar se escogió un único sensor que permite medir pequeñas deformaciones de forma precisa y fácil, para ello se optó por usar las galgas extensiométricas.

Las galgas extensiométricas son sensores que pueden ayudar a medir la deformación en un material en específico (acero para el proyecto), el propósito de ella es estudiar su cambio de resistencia al momento en que el material tenga alguna tensión o deformación mecánica sobre su superficie.

Teniendo en cuenta que la galga extensiométrica es el sensor que se usó para medir los cambios de resistencia cuando se tense la lámina de acero, es necesario hacer un correcto montaje de esta sobre la superficie a medir de la viga de acero. Esto es importante ya que esta se tiene que adherir muy bien a la superficie que se requiere medir.

Para poder obtener una medición exacta sobre el cambio de resistencia en la medición, se debe implementar un circuito en configuración de Puente de Wheatstone en donde el sensor se convertirá la señal de cambio de resistencia, en una señal de salida dada en voltaje. Que a su vez se amplificará con un amplificador de instrumentación mediante una ganancia definida para poder amplificar esos valores de salida.

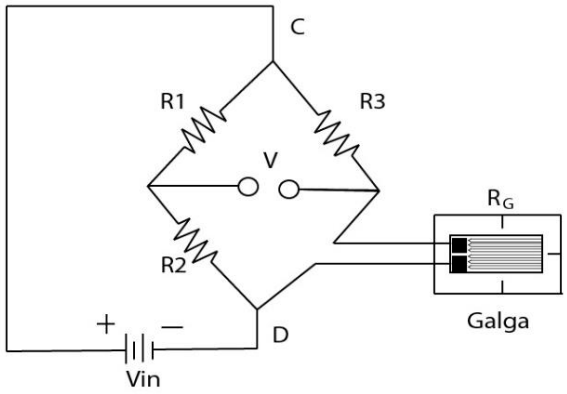

Figura 29. Circuito de Acople de Galga Extensiométrica para obtener variaciones de voltaje.

Como sensor de medida principal se escogió una galga extensiométrica de referencia BF350-3AA la cual tiene las siguientes especificaciones relevantes:

Tabla 8. Datos y Características relevantes Galga Extensiométrica BF350-3AA.

Especificación	Dato
Resistencia Nominal	Puede tener valores de 120Ω, 350Ω y 1000Ω.
Tolerancia de la Resistencia	<±0.1%
Factor Galga	Tiene un base de 2.0, pero puede también tener un valor aproximado a 2.2.
Límite de Estrés	2.0%
Rango de Temperatura en la que opera el sensor.	Tiene un rango aproximado de -30° a 80°C.

En la figura 30 se presenta la galga extensiométrica de forma detallada.

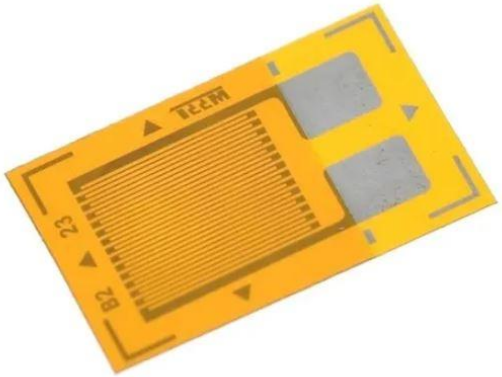

Figura 30. Galga Extensiométrica [41].

Dentro del campo de las galgas existen muchos tipos de galgas, unas son específicas para un material en concreto, o simplemente tienen la posibilidad de medir deformación en varios ángulos o posiciones gracias a sus rejillas [41]. Para la viga en específico se decidió usar una sola galga la cual tiene una resistencia nominal de 350Ω y es la que se usó para poder medir las pequeñas deformaciones hechas a la viga de acero por medio de deformación mecánica.

4.3.4 Elección tarjetas de control

4.3.4.1 Elección tarjeta de sistema de temperatura

Para el sistema de temperatura, se analizaron las limitaciones presentadas donde su rango de variación de temperatura varía entre 30° a 50° C. A su vez, dicha tarjeta tiene que ser capaz de realizar conversiones análogo-digitales con una gran facilidad ya que el sensor de temperatura da una medida análoga obtenida del Puente de Wheatstone. Otra característica fundamental es que se pueda conectar con la tarjeta del sistema de deformación para así poder unificar los datos para su visualización. Dicha conexión puede ser mediante un servidor, una transmisión y recepción de datos por Bluetooth, una creación de un servidor en Python y otras alternativas presentes.

Se escogió la tarjeta Raspberry Pi Pico W porque posee 3 pines los cuales se pueden utilizar como pines de conversión análogo-digitales, posee capacidad de conexión Wifi lo que implica que se puede crear un servidor para la transmisión de los datos de temperatura a la tarjeta de control del sistema de deformación, posee 26 pines GPIO multifuncionales, posee Bluetooth y tamaño pequeño.

También se escogió esta tarjeta ya que su lenguaje de programación es MicroPython, brindando así beneficios para programar servidores, interconexión entre tarjetas, módulos externos como conversores análogo-digitales y posee una gran facilidad para programar.

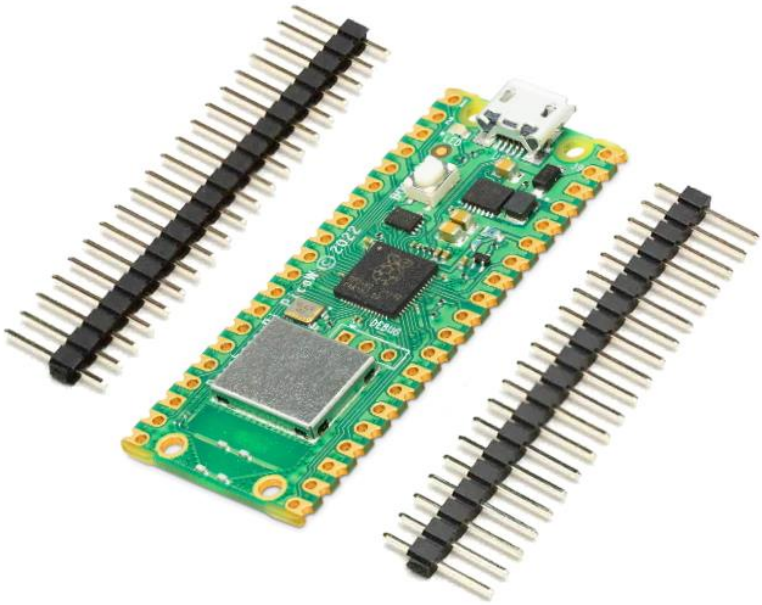

Figura 31. Raspberry Pi Pico W [32].

4.3.4.2 Elección tarjeta de sistema de deformación

En el sistema de control de deformación es importante tener una tarjeta embebida que sea polivalente y que dentro del flujo de proceso permita que sea fácil y sencillo controlar las entradas y salidas del sistema completo. En este caso, la ESP32 es una tarjeta perfecta para estas funciones, además de su gran capacidad de procesamiento esto gracias a su procesador Xtensa LX6 de 32 Bits con una arquitectura amplia de procesamiento [42], también es una tarjeta que ofrece un soporte amplio para variedad de funcionalidades. Las razones por las cuales se escogió esta tarjeta son las siguientes:

1) Control PWM: Tiene puertos IN/OUT que funcionan y controlan este tipo de señales, para el proyecto es muy importante porque se necesitó controlar la posición de giro de un Motor DC, dentro del control de pines se puede configurar hasta 16 puertos, por lo tanto, tiene una amplia utilidad.
2) Entradas ADC: El ESP32 también cuenta con entradas y salidas ADC, cada una de ellas le permite al módulo leer señales en una resolución de 12 Bits, alternativa inicial de adquisición de datos que finalmente se reemplazó por el ADS1115 que es

un ADC externo que ofrece un mayor número de bits y por ende una resolución de 16 bits, equivalente a 65535 valores que logra representar.

3) I2C: Este protocolo es muy compatible con esta tarjeta, permite la utilización de un ADC externo al ESP32 y por este medio mapear los datos que se midieron.
4) WIFI: Dentro sus utilidades también poseen un módulo WIFI incorporado de gran rendimiento, que permite por medio de la tarjeta conectar y enviar datos a otro dispositivo, para el proyecto es necesario ya que es un medio con el que se hizo el envío de los datos leídos por el sensor de deformación.
5) Compatibilidad de Lenguaje de programación: Este punto es muy importante ya que la ESP32 tiene compatibilidad con IDE Arduino, C++, Java, y MicroPython. Para este proyecto la tarjeta se programó 100% en MicroPython debido a que es un lenguaje óptimo y sencillo de programar.

Teniendo en cuenta las características de la tarjeta, queda claro el por qué se escogió y por qué se utilizó como centralizador para implementar todo el sistema de control de deformación. La figura 32 muestra detalladamente los pines del ESP32 y sus características.

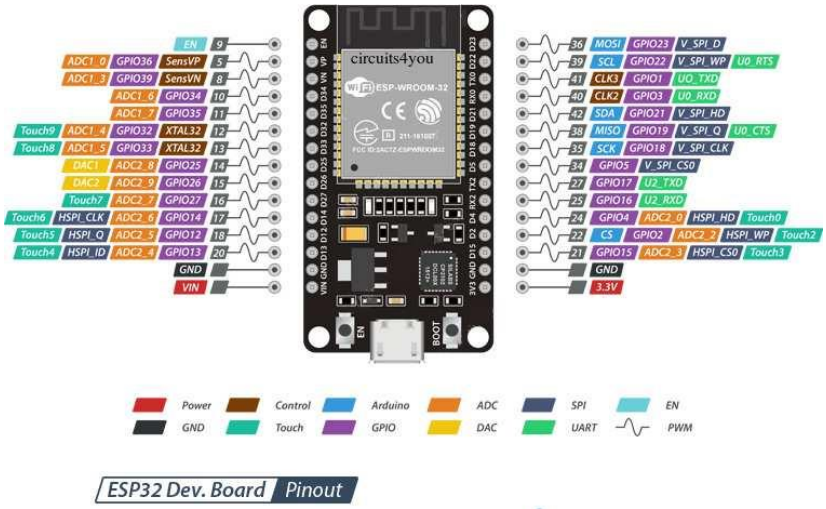

Figura 32. ESP32 tarjeta de control [43].

4.3.5 Caracterización del sistema de temperatura

El sistema de temperatura se caracterizó utilizando una fuente de alimentación de 12V DC a 2Amperios. La caracterización consistió en introducir el máximo voltaje al cartucho calefactor y con el termistor tomar muestras de temperatura cada medio segundo hasta que la medida se estabilizara. Todo el proceso de caracterización duró 40 minutos con 15 segundos y los datos de esta prueba se encuentran en el anexo 3. Se realizó dicho proceso con el objetivo de obtener una representación matemática del sistema el cual nos permitiera modelar el controlador PID.

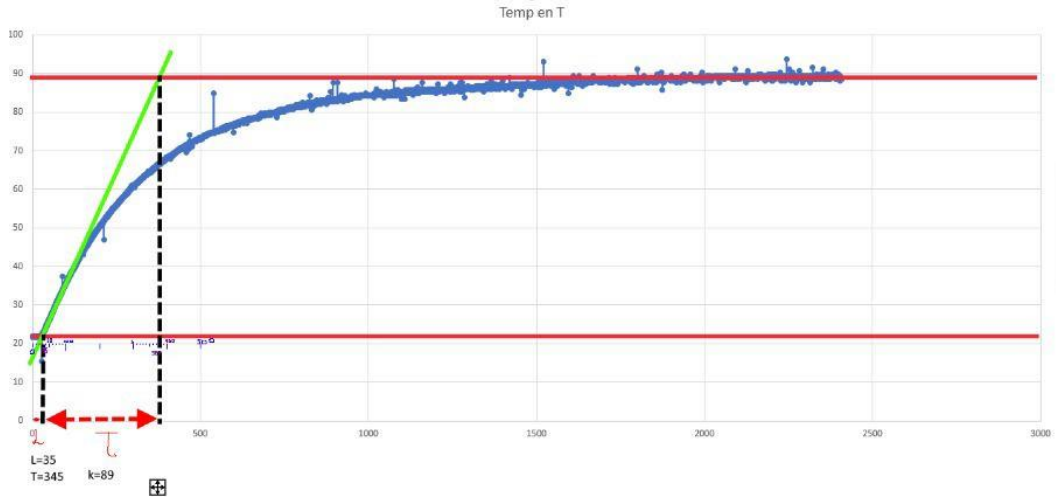

Figura 33. Resultado de prueba de la caracterización del sistema de temperatura.

A partir de la gráfica obtenida de la caracterización, se extraen las constantes fundamentales para el control PID y se discretizan para su implementación en un sistema digital de control que será desarrollado en MicroPython, dichas constantes son expuestas en la sección 4.3.6 cuando se explica detalladamente el código implementado.

4.3.6 Integración para el control de temperatura

Teniendo los componentes establecidos para el sistema de temperatura, se presenta el diagrama de bloques del sistema en el cual se muestra la forma del sistema.

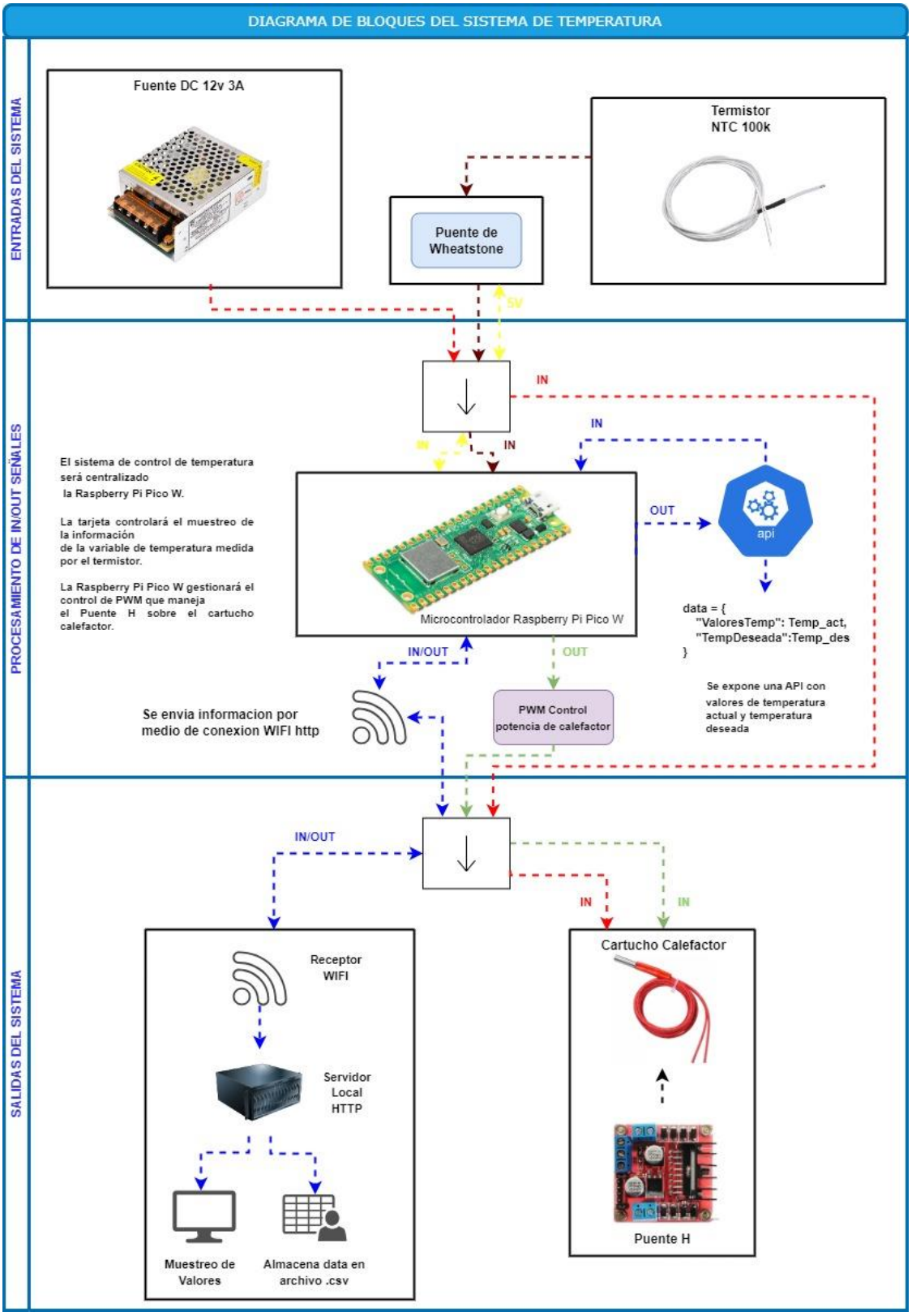

Figura 34. Diagrama de bloques del sistema de temperatura.

Con el diagrama de bloques, se procede a realizar el diagrama esquemático del sistema de control de temperatura.

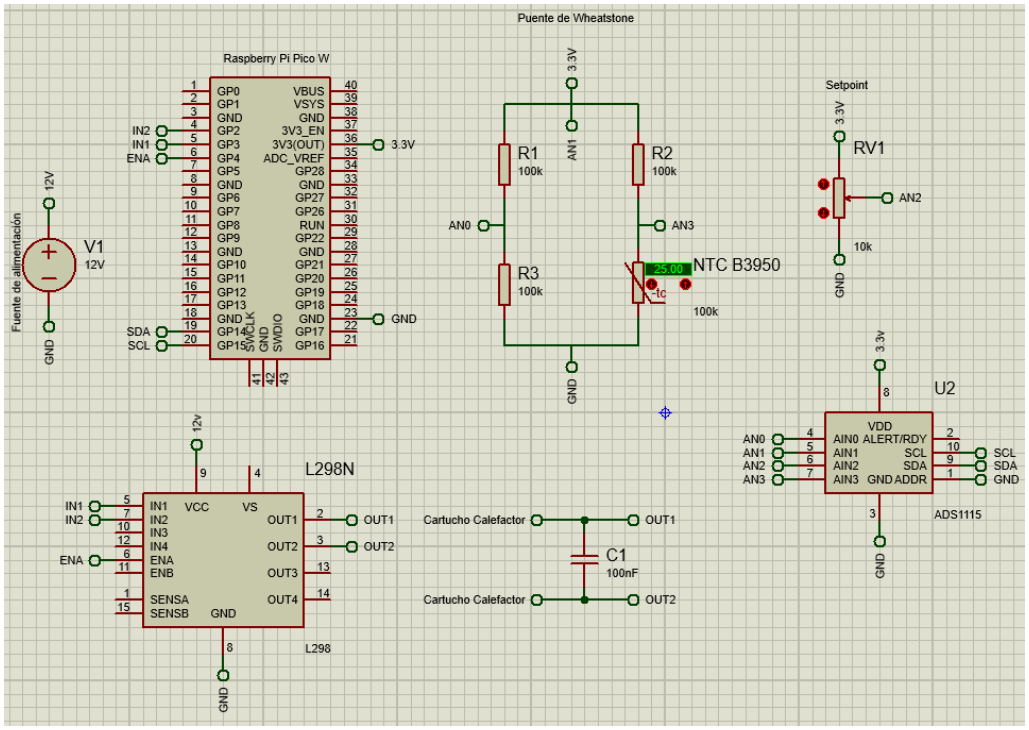

Figura 35. Diagrama esquemático del sistema de temperatura elaborado en Proteus.

Teniendo el diagrama esquemático del sistema, se procede a obtener las medidas de los componentes para realizar el respectivo diseño del diagrama PCB que incluirá todos los componentes a utilizar. El diseño del diagrama PCB se expone en la figura 36.

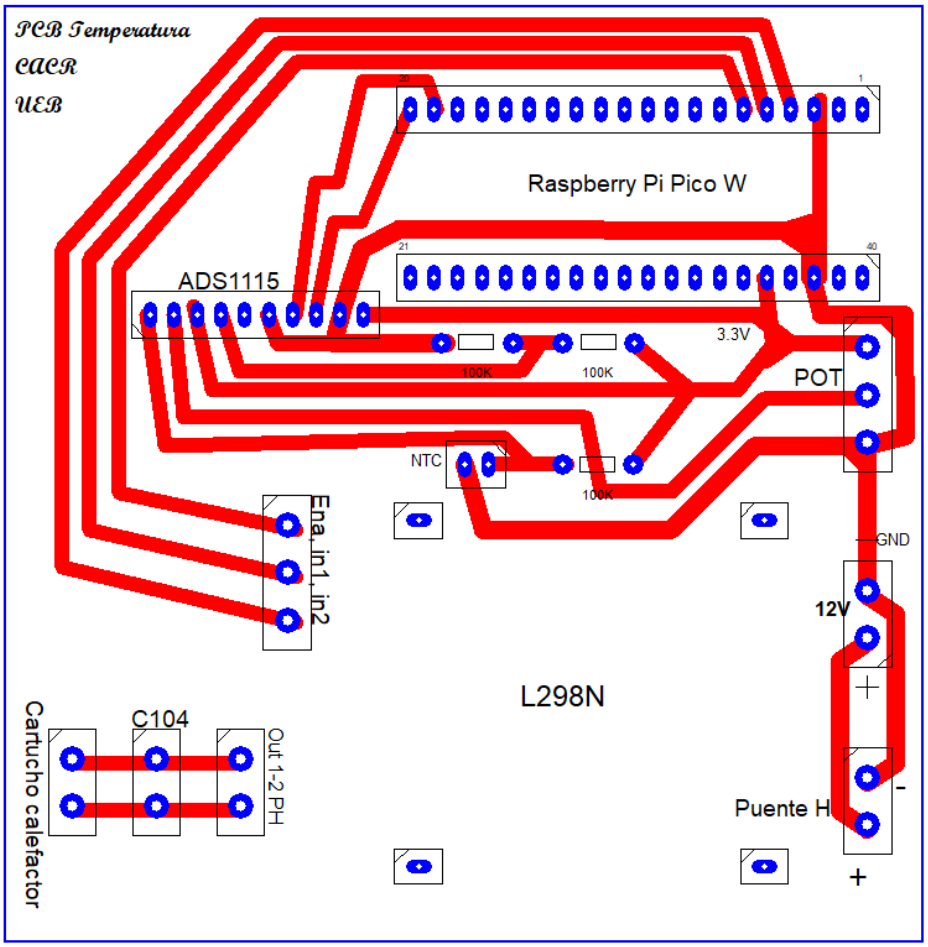

Figura 36. Diagrama PCB elaborado en PCB Wizard.

Tabla 9. Componentes del sistema de temperatura.

Componente	**Cantidad**	**Costo**
Raspberry Pi Pico W con regleta	1	$39.000 pesos colombianos
ADS1115	1	$43.000 pesos colombianos
Driver L298N	1	$22.000 pesos colombianos
Resistencias 100K	10	$100 pesos colombianos c/u
Hotend Volcano	1	$27.000 pesos colombianos
Potenciómetro 10K	2	$1.000 pesos colombianos
Condensador 104	2	$100 pesos colombianos c/u
Regletas hembra	10	$2.000 pesos colombianos c/u
Regletas macho	5	$1.500 pesos colombianos c/u
Jumpers variados	4 paquetes (paquete por 20 unidades)	$5.000 pesos colombianos c/u

Con el fin de realizar la obtención del voltaje del puente de Wheatstone, se calcula dicho voltaje usando (7), los valores de resistencias son reemplazados por los valores medidos de

estas. V_2 es el voltaje medido el cual es ingresado al ADC donde se toma su lectura positiva que está ubicada entre R_3 y R_x y su lectura negativa ubicada entre R_1 y R_2. V_1 también es medido con el ADS1115 ya que se buscó identificar cuál es el voltaje que está entregando la Raspberry Pi Pico W en su salida de 3.3V, se realiza de esta manera para buscar una medida de resistencia y temperatura lo más precisa posible.

Para hallar la temperatura leída por el termistor, se utiliza (6) estableciendo que $R_x = R_T$ donde R_T es la resistencia del termistor calculada en tiempo real por código. Con (6) se realiza el despeje de T_{NTC} la cual es la variable que indicará el valor de temperatura actual.

$$T_{NTC} = \frac{1}{\frac{\ln\left(\frac{R_T}{R_{25}}\right)}{B} + \frac{1}{T_{25}}} - 273{,}15 \qquad (8)$$

El valor de 273,15 se le resta debido a que se requiere expresar la temperatura en términos de °C. Con el uso de (6) y (7) se calcularon mediante código las incógnitas de resistencia y temperatura actuales logrando así el control para las temperaturas deseadas establecidas por el usuario en tiempo real.

El código implementado para el sistema de control de temperatura es muy extenso, por lo que se procederá a explicar las partes importantes de este y el código completo se encontrará en el anexo 4 de este documento.

Las primeras nueve líneas del código consisten en la importación de las librerías a utilizar en todo el programa, se destaca la importación de la librería math ya que esta ayudará a realizar los cálculos necesarios para la obtención del valor de temperatura leído por el termistor.

```
1. #Importación de librerías a usar
2. from machine import ADC, Pin, PWM, Timer, I2C
3. from utime import sleep_ms
4. from time import sleep
5. import math
6. import time
7. import uos
8. import network
9. import urequests
```

El siguiente fragmento de código incluye la conexión del protocolo I2C del ADS1115 con la Raspberry Pi Pico W y sus parámetros de lectura ya que se realiza una conversión para así mostrar los valores requeridos para efectuar los cálculos para la obtención de

temperatura actual. Las líneas de código que llevan un numeral al inicio, brindan más información acerca del contenido del código.

```
#Conexion del protocolo de I2C para el conversor análogo-digital ADS1115
ads = I2C(1, scl=Pin(15), sda=Pin(14))

print(ads.scan())
global channel}
#Dirección establecida del ADS1115 para la identificación en la Raspberry Pi Pico
address = 72

def readConfig():
    ads.writeto(address, bytearray([1]))
    result = ads.readfrom(address, 2)
    return result[0]<<8 | result[1]

#Lectura de los valores de las entradas análogas del ADC
def readValueFrom(channel):
    config = readConfig()
    config &= ~(7<<12)# Clear MUX bits
    config &= ~(7<<9)# Clear PGA
    config |= (7 & (4+ channel))<<12
    config |= (1<<15) # Trigger next conversion
    config |= (1<<9) #gain 4.096volts
    config = [int(config>>i & 0xff) for i in [8,0]]
    ads.writeto(address, bytearray([1] + config))

    config = readConfig()
    while (config & 0x8000) == 0:
        config = readConfig()
    ads.writeto(address, bytearray([0]))
    result = ads.readfrom(address, 2)
    return result[0]<<8 | result[1]

#Conversión del valor análogo a valor de voltaje de las entradas que lo requieran
def readVoltsFrom(channel):
    value = readValueFrom(channel)

    return ((4.096 * 2) / 0xffff) * value

#Array de valores análogos del ADC
val = [0,0,0,0]
```

La siguiente parte del código consiste en la declaración de variables para el PWM del puente h, las variables requeridas para el control PID, la creación de los vectores necesarios que son aquellos que guardan los valores de los errores y la señal de control. A su vez, se realiza la medición de los valores de las resistencias del Puente de Wheatstone para el termistor y se ponen sus valores reales buscando una mayor precisión para el cálculo de las ecuaciones. Se obtienen los parámetros del termistor NTC B3950 100k del datasheet y se establecen como variables, y por último se agregan dos ecuaciones de temperatura.

```
#Declaracion de voltaje de alimentacion, puertos ADC, PWM y los ENABLE para Puente   #H
del cartucho calefactor
pwm = PWM(Pin(4))
pwm.freq(40000)
```

```
68. IN1 = Pin(3, Pin.OUT)
69. IN2 = Pin(2, Pin.OUT)
70.
71. #Modelo del Sistema
72. K = 89 #Ganancia
73. tau = 345
74. theta = 35 #Retardo
75. Ts = 10; #Periodo de muestreo
76. L = theta + Ts/2
77. e = [0,0,0]    #Vector de error
78. u = [0,0]      #Vector de Ley de Control
79.
80. #Resistencias del puente de Wheatstone para NTC
81. R1 = 96200
82. R2 = 97700
83. R3 = 96300
84.
85. #Parámetros de NTC 100K B3950
86. bResistance = 3950 #Valor B dado por fabricante
87. t25Resistance = 100000 #Valor de resistencia del termistor a temperatura de 25°C
88.
89. #Ecuaciones de Temperatura
90. t0 = 273.15
91. t25 = t0 + 25
```

La siguiente parte consiste en la especificación del mapeo del setpoint de temperatura. Este setpoint es el rango de temperatura deseada establecida por el usuario. El valor máximo es de 50° C y el valor mínimo es de 30° C.

```
93. #Función de mapeo del setpiont, permite establecer el mínimo y máximo
94. def map(x, in_min, in_max, out_min, out_max):
95.     return int((x-in_min) * (out_max-out_min) / (in_max-in_min) + out_min)
```

Los siguientes bloques de código consisten en todo el control PID discreto que se utilizó para el control de temperatura. La función update_past consiste en la actualización de los vectores de la ley de control y del pasado donde lo que hace es ir actualizando el valor pasado cada que se llame a esta función.

```
97. #Ciclo for que nos permite actualizar el pasado de la ley de control y del error
98. def update_past(v,kT):
99.    for i in range(1,kT,1):
100.       v[i-1]=v[i]
101.   return v
```

A continuación, se define la función que lleva consigo la ley de control discreta, donde se presentan los valores presentes y los valores pasados, que son aquellos que van guardando los vectores *e* y *u*.

```
103. #Función de Ley de Control PID Discreto
```

```
104. def PID_Controller(u, e, q0, q1, q2):
105.     lu = u[0] + q0*e[2] + q1*e[1] + q2*e[0]; #Ley del controlador PID discreto
106.     # Anti - Windup
107.     if (lu >= 100.0):
108.         lu = 100.0
109.
110.     if (lu <= 0.0):
111.         lu = 0.0
112.
113.     return(lu)
```

Esta siguiente función consiste en la acción de ejecutar las dos funciones anteriores que son PID_Controller y update_past. La función temporizador a su vez calcula el error presente entre el setpoint y la temperatura actual, logrando así identificar la diferencia entre temperaturas, buscando efectuar el control PID donde se le mande un PWM necesario para cada caso, si el setpoint posee una temperatura inferior a la temperatura leída, el controlador PID mandará un valor nulo o casi nulo de PWM para que el Puente H sepa que debe dejar de entregar energía al cartucho calefactor. Si el setpoint es mayor a la temperatura actual, se calcula el error y se procede a efectuar el proceso de control donde la Raspberry identifica que debe establecer un valor de PWM superior para que el Puente H sepa que debe entregar energía al cartucho calefactor.

```
115. #Función que lee el error y la ley de control, calcula el error entre el setpoint y #la
temperatura actual y procede a establecer el PWM para el control del cartucho #calefactor
116. def temporizador(timer):
117.     global u, e, q0, q1, q2
118.     #Actualiza los vectores u y e
119.     u = update_past(u,len(u));
120.     e = update_past(e,len(e));
121.
122.     #Calcula el error actual
123.     e[len(e)-1] = setpoint - temp_x;
124.
125.     #Calcula la Acción de Control PID
126.     u_end = PID_Controller(u, e, q0, q1, q2);  #Max= 100, Min=0
127.     u[len(u)-1] = u_end
128.     velocidad = int(u[len(u)-1] * 65535 /100);
129.
130.     #Aplica la acción de control en el PWM
131.     pwm.duty_u16(velocidad)
```

Por último, se tiene el siguiente bloque que es la función main, dicha función es la que siempre ejecuta la Raspberry. Se definen variables globales ya que son utilizadas en otras funciones, se llama periódicamente la función temporizador cada 10 segundos, buscando que se ejecute todo el proceso de obtención del error y control de la temperatura, este

tiempo es escogido ya que un proceso de temperatura. Se tienen las ecuaciones de los factores de proporcional, integral y derivativo de un control PID.

Se inicia el ciclo while True fundamental para el cálculo de las variables de temperatura, se especifican las salidas del Puente H que es donde está conectado el cartucho calefactor, la especificación de cada lectura de los valores del ADS1115, se establecen dos variables requeridas para la obtención de temperatura.

Se realiza un filtro de promedio móvil en el que se realiza el cálculo del voltaje en el Puente de Wheatstone, se obtiene el valor de resistencia actual del termistor, con el valor de resistencia del termistor, se procede a ubicar (8) para así obtener el valor de temperatura utilizando los factores del termistor y a la temperatura actual que se encuentra como temp_sum en el código, se realiza un promedio entre 50 muestras leídas para así poder obtener un valor más preciso de la temperatura actual.

```
133. def main():
134.     global setpoint, q0, q1, q2, u, e
135.
136.     #Llamado de la función temporizador de forma periódica cada 10 segundos
137.     tim = Timer()
138.     tim.init(period= 10000, mode=Timer.PERIODIC, callback=temporizador)
139.
140.     #Factores para la Ley de Control
141.     kp=(1.2*tau)/(K*L)
142.     ti=2*L
143.     td=0.5*L
144.
145.     q0=kp*(1+Ts/(2*ti)+td/Ts)
146.     q1=-kp*(1-Ts/(2*ti)+(2*td)/Ts)
147.     q2=(kp*td)/Ts
148.
149.     while True:
150.         global temp_x
151.         IN1.high()
152.         IN2.low()
153.         temp_sum = 0
154.         val[0] = readVoltsFrom(0) #Puente de Wheatstone negativo (voltaje)
155.         val[1] = readVoltsFrom(1) #Voltaje de alimentación
156.         val[2] = readValueFrom(2) #Valor potenciometro Setpoint en 16 bits
157.         val[3] = readVoltsFrom(3) #Puente de Wheatstone positivo (voltaje)
158.         Vs = val[1]
159.         #filtro de promedio movil para mejorar el valor de medición de la temperatura
160.         for i in range(50):
161.             Vout = val[3] - val[0] #Voltaje de la diferencia del puente de Wheatstone
162.             resistance = (R2*R3+R3*(R1+R2)*Vout/Vs)/(R1-(R1+R2)*Vout/Vs) #Cálculo de
resistencia del termistor, se obtiene del Puente de Wheatstone
163.             temp_sum += 1/((math.log(resistance/t25Resistance)/bResistance)+(1/t25))-t0
#Obtención del valor de temperatura actual mediante despeje de la ecuación de los termistores
164.         temp_x = temp_sum/50 # Resultado del filtro de promedio movil
165.
166.         val_pot = val[2] #Obtención del valor del ADC para realizar la conversión de sus
valores al valor de mapeo de 30°C a 50°C
167.         setpoint = map(val_pot,0,25900,30,50)
168.         time.sleep(1)
```

```
169.
170. if __name__ == '__main__':
171.     main()
```

Este código presentado no tiene la conexión a internet ni el envío de datos mediante internet, esta parte será explicada más adelante, en la sección 4.3.7. El ANEXO 4 presenta el código completo con la conexión a internet y el envío de datos.

Para la fijación del sensor de fibra óptica, se optó por utilizar pasta térmica la cual estaría entre el sensor FBG y el bloque, este funcionaría como forma de transmisión de la temperatura presente en el bloque al sensor de fibra óptica.

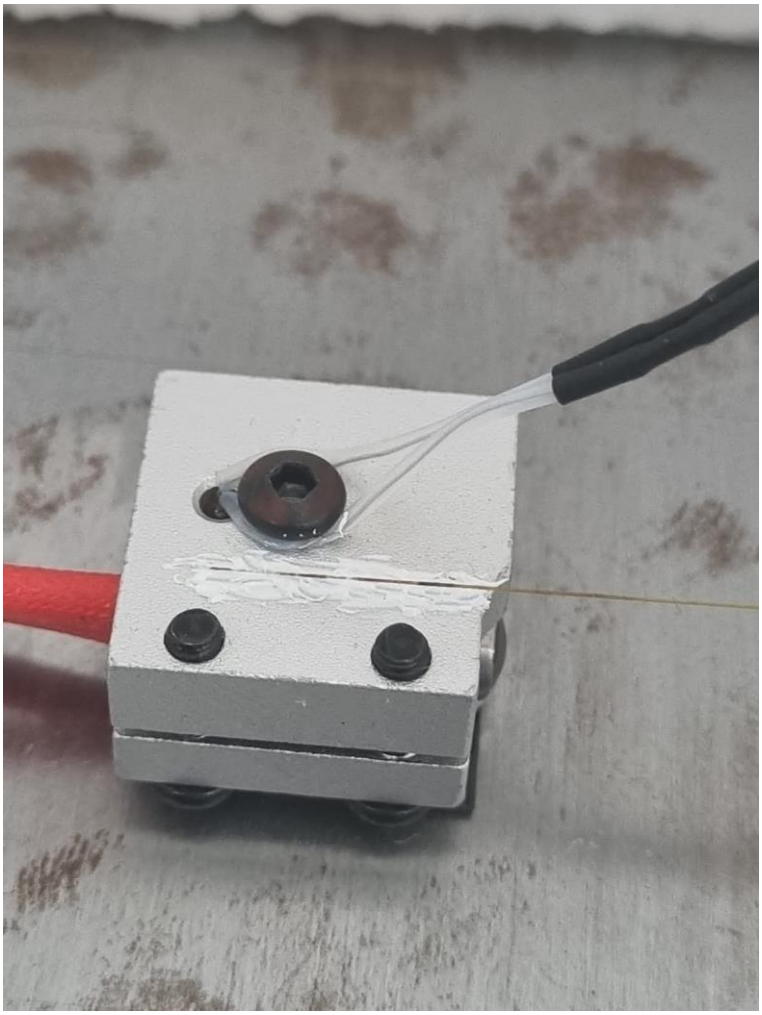

Figura 37. Sensor FBG de color amarillo con pasta térmica para una fijación al bloque de aluminio.

4.3.7 Caracterización e Integración del sistema de deformación.

Para poder integrar el sistema de control de deformación es necesario entender cuáles son las necesidades del proceso que se quiere implementar, qué tipo de actuadores se necesitan, qué se necesita procesar, y cuál será el resultado final de todo el proceso. Para explicarlo brevemente, el sistema de control de deformación que se requiere es dentro de un esquema de deformación que se encuentra físicamente en la estructura, un sensor tipo galga extensiométrica que sea capaz de medir la deformación que se puede provocar al tensar una viga de acero la cual se controlará por medio de un motor DC. La señal de cambio de resistencia que causa la deformación del material y de la lectura del sensor, se procesarán en unidades de voltaje, la señal se amplificará y procesará digitalmente para que la tarjeta de adquisición de datos que en este caso es la ESP32 se pueda encargar de tomar ese valor y por medio de código procesar la medida y convertirla en unidades de Strain (ε) donde podremos obtener el valor de deformación del material de acero. Este flujo completo se explica e ilustra en la figura 38.

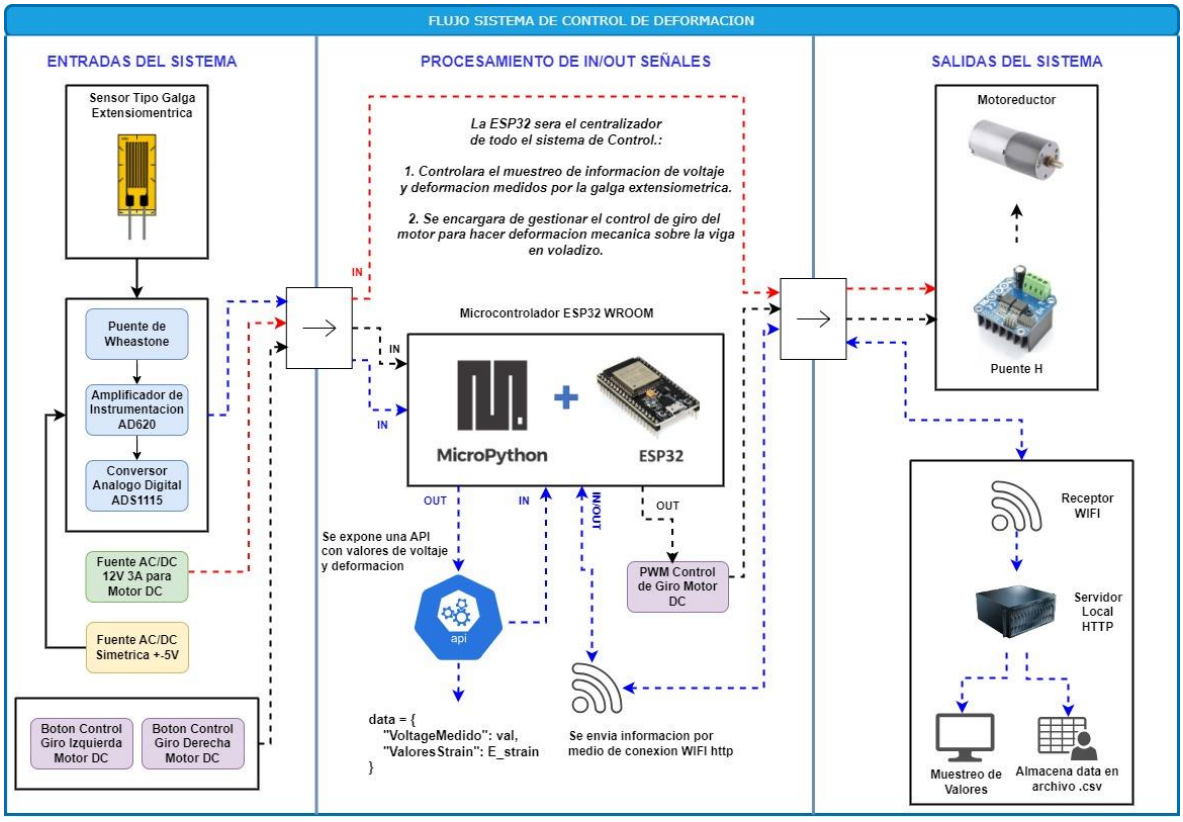

Figura 38. Diagrama de bloques del sistema de control de deformación.

Como primer parte del proceso de integración del sistema de control, es importante empezar a definir cada uno de los flujos internos que hacen parte del proceso, para ello se empezó con el control de giro del motor el cual tiene como propósito mover la viga a la izquierda si se quiere tensar la viga de acero lo más que se pueda o a la derecha si se quiere dejar de tensar dejándola de nuevo en su posición original.

Para la implementación de dicho flujo de control de motor, fue necesario conseguir los componentes de la tabla 10.

Tabla 10. Costos Sistema de Control Giro de motor DC.

Componente	Costo
ESP32 DEVKIT	$30.000 pesos colombianos
Puente H BTS7960	$40.000 pesos colombianos
2 switches Pulsador	$600 pesos colombianos
Fuente 12V 3A Independiente para alimentación del motor.	$25.000 pesos colombianos
Jumpers de Conexión Macho Hembra	$5.000 pesos colombianos

Con los componentes descritos anteriormente, la implementación del circuito se muestra en la figura 39.

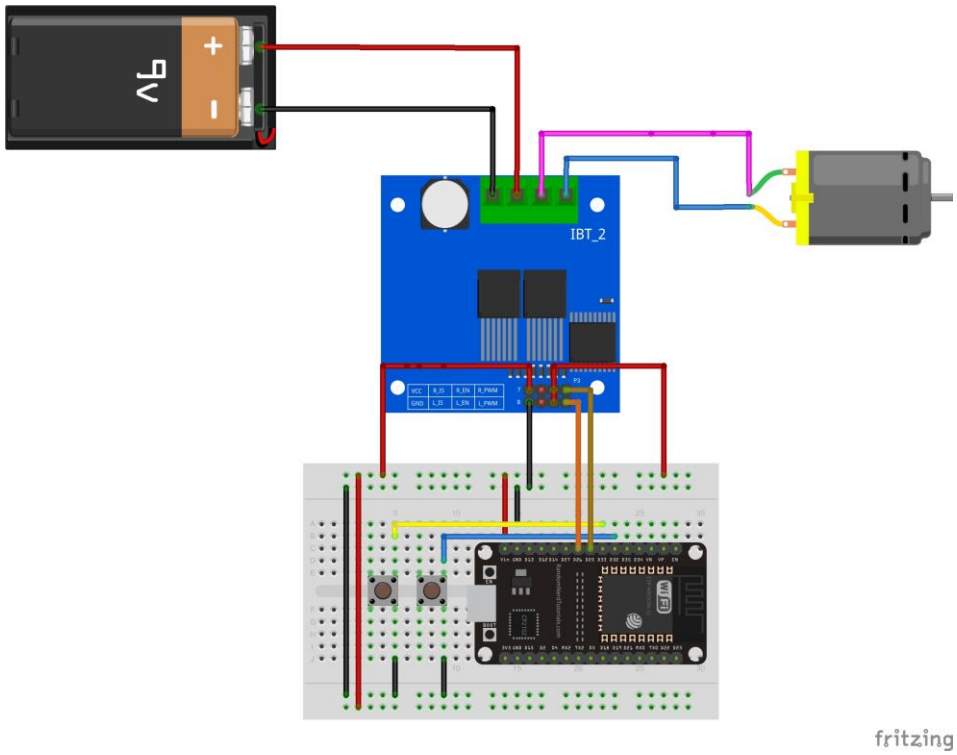

Figura 39. Diagrama Circuito Control de Motor.

El control del motor se da cuando sea necesario oprimir cualquiera de los 2 pulsadores, cada uno de ellos controlará el sentido de giro del motor. El puente H ayudó a controlar ese giro dando al motor la señal cuando se le especifique que lo haga. Se debe tener en cuenta que la ESP32 está usando los puertos GPIO 33 y 32 para las entradas de los pulsadores. Y los GPIO 26 y 25 son las salidas de PWM para el control del motor, y estas se conectaron directamente a los puertos R_PWM y L_PWM del Puente H. Para los Enable del motor, estos no se controlaron por medio del código, si no que siempre estuvieron en un estado lógico 1, lo que conlleva a que siempre estén conectados a VCC.

Para llevar a cabo toda esta implementación fue necesario diseñar una PCB que administró las entradas y salidas del ESP32, e adicionalmente de una vez se pudiera identificar las entradas y salidas para el sistema de control del motor. La PCB se implementó a partir de diagrama esquemático elaborado en Proteus.

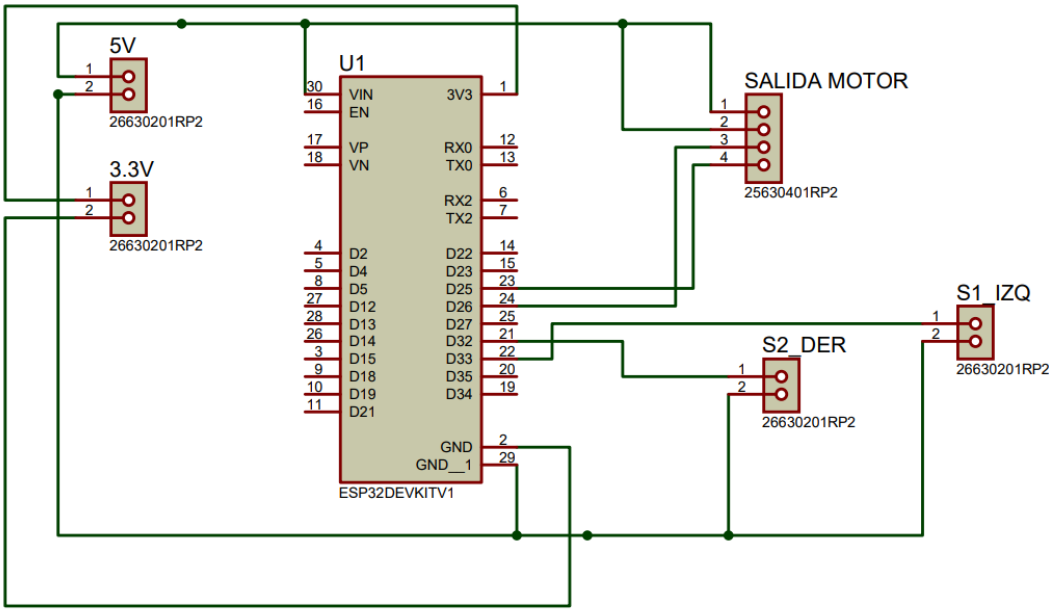

Figura 40. Diagrama esquemático del circuito de control de motor para su diagrama PCB.

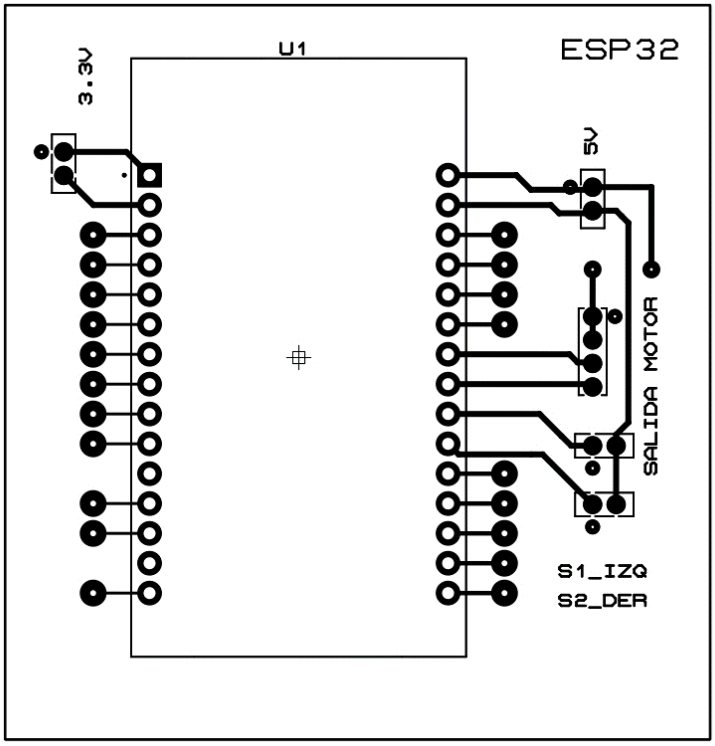

Figura 41. Diagrama PCB sistema control de motor.

Con los esquemáticos de los circuitos del sistema de control para el motor, se programó la lógica interna para el control del giro del motor haciendo uso de los pulsadores. Para ello de definió lo siguiente:

1) Como se definió anteriormente los pines de entrada para los pulsadores serán el GPIO 33 y 32.
2) Los pines de salida de la ESP32 al puente H son los pines GPIO 25 y 26.
3) Se parametriza un PWM o Duty Cycle con Valor de 193, que representa un aproximado del 75% del ciclo completo de giro del motor, el cual su máximo es 255.
4) La frecuencia de giro del motor es de 200Hz, con un retardo entre pulsaciones de 0.2 segundos.
5) Se implementó un contador en donde por cada giro sea derecha o izquierda, solo podrá girar en cada sentido 3 veces. Una vez pasadas los 3 giros cuando se oprima de nuevo, se limita el control de giro del motor.

Todas las parametrizaciones mencionadas anteriormente se detallan en el paso a paso de cómo se construyó el código, explicando las partes más importantes del mismo. Primero que todo se deben importar las librerías y variables del código:

```
1. import machine # Librería para controlar las salidas y entradas GPIO, PWM
 2. import time # Librería para determinar los tiempos
 3.
 4. # Configuración de los pines de entrada y salida
 5. button_left = machine.Pin(33, machine.Pin.IN, machine.Pin.PULL_UP) #Entrada botonIzq.
 6. button_right = machine.Pin(32, machine.Pin.IN, machine.Pin.PULL_UP) #Entrada botonDerecha
 7. motor_left = machine.PWM(machine.Pin(26)) # Salida PWM Motor Giro Izquierda
 8. motor_right = machine.PWM(machine.Pin(25)) # Salida PWM Motor Giro Derecha
 9.
10. # Configuración de los valores de PWM
11. motor_left.freq(200)  # Frecuencia de PWM izquierda (Hz)
12. motor_right.freq(200) # Frecuencia de PWM derecha (Hz)
13.
14. # Velocidad del motor
15. speed_reduced = 193 # Valor PWM
16.
17. debounce_delay = 0.25 #Valor de Retardo de PWM
18. last_left_state = True
19. last_right_state = True
20. left_count = 0
21. right_count = 0
22. max_count = 3 # Conteo Máximo para limitar giros
23.
```

Se crean las funcionas para determinar cuándo:

- Deben detenerse los motores

```
24. def stop_motors():
25.     motor_left.duty(0)
26.     motor_right.duty(0)
```

Los estados en 0 mantienen para cada giro cuando debe detenerse, esto en caso de que no sea necesario mover el motor.

- Giro hacia la izquierda:

```
28. def move_left():
29.     global left_count, right_count #Determina el conteo de izquierda y derecha
30.     if left_count < max_count: # Valida si el valor es menor a 3-que es el valor
limite
31.         motor_left.duty(speed_reduced) # Varia dependiendo del PWM definido valor 193
32.         motor_right.duty(0)
33.         left_count += 1 #Suma una unidad cada vez que se oprime el boton de giro
34.         right_count = 0
35.         print("Motor girando a la izquierda")
```

Dependiendo del contados de 3, y el pulso que se dé, determina cuando debe girar hacia la izquierda, cuando el contado sea mayor a 3, el estado será 0 y no se moverá más de su lugar, obligará a que se tenga que girar en sentido contrario.

- Giro hacia la derecha:

```
37. def move_right():
38.     global left_count, right_count
39.     if right_count < max_count: : # Valida si el valor es menor a 3-que es el valor
limite
40.         motor_left.duty(0) # Varia dependiendo del PWM definido valor 193
41.         motor_right.duty(speed_reduced)
42.         right_count += 1 #Suma una unidad cada vez que se oprime el boton de giro
43.         left_count = 0
44.         print("Motor girando a la derecha")
```

Funcionamiento similar a la función de giro izquierda, solo que esta controla totalmente el giro hacia la derecha, utilizando la misma lógica de conteo. Cuando esta llegue al límite de 3 giros, obligará al usuario a girar al lado contrario.

- Bucle Principal:

```
46. # Bucle principals
47. while True:
48.     left_state = button_left.value() # Almacena los estados de los botones en
variables
49.     right_state = button_right.value()
50.
51.     if not left_state and left_state != last_left_state:
52.         move_left() # Valida con el if el estado del contador
53.         time.sleep(debounce_delay) # Pequeño delay para controlar posibles errores
54.
55.     elif not right_state and right_state != last_right_state:
56.         move_right()
57.         time.sleep(debounce_delay)
58.
59.     else:
60.         stop_motors() # Detiene motor si el Contador esta completo
61.
62.     last_left_state = left_state # Restablece los estados
63.     last_right_state = right_state
64.
65.     time.sleep(0.05) # Tiempo de ejecucion del bucle
```

El bucle principal determina cuando debe girar el motor izquierda o derecha, y el momento en que debe detenerse, esto teniendo en cuenta el conteo. Y controla el tiempo de sleep (reposo) de los motores en el giro.

Con este código se controla toda la parte del giro del motor para la deformación de la viga de acero. Si se requiere ver todo el código completo este se encuentra disponible en el ANEXO 5 del documento.

Seguido al código se implementó un circuito que permite acoplar la señal de medición tomada por la galga extensiométrica, para ello nos apoyaremos en un arreglo de resistencias en tipo puente de Wheatstone que ayudó a obtener el cambio de resistencia en una señal de voltaje a la salida. Para ello se define la siguiente ecuación para determinar el valor de las resistencias y el voltaje de salida del puente [44].

Ecuación para voltaje de salida:

$$V_A - V_b = \left(\frac{R_2}{R_1 + R_2} - \frac{R_G}{R_3 + R_G}\right) X\ V_{CC} \quad (9)$$

Ecuación para igualdad de las resistencias:

$$\frac{R_1}{R_2} = \frac{R_3}{R_G} \quad (10)$$

En este caso para el Puente de Wheatstone implementado el valor de R_G ya está dado por la resistencia de la galga extensiométrica el cual tiene un valor de 350Ω. Para las demás resistencias se decidió dejar $R_1 = 100\Omega$, $R_2 = 330\Omega + 20\Omega = 350\Omega$, $R_3 = 100\Omega$, y con ello también un voltaje de entrada fuente de $V_{cc} = 5V$. Esto con el fin de dejar el puente en estado de equilibrio esperando que a su salida del voltaje entre $V_A - V_B = 0$, para ello el puente se plantea de la siguiente forma:

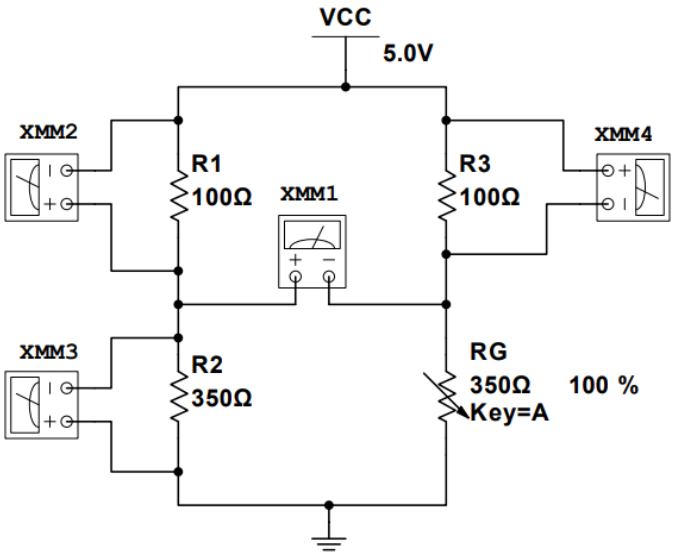

Figura 42. Circuito Puente de Wheatstone implementado.

Con este circuito implementado, se añadió una etapa de amplificación haciendo uso de un amplificador AD620, cuya ganancia está definida en G=100, donde para este tipo de ganancia se debe poner al amplificador una resistencia de ganancia con valor de 499Ω. El circuito completo se muestra en la figura 43.

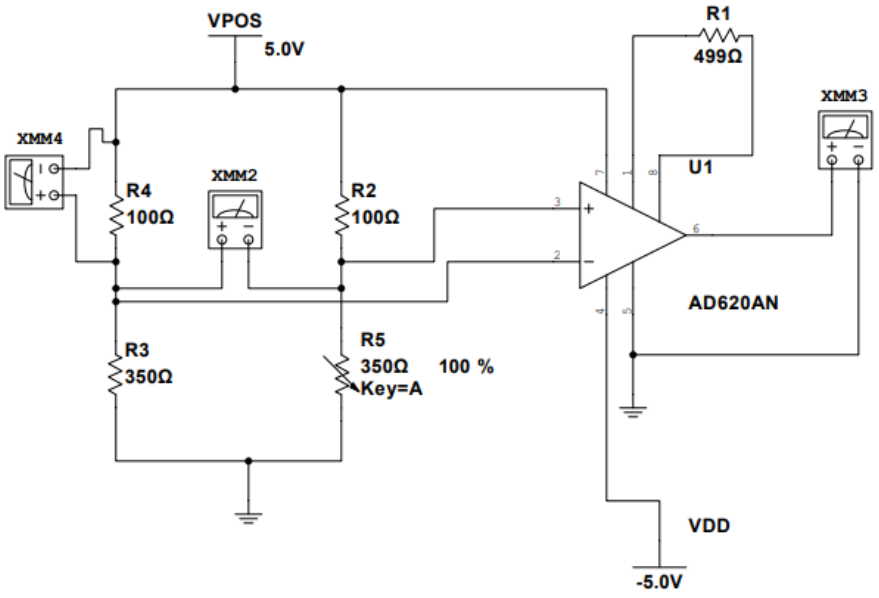

Figura 43. Circuito Puente de Wheatstone Implementado + Amplificación de Voltaje.

Con el circuito totalmente implementado lo que se hace es realizar la adición de un ADC (Conversor análogo digital), en este caso se utilizó el ADS1115 el cual tomará la señal de

salida del amplificador. Y por medio del protocolo I2C se calcula y visualiza el voltaje obtenido en cada variación de deformación que está leyendo el sensor galga. Para ello se implementó el esquemático y diagrama PCB presentes en las figuras 44 y 45 respectivamente.

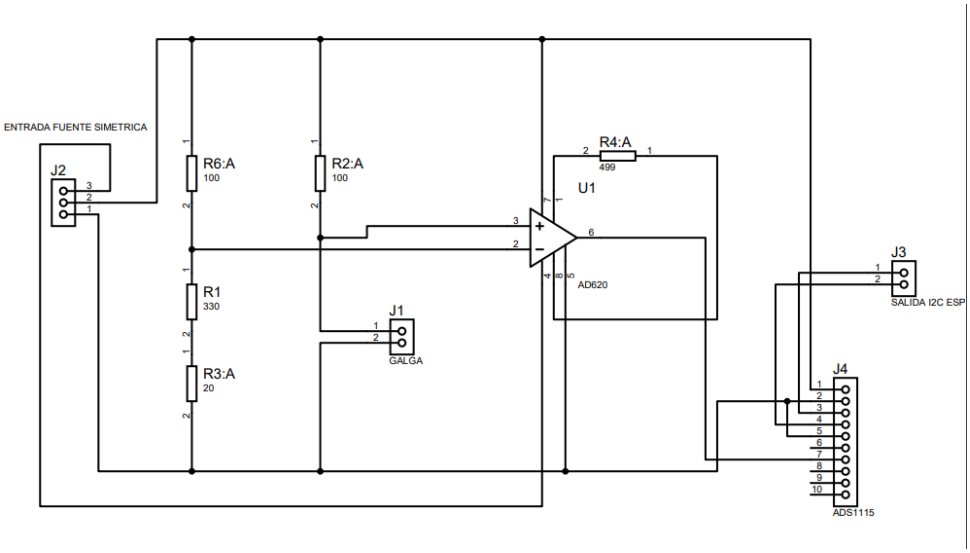

Figura 44. Diagrama esquemático Puente de Wheatstone + Amplificación + ADC.

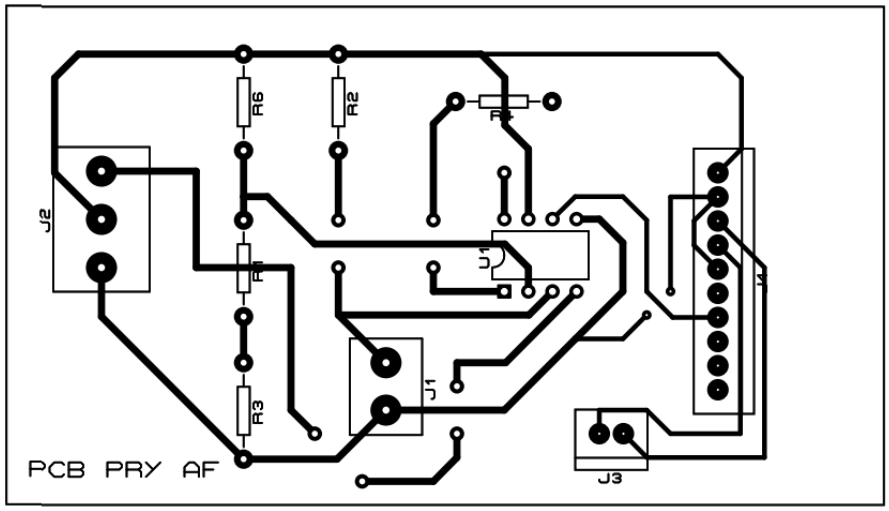

Figura 45. Diagrama PCB Puente de Wheatstone + Amplificación + ADC.

Para terminar la integración de circuitos, el circuito de la Figura 41 se tiene que conectar a la ESP32 a los puertos SCL GPIO 22 y SDA GPIO 21 los cuales leerán el voltaje medido

por medio del protocolo I2C, pero también es importante que se agregue la conversión que es necesaria para convertir esa medición a la cantidad de deformación que se está sometiendo a la viga de acero. Para realizar la relación de voltaje medido y deformación medida, se usaron 2 ecuaciones para realizarlo [45].

Tensión Relativa:

$$V_R = \frac{V_o\ tensa - V_o\ no\ tensa}{V_{ex}} \quad (11)$$

Donde $V_o\ tensa$ es el valor de voltaje que varía dependiendo si se tensa la viga de acero, $V_o\ no\ tensa$ es el valor inicial de voltaje en el cual la viga de aluminio no se tensa y V_{ex} es el voltaje de entrada de excitación del Puente de Wheatstone.

Ecuación Deformación:

$$\varepsilon = \left(1 + \frac{R_L}{R_G}\right) x\ \frac{-4V_R}{[GF * (1 + 2V_R)]} \quad (12)$$

$$R_L = \frac{\rho * l}{a} \quad (13)$$

Donde R_L es la resistencia del cable, ρ es la resistividad, l es la longitud del cable, a es el área transversal del cable, R_G es la resistencia de la galga extensiométrica, V_R es la tensión relativa y GF es el factor de galga.

Aplicando esta información lo que hacemos es generar el código en MicroPython, donde utilizaremos las ecuaciones anteriores para poder obtener el valor de deformación medida. El código detallado implementado para la lectura y cálculo de la deformación medida es la siguiente:

1. Como primera instancia se definen las librerías que usara la ESP32 para comunicarse por I2C, configuración de uso de pines de entrada y salida. Así como

también las variables donde se especifican los pines de lectura del ADC, y la dirección que se establece principalmente para la lectura del conversor análogo digital.

```
1. import utime
 2. import urequests
 3. import network
 4. import uos
 5. import sys
 6. from machine import I2C, Pin
 7.
 8. dev = I2C(1, scl=Pin(22), sda=Pin(21)) # Configuracion de Pines para I2C.
 9. print(dev.scan()) # Como la direccion esta conectada a tierra siempre sera 72 el valor
leido.
10.
11. address = 72
```

2. Se plantea la función que determina la configuración de lectura del adc por el protocolo I2C, donde se establece la dirección, y retorna el resultado para ser leído.

```
13. def readConfig():
14.     dev.writeto(address, bytearray([1]))
15.     result = dev.readfrom(address, 2) # Lee direccion de lectura del I2C
16.     return result[0] << 8 | result[1] # Retorna el resultado de valor medido
```

3. Con la función de lectura definida, ahora se creo una nueva función, que se encarga de calcular el valor del ADC medido en la lectura del canal de entrada, haciendo uso de un valor de ganancia hasta 4.1 Volts.

```
18. def readValueFrom(channel):
19.     config = readConfig()
20.     config &= ~(7 << 12)  # clear MUX bits
21.     config &= ~(7 << 9)   # clear PGA
22.     config |= (7 & (4 + channel)) << 12
23.     config |= (1 << 15) | (1 << 9)  # ganancia de 4.1 volts
24.     config = [int(config >> 8), int(config & 0xff)]
25.     dev.writeto(address, bytearray([1] + config))
26.     config = readConfig()
27.     while (config & 0x8000) == 0:
28.         config = readConfig()
29.     dev.writeto(address, bytearray([0]))
30.     result = dev.readfrom(address, 2)
31.     return result[0] << 8 | result[1]
```

4. Retornado el valor del ADC, se crea otra función para determinar el valor del ADC, y diferenciar el voltaje medido en el puente de Wheatstone. Y ese valor es el que se devuelve al final para el cálculo de la deformación medida.

```
33. def readVoltsFrom(channel):
34.     value = readValueFrom(channel)
35.     return (value * 4.096) / 32767 # Se  hace diferenciacion para calcular voltage
medido por el puente de Wheastone
```

5. Para finalizar el valor de voltaje medido devuelto se usó dentro de un bucle que hace utiliza (12) para realizar el cálculo de la deformación medida en la variable E_strain.

```
37. while True:
38.     val = readVoltsFrom(0)
39.     ro = 1.72e-8
40.     l = 0.82  # Longitud en metros (82 cm convertidos a metros)
41.     a = 0.259e-6  # Área transversal en metros cuadrados (0.259 mm^2 convertidos a
metros cuadrados)
42.     RL = ro * l / a
43.     RG = 350  # Resistencia del galvanómetro (Ohms)
44.     Vex = 5.0  # Voltaje de excitación (Voltios)
45.     GF = 2.0  # Gauge Factor
46.
47.     Vr = (val - 0) / Vex  # Tensión relativa (mV/V)
48.     E_strain = (1 + RL / RG) * (-4 * Vr) / (GF * (1 + 2 * Vr))
49.     print(f'Voltaje (V): {val:.2f} | Deformación (με): {E_strain:.2f}')
50.
51.     utime.sleep(1)
```

Con este código, la idea es obtener cada un segundo la medida de deformación obtenida en el proceso que se pueda estar tensando o no la viga de acero por medio del motor, el código completo se puede encontrar completo en el ANEXO 6.

Para poder obtener todos estos datos de deformación medida y almacenarlos, se creó una API (Application Programming Interface) y un cliente/servidor para enviar, recibir y guardar en un archivo de extensión CSV los datos leídos con el propósito de que se puedan procesar de forma gráfica, y que de forma visual también se pueda observar la medición de dichos valores.

Para la construcción del servidor fue necesario instalar el componente Flask con el cual se puede ejecutar el servidor haciendo uso del código Python, para instalarlo se hizo uso del comando:

```
1. Pip install flask
```

Una vez instalado se crea el código en Python para recibir por medio del servidor los datos y una interfaz dinámica en Tkinter que permite visualizar las mediciones ya obtenidas, a su vez también se intenta que estos datos obtenidos se almacenen en un archivo CSV.

Código Recepción Servidor:

1) Se importan las librerías Tkinter para la creación de la ventana, Flask para exponer el código por medio de un servidor, y la librería Threads que permite al código trabajar con múltiples Hilos de Conexión.

```
1. import tkinter as tk
 2. from flask import Flask, request, jsonify
 3. from threading import Thread
 4. import csv
```

2) Se crea una variable app para ejecutar la aplicación por medio del servidor, y se crea también un diccionario dentro de un arreglo que almacenará toda la información recibida para guardarla en un archivo CSV.

```
6. app = Flask(__name__) # Comando para ejecutar la aplicacion por medio del servidor
 7.
 8. datos_recibidos = []
```

3) Definimos la función para crear la ventana que muestra los datos medidos, se usa Tk para definir la geometría de la ventana, los labels, títulos, los textos a mostrar, y la información que debe ir almacenando para generar la visualización.

```
10. def crear_ventana(): # Ventana Dinamica haciendo uso de Tkinter
11.     ventana = tk.Tk()
12.     ventana.title("Valores Recibidos")
13.     ventana.geometry("400x170")  # Ancho x Alto
14.
15.     titulo = tk.Label(ventana, text="Valores Deformación:", font=("Helvetica", 11,
"bold"))
16.     titulo.pack()
17.
18.     espacio = tk.Label(ventana, text="")
19.     espacio.pack()
20.
21.     campos_texto = {} # Diccionario donde se almacenaran los datos
```

4) Se crea otra sub-función que actualizará los datos mostrados en la ventana cada 1 segundo, y que a medida que va muestreando, se irá almacenando en el archivo CSV.

```
23.     def actualizar_valores():
24.         if datos_recibidos:
25.             data = datos_recibidos.pop(0)
26.             guardar_en_csv(data, "datos_deformacion.csv") # Funcion para almacenar los
datos en .csv
27.             for key, value in data.items():
28.                 if key not in campos_texto:
29.                     campos_texto[key] = tk.Text(ventana, height=1, width=30,
font=("Helvetica", 16), state="disabled")
30.                     campos_texto[key].pack()
31.                 campos_texto[key].config(state="normal")
32.                 campos_texto[key].delete(1.0, tk.END)
33.                 campos_texto[key].insert(tk.END, f"{key}: {value}")
34.                 campos_texto[key].config(state="disabled")
35.             ventana.update()
36.         ventana.after(1000, actualizar_valores)  # Actualizar cada 1 segundo
37.
38.     actualizar_valores()  # Llamar a la función por primera vez
39.     ventana.mainloop()
```

5) Creadas las funciones de muestreo, se define otra función que se encarga de leer, guardar y escribir sobre el archivo CSV, aquí se define la extensión, y los campos a almacenar dentro del CSV.

```
41. def guardar_en_csv(data, filename): # Funcion que permite leer, guardar y escribir
sobre archivos CSV
42.     with open(filename, "a", newline='') as archivo_csv:
43.         campos = data.keys()
44.         writer = csv.DictWriter(archivo_csv, fieldnames=campos)
45.
46.         if archivo_csv.tell() == 0:
47.             writer.writeheader()
48.
49.         writer.writerow(data)
```

6) Por último, se crea el API con un método POST como canal de escucha para recibir los datos enviados por la ESP32, verificando la correcta conexión con el cliente, y verificando si recibió los datos de forma correcta por la IP y el puerto expuesto.

7)

```
51. @app.route('/data', methods=['POST']) # API que se encarga de recibir los datos
enviados por la ESP32
52. def recibir_datos():
53.     data = request.get_json()
54.     datos_recibidos.append(data)
55.     return jsonify({"mensaje": "Datos recibidos con éxito"})
56.
57. def ejecutar_servidor():
58.     app.run(host='0.0.0.0', port=8000) # IP y puerto donde se abre la conexion de
escucha para recibir conexion.
59.
60. servidor_thread = Thread(target=ejecutar_servidor) # Se usan Hilos para poder procesar
toda la informacion de forma mas optima.
```

```
servidor_thread.daemon = True
servidor_thread.start()

if __name__ == '__main__':
    crear_ventana()
```

El código completo de esta implementación se puede encontrar en el ANEXO 7 del documento.

Este código se ejecuta por medio de consola CMD usando el siguiente comando:

```
python server2.py
```

Y para ello es importante que el servidor se ejecute antes con el siguiente comando:

```
python -m http.server 8000
```

Identificando el puerto con el cual se quiere recibir la conexión, una vez ejecutado el servicio API estará listo para recibir datos cuando la punta que envía los necesite hacer.

Por último, se hizo la integración de un código completo en donde se ejecuta el control del motor y a su vez se puedan tomar los datos medidos por el sensor tipo galga extensiométrica, y que además pueda enviar esos datos por medio del API hacia el servidor local. Para ello se implementó el siguiente código completo:

1) Se importan librerías para envío de datos, conexión a internet (WIFI), definición de puertos para PWM y I2C, además de que se anexa también librería para que la ESP32 trabaje con hilos paralelos. Se crean las variables para conexión a wifi, definición de entradas y salidas para los pulsadores, configuración del motor, y conversor análogo digital.

```
#import time
import utime as time
import urequests
import network
import uos
import sys
from machine import I2C, Pin, PWM
import threading

# Configurar la conexión Wi-Fi
ssid = "user"
password = "password"
```

```
13.
14. # Conectar a la red Wi-Fi
15. wifi = network.WLAN(network.STA_IF)
16. wifi.active(True)
17. wifi.connect(ssid, password)
18.
19. # Esperar a que se conecte a la red Wi-Fi
20. while not wifi.isconnected():
21.     pass
22.
23. print("Conectado a la red Wi-Fi")
24.
25. # Configuración de los pines para pulsadores
26. button_left = Pin(33, Pin.IN, Pin.PULL_UP)
27. button_right = Pin(32, Pin.IN, Pin.PULL_UP)
28. motor_left = PWM(Pin(26))
29. motor_right = PWM(Pin(25))
30.
31. # Configuración de los valores de PWM
32. motor_left.freq(200)  # Frecuencia de PWM (Hz)
33. motor_right.freq(200)
34.
35. # Velocidad del motor
36. speed_reduced = 193 # Valor definido de PWM
37.
38. debounce_delay = 0.3  # Tiempo entre giro de motor
39. last_left_state = True
40. last_right_state = True
41. left_count = 0
42. right_count = 0
43. max_count = 3 # Contador para limitar giro derecha e izquierda
44.
45. dev = I2C(1, scl=Pin(22), sda=Pin(21)) # Pines de Conexion I2C
46. address = 72 # Direccion de lectura cuando el addr esta conectado a tierra, siempre
sera 72 su valor.
47.
```

2) Se mapea dentro del código las funciones para la lectura y conversión de la señal análoga a digital, retornando el voltaje leído por el ADC en el puente de Wheatstone.

```
48. def readConfig():
49.     dev.writeto(address, bytearray([1])) # Procesa la lectura de entrada del I2C.
50.     result = dev.readfrom(address, 2)
51.     return result[0] << 8 | result[1] # Retorna el resultado de la medida tomada.
52.
53. def readValueFrom(channel):
54.     config = readConfig()
55.     config &= ~(7 << 12)  # clear MUX bits
56.     config &= ~(7 << 9)   # clear PGA
57.     config |= (7 & (4 + channel)) << 12
58.     config |= (1 << 15) | (1 << 9)  # ganancia de 4.1 volts
59.     config = [int(config >> 8), int(config & 0xff)]
60.     dev.writeto(address, bytearray([1] + config))
61.     config = readConfig()
62.     while (config & 0x8000) == 0:
63.         config = readConfig()
64.     dev.writeto(address, bytearray([0]))
65.     result = dev.readfrom(address, 2)
66.     return result[0] << 8 | result[1]
```

```
67.
68. def readVoltsFrom(channel):
69.     value = readValueFrom(channel)
70.     return (value * 4.096) / 32767
```

3) Se anexan ahora las funciones para el funcionamiento de los botones y el control de giro del motor, aquí una de las ventajas es que se optimiza el funcionamiento haciendo uso de los Threads para ejecutar los procesos en paralelo en la ESP32.

```
71. # Logica de Giro del Motor
72. def stop_motors():
73.     motor_left.duty(0)
74.     motor_right.duty(0)
75.
76. def move_left():
77.     global left_count, right_count
78.     if left_count < max_count:
79.         motor_left.duty(speed_reduced)
80.         motor_right.duty(0)
81.         left_count += 1
82.         right_count = 0
83.         #print("Motor girando a la izquierda")
84.
85. def move_right():
86.     global left_count, right_count
87.     if right_count < max_count:
88.         motor_left.duty(0)
89.         motor_right.duty(speed_reduced)
90.         right_count += 1
91.         left_count = 0
92.         #print("Motor girando a la derecha")
93.
94. def button_thread():
95.
96.     global left_count, right_count
97.
98.     while True:
99.         left_state = button_left.value()
100.        right_state = button_right.value()
101.
102.        if not left_state and left_state != last_left_state:
103.            move_left()
104.            time.sleep(debounce_delay)
105.
106.        elif not right_state and right_state != last_right_state:
107.            move_right()
108.            time.sleep(debounce_delay)
109.
110.        else:
111.            stop_motors()
112.
113.        last_left_state = left_state
114.        last_right_state = right_state
115.
116.        time.sleep(0.05)
117.
118. # Crear y ejecutar el hilo para control de botones
119. button_thread = threading.Thread(target=button_thread)
120. button_thread.start()
```

4) Creamos el bucle para determinar el valor de deformación medido, esto mientras ejecutamos en paralelo el control de giro del motor. Este estará leyendo los datos del adc, calculando los valores de deformación, e imprimiendo el voltaje medido y la deformación en microstrain.

```
122. while True:
123.     # Leer el valor del sensor y enviar datos
124.     val = readVoltsFrom(0)
125.     ro = 1.72e-8
126.     l = 0.82  # Longitud en metros (82 cm convertidos a metros)
127.     a = 0.259e-6  # Área transversal en metros cuadrados (0.259 mm^2 convertidos a
metros cuadrados)
128.     RL = ro * l / a
129.     RG = 347  # Resistencia del galvanómetro (Ohms)
130.     Vex = 5.0  # Voltaje de excitación (Voltios)
131.     GF = 2.0  # Gauge Factor
132.     Vr = (val - 0) / Vex  # Tensión relativa (mV/V)
133.     E_strain = (1 + RL / RG) * (-4 * Vr) / (GF * (1 + 2 * Vr))
134.
135.     print(f'Voltaje (V): {val:.2f} | Deformación (με): {E_strain:.2f}')
```

5) En el mismo bucle, se define y se especifica a la ESP32 que debe conectarse por protocolo WIFI al servidor expuesto, y establecer una conexión cliente/servidor, haciendo envío por medio de un request REST la información del voltaje medido y deformación medida. La misma ESP32 definirá si fue capaz de conectarse con el servidor y si el dato por segundo fue enviado con éxito.

```
137.     pc_ip = "192.168.115.150:8000"  # La dirección IP de la PC en la cual esta
desplegado el servidor de recepcion
138.     url = "http://" + pc_ip + "/data"
139. # Datos a enviar en un JSON
140.     data = {
141.         "VoltageMedido": val,
142.         "ValoresStrain": E_strain
143.     }
144.
145.     try:
146.         response = urequests.post(url, json=data)
147.         if response.status_code == 200:
148.             print("Datos enviados con éxito")
149.         else:
150.             print("Error al enviar datos:", response.status_code)
151.         response.close()
152.     except Exception as e:
153.         print("Error al enviar datos:", str(e))
154.
155.     time.sleep(1)
```

Con esta configuración y código final unificado es con el cual se trabajó todo en una misma tarjeta de control ESP32, este mismo se podrá encontrar completo en el ANEXO 8 del documento. Logrando obtener los valores de medida y también poder visualizarlos en el mismo instante de tiempo. Para poder ejecutar los procesos al mismo tiempo, a la ESP32 se le ejecutó e instaló una librería que permite trabajar con hilos o multiprocesos, esto es de gran ayuda ya que permite que la tarjeta pueda procesar tareas en paralelo sin afectar el funcionamiento de estas.

5. RESULTADOS Y ANÁLISIS DE RESULTADOS

5.1 Temperatura

Para la obtención de la medición de la temperatura, se realizaron dos pruebas de obtención y medición de datos en el laboratorio de comunicaciones de la Universidad Nacional de Colombia. La primera prueba consistió en realizar el incremento de temperatura del sistema de control cada 2°C entre el rango de 30°C a 50°C y luego apagar el sistema para ver sus gráficas de respuesta.

La gráfica obtenida para la prueba 1 se encuentra en la figura 46.

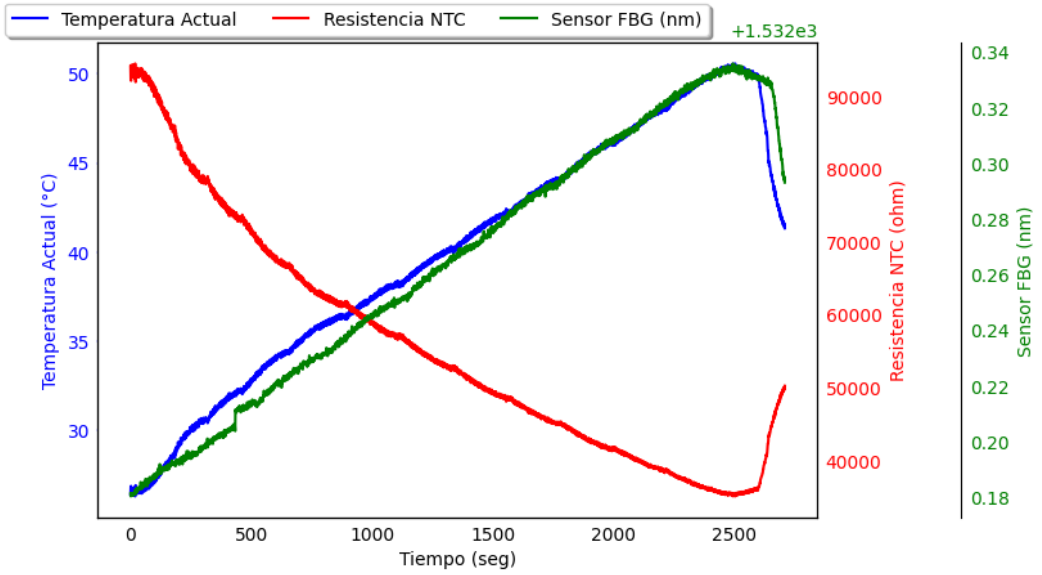

Figura 46. Gráfica de resultados de la prueba 1 del sistema de temperatura.

La figura 46 contiene 3 mediciones comparadas en el tiempo, la primera es la medición de Temperatura Actual (°C) que es la lectura de temperatura leída por el termistor NTC identificada de color azul, la segunda es la medida de resistencia actual del termistor que es calculada mediante programación y se identifica de color rojo y la última medida es la medición de la longitud de onda obtenida por el interrogador óptico distinguida de color verde. Como factor a resaltar, al desconectar la fuente en el momento que llegó a los 50°C, vemos que, al irse disminuyendo la temperatura, la línea verde empieza a tomar un poco más de tiempo en reducir su medición.

La segunda prueba consistió en realizar el mismo incremento de temperatura desde 32°C en pasos de 2°C hasta llegar a 50°C y luego realizar el decrecimiento de temperatura desde 50°C hasta 30.5°C en pasos de 5°C.

Para la segunda prueba, se obtuvo la gráfica presente en la figura 47.

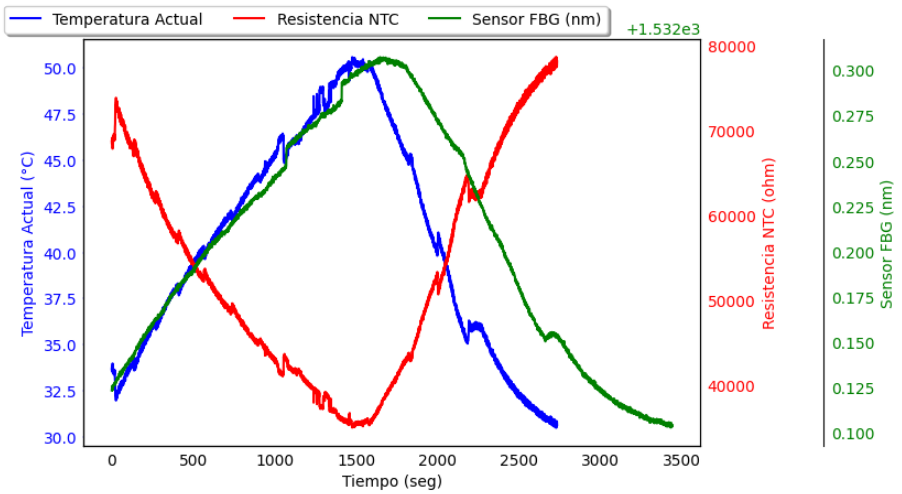

Figura 47. Gráfica de resultados de la prueba 2 del sistema de temperatura.

Los colores de identificación de las líneas se mantienen igual a la figura 46 de la prueba 1. Analizando a detalle ambas gráficas de resultados, se identifican algunos factores cruciales del por qué el comportamiento de la línea del sensor FBG. Como lo muestra la figura 37, el sensor de fibra óptica está embebido en la pasta térmica, dando como resultado la generación de inercia térmica para la lectura de la temperatura. Esta inercia térmica se puede entender como un retardo presentado en los cambios de variación de temperatura, haciendo que la pasta térmica conserve por un tiempo la temperatura generando el retraso de valores de lectura. Para el termistor NTC no ocurre esto ya que este está en contacto directamente con el metal del bloque de aluminio, por lo que se logra identificar una respuesta más rápida en la lectura de la temperatura.

Otro factor crucial reconocido consiste en la ubicación del sensor FBG y del termistor. Como se evidencia en la figura 37, el termistor (sensor de color blanco ubicado en la parte superior del bloque sujeto con un tornillo) está ingresando al bloque buscando la posición más cercana al cartucho calefactor, eso con el fin de poder obtener la mejor medida posible de temperatura presente en el bloque reduciendo así que intervenciones externas tales como corrientes de aire afecten la medición. En cambio, el sensor de fibra óptica está ubicado en

la parte superior del bloque, no es mucha la diferencia de distancia, pero si es notoria ya que la temperatura que llegue a medir el termistor tendrá un mínimo retardo por dicha diferencia de posición.

Tomando los valores obtenidos de resistencia del termistor y de temperatura leída por el termistor de la prueba 1 se realiza un análisis de dichas lecturas con los valores presentados en el datasheet del termistor.

Se tomaron valores aleatorios de temperatura entre 30°C a 50°C presentes en el datasheet y se calculó el porcentaje de error presente entre el valor teórico y el valor experimental, sus resultados se exponen en la tabla 12.

Tabla 11. Tabla comparativa de medidas de temperatura experimentales y teóricas del termistor de la prueba 1 [40].

Temp(°C) datasheet	**Resistencia termistor mínima**	**Resistencia termistor central**	**Resistencia termistor máxima**	**Temp(°C) calculada por código**	**Resistencia termistor calculada**
32	73154	74115	75081	31.94	73997
37	59318	60222	61134	37.07	59713
40	52436	53300	54173	39.99	53024
43	46437	47259	48090	43.05	46933

Con estos valores, se calcula el porcentaje de error que está definido por la siguiente ecuación:

$$\%Error = \frac{|valor\ experimental - valor\ teórico|}{valor\ teórico} \quad (13)$$

Tabla 12. Porcentajes de error de la prueba 1 con medidas aleatorias.

Temperatura (°C)	**Porcentaje de error del valor de resistencia**
32	0,1592%
37	0,8452%
40	0,5178%
43	0,6898%

Para la prueba 1, se logra identificar que el porcentaje de error presente en las medidas tomadas es mínimo, logrando así una buena lectura de temperatura obtenida mediante (6) y (7). Para la prueba 2, se tomaron medidas de temperatura específicamente cuando se le está reduciendo la temperatura deseada.

Tabla 13. Tabla comparativa de las medidas experimentales y teóricas de la prueba 2 [40].

Temp(°C) datasheet	Resistencia termistor mínima	Resistencia termistor central	Resistencia termistor máxima	Temp(°C) calculada por código	Resistencia termistor calculada
50	35119	35840	36571	50.18	35635
45	42865	43659	44462	45.1	43316
40	52436	53300	54173	40.28	52411
35	64467	65395	66330	35.10	64771

Con la tabla de las mediciones, se calculan los porcentajes con (13) y se exponen en la tabla 14.

Tabla 14. Porcentajes de error de la prueba 2.

Temperatura (°C)	**Porcentaje de error del valor de resistencia**
50	0,572%
45	0,7856%
40	1,6679%
35	0,9542%

Para la prueba uno, donde se escogieron temperaturas cuando se estaba calentando el bloque de aluminio, los resultados de los porcentajes de error son inferiores al 1% demostrando así que (7) es una fórmula adecuada para hallar el valor de resistencia del termistor en un Puente de Wheatstone.

En la prueba 2, donde se escogió directamente tomar medidas cuando la temperatura disminuía, los porcentajes obtenidos para la temperatura de 40°C y 35°C empezaron a tener un incremento donde hasta el de 40°C superó el 1,5%. A pesar de esto, siguen siendo medidas confiables y precisas.

Mediante estas comparaciones, se cumple el requerimiento el cual dice que el sistema de temperatura debe presentar un error mínimo.

5.2 Deformación

Para el sistema de control de deformación los primeros resultados obtenidos fueron los voltajes que se pudieron obtener tanto de forma teórica como de forma práctica en la caracterización del sensor tipo galga extensiométrica, para ello de forma teórica teniendo en cuenta los valores de resistencia escogidos:

$R_1 = 100\Omega$
$R_2 = 350\Omega$
$R_3 = 100\Omega$
$R_G = 350\Omega$

$$V_A - V_B = \left(\frac{350\Omega}{100\Omega + 350\Omega} - \frac{350\Omega}{100\Omega + 350\Omega}\right) * 5V = 0V$$

Si fijamos el resultado el voltaje de salida con los valores de resistencia todos iguales el puente de Wheatstone da como resultado un voltaje de 0V, lo que indica que este estará en equilibrio. Pero teniendo en cuenta que en valores reales una resistencia es aproximada y casi siempre es difícil encontrar un valor exacto, dentro de los valores medidos con las resistencias usadas en el circuito de medición los valores obtenidos fueron los siguientes:

$R_1 = 98.3\Omega$
$R_2 = 347.1\Omega$
$R_3 = 98.7\Omega$
$R_G = 348\Omega$

$$V_A - V_B = \left(\frac{347.1\Omega}{98.3\Omega + 347.1\Omega} - \frac{348\Omega}{98.7\Omega + 348\Omega}\right) * 5V = 1.2658mV$$

El valor de R_G se midió con un valor de 348Ω, teniendo en cuenta que la viga al estar soportando el peso que le genera la cadena con el eje del motor provoca que se tense un poco y por lo tanto el valor de la galga extensiométrica en resistencia sea menor al valor

inicial que es 350Ω. Teniendo en cuenta esto cada vez que se tense la viga de acero esta provocará que la resistencia disminuya y el voltaje aumente, por lo tanto si realizamos el cálculo teórico de los posibles voltajes bajando el valor de la resistencia de la galga extensiométrica daría lo siguiente:

Tabla 15. Tabla valores resistencia caracterización deformación.

Valor Resistencia Galga	**Voltaje Medido a la salida**	**Voltaje Amplificado con Ganancia de 100.**
348Ω	1.2658mV	0.1265V
347Ω	3.7445mV	0.3744V
346Ω	6.2344mV	0.6234V
345Ω	8.7355mV	0.8735V
344Ω	11.2479mV	1.12V

Los valores anteriores si se toma el valor de 0.1265V correspondiente al valor de resistencia de 348Ω, midiendo realmente en la práctica el voltaje de salida del amplificador en donde la viga está estática sin tensarse, se puede medir un voltaje de 0.1196V, lo que quiere decir que hay un pequeño porcentaje de error en la medida teórica a la práctica. Se calculará con la ecuacion (13) a continuación:

$$\%Error = \frac{|0.1196V - 0.1265V|}{0.1265V} * 100\%$$

$$\%Error = 5.45\%$$

Este porcentaje del 5.45% puede deberse a que las resistencias del puente de Wheatstone puedan tener una variación a la resistencia medida originalmente, algunas resistencias pueden cambiar su valor nominal con el tiempo debido a factores, como la temperatura, humedad si la llegase a haber y estrés eléctrico. Se evidencia que las variaciones en el voltaje son pequeñas y que fácilmente se pueden llegar a aproximar al valor teórico calculado.

Siguiente a esto, lo primero que se hizo para comprobar este tipo de resultados fue hacer una caracterización conjunta entre el sensor galga y el sensor FBG ambos midiendo en el mismo tiempo un peso en específico, en uno de ellos se tensó la viga hasta deformarla en el

rango de deformación obtenido para la planta didáctica, y lentamente se dejó de tensar hasta volver a su posición original.

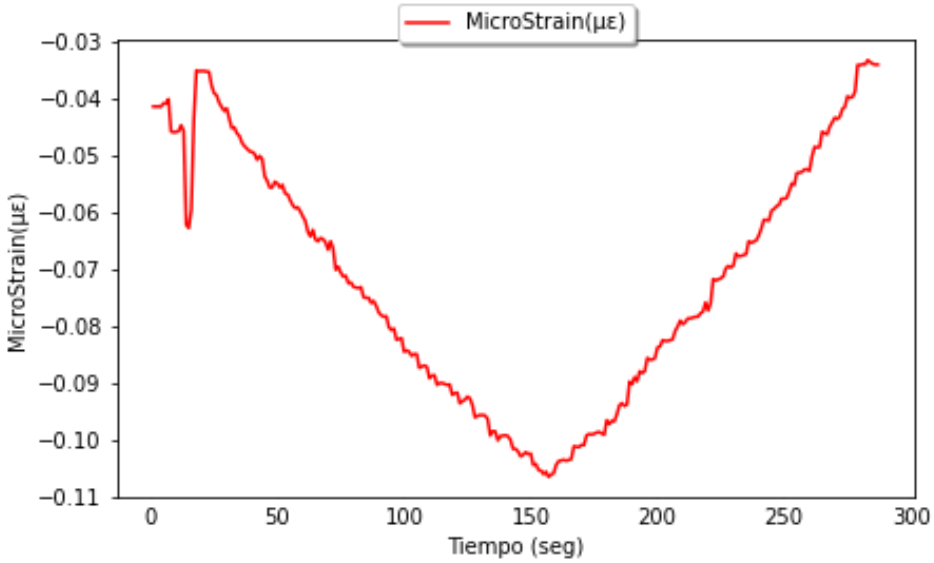

Figura 48. Gráfica de resultados de caracterización de la prueba del sistema de deformación con galga.

En la figura 48 se puede observar la curva Strain/Tiempo donde se puede ver el momento donde se tensa la viga en un tiempo de 150Seg, alcanzando un aproximado de -0.11µStrain en la medición, y como a medida que dejamos de tensar la viga este valor disminuye de nuevo al valor nominal o inicial. Se hizo la misma tarea con el sensor de fibra óptica obteniendo el comportamiento que se muestra en la figura 49.

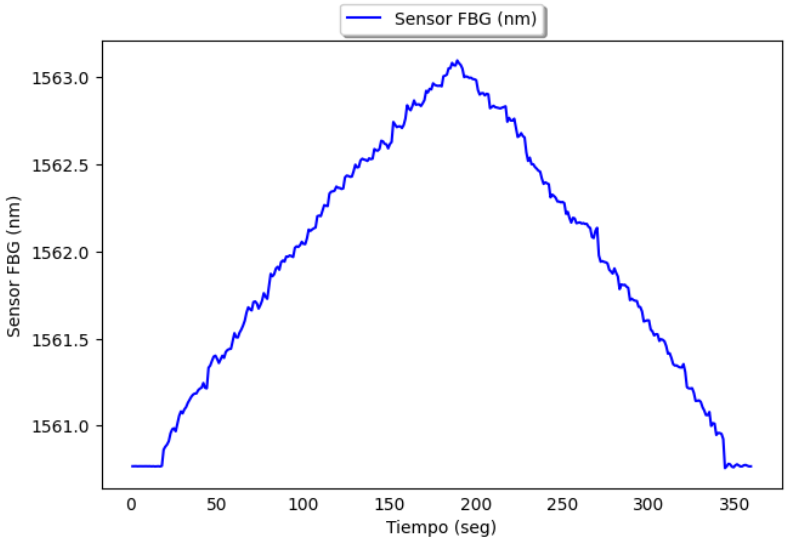

Figura 49. Gráfica de resultados de la prueba caracterización del sistema de deformación con sensor FBG.

El comportamiento de la gráfica en la figura 49 es muy similar al obtenido en la medición de la figura 48, y si nos fijamos cada vez que tensamos la viga la longitud de onda crece bastante en el tiempo, y cuando esta se deja de tensar la longitud de onda disminuye regresando a su posición original. Si comparamos ambas mediciones obtenemos la siguiente gráfica:

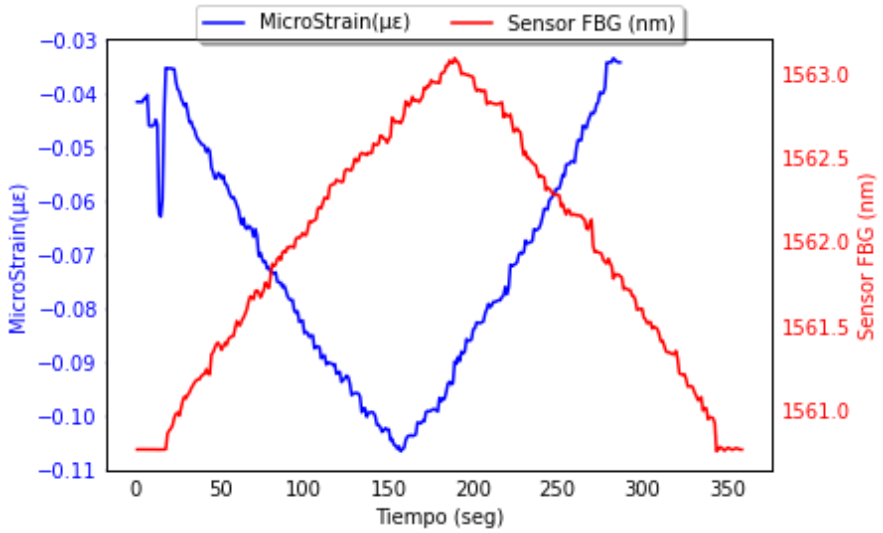

Figura 50. Gráfica de resultados de la caracterización del sistema de deformación con galga y sensor FBG.

Si se observa la figura 50, las gráficas convergen en sentido diferente esto se debe a la magnitud negativa que se tiene en la medición de deformación, debido al posicionamiento que tiene la galga extensiométrica dentro del puente de Wheatstone, donde la galga en este caso su sentido de compresión está para que el voltaje incremente y la lectura de estrés sea negativa [45]. Para poder entenderlo de mejor manera la figura 51 explica dicho comportamiento.

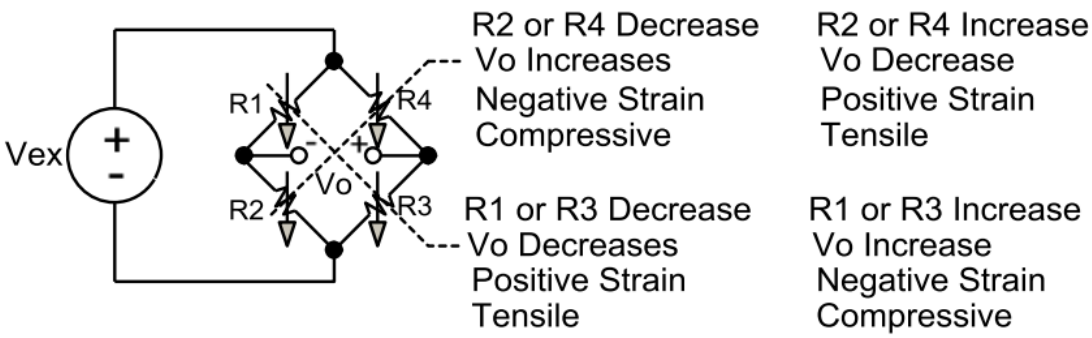

Figura 51. Posicionamiento Lectura Negativa Estrés en la viga [45].

El primer ejercicio realizado para corroborar las mediciones del sensor de fibra, como de la galga extensiométrica fue determinar con el giro de motor, 4 fases donde el motor deformaría la viga de acero en 3 diferentes posiciones, y luego esta regresaría a su posición original, esto 4 veces.

Se realiza la primera medida tomando los valores medidos por el sensor de fibra óptica, tomando medidas en magnitud de nm(nanómetros) contra tiempo (Segundo), obteniendo la siguiente grafica como resultado:

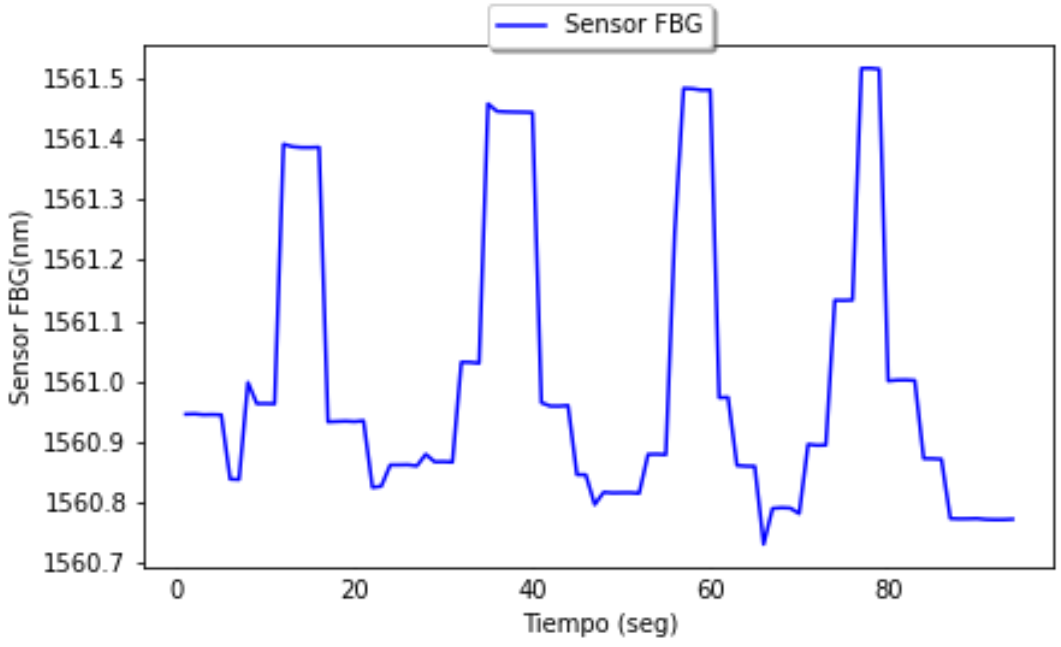

Figura 52. Medición Deformación 4 Fases Viga de Acero con sensor FBG.

Como se puede observar en la figura 52, cada pico corresponde a una de las 4 fases en donde la viga de acero de deforma, después de cierto tiempo se deja de deformar para retornar a su posición original. Si nos fijamos también los picos mantienen una longitud de onda entre 1561.5nm y 1560.7nm, siendo el último pico el que tiene una longitud mayor. Estos pequeños cambios y ruidos que se pueden ver en las bases de los picos suelen deberse también a que el motor genera un movimiento algo brusco para deformar la viga y esto provoca variaciones en la medida, pero no afecta gravemente las medidas tomadas.

Ahora comparando con el sensor de galga extensiométrica se realiza el mismo ejercicio y se identifica que al realizar el mismo ejercicio de deformar con el motor en 4 fases distintas, este tiene un comportamiento similar al medido con el sensor de fibra, solo que esta vez midiendo la magnitud en Deformación (microstrain) contra tiempo, logrando observar la gráfica presente en la figura 53.

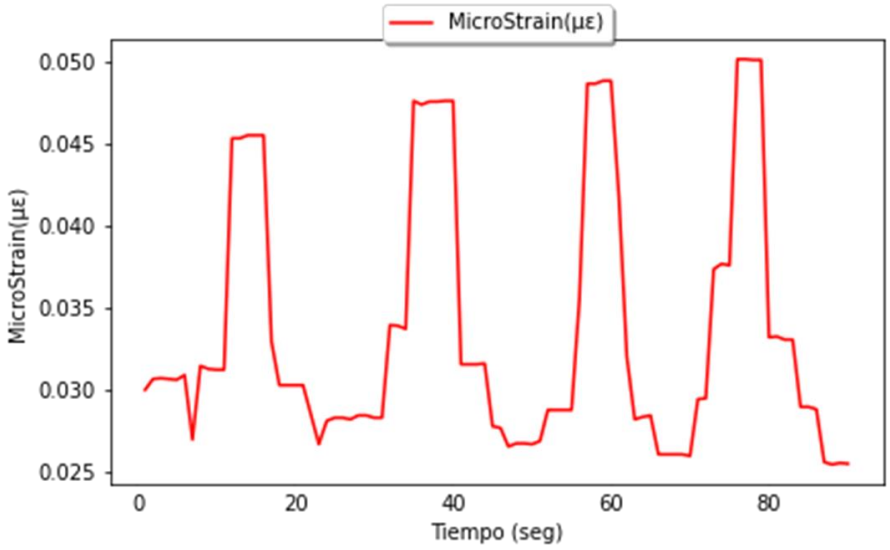

Figura 53. Medición Deformación 4 Fases Viga de Acero con galga extensiométrica.

En la gráfica 53 se observa que el pico más alto de 0.050$\mu\varepsilon$ y el más bajo de 0.025$\mu\varepsilon$, el comportamiento se ve con las 4 fases que la sensibilidad es menor, ya que se ve menos el ruido realizado por el motor durante la deformación de la viga de acero. Se debe tener presente que las unidades medidas en $\mu\varepsilon$ son negativas esto debido al cálculo de compresión que se hizo con la medida tomada en el Puente de Wheatstone y esto por la posición en la que se ubicó la galga, comportamiento expuesto en la figura 51. Para

comparar las gráficas del sensor de fibra con el de galga extensiométrica, se normalizaron las medidas para obtener valores positivos, esto con el fin de hacer una superposición entre las 2 gráficas y verificar su comportamiento. Logrando como resultado la gráfica de la figura 54.

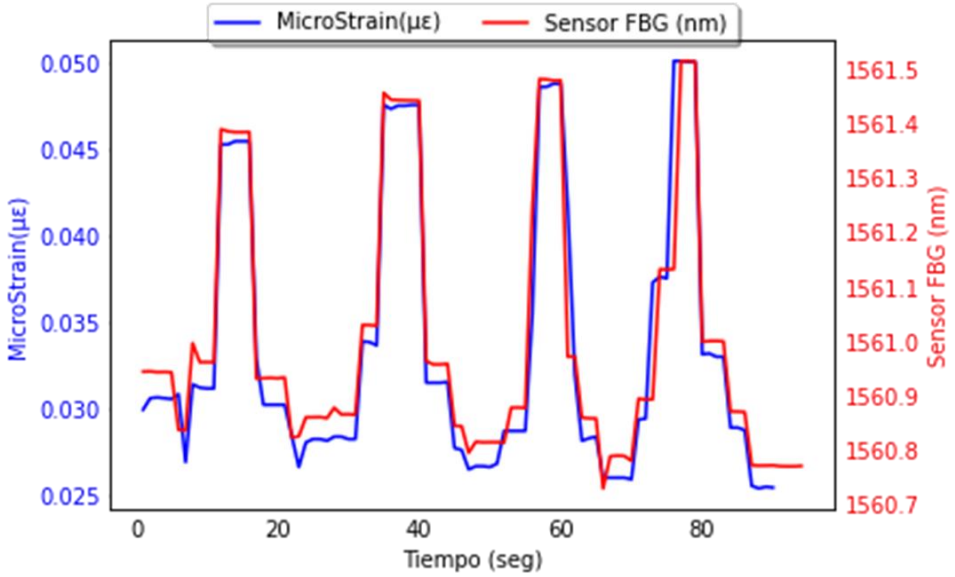

Figura 54. Medición Deformación 4 Fases Viga de Acero con galga extensiométrica y sensor FBG.

En la figura 54, podemos observar que, al hacer superposición entre el comportamiento de ambos sensores, se puede ver que el comportamiento es relativamente similar si vemos como se ejecutó cada una de las fases, con esto concluyendo que ambos sensores están midiendo la deformación realizada sobre la viga de forma correcta, correspondiendo a la magnitud y al tiempo en el que fueron tomadas las muestras. Por último, se puede evidenciar que en ambas graficas se muestra algo de ruido provocado por el motor, algo ya evidenciado en las gráficas individuales.

6. CONCLUSIONES

Respecto del sistema de control de temperatura, las mediciones realizadas tanto con sensores eléctricos como con sensores ópticos muestran un comportamiento concordante. En trabajos futuros, se debe cuantificar el error presente en la relación entre longitud de onda y temperatura ya que se presentan ligeras diferencias en las mediciones obtenidas.

Las mediciones de deformación realizadas con la planta evidencian un comportamiento coherente entre los sensores FBG y galgas extensiométricas. Se propone como trabajo a futuro la cuantificación de la relación entre la deformación y la longitud de onda. Además, se recomienda verificar la medida de deformación con un sensor de referencia correctamente calibrado.

Para la caracterización y comparación de medidas de deformación, se recomienda relacionar los valores de microstrain medidos por el sensor de fibra óptica realizando el cálculo con la ecuación suministrada por la hoja de datos del sensor.

Se construyó una planta didáctica con secciones separadas para la medición de deformación y temperatura independientemente con lo cual se simplificó el proceso de medición ya que, bajo condiciones controladas, no se requieren sensores adicionales de compensación. Si se considera necesario, en un trabajo futuro se podría plantear el uso de un sensor adicional compensando la temperatura para mejorar el sistema.

En los dos sistemas de medición se implementaron dos Puente de Wheatstone que permiten obtener una medida adecuada de cada una de las variables. Una posible mejora consiste en la elección de resistencias con una precisión menor al 5%.

7. RECOMENDACIONES

Para mejorar las mediciones del sensor FBG con el sistema de temperatura, se sugiere buscar la forma de ubicar la fibra en una posición donde se encuentre más cercano al termistor, buscando así reducir los errores de mediciones a lo largo de las pruebas. Como segunda sugerencia, se propone buscar otra forma de pegar o al menos de sujetar fijamente el sensor FBG en el bloque de aluminio, esto con el objetivo de erradicar la inercia térmica generada por la pasta térmica. Teniendo en cuenta el circuito, se recomienda realizar una regulación del voltaje suministrado al Puente de Wheatstone, ya que este voltaje influye directamente en el cálculo de la resistencia del termistor y del cálculo de la temperatura. Es por esto se sugiere certificar que el circuito se alimente con 3.3V constantes buscando reducir la variación mínima presentada en los valores de temperatura medidos por el termistor.

Desde la implementación del sistema de control de deformación, se ha identificado que el proceso de medición es fiable. No obstante, es posible hacer varias recomendaciones para mejorar la calidad y la precisión de la medición. En este sentido, se recomienda realizar pruebas utilizando siempre varios sensores, teniendo en cuenta que algunos ofrecen una mejor percepción en la medición y pueden contribuir a que los cambios de resistencia sean más precisos. También es importante verificar los valores de resistencia utilizados en el puente de Wheatstone, ya que pueden afectar el voltaje de salida, lo que a su vez podría generar un pequeño porcentaje de error en la medición teórica en comparación con la práctica. Lo ideal es minimizar al máximo ese porcentaje de error.

Como en ambos sistemas se utilizó el Puente de Wheatstone, se recomienda implementar resistencias que tengan el valor más cercano al ideal, tratando de reducir las variaciones presentes en las medidas, al igual que certificar que ambos circuitos estén conectados a fuentes de alimentación que entreguen un valor de voltaje constante.

Por último, es crucial destacar la importancia de una correcta instalación de los sensores de galga extensiométrica y los sensores de fibra óptica. La instalación adecuada de estos sensores en la superficie que se deformará es crucial para garantizar una medición precisa.

8. REFERENCIAS BIBLIOGRÁFICAS

[1] F. Navarro-Henríquez, «Sensores de fibra óptica FBG para el monitoreo de la salud estructural de los puentes», *Rev. Tecnol. En Marcha*, vol. 27, n.º 4, p. 3, nov. 2014, doi: 10.18845/tm.v27i4.2080.

[2] S. Silvestre, «Optoelectrónica, fotónica y sensores». Prague: European Virtual Learning Platform for Electrical and Information Engineering, 2016. [En línea]. Disponible en: https://upcommons.upc.edu/handle/2117/103770?locale-attribute=en

[3] fortiz, «Historia De La Fibra óptica - Una Breve Cronología de la Fibra óptica», La fibra optica | los cables de fibra optica | Cursos de Fibra optica. Accedido: 26 de febrero de 2024. [En línea]. Disponible en: https://lafibraopticaperu.com/historia-de-la-fibra-optica/

[4] «Las ventajas y desventajas de la fibra óptica | Comunidad FS», Knowledge. Accedido: 26 de febrero de 2024. [En línea]. Disponible en: https://community.fs.com/es/article/the-advantages-and-disadvantages-of-fiber-optic-transmission.html

[5] G. Rajan, D. Callaghan, Y. Semenova, M. McGrath, E. Coyle, y G. Farrell, «A Fiber Bragg Grating-Based All-Fiber Sensing System for Telerobotic Cutting Applications», *IEEE Sens. J.*, vol. 10, n.º 12, pp. 1913-1920, dic. 2010, doi: 10.1109/JSEN.2010.2049260.

[6] A. M. Juan Jiménez, «DISPOSITIVO A BASE DE FIBRA ÓPTICA PARA EL MONITOREO DE LA DEFORMACIÓN EN VIGAS ESTRUCTURALES EN GRANDES CLAROS», Instituto Tecnológico de Tuxtla Gutiérrez, TUXTLA GUTIERREZ, 2020. [En línea]. Disponible en: http://repositoriodigital.tuxtla.tecnm.mx/xmlui/handle/123456789/2039

[7] D. Samaniego Chamba, «Desarrollo de un sensor óptico para detección de fugas», *Ing. Agua*, vol. 18, n.º 1, p. ix, sep. 2014, doi: 10.4995/ia.2014.3293.

[8] F. J. Hoyos Vélez, C. M. Serpa Imbet, y N. D. Gómez Cardona, «Medición de microdeformaciones en losas viales usando sensores de redes de Braggen fibras ópticas», 2010, [En línea]. Disponible en: https://repository.eafit.edu.co/server/api/core/bitstreams/93b75ee3-4321-4ea5-87b8-1569a79174bc/content

[9] C. A. Triana Infante, «Diseño e implementación de un sensor óptico basado en redes de difracción de Bragg para el monitoreo de estructuras», Universidad Nacional de Colombia, Bogotá, Cundinamarca, 2019. [En línea]. Disponible en: https://repositorio.unal.edu.co/handle/unal/21130

[10] J. E. A. Gil, «ESQUEMA PARA LA IMPLEMENTACIÓN DE MEDICIÓN DE DEFORMACIONES EN EDIFICACIONES DE HORMIGÓN ANDREA CORREA URIBE», 2015, [En línea]. Disponible en: https://repository.eia.edu.co/handle/11190/2045

[11] F. E. Barón Moreno, «Diseño de un sistema de medición de temperatura para líneas de transmisión y distribución de energía utilizando sensores ópticos basados en redes de difracción de Bragg», Universidad Nacional de Colombia, Bogotá, Cundinamarca, 2020. [En línea]. Disponible en: https://repositorio.unal.edu.co/handle/unal/77535

[12] D. F. A. Aillon, «ANALISIS DE LA VIABILIDAD PARA LA IMPLEMENTACIÓN DE REDES DE SENSORES DE FIBRA ÓPTICA EN EL SECTOR INDUSTRIAL DE SAN JOSÉ DE CÚCUTA.», Universidad Libre Seccional Cúcuta, Cúcuta, Norte de Santander, 2022. [En línea]. Disponible en: https://repository.unilibre.edu.co/handle/10901/23775?locale-attribute=en

[13] K. O. Hill y G. Meltz, «Fiber Bragg grating technology fundamentals and overview», *J. Light. Technol.*, vol. 15, n.º 8, pp. 1263-1276, ago. 1997, doi: 10.1109/50.618320.

[14] C. A. Triana-Infante, M. Varón-Durán, y D. Pastor-Abellán, «Validación de sensores basados en redes de difracción de bragg (fbgs) para deformación y temperatura», *ITECKNE*, vol. 11, n.º 2, pp. 172-182, dic. 2014, doi: 10.15332/iteckne.v11i2.730.

[15] «sm125.pdf».

[16] R.- ASALE y RAE, «temperatura | Diccionario de la lengua española», «Diccionario de la lengua española» - Edición del Tricentenario. Accedido: 4 de octubre de 2023. [En línea]. Disponible en: https://dle.rae.es/temperatura

[17] «Como elegir el hotend correcto». Accedido: 4 de octubre de 2023. [En línea]. Disponible en: https://filament2print.com/es/blog/96_Como-elegir-el-hotend-correcto.html

[18] «Hotend Volcano 12/24V - Zuluprints Bogotá, Colombia», Zuluprints. Accedido: 3 de octubre de 2023. [En línea]. Disponible en: https://zuluprints.co/producto/hotend-volcano-12-24v/

[19] B. J.L, «Termistor: Qué es, Funcionamiento y Aplicaciones», Electrónica Online. Accedido: 5 de octubre de 2023. [En línea]. Disponible en: https://electronicaonline.net/componentes-electronicos/resistor/termistor/

[20] «¿Qué significa NTC en un termistor y cómo funciona? | SDI». Accedido: 5 de octubre de 2023. [En línea]. Disponible en: https://sdindustrial.com.mx/blog/que-significa-ntc-en-un-termistor/

[21] «Termistor NTC 100K - Zuluprints Bogotá». Accedido: 5 de octubre de 2023. [En línea]. Disponible en: https://zuluprints.co/producto/termistor-ntc-100k/

[22] «termistores_NTC_1.pdf». Accedido: 20 de octubre de 2023. [En línea]. Disponible en: http://dfs.uib.es/GTE/education/industrial/tec_electronica/teoria/termistores_NTC_1.pdf

[23] K. Ogata, «Ingeniería de control moderna», vol. 5, pp. 567-614, 2010.

[24] «Optimizando Desempeño de Deformación Mecánica para Materiales No Homogéneos», https://es.omega.com/. Accedido: 28 de febrero de 2024. [En línea]. Disponible en: https://es.omega.com/technical-learning/optimizando-el-desempeno-de-sensores-de-deformacion-mecanica.html

[25] ana ramirez, «Deformacion y resistencia de los materiales», Monografias.com. Accedido: 23 de octubre de 2023. [En línea]. Disponible en: https://www.monografias.com/trabajos96/deformacion-y-resistencia-materiales/deformacion-y-resistencia-materiales

[26] carakenio73, «Driver de motor DC – Electrónica de potencia», dademuchconnection. Accedido: 5 de octubre de 2023. [En línea]. Disponible en: https://dademuchconnection.wordpress.com/2018/04/26/driver-de-motor-dc-electronica-de-potencia/

[27] «Tutorial de Uso del Módulo L298N», Naylamp Mechatronics - Perú. Accedido: 5 de octubre de 2023. [En línea]. Disponible en: https://naylampmechatronics.com/blog/11_tutorial-de-uso-del-modulo-l298n.html

[28] «BTS7960 Controlador Motor 43A de Alta Potencia Puente H», Electronilab. Accedido: 5 de octubre de 2023. [En línea]. Disponible en: https://electronilab.co/tienda/bts7960-controlador-motor-43a-de-alta-potencia-puente-h/

[29] «BTS7960 Driver motor 43A de alta potencia BricoGeek | BricoGeek.com». Accedido: 5 de octubre de 2023. [En línea]. Disponible en: https://tienda.bricogeek.com//controladores-motores/1560-bts7960-driver-motor-43a-de-alta-potencia.html

[30] P. Quintero y N. Fernando, *Microcontroladores Microchip, Atmel, NXP Freescale y Texas Instruments. Pasos para una programación con éxito*. Universidad Piloto de Colombia, 2019. Accedido: 5 de octubre de 2023. [En línea]. Disponible en: http://repository.unipiloto.edu.co/handle/20.500.12277/7080

[31] «Random Nerd Tutorials | Learn ESP32, ESP8266, Arduino, and Raspberry Pi». Accedido: 5 de octubre de 2023. [En línea]. Disponible en: https://randomnerdtutorials.com/

[32] «Raspberry Pi Pico W - RP2040 WiFi + Regletas Sin Soldar», Electronilab. Accedido: 5 de octubre de 2023. [En línea]. Disponible en: https://electronilab.co/tienda/raspberry-pi-pico-w-rp2040-wifi-regletas-sin-soldar/

[33] «Raspberry Pi Pico: Programmieren mit dem günstigen Mikrocontroller». Accedido: 5 de octubre de 2023. [En línea]. Disponible en: https://tutorials-raspberrypi.de/raspberry-pi-pico-mikrocontroller-programmieren/

[34] «conversor analogo digital ads1115», Dualtronica. Accedido: 21 de octubre de 2023. [En línea]. Disponible en: https://dualtronica.com/inicio/439-conversor-analogo-digital-ads1115-16-bit-4-canales-i2c.html

[35] «Puente de Wheatstone | José Casares». Accedido: 21 de octubre de 2023. [En línea]. Disponible en: https://josecasares.com/puente-de-wheatstone/

[36] «ADS1115 convertidor analógico digital ADC para Arduino y ESP8266». Accedido: 21 de octubre de 2023. [En línea]. Disponible en: https://programarfacil.com/blog/arduino-blog/ads1115-convertidor-analogico-digital-adc-arduino-esp8266/

[37] C. Rubio Ramírez, G. M. Lizarazo, y E. Vera Duarte, «Termoelectricidad: uso de las celdas peltier en el campo de la refrigeración y sus principales aplicaciones», *INVENTUM*, vol. 12, n.º 22, pp. 9-16, nov. 2017, doi: 10.26620/uniminuto.inventum.12.22.2017.9-16.

[38] «Cartucho calefactor 40W 12/24V - Zuluprints Bogotá», Zuluprints. Accedido: 20 de octubre de 2023. [En línea]. Disponible en: https://zuluprints.co/producto/cartucho-calefactor-40w/

[39] «Celda Celula Peltier 12705 Tec1-12705 Dc 12V 5A 60W Disipador - yorobotics». Accedido: 20 de octubre de 2023. [En línea]. Disponible en: https://yorobotics.co/producto/celda-celula-peltier-12705-60w-tec1-12705-dc12v-5a-disipador

[40] «NTCM-100K-B3950.pdf».

[41] I. Urgilés y P. Xavier, «Aplicación de galgas extensiométricas en el laboratorio de mecánica de materiales de la carrera de Ingeniería Mecánica para la obtención de deformaciones en elementos sometidos a cargas combinadas».

[42] «esp32-wroom-32_datasheet_en.pdf».

[43] jameswilson, «ESP32 Pinout, Datasheet, Features & Applications - The Engineering Projects». Accedido: 23 de octubre de 2023. [En línea]. Disponible en: https://www.theengineeringprojects.com/2020/12/esp32-pinout-datasheet-features-applications.html

[44] «Cálculos de resistencias en puente de Wheatstone - Resistencias en electrónica». Accedido: 23 de octubre de 2023. [En línea]. Disponible en: https://www.aulafacil.com/cursos/electronica/resistencias-en-electronica/calculos-de-resistencias-en-puente-de-wheatstone-l34232

[45] «White-Paper-Intro-StrainGauge_699B.pdf».

ANEXOS

ANEXO 1: REFERENCIA [40] NTCM-100K-B3950.pdf

NTCM-100K-B3950.pdf

ANEXO 2: REFERENCIA [45] White-Paper-Intro-StrainGauge_699B.pdf

White-Paper-Intro-StrainGauge_699B.pdf

ANEXO 3: Datos de caracterización del sistema de temperatura (Sección 4.3.5)

datos_caracterizacion_sist_temp.xlsx

ANEXO 4: CÓDIGO COMPLETO DE SISTEMA DE TEMPERATURA

```
#Importación de librerías a usar
from machine import ADC, Pin, PWM, Timer, I2C
from utime import sleep_ms
from time import sleep
import math
import time
import uos
# import network
# import urequests

#Conexion del protocolo de I2C para el conversor análogo-digital ADS1115
ads = I2C(1, scl=Pin(15), sda=Pin(14))

print(ads.scan())
global channel}
#Dirección establecida del ADS1115 para la identificación en la Raspberry Pi Pico
address = 72

def readConfig():
    ads.writeto(address, bytearray([1]))
    result = ads.readfrom(address, 2)
    return result[0]<<8 | result[1]

#Lectura de los valores de las entradas análogas del ADC
def readValueFrom(channel):
    config = readConfig()
    config &= ~(7<<12)# Clear MUX bits
    config &= ~(7<<9)# Clear PGA
    config |= (7 & (4+ channel))<<12
    config |= (1<<15) # Trigger next conversion
    config |= (1<<9) #gain 4.096volts
    config = [int(config>>i & 0xff) for i in [8,0]]
    ads.writeto(address, bytearray([1] + config))

    config = readConfig()
    while (config & 0x8000) == 0:
        config = readConfig()
```

```
    ads.writeto(address, bytearray([0]))
    result = ads.readfrom(address, 2)
    return result[0]<<8 | result[1]

#Conversión del valor análogo a valor de voltaje de las entradas que lo requieran
def readVoltsFrom(channel):
    value = readValueFrom(channel)

    return ((4.096 * 2) / 0xffff) * value

#Array de valores análogos del ADC
val = [0,0,0,0]

# Configurar la conexión Wi-Fi
ssid = "Andresito"
password = "97072817021**"

# Conectar a la red Wi-Fi
wifi = network.WLAN(network.STA_IF)
wifi.active(True)
wifi.connect(ssid, password)

# Esperar a que se conecte a la red Wi-Fi
while not wifi.isconnected():
    pass
print("Conectado a la red Wi-Fi")

#Declaracion de voltaje de alimentacion, puertos ADC, PWM y los ENABLE para Puente   #H
del cartucho calefactor
pwm = PWM(Pin(4))
pwm.freq(40000)
IN1 = Pin(3, Pin.OUT)
IN2 = Pin(2, Pin.OUT)

#Modelo del Sistema
K = 89
tau = 345
theta = 35
Ts = 10;
L = theta + Ts/2
e = [0,0,0]    #Vector de error
u = [0,0]      #Vector de Ley de Control

#Resistencias del puente de Wheatstone para NTC
R1 = 96200
R2 = 97700
R3 = 96300

#Parámetros de NTC 100K B3950
bResistance = 3950 #Valor B dado por fabricante
t25Resistance = 100000 #Valor de resistencia del termistor a temperatura de 25°C

#Ecuaciones de Temperatura
t0 = 273.15
t25 = t0 + 25

#Función de mapeo del setpiont, permite establecer el mínimo y máximo
def map(x, in_min, in_max, out_min, out_max):
    return int((x-in_min) * (out_max-out_min) / (in_max-in_min) + out_min)

#Ciclo for que nos permite actualizar el pasado de la ley de control y del error
def update_past(v,kT):
   for i in range(1,kT,1):
      v[i-1]=v[i]
```

```
    return v

#Función de Ley de Control PID Discreto
def PID_Controller(u, e, q0, q1, q2):
    lu = u[0] + q0*e[2] + q1*e[1] + q2*e[0]; #Ley del controlador PID discreto
    # Anti - Windup
    if (lu >= 100.0):
        lu = 100.0

    if (lu <= 0.0):
        lu = 0.0

    return(lu)

#Función que lee el error y la ley de control, calcula el error entre el setpoint y #la temperatura actual y procede a establecer el PWM para el control del cartucho #calefactor
def temporizador(timer):
    global u, e, q0, q1, q2
    #Actualiza los vectores u y e
    u = update_past(u,len(u));
    e = update_past(e,len(e));

    #Calcula el error actual
    e[len(e)-1] = setpoint - temp_x;

    #Calcula la Acción de Control PID
    u_end = PID_Controller(u, e, q0, q1, q2);  #Max= 100, Min=0
    u[len(u)-1] = u_end
    velocidad = int(u[len(u)-1] * 65535 /100);

    #Aplica la acción de control en el PWM
    pwm.duty_u16(velocidad)

def main():
    global setpoint, q0, q1, q2, u, e

    #Llamado de la función temporizador de forma periódica cada 10 segundos
    tim = Timer()
    tim.init(period= 10000, mode=Timer.PERIODIC, callback=temporizador)

    #Factores para la Ley de Control
    kp=(1.2*tau)/(K*L)
    ti=2*L
    td=0.5*L

    q0=kp*(1+Ts/(2*ti)+td/Ts)
    q1=-kp*(1-Ts/(2*ti)+(2*td)/Ts)
    q2=(kp*td)/Ts

    while True:
        global temp_x
        IN1.high()
        IN2.low()
        temp_sum = 0
        val[0] = readVoltsFrom(0) #Puente de Wheatstone negativo (voltaje)
        val[1] = readVoltsFrom(1) #Voltaje de alimentación
        val[2] = readValueFrom(2) #Valor potenciometro Setpoint en 16 bits
        val[3] = readVoltsFrom(3) #Puente de Wheatstone positivo (voltaje)
        Vs = val[1]
        #filtro de promedio movil para mejorar el valor de medición de la temperatura
        for i in range(50):
            Vout = val[3] - val[0] #Voltaje de la diferencia del puente de Wheatstone
            resistance = (R2*R3+R3*(R1+R2)*Vout/Vs)/(R1-(R1+R2)*Vout/Vs) #Cálculo de resistencia del termistor, se obtiene del Puente de Wheatstone
```

```
163.             temp_sum += 1/((math.log(resistance/t25Resistance)/bResistance)+(1/t25))-t0
#Obtención del valor de temperatura actual mediante despeje de la ecuación de los termistores
164.         temp_x = temp_sum/50 # Resultado del filtro de promedio movil
165.
166.         val_pot = val[2] #Obtención del valor del ADC para realizar la conversión de sus
valores al valor de mapeo de 30°C a 50°C
167.         setpoint = map(val_pot,0,25900,30,50)
168.
169.         print("Temp deseada:", setpoint, "Temp actual:", temp_x, "Resistencia:",
resistance, "Vout: ",Vout, val[3], val[0], val[1], val[2])
170.
171.         pc_ip = "192.168.115.150:5000"  # La dirección IP de tu PC del servidor
172.         url = "http://" + pc_ip + "/data"
173.
174.         data = {
175.             "Temperatura_Actual": temp_x,
176.             "Temperatura_Deseada": setpoint,
177.             "Resistencia_NTC": resistance
178.         }
179.
180.         try:
181.             response = urequests.post(url, json=data)
182.             if response.status_code == 200:
183.                 print("")
184.             else:
185.                 print("Error al enviar datos:", response.status_code)
186.             response.close()
187.         except Exception as j:
188.             print("Error al enviar datos:", str(j))
189.         time.sleep(1)
190.
191. if __name__ == '__main__':
192.     main()
193.
```

ANEXO 5: Código completo control giro de motor

```
1. import machine
 2. import time
 3.
 4. # Configuración de los pines de entrada y salida
 5. button_left = machine.Pin(33, machine.Pin.IN, machine.Pin.PULL_UP)
 6. button_right = machine.Pin(32, machine.Pin.IN, machine.Pin.PULL_UP)
 7. motor_left = machine.PWM(machine.Pin(26))
 8. motor_right = machine.PWM(machine.Pin(25))
 9.
10. # Configuración de los valores de PWM
11. motor_left.freq(200)  # Frecuencia de PWM izquierda (Hz)
12. motor_right.freq(200) # Frecuencia de PWM izquierda (Hz)
13.
14. # Velocidad del motor
15. speed_reduced = 193 # Valor PWM
16.
17. debounce_delay = 0.25 #Valor de Retardo de PWM
18. last_left_state = True
19. last_right_state = True
20. left_count = 0
21. right_count = 0
22. max_count = 3 # Conteo Máximo para limitar giros
23.
24. def stop_motors():
25.     motor_left.duty(0)
```

```
    motor_right.duty(0)

def move_left():
    global left_count, right_count
    if left_count < max_count:
        motor_left.duty(speed_reduced)
        motor_right.duty(0)
        left_count += 1
        right_count = 0
        print("Motor girando a la izquierda")

def move_right():
    global left_count, right_count
    if right_count < max_count:
        motor_left.duty(0)
        motor_right.duty(speed_reduced)
        right_count += 1
        left_count = 0
        print("Motor girando a la derecha")

# Bucle principal
while True:
    left_state = button_left.value()
    right_state = button_right.value()

    if not left_state and left_state != last_left_state:
        move_left()
        time.sleep(debounce_delay)

    elif not right_state and right_state != last_right_state:
        move_right()
        time.sleep(debounce_delay)

    else:
        stop_motors()

    last_left_state = left_state
    last_right_state = right_state

    time.sleep(0.05)
```

ANEXO 6: Código completo lectura y muestro de valores de deformación.

```
import utime
import urequests
import network
import uos
import sys
from machine import I2C, Pin

dev = I2C(1, scl=Pin(22), sda=Pin(21)) # Configuracion de Pines para I2C.
print(dev.scan()) # Como la direccion esta conectada a tierra siempre sera 72 el valor leido.

address = 72

def readConfig():
    dev.writeto(address, bytearray([1]))
    result = dev.readfrom(address, 2) # Lee direccion de lectura del I2C
    return result[0] << 8 | result[1] # Retorna el resultado de valor medido

def readValueFrom(channel):
```

```
    config = readConfig()
    config &= ~(7 << 12)  # clear MUX bits
    config &= ~(7 << 9)   # clear PGA
    config |= (7 & (4 + channel)) << 12
    config |= (1 << 15) | (1 << 9)  # ganancia de 4.1 volts
    config = [int(config >> 8), int(config & 0xff)]
    dev.writeto(address, bytearray([1] + config))
    config = readConfig()
    while (config & 0x8000) == 0:
        config = readConfig()
    dev.writeto(address, bytearray([0]))
    result = dev.readfrom(address, 2)
    return result[0] << 8 | result[1]

def readVoltsFrom(channel):
    value = readValueFrom(channel)
    return (value * 4.096) / 32767 # Se  hace diferenciacion para calcular voltage medido por el puente de Wheastone

while True:
    val = readVoltsFrom(0)
    ro = 1.72e-8
    l = 0.82  # Longitud en metros (82 cm convertidos a metros)
    a = 0.259e-6  # Área transversal en metros cuadrados (0.259 mm^2 convertidos a metros cuadrados)
    RL = ro * l / a
    RG = 350  # Resistencia del galvanómetro (Ohms)
    Vex = 5.0  # Voltaje de excitación (Voltios)
    GF = 2.0  # Gauge Factor

    Vr = (val - 0) / Vex  # Tensión relativa (mV/V)
    E_strain = (1 + RL / RG) * (-4 * Vr) / (GF * (1 + 2 * Vr))
    print(f'Voltaje (V): {val:.2f} | Deformación (με): {E_strain:.2f}')

    utime.sleep(1)
```

ANEXO 7: Código completo APP servidor para recepción de datos.

```
import tkinter as tk
from flask import Flask, request, jsonify
from threading import Thread
import csv

app = Flask(__name__) # Comando para ejecutar la aplicacion por medio del servidor

datos_recibidos = []

def crear_ventana(): # Ventana Dinamica haciendo uso de Tkinter
    ventana = tk.Tk()
    ventana.title("Valores Recibidos")
    ventana.geometry("400x170")  # Ancho x Alto

    titulo = tk.Label(ventana, text="Valores Deformación:", font=("Helvetica", 11, "bold"))
    titulo.pack()

    espacio = tk.Label(ventana, text="")
    espacio.pack()

    campos_texto = {} # Diccionario donde se almacenaran los datos

    def actualizar_valores():
```

```
24.         if datos_recibidos:
25.             data = datos_recibidos.pop(0)
26.             guardar_en_csv(data, "datos_deformacion.csv") # Funcion para almacenar los
datos en .csv
27.             for key, value in data.items():
28.                 if key not in campos_texto:
29.                     campos_texto[key] = tk.Text(ventana, height=1, width=30,
font=("Helvetica", 16), state="disabled")
30.                     campos_texto[key].pack()
31.                 campos_texto[key].config(state="normal")
32.                 campos_texto[key].delete(1.0, tk.END)
33.                 campos_texto[key].insert(tk.END, f"{key}: {value}")
34.                 campos_texto[key].config(state="disabled")
35.             ventana.update()
36.         ventana.after(1000, actualizar_valores)  # Actualizar cada 1 segundo
37.
38.     actualizar_valores()  # Llamar a la función por primera vez
39.     ventana.mainloop()
40.
41. def guardar_en_csv(data, filename): # Funcion que permite leer, guardar y escribir sobre
archivos CSV
42.     with open(filename, "a", newline='') as archivo_csv:
43.         campos = data.keys()
44.         writer = csv.DictWriter(archivo_csv, fieldnames=campos)
45.
46.         if archivo_csv.tell() == 0:
47.             writer.writeheader()
48.
49.         writer.writerow(data)
50.
51. @app.route('/data', methods=['POST']) # API que se encarga de recibir los datos enviados
por la ESP32
52. def recibir_datos():
53.     data = request.get_json()
54.     datos_recibidos.append(data)
55.     return jsonify({"mensaje": "Datos recibidos con éxito"})
56.
57. def ejecutar_servidor():
58.     app.run(host='0.0.0.0', port=8000) # IP y puerto donde se abre la conexion de escucha
para recibir conexion.
59.
60. servidor_thread = Thread(target=ejecutar_servidor) # Se usan Hilos para poder procesar
toda la informacion de forma mas optima.
61. servidor_thread.daemon = True
62. servidor_thread.start()
63.
64. if __name__ == '__main__':
65.     crear_ventana()
66.
```

ANEXO 8: Código control de deformación y motor unificado + Envió de información por protocolo WIFI.

```
1. #import time
2. import utime as time
3. import urequests
4. import network
5. import uos
6. import sys
```

```
from machine import I2C, Pin, PWM
import threading

# Configurar la conexión Wi-Fi
ssid = "user"
password = "password"

# Conectar a la red Wi-Fi
wifi = network.WLAN(network.STA_IF)
wifi.active(True)
wifi.connect(ssid, password)

# Esperar a que se conecte a la red Wi-Fi
while not wifi.isconnected():
    pass

print("Conectado a la red Wi-Fi")

# Configuración de los pines para pulsadores
button_left = Pin(33, Pin.IN, Pin.PULL_UP)
button_right = Pin(32, Pin.IN, Pin.PULL_UP)
motor_left = PWM(Pin(26))
motor_right = PWM(Pin(25))

# Configuración de los valores de PWM
motor_left.freq(200)  # Frecuencia de PWM (Hz)
motor_right.freq(200)

# Velocidad del motor
speed_reduced = 193 # Valor definido de PWM

debounce_delay = 0.3  # Tiempo entre giro de motor
last_left_state = True
last_right_state = True
left_count = 0
right_count = 0
max_count = 3 # Contador para limitar giro derecha e izquierda

dev = I2C(1, scl=Pin(22), sda=Pin(21)) # Pines de Conexion I2C
address = 72 # Direccion de lectura cuando el addr esta conectado a tierra, siempre sera 72 su valor.

def readConfig():
    dev.writeto(address, bytearray([1])) # Procesa la lectura de entrada del I2C.
    result = dev.readfrom(address, 2)
    return result[0] << 8 | result[1] # Retorna el resultado de la medida tomada.

def readValueFrom(channel):
    config = readConfig()
    config &= ~(7 << 12)  # clear MUX bits
    config &= ~(7 << 9)   # clear PGA
    config |= (7 & (4 + channel)) << 12
    config |= (1 << 15) | (1 << 9)  # ganancia de 4.1 volts
    config = [int(config >> 8), int(config & 0xff)]
    dev.writeto(address, bytearray([1] + config))
    config = readConfig()
    while (config & 0x8000) == 0:
        config = readConfig()
    dev.writeto(address, bytearray([0]))
    result = dev.readfrom(address, 2)
    return result[0] << 8 | result[1]

def readVoltsFrom(channel):
    value = readValueFrom(channel)
```

```
70.     return (value * 4.096) / 32767
71. # Logica de Giro del Motor
72. def stop_motors():
73.     motor_left.duty(0)
74.     motor_right.duty(0)
75. 
76. def move_left():
77.     global left_count, right_count
78.     if left_count < max_count:
79.         motor_left.duty(speed_reduced)
80.         motor_right.duty(0)
81.         left_count += 1
82.         right_count = 0
83.         #print("Motor girando a la izquierda")
84. 
85. def move_right():
86.     global left_count, right_count
87.     if right_count < max_count:
88.         motor_left.duty(0)
89.         motor_right.duty(speed_reduced)
90.         right_count += 1
91.         left_count = 0
92.         #print("Motor girando a la derecha")
93. 
94. def button_thread():
95. 
96.     global left_count, right_count
97. 
98.     while True:
99.         left_state = button_left.value()
100.         right_state = button_right.value()
101. 
102.         if not left_state and left_state != last_left_state:
103.             move_left()
104.             time.sleep(debounce_delay)
105. 
106.         elif not right_state and right_state != last_right_state:
107.             move_right()
108.             time.sleep(debounce_delay)
109. 
110.         else:
111.             stop_motors()
112. 
113.         last_left_state = left_state
114.         last_right_state = right_state
115. 
116.         time.sleep(0.05)
117. 
118. # Crear y ejecutar el hilo para control de botones
119. button_thread = threading.Thread(target=button_thread)
120. button_thread.start()
121. 
122. while True:
123.     # Leer el valor del sensor y enviar datos
124.     val = readVoltsFrom(0)
125.     ro = 1.72e-8
126.     l = 0.82  # Longitud en metros (82 cm convertidos a metros)
127.     a = 0.259e-6  # Área transversal en metros cuadrados (0.259 mm^2 convertidos a metros
cuadrados)
128.     RL = ro * l / a
129.     RG = 347  # Resistencia del galvanómetro (Ohms)
130.     Vex = 5.0  # Voltaje de excitación (Voltios)
131.     GF = 2.0  # Gauge Factor
132.     Vr = (val - 0) / Vex  # Tensión relativa (mV/V)
```

```
133.     E_strain = (1 + RL / RG) * (-4 * Vr) / (GF * (1 + 2 * Vr))
134.
135.     print(f'Voltaje (V): {val:.2f} | Deformación (με): {E_strain:.2f}')
136.
137.     pc_ip = "192.168.115.150:8000"  # La dirección IP de la PC en la cual esta desplegado
el servidor de recepcion
138.     url = "http://" + pc_ip + "/data"
139. # Datos a enviar en un JSON
140.     data = {
141.         "VoltageMedido": val,
142.         "ValoresStrain": E_strain
143.     }
144.
145.     try:
146.         response = urequests.post(url, json=data)
147.         if response.status_code == 200:
148.             print("Datos enviados con éxito")
149.         else:
150.             print("Error al enviar datos:", response.status_code)
151.         response.close()
152.     except Exception as e:
153.         print("Error al enviar datos:", str(e))
154.
155.     time.sleep(1)
156.
```

ANEXO 9: REFERENCIA [15] «sm125.pdf».

sm125.pdf