Neiva, 23 de agosto de 2024

Señores

CENTRO DE INFORMACIÓN Y DOCUMENTACIÓN

UNIVERSIDAD SURCOLOMBIANA

Neiva, Huila

El (Los) suscrito(s):

__**Nathash Abbarig Guerrero Gonzalez**_____, con C.C. No. __**1075315565**_____,

__**Juan David Daza Perez**_____, con C.C. No. __**1075309776**_____,

_____, con C.C. No. _____,

_____, con C.C. No. _____,

Autor(es) de la tesis y/o trabajo de grado _____

titulado___ **DISEÑO E IMPLEMENTACIÓN DE UN LABORATORIO PERSONAL DE TEMPERATURA PARA LA ASIGNATURA DE CONTROL** _____

presentado y aprobado en el año __**2024**___ como requisito para optar al título de

_____**Ingeniero Electrónico**_____;

Autorizo (amos) al CENTRO DE INFORMACIÓN Y DOCUMENTACIÓN de la Universidad Surcolombiana para que, con fines académicos, muestre al país y el exterior la producción intelectual de la Universidad Surcolombiana, a través de la visibilidad de su contenido de la siguiente manera:

- Los usuarios puedan consultar el contenido de este trabajo de grado en los sitios web que administra la Universidad, en bases de datos, repositorio digital, catálogos y en otros sitios web, redes y sistemas de información nacionales e internacionales “open access” y en las redes de información con las cuales tenga convenio la Institución.

- Permita la consulta, la reproducción y préstamo a los usuarios interesados en el contenido de este trabajo, para todos los usos que tengan finalidad académica, ya sea en formato Cd-Rom o digital desde internet, intranet, etc., y en general para cualquier formato conocido o por conocer, dentro de los términos establecidos en la Ley 23 de 1982, Ley 44 de 1993, Decisión Andina 351 de 1993, Decreto 460 de 1995 y demás normas generales sobre la materia.

- Continúo conservando los correspondientes derechos sin modificación o restricción alguna; puesto que, de acuerdo con la legislación colombiana aplicable, el presente es un acuerdo jurídico que en ningún caso conlleva la enajenación del derecho de autor y sus conexos.

De conformidad con lo establecido en el artículo 30 de la Ley 23 de 1982 y el artículo 11 de la Decisión Andina 351 de 1993, “Los derechos morales sobre el trabajo son propiedad de los autores” , los cuales son irrenunciables, imprescriptibles, inembargables e inalienables.

EL AUTOR/ESTUDIANTE: **Juan David Daza Perez**

Firma: _____

EL AUTOR/ESTUDIANTE: **Nathash Abbarig Guerrero González**

Firma: Nathash Guerrero

TÍTULO COMPLETO DEL TRABAJO:

DISEÑO E IMPLEMENTACIÓN DE UN LABORATORIO PERSONAL DE TEMPERATURA PARA LA ASIGNATURA DE CONTROL

AUTOR O AUTORES:

Primero y Segundo Apellido	Primero y Segundo Nombre
GUERRERO GONZALEZ	NATHASH ABBARIG
DAZA PEREZ	JUAN DAVID

DIRECTOR Y CODIRECTOR TESIS:

Primero y Segundo Apellido	Primero y Segundo Nombre
SENDOYA LOSADA	DIEGO FERNANDO

ASESOR (ES):

Primero y Segundo Apellido	Primero y Segundo Nombre

PARA OPTAR AL TÍTULO DE: INGENIERO ELECTRONICO

FACULTAD: INGENIERIA

PROGRAMA O POSGRADO: ELECTRONICA

CIUDAD: NEIVA **AÑO DE PRESENTACIÓN:** 2024 **NÚMERO DE PÁGINAS:** 69

TIPO DE ILUSTRACIONES (Marcar con una **X**):

Diagramas_X_ Fotografías_X__ Grabaciones en discos___ Ilustraciones en general___ Grabados___ Láminas___ Litografías___ Mapas___ Música impresa___ Planos_X_ Retratos___ Sin ilustraciones___ Tablas o Cuadros_X_

SOFTWARE requerido y/o especializado para la lectura del documento: MATLAB para la ejecución de aplicativo

MATERIAL ANEXO: Anexo A: Manual de usuario, Anexo B: Código aplicación, Anexo C: Fotografías del acople de la planta de temperatura con el Arduino Uno.

PREMIO O DISTINCIÓN *(En caso de ser LAUREADAS o Meritoria)*:

PALABRAS CLAVES EN ESPAÑOL E INGLÉS:

	Español	Inglés		Español	Inglés
1.	___i2c____	_____ i2c _____	6.	____PWM_____	_____ PWM _____
2.	___kicad____	____ kicad _____	7.	___SMD_____	_____ SMD _____
3.	multivariable	multivariable	8.	___THT_____	_____ THT _____
4.	__PCB___	_____ PCB ____	9.	_____	_____
5.	____PID_____	_____ PID _____	10.	_____	_____

RESUMEN DEL CONTENIDO: (Máximo 250 palabras)

El objetivo de este proyecto es diseñar un dispositivo y su interfaz gráfica, el cual pueda ser utilizado fácilmente por el estudiante para realizar pruebas y aplicaciones de control sin la necesidad de asistir a un laboratorio presencial y utilizar los elementos presentes en ese espacio. Este trabajo se realiza con el fin de mitigar las dificultades que surgieron durante la pandemia, modernizando la forma en la que se desarrollan usualmente los laboratorios de control y brindando una nueva experiencia al estudiante.

Para cumplir con este propósito se realiza el diseño y la implementación de una planta de temperatura personal y compacta, que brindará las mismas características que las plantas robustas presentes en las instalaciones de la universidad.

ABSTRACT: (Máximo 250 palabras)

The objective of this project is to design a device and its graphical interface, which can be easily used by the student to perform tests and control applications without the need to attend an on-site laboratory and use the elements present in that space. This work is done to mitigate the difficulties that arose during the pandemic, modernizing the way in which control laboratories are usually developed and providing a new experience to the student.

To fulfill this purpose, the design and implementation of a personal and compact temperature plant is carried out, which will provide the same characteristics as the robust plants present in the university facilities.

APROBACION DE LA TESIS

Nombre Presidente Jurado: Diego Fernando Sendoya Losada

Firma:

Nombre Jurado: José de Jesús Salgado Patrón

Firma:

Nombre Jurado: Fabián Robayo.

Firma:

DISEÑO E IMPLEMENTACIÓN DE UN LABORATORIO PERSONAL DE TEMPERATURA PARA LA ASIGNATURA DE CONTROL

JUAN DAVID DAZA PÉREZ
NATHASH ABBARIG GUERRERO GONZÁLEZ

UNIVERSIDAD SURCOLOMBIANA
FACULTAD DE INGENIERÍA
PROGRAMA DE INGENIERÍA ELECTRÓNICA
NEIVA – HUILA
2024

DISEÑO E IMPLEMENTACIÓN DE UN LABORATORIO PERSONAL DE TEMPERATURA PARA LA ASIGNATURA DE CONTROL

JUAN DAVID DAZA PÉREZ Cod. 20161148107
NATHASH ABBARIG GUERRERO GONZÁLEZ Cod. 20161146145

Trabajo de grado para aplicar
al título de ingeniero electrónico

Director:
Mag. Diego Fernando Sendoya

UNIVERSIDAD SURCOLOMBIANA
FACULTAD DE INGENIERÍA
PROGRAMA DE INGENIERÍA ELECTRÓNICA
NEIVA – HUILA
2024

Notas de aceptación

_____
_____
_____
_____
_____
_____
_____

_____
Firma del director de Tesis

_____
Firma del Jurado

_____
Firma del Jurado

Neiva, 20 de mayo de 2024

Desde este momento, veo mi vida desde otra perspectiva y puedo ver claramente el camino que me ha traído hasta aquí, incluso soy consciente de las personas a las que este título les pertenece tanto como a mí. Es difícil encontrar las palabras para agradecer, sin embargo, aquí va un intento desde lo profundo de mi corazón.

A mi abuelo, Jorge González le agradeceré siempre lo hecho por mí, las tardes de pláticas sobre circuitos y electrónica que nadie más entendía y toda la sabiduría que me infundo, puedo imaginar su sonrisa, la felicidad de saber que al fin lo logré y aunque ya no esté más aquí, espero que desde el cielo este muy orgulloso. A mí madre, Claudia González, por ser siempre mi ejemplo, mi más fiel compañera en este camino lleno de desafíos, por recordarme siempre vivir todas las etapas y por hacerme mi mejor versión. A mí padre, Vismark Guerrero, por todos los conocimientos, por mostrarme todo lo que puedo hacer y a ambos, por la paciencia, por el cariño y por fundar las bases para lo que hoy soy. A mi colega y amigo, David Alejandro Murcia cuyo recuerdo quedará siempre en mi alma, gracias por enseñarme tanto, por las ocurrencias, los asados y sobre todo por mostrarme una parte de mí misma que no conocía. A los demás amigos que conocí en la carrera, Luis, Mar, Andrés, Paula, Howard, entre otros, con los que nos reuníamos en las tardes a estudiar y a hacer tareas, por las trasnochas y el café, fueron indispensables en todo este proceso, a mi compañero y pareja, Juan David, por la excepcional compañía a través de los años, por cumplir esta meta juntos y por el futuro que nos espera.

Quiero agradecer a don Juan y a doña Angelica, padres de Alejo, por recibirnos en su hogar, por vernos crecer y por darnos el cariño que solo los padres saben brindar a un hijo. Al ingeniero Jimy Aguirre, por ofrecerme sus conocimientos, su apoyo y por terminar de formar a la profesional en mí.

Agradezco a la vida por permitirme conocerlos y vivir todo esto, ha sido maravilloso, ahora llevo algo de cada uno que me pertenece y estará para siempre en mí.

Nathash

Quiero primero agradecerle a mi madre por todo el apoyo que me ha brindado a lo largo de mi vida, por estar ahí en cada momento porque, aunque no fue fácil hacerlo sola, siempre encontró la forma de asegurarse de que ni a mí ni a mis hermanos nos hiciera falta nada, por proporcionarme todo lo necesario para sacar adelante esta carrera, por darme consejo y ánimo cuando me hizo falta. Agradezco a mis hermanos por su cariño y los buenos momentos que hemos compartido, por escucharme cuando quería hablar.

Quiero agradecer a mi familia, a mis tíos, mis primos y abuelos que siempre me aconsejaron y animaron con cariño, los cuales pusieron su grano de arena en mi para terminar esta carrera. A mi pareja Nathash, por brindarme su amor y apoyo incondicional en todo este proceso, porque al ser mi compañera pudimos disfrutar de una experiencia muy agradable en esta carrera. Les agradezco a todos por la paciencia y por esa fe puesta en mí que me impulsó a esforzarme más para pasar los obstáculos que muchas veces yo mismo me ponía.

Quiero agradecer a todas las personas que aportaron a mi crecimiento académico, a todos los docentes que me dieron su sabiduría en cada tema impartido y que siempre recordare con gran estima, a cada persona que me ayudo cuando tal vez me costaba entender algo en clase, a mis compañeros que hicieron todo este proceso más ameno y cálido, recordando las noches de estudio y reuniones para hacer trabajos, apoyándonos mutuamente y dándonos ánimos cuando tal vez no alcanzábamos las notas que queríamos.

En este espacio quiero mencionar a un compañero en especial, el cual fue un pilar demasiado importante en mi proceso académico y vida personal, porque me brindo su amistad sincera convirtiéndose en una de las personas más cercanas a mí. David Alejandro Murcia Llanos fue una persona indispensable a lo largo de estos años porque, aunque ya no este con nosotros siempre será para mí una referencia de felicidad y esfuerzo. Gracias a él empecé a diseñar PCB's y esto fue la base fundamental para realizar con más facilidad este proyecto. Quiero mencionar a don Juan y a doña Angelica, padres de Alejo, quiero agradecerles por cada palabra y cada momento que pudimos compartir y seguiremos compartiendo, por abrirme las puertas de su hogar, por las noches en las que nos tocaba estudiar y ustedes con su hospitalidad nos hacían más agradable los ratos, por hacerme sentir como parte de su familia. Al ingeniero Jimmy Alexander Aguirre por su consejo, por orientarme y ayudarme en muchas ocasiones, además de brindarme la oportunidad de iniciar a desempeñarme como ingeniero. Él me brindo conocimientos esenciales que me ayudaron en la elaboración de esta tesis, además de una que otra jalada de orejas cuando fue necesario.

Finalmente, gracias a todas las personas que, aunque no mencioné explícitamente en estas líneas, comparten mi felicidad y emoción con la finalización de esta etapa de mi vida.

Juan David

AGRADECIMIENTOS

Agradecemos a todos los docentes del programa de ingeniería electrónica por contribuir en nuestro desarrollo como profesionales, principalmente al ingeniero Diego Fernando Sendoya por su guía en el proceso de la elaboración de este proyecto, por sus consejos y mentoría en cada paso, por su paciencia y por siempre estar atento a los momentos en que lo necesitamos. A Sonia por su amabilidad y por siempre brindarnos una sonrisa y la colaboración cuando se necesitó.

Agradecemos a todos nuestros compañeros y amigos que han sido un pilar importante para cumplir nuestro objetivo, haciendo más agradable y glorioso este recorrido.

TABLA DE CONTENIDO

Pág.

LISTA DE FIGURAS

Pág.

LISTA DE TABLAS

Pág.

LISTA DE ANEXOS

GLOSARIO

I2C: Es un puerto y protocolo de comunicación serial, el cual incluye dos cables de comunicación, SDA y SCL, permite conectar hasta 127 dispositivos esclavos.

KICAD: Es un paquete de software de código abierto para la automatización del diseño electrónico (EDA). Los programas gestionan la captura de esquemas y el diseño de PCB con salida Gerber e IPC-2581. El paquete funciona en Windows, Linux y macOS y está licenciado bajo GNU GPL v3.

MULTIVARIABLE: Es una característica de un sistema o proceso físico que presenta múltiples entradas y salidas que están interconectadas y pueden influenciarse mutuamente.

P: Es un dispositivo que permite controlar un sistema en lazo cerrado para generar a la salida el estado deseado, está compuesto por un elemento que proporciona una acción proporcional.

PCB: Es un circuito cuyos componentes y conductores están contenidos dentro de una estructura mecánica, cumple su función a través de trazas de cobre, terminales, disipadores de calor o conductores planos y su estructura se realiza con material laminado aislante entre capas de material conductivo (placa de circuito impreso).

PI: Es un dispositivo que permite controlar un sistema en lazo cerrado para generar a la salida el estado deseado, está compuesto por dos elementos que proporcionan una acción proporcional e integral.

PID: Es un dispositivo que permite controlar un sistema en lazo cerrado para generar a la salida el estado deseado, está compuesto por tres elementos que proporcionan una acción proporcional, integral y derivativa.

PWM: Es un tipo de señal de control que cuenta con una frecuencia fija y una amplitud variable.

SMD: Es una forma de ensamblar y soldar los componentes electrónicos directamente sobre la placa de circuito impreso (tecnología de montaje superficial).

THT: Es una forma de ensamblar y soldar los componentes electrónicos atravesando la placa de circuito impreso (orificios pasantes).

RESUMEN

El objetivo de este proyecto es diseñar un dispositivo y su interfaz gráfica, el cual pueda ser utilizado fácilmente por el estudiante para realizar pruebas y aplicaciones de control sin la necesidad de asistir a un laboratorio presencial y utilizar los elementos presentes en ese espacio. Este trabajo se realiza con el fin de mitigar las dificultades que surgieron durante la pandemia, modernizando la forma en la que se desarrollan usualmente los laboratorios de control y brindando una nueva experiencia al estudiante.

Para cumplir con este propósito se realiza el diseño y la implementación de una planta de temperatura personal y compacta, que brindará las mismas características que las plantas robustas presentes en las instalaciones de la universidad.

ABSTRACT

The objective of this project is to design a device and its graphical interface, which can be easily used by the student to perform tests and control applications without the need to attend an on-site laboratory and use the elements present in that space. This work is done to mitigate the difficulties that arose during the pandemic, modernizing the way in which control laboratories are usually developed and providing a new experience to the student.

To fulfill this purpose, the design and implementation of a personal and compact temperature plant is carried out, which will provide the same characteristics as the robust plants present in the university facilities.

1. INTRODUCCIÓN

En respuesta a los desafíos impuestos por la pandemia del año 2019 y las restricciones asociadas al distanciamiento social, surge la necesidad de encontrar soluciones innovadoras que permitan a los estudiantes de ingeniería electrónica en la Universidad Surcolombiana continuar con su formación práctica en el campo del control, sin la necesidad de asistir físicamente a un laboratorio. En este contexto, este proyecto se propone diseñar e implementar un dispositivo junto con su correspondiente interfaz gráfica, con el objetivo de proporcionar una plataforma accesible y flexible para la realización de prácticas de control en remoto.

La propuesta se centra en el desarrollo de una planta de temperatura multivariable, compacta y versátil, que emule las características y funcionalidades de los equipos presentes en los laboratorios convencionales de la Universidad Surcolombiana. Este enfoque busca no solo mitigar los obstáculos surgidos durante la pandemia, sino también modernizar y enriquecer la experiencia de aprendizaje de los estudiantes en el área de control. En este documento se presenta una visión integral del proyecto, el desarrollo de la metodología propuesta para el diseño, la interfaz gráfica de usuario y la implementación de la planta de temperatura.

La ejecución exitosa de esta solución no solo beneficia a los estudiantes al ofrecerles una herramienta útil y eficiente para la realización de prácticas de control, sino que también contribuye al avance de la educación en ingeniería al proporcionar una alternativa eficiente y adaptable a los desafíos actuales. Este proyecto representa un paso significativo hacia la transformación digital de los métodos de enseñanza en ingeniería, promoviendo la accesibilidad, la flexibilidad y la calidad en la formación de los futuros ingenieros en la Universidad Surcolombiana.

2. OBJETIVOS

2.1 OBJETIVO GENERAL

Diseñar e implementar una planta de temperatura multivariable que mediante una interfaz gráfica permite la visualización y toma de datos para facilitar al estudiante la realización práctica de los laboratorios de las asignaturas de control.

2.2 OBJETIVOS ESPECÍFICOS

- Diseñar una planta de temperatura multivariable.
- Diseñar el hardware para el acople de señales en Arduino.
- Diseñar la interfaz gráfica para interacción con el usuario.
- Diseñar e implementar el circuito impreso PCB.
- Realizar el manual de usuario.
- Realizar prueba de la planta utilizando Control Digital explicado paso a paso.

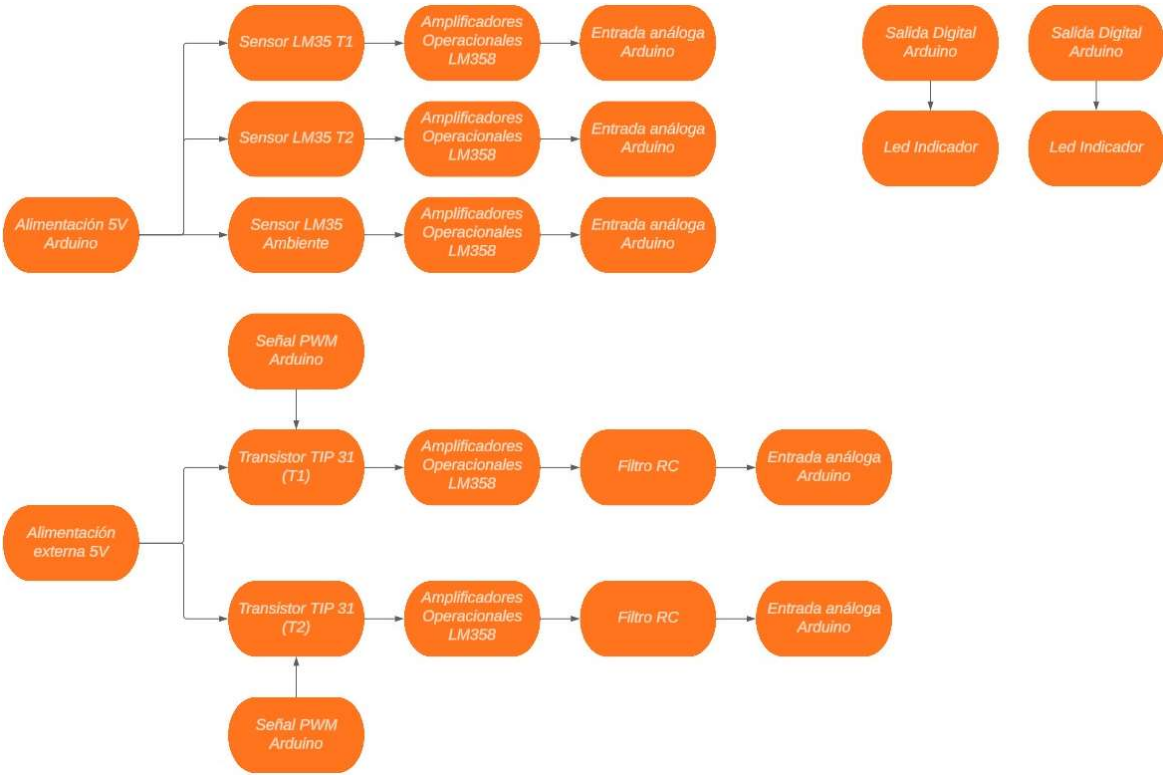
3 DISEÑO DE LA PLANTA DE TEMPERATURA

En el diseño de la planta se presenta el interés de cumplir con tres características importantes, primero la planta debe ser compacta debido a que lo que se pretende es que cada estudiante pueda usarla en cualquier espacio sin necesidad de accesorios extensos, facilitando así la interacción para realizar las pruebas de control exento de asistir a las instalaciones de la universidad, segundo y ligada a la anterior esta debe ser de bajo costo con la finalidad de ser asequible y de claro funcionamiento, proporcionando al estudiante la posibilidad de realizar por sí mismo su ensamble, tercero esta debe ser multivariable con el fin de garantizar que los factores que interactúan en la misma sean de mayor complejidad, mejorando la calidad de los controles que se pueden aplicar en las plantas presentes en los laboratorios de la universidad, para esto es necesario que se tengan dos o más variables de entrada y salida y estén interconectadas de manera en que el más mínimo cambio en una, afecte a la otra.

Con base en las características anteriores se propone una planta (ver Figura 1.) con dos calentadores y tres sensores de temperatura, la cual debe ser compatible con el hardware Arduino. Para cumplir con las proporciones de tamaño se define que los calentadores son remplazados por transistores y es que además estos permiten entrelazar su funcionamiento con el hardware a utilizar. Un transistor de potencia se puede utilizar como un generador de calor por la disipación que presenta debido a la gran corriente que maneja. En este caso se planea el uso del transistor TIP31 y el sensor de temperatura LM35 por el cumplimiento de las características necesarias en el diseño (dispositivo compacto, de bajo costo y multivariable).

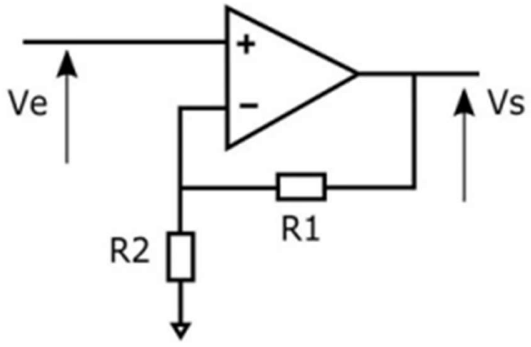
Físicamente la planta se proyecta de forma en que los dos transistores estén uno al lado del otro, para que de ese modo la variación en la temperatura de uno afecte al otro, junto a cada transistor debe haber un sensor de temperatura y adicionalmente un sensor que indique la temperatura ambiente. Teniendo en cuenta la manera en que se desea entrelazar la planta y el hardware de Arduino, se realizan los cálculos para implementar amplificadores operacionales no inversores en las salidas respectivas de los sensores de temperatura y amplificadores operacionales seguidores para las salidas de corriente de los transistores buscando adaptar el rango de los datos obtenidos según la resolución de trabajo del Arduino, que en los pines analógicos es de 10 bits, así que se puede dividir el rango de voltajes de entrada en 1024 niveles distintos, lo significa que el ADC (Analog to Digital Converter) puede convertir un voltaje de entrada analógico en un número entero entre 0 y 1023.

Figura 1. Diagrama de bloques del diseño de la planta.



Normalmente el rango de voltaje de funcionamiento es el mismo rango en el que se obtienen los datos, pero en este caso teniendo en cuenta que se va a trabajar con el amplificador operacional LM358 se define que, con el objetivo de evitar la saturación de estos, la toma de datos se realiza en el rango de 0 a 4V.

Figura 2. Diseño básico de amplificador operacional no inversor.



En este diseño (ver Figura 2) el voltaje de entrada es aplicado al pin no inversor, y el pin inversor está conectado a tierra (GND). Esta configuración se determina

porque permite la amplificación de la señal y la ecuación para determinar la salida es la siguiente:

$$Vs = Ve\left(1 + \frac{R1}{R2}\right) \quad \text{Ecuación (1)}$$

Se determina el rango de funcionamiento de los sensores de temperatura (ver Figura 5), según el datasheet del LM35 el sensor entrega 10mV por cada grado centígrado medido, es decir que, si los sensores de los transistores miden en un rango entre 20 y 100 °C respectivamente se obtiene del sensor 0,2V y 1V y la salida que se requiere debe estar en un rango entre 0V Y 4V, por ende, la adaptación del rango se realiza con los amplificadores operacionales usando una ganancia de 4, definiendo R2 de un valor de 10KΩ, se procede a hallar R1 despejando la ecuación (1).

$$R1 = R2\left(\frac{Vs}{Ve} - 1\right) \text{ Ecuación (2)}$$

$$R1 = 10K\Omega\left(\frac{4V}{1V} - 1\right) \text{ Ecuación (3)}$$

Según la ecuación (3), el valor de R1 es de 30KΩ. Seguidamente se realiza la comprobación con los rangos de temperatura definidos:

$$0{,}2V\left(1 + \frac{30K\Omega}{10K\Omega}\right) = 0{,}8V \quad \text{Ecuación (4)}$$

$$1V\left(1 + \frac{30K\Omega}{10K\Omega}\right) = 4V \quad \text{Ecuación (5)}$$

Con la temperatura en 20 °C se tiene una salida de 0.8V como se muestra en la ecuación (4) y con la temperatura máxima estimada en 100 °C se tiene una salida de 4V como se muestra en la ecuación (5).

El sensor de temperatura ambiente (ver Figura 6) no va a tener medidas por encima de los 50°C, así que se define un rango entre 20 y 50°C donde respectivamente se obtiene del sensor 0,2V y 0,5V y la salida que se requiere debe estar en un rango entre 0V Y 4V, por ende, la adaptación del rango se realiza con el amplificador operacional usando una ganancia de 8, definiendo R2 de un valor de 20KΩ y tomando la ecuación (2) se procede a hallar R1 en este caso:

$$R1 = 20K\Omega\left(\frac{4V}{0{,}5V} - 1\right) \text{ Ecuación (6)}$$

Según la ecuación (6), el valor de R1 es de 140KΩ. Seguidamente se realiza la comprobación con los rangos de temperatura definidos:

$$0{,}2V\left(1+\frac{140K\Omega}{20K\Omega}\right)=1{,}6V \quad \text{Ecuación (7)}$$

$$0{,}5V\left(1+\frac{140K\Omega}{20K\Omega}\right)=4V \quad \text{Ecuación (8)}$$

Con la temperatura en 20 °C se tiene una salida de 1.6V como se muestra en la ecuación (7) y con la temperatura máxima estimada en 50 °C se tiene una salida de 4V como se muestra en la ecuación (8).

El circuito del transistor (ver Figura 7) debe diseñarse con amplificadores operacionales seguidores, debido a que en este caso se pretende obtener la corriente que pasa a través de este mientras está en funcionamiento, con el fin de dejar abierta la posibilidad de realizar otro tipo de control; para poder registrar la corriente se coloca una resistencia de precisión de 1 ohm en el pin del emisor a tierra. La señal de PWM para activar el transistor se coloca en la base y se debe calcular la resistencia que evita que se exceda la corriente que puede entregar el Arduino, además de definir la zona de trabajo del transistor, la cual funciona en la región activa. Para hallar la resistencia se tienen presentes los datos importantes del datasheet como lo es la corriente de base máxima (la cual no debe superar 1A) y el voltaje base emisor en la región activa (1.8V). Con esos datos presentes, se define que la corriente I_b debe ser menor a 20mA (salida máxima de un pin del Arduino), para el diseño se toman 10mA.

Primero se determina la caída de voltaje en la resistencia y esta se puede calcular como la diferencia entre el voltaje de entrada a la base y el voltaje base-emisor así:

$$VR = Ventrada - VBE \quad \text{Ecuación (9)}$$

$$5V - 1.8Vmáx = 3.2V \text{ Ecuación (10)}$$

Teniendo esto, usando la ley de Ohm se procede a encontrar el valor de la resistencia:

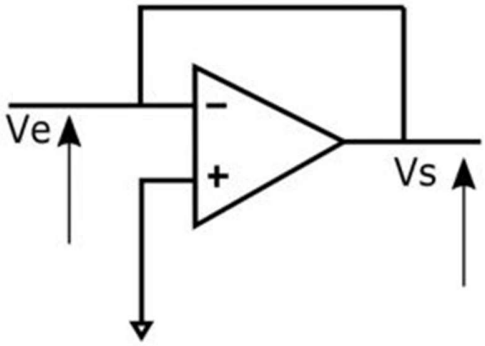
$$Rb = \frac{VR}{Ib} \text{ Ecuación (11)}$$

$$Rb = \frac{3.2Vmáx}{10mA} = 320\Omega \text{ Ecuación (12)}$$

Para garantizar que la corriente no supere los 10mA, se toma una resistencia de 470Ω.

Teniendo en cuenta que los transistores podrán llegar a temperaturas muy altas, el suministro de corriente exigido puede llegar hasta 3A por ende no puede ser proporcionado por el Arduino, ya que este supera los 500mA máximos recomendados para evitar daños en la placa de desarrollo. Por esta razón, para evitar sobrecargas se plantea una alimentación externa para la etapa de alimentación de los transistores, aun cuando todo el diseño funciona a 5V. Finalmente los sensores y amplificadores operacionales pueden trabajar con la alimentación suministrada por el Arduino y los transistores con la alimentación externa proporcionada por el USB asegurando el correcto funcionamiento de todo el sistema.

Figura 3. Diseño básico de amplificador operacional seguidor.



La salida del amplificador operacional cuenta con un circuito RC (ver Figura 7), que busca filtrar y suavizar la señal antes de ser recibida por el Arduino, para esto se define un capacitor de 100µF y se procede a encontrar la resistencia así:

Con τ = 1 segundo, usando la fórmula de la constante de tiempo:

$$\tau = R \times C \quad \text{Ecuación (13)}$$

Despejando R:

$$R = \frac{\tau}{C} = \frac{1seg}{100\mu F} = 10K\Omega \quad \text{Ecuación (14)}$$

Adicionalmente se colocan dos leds en el diseño (ver Figura 10), los cuales proporcionan la posibilidad de interactuar con la placa de otra forma, principalmente como indicadores.

Para hallar la resistencia de los leds se usa la siguiente ecuación y se obtienen los datos del datasheet del led escogido:

$$RL = \frac{Vs - Vled}{Iled} \quad \text{Ecuación (15)}$$

$$RL = \frac{5V - 1.8V}{10mA} = 320\Omega$$ Ecuación (16)

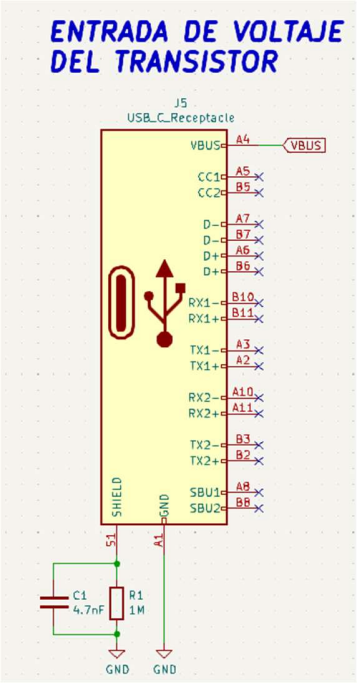
Para garantizar que la corriente no supere los 10mA, se toma una resistencia de 470Ω.

4 IMPLEMENTACIÓN DEL CIRCUITO IMPRESO (PCB)

Al momento de realizar la implementación del circuito, se decide usar un software libre de diseño como lo es KiCad. La mayoría de los símbolos de los elementos utilizados en el diseño vienen previamente cargados en la biblioteca del software. Anteriormente se realiza un análisis para elegir la forma de la placa y se define que para lograr una implementación sencilla con la placa de desarrollo de Arduino esta va a tener la misma forma y organización de sus conectores externos, además de facilitar el uso del Arduino aun con la PCB de la planta de temperatura conectada.

Inicialmente se desarrolla la organización del esquemático, el cual incluye todas las conexiones físicas del circuito mismas que se dividen en secciones según su funcionamiento.

Figura 4. Conexiones físicas del USB C de alimentación.



Como se describe en la sección anterior, la alimentación de los transistores (ver figura 4) debe realizarse de forma externa debido a la gran corriente que exigen para su funcionamiento, se determina el uso de un USB C pensando en que los adaptadores de corriente más comúnmente usados en el presente son los mismos utilizados para cargar los teléfonos inteligentes.

La etiqueta llamada VBUS se encarga de nombrar la conexión de los 5V de alimentación proporcionados por este. El pin SHIELD forma parte del recubrimiento del conector, este por defecto puede dejarse flotante, pero en este caso se utiliza

un circuito RC (ver figuras 5 y 6) para la estabilización de la señal y evitar la interferencia electromagnética.

Figura 5. Conexiones físicas de los sensores de temperatura de los transistores con el amplificador operacional no inversor.

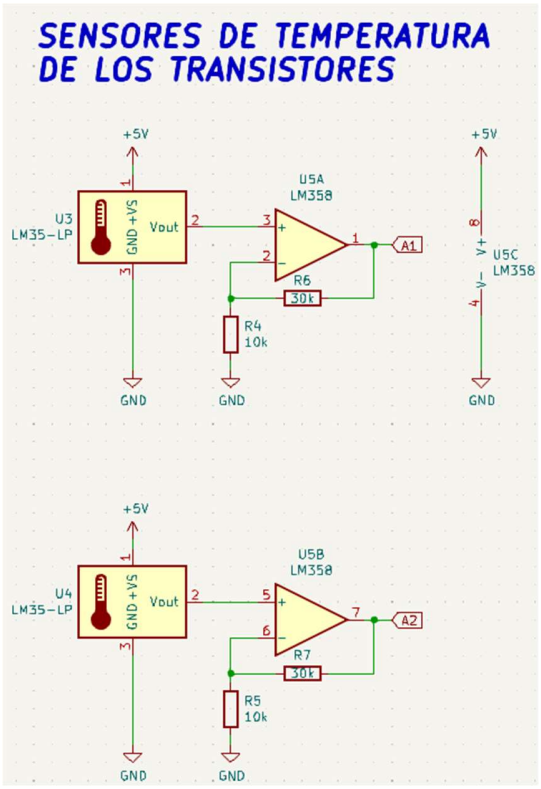
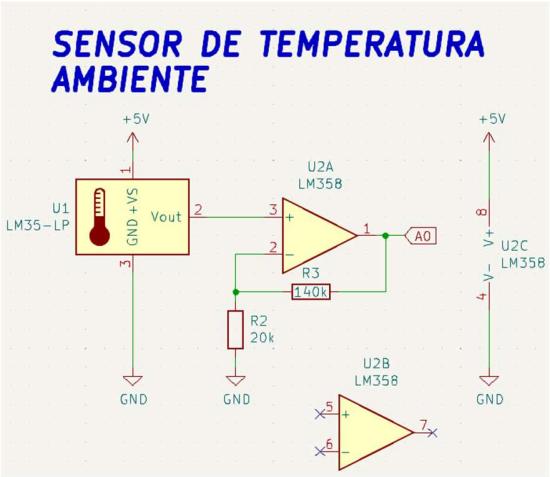


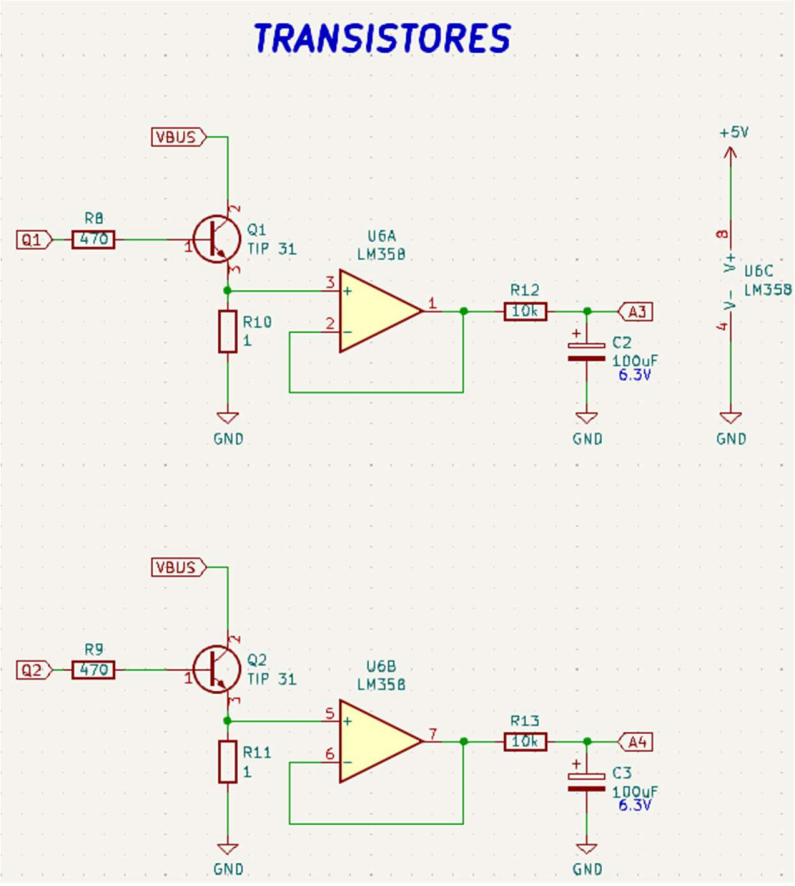
Figura 6. Conexiones físicas del sensor de temperatura ambiente con el amplificador operacional no inversor.



Se inserta el símbolo del sensor LM35 el cual se conecta en el pin 1 a la alimentación de 5V proveniente del Arduino (por esta razón la etiqueta es diferente), el pin 3 se conecta a la tierra global del circuito y el pin 2, que es la salida de la medición de temperatura a la entrada no inversora del amplificador operacional. Previamente se determina el uso del amplificador LM358 debido a su bajo costo y a que en su encapsulado vienen dos amplificadores, en el esquema se pueden diferenciar por el nombre que posee, en el caso de la figura 3 el elemento es llamado U5 y en el caso de la figura 4 el elemento es llamado U2, para la diferenciación del amplificador operacional dentro del encapsulado se utiliza adicionalmente la letra A o B, la alimentación se indica con la letra C.

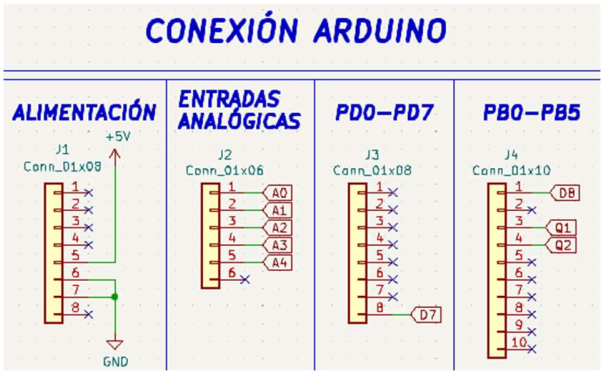
Seguidamente se realiza la conexión con las resistencias según la configuración y el valor definido en los cálculos de la sección anterior. Las salidas de los amplificadores tienen por nombre etiquetas según los pines analógicos del Arduino a donde van conectadas.

Figura 7. Conexiones físicas de los transistores con el amplificador operacional seguidor.



Se añaden los TIP31 al esquema, se alimentan desde VBUS en el colector (pin 2), la base (pin 1) se conecta a las etiquetas llamadas Q1 y Q2 las cuales son las señales PWM provenientes de los pines digitales del Arduino, la resistencia de la base la cual regula Ib, el emisor (pin 3) se conecta a la entrada no inversora del amplificador y a su vez este se conecta a una resistencia pequeña (1Ω) con la finalidad de registrar la corriente que pasa por este nodo e ingresa al amplificador operacional. A la salida del amplificador operacional se encuentra un filtro RC (ver figura 7), el cual sirve para limpiar la señal antes de que las etiquetas las conecten al Arduino.

Figura 8. Conexiones físicas de los conectores externos del Arduino mediante los conectores en la PCB diseñada.

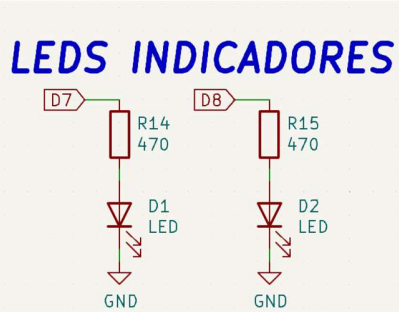


Las etiquetas previamente mencionadas (ver figura 8) se conectan a cada conector dependiendo de la organización predefinida del Arduino para que este encaje perfectamente encima. En el conector J1 se puede encontrar la alimentación del resto de la placa, la cual proviene del Arduino.

Figura 9. Conexiones físicas de los conectores externos I2C del Arduino mediante los conectores en la PCB diseñada.

Se implementan las conexiones de los conectores I2C en la PCB (ver figura 9) para cumplir con el propósito de dejar los pines del Arduino totalmente asequibles aun cuando esté conectado a la planta de temperatura.

Figura 10. Conexiones físicas de los leds indicadores de temperatura en la PCB diseñada.



La señal D7 y D8 vienen de etiquetas manejadas a través del Arduino (ver figura 10), esto con la finalidad de determinar una temperatura máxima para tener precaución, puesto que cada transistor puede llegar a una temperatura de 100 °C y se requiere informar que es sumamente peligroso tocarlo después de que el LED este encendido.

Después de completar el esquemático se procede a realizar el llenado del BOM o la lista de materiales utilizados en el diseño (ver tabla 1), como anteriormente se menciona, a cada elemento se le asigna una letra y un número, cuando se abre esta pestaña en el software se puede ingresar detalles acerca del elemento, algunos son opcionales y otros son necesarios como por ejemplo el footprint o huella, esto indica la forma que debe incluirse en la placa para poder soldar el elemento real en ella, otro ejemplo es el valor, en elementos como las resistencias es por ejemplo 10KΩ, los datos opcionales son el costo, el fabricante, el vendedor y el datasheet.

Es importante verificar el datasheet u hoja de datos de cada elemento para comprobar que la huella escogida sea acorde al elemento vendido por el fabricante que se va a utilizar.

Tabla 1: BOM del diseño realizado.

REFERENCE	VALUE	FOOTPRINT	DATASHEET	COST (usd)	Mfg PN	Vendor link	Vendor PN	Cantidad
C1	4.7nF	Capacitor_SMD:C_0805_2012Metric	https://datasheet.lcsc.c	0.02	CC0805KRX7RBBB472	https://jlcpcb.com/par	C113868	1
> C2, C3	100uF	Capacitor_Tantalum_SMD:CP_EIA-3528-15_AVX-H	https://datasheet.lcsc.c	0.12	T491B107K006AT	https://jlcpcb.com/par	C115391	2
> D1, D2	LED	LED_SMD:LED_0805_2012Metric	https://datasheet.lcsc.c	0.05	HL-PSC-2012S35FC-CZ	https://jlcpcb.com/par	C497953	2
> J1, J3	Conn_01x08	Connector_PinSocket_2.54mm:PinSocket_1x08_P2.54mm_Vertical	~					2
J2	Conn_01x06	Connector_PinSocket_2.54mm:PinSocket_1x06_P2.54mm_Vertical	~					1
J4	Conn_01x10	Connector_PinSocket_2.54mm:PinSocket_1x10_P2.54mm_Vertical	~					1
J5	USB_C_Receptacle	SamacSys_Parts:MOLEX_105450-0101	https://datasheet.lcsc.c	0.66	1054500101	https://jlcpcb.com/par	C134092	1
> J6, J7	Conn_02x03	Connector_PinSocket_2.54mm:PinSocket_2x03_P2.54mm_Vertical	~					2
> Q1, Q2	TIP 31	Package_TO_SOT_THT:TO-220-3_Vertical						2
R1	1M	Resistor_SMD:R_0805_2012Metric	https://datasheet.lcsc.c	0.01	RMC 0805 1M F N	https://jlcpcb.com/par	C325768	1
R2	20k	Resistor_SMD:R_0805_2012Metric	https://datasheet.lcsc.c	0.11	AT0805BRD0720KL	https://jlcpcb.com/par	C723531	1
R3	140k	Resistor_SMD:R_0805_2012Metric	https://datasheet.lcsc.c	0.1	AC0805FR-07140KL	https://jlcpcb.com/par	C228303	1
> R4, R5, R12, R13	10k	Resistor_SMD:R_0805_2012Metric	https://datasheet.lcsc.c	0.01	AC0805JR-0710KL	https://jlcpcb.com/par	C138260	4
> R6, R7	30k	Resistor_SMD:R_0805_2012Metric	https://datasheet.lcsc.c	0.01	AC0805FR-0730KL	https://jlcpcb.com/par	C144565	2
> R8, R9, R14, R15	470	Resistor_SMD:R_0805_2012Metric	https://datasheet.lcsc.c	0.01	AC0805JR-07470RL	https://jlcpcb.com/par	C229215	4
> R10, R11	1	Resistor_SMD:R_2512_6332Metric	https://datasheet.lcsc.c	0.1	AECR2512F1R00T9	https://jlcpcb.com/par	C328414	2
> U1, U3, U4	LM35-LP	Package_TO_SOT_THT:TO-92L_Inline						3
> U2, U5, U6	LM358	Package_SO:SOIC-8_3.9x4.9mm_P1.27mm	https://datasheet.lcsc.c	0.12	LM358DR2G	https://jlcpcb.com/par	C7950	3

El vendedor determinado para la compra y además ensamble de los elementos SMD es JLCPCB, este vendedor provee la fabricación de los circuitos impresos en China, es reconocido por su calidad y buen precio, adicionalmente también provee

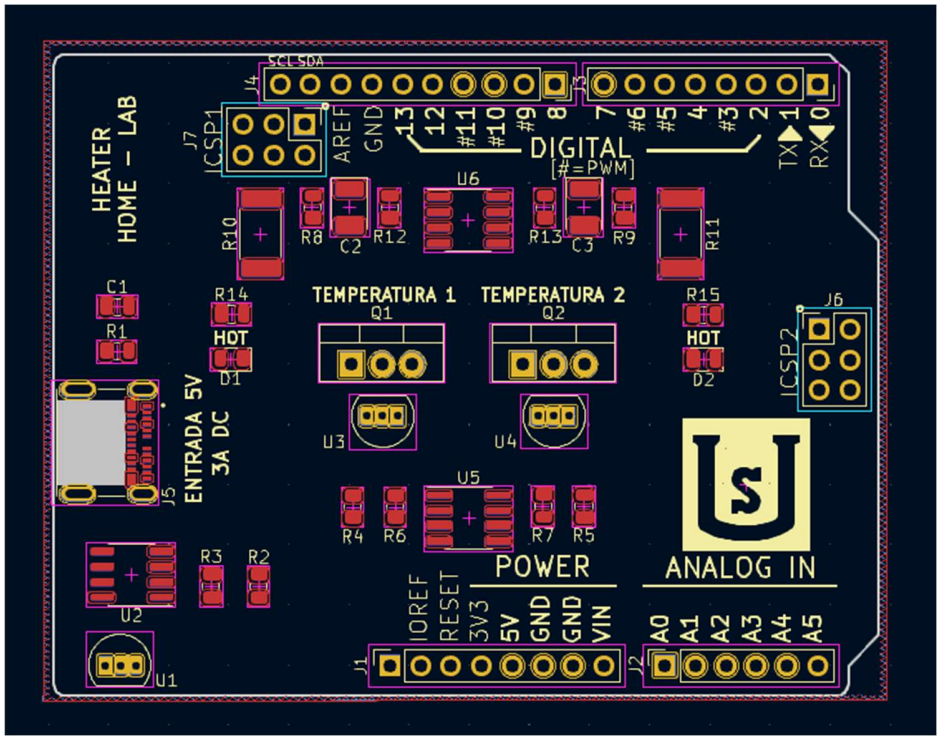
el servicio de ensamble, la mayoría de los elementos se eligen SMD por su tamaño y costo, cumpliendo con las características importantes en el desarrollo de este proyecto. Los elementos restantes, que no poseen detalles del vendedor, son aquellos que son fáciles de conseguir en el Huila, sencillos de soldar por ser THT y además los que tienen más posibilidades de generar un daño o falla, así mismo son de simple remplazo.

Seguidamente se realiza el diseño de la PCB, para esto se accede a la base de datos de Arduino en su página oficial, al ser un producto tan comercial y de uso libre se puede tener acceso al diseño de la placa de desarrollo en el mismo software utilizado para diseñar la PCB de la planta de temperatura.

Figura 11. Líneas externas de corte de la placa de desarrollo de Arduino, basada en sus archivos originales.

Con los archivos de diseño descargados, se toma la forma de la placa de desarrollo (ver figura 11) y la ubicación de sus conectores como base para realizar el diseño de la planta, teniendo esto se realiza la carga de los elementos definidos en el esquema a la placa, este software no acomoda automáticamente los elementos, esto se debe realizar de forma manual.

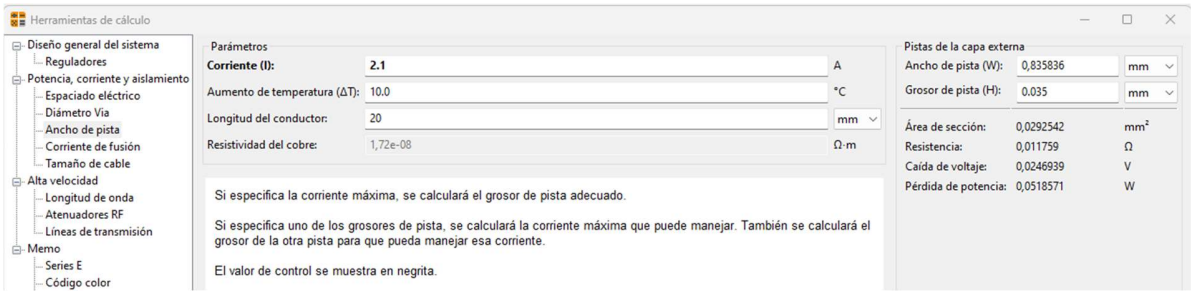
Figura 12. Líneas externas de corte de la placa diseñada junto a los elementos cargados y organizados.



Se realiza la acomodación de los elementos en el espacio de la placa (ver figura 12) y se inicia el ruteo de las líneas, se encuentran disponibles dos capas, la superior también llamada F.Cu y la inferior también llamada B.Cu, sus colores predeterminados por el software son rojo y azul respectivamente, se define el tamaño de la traza en cuanto al ancho, esto por lo general se realiza teniendo en cuenta la corriente que va a pasar a través de esta, en este caso todas las líneas pueden ser del ancho mínimo que predeterminadamente viene en 0,2mm a excepción de las líneas conectadas a la resistencia de 1Ω del transistor TIP31, ya que aproximadamente la corriente que maneja este es de 2,1A. El mismo software posee una calculadora integrada (ver figura 13) para realizar el cálculo del ancho de la pista, el cual según la información brindada es de 0.8mm.

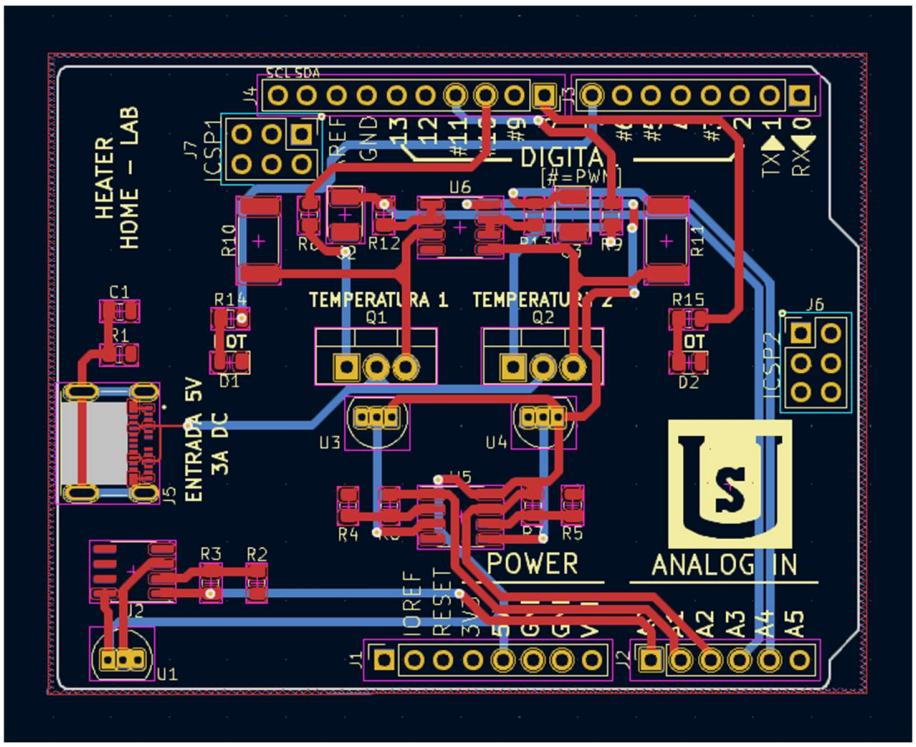
Para aprovechar el espacio presente en la PCB y para generalizar el tamaño de las trazas, todas las líneas se rutean con un ancho de 0.8mm. Adicionalmente se agregan zonas de relleno en ambas capas con la línea de tierra para favorecer la disipación del calor si se requiere.

Figura 13. Calculadora de KiCad para definir el ancho de la traza del ruteo.



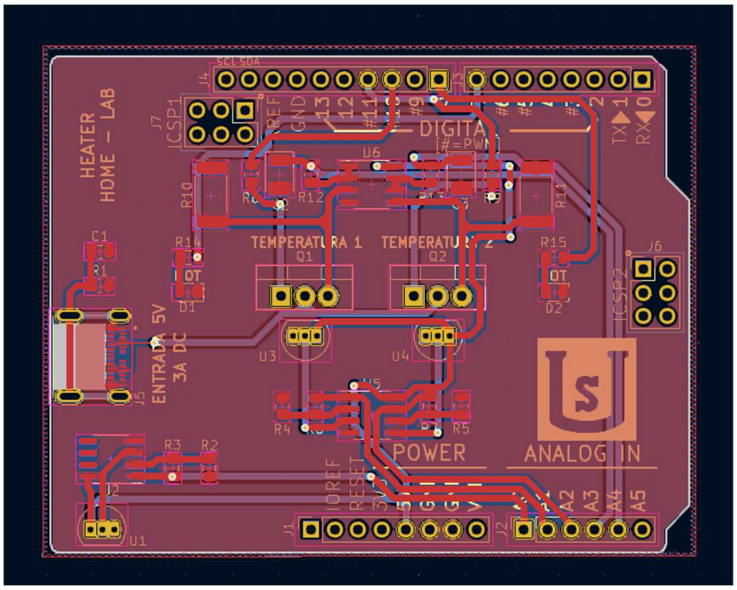
Finalmente se realiza el ruteo de la placa y se agregan nombres para diferenciar las conexiones con el Arduino y otros elementos importantes de la placa, además del nombre de la planta y el logo de la Universidad Surcolombiana para hacer la personalización de la PCB.

Figura 14. Diseño de la PCB ruteada.



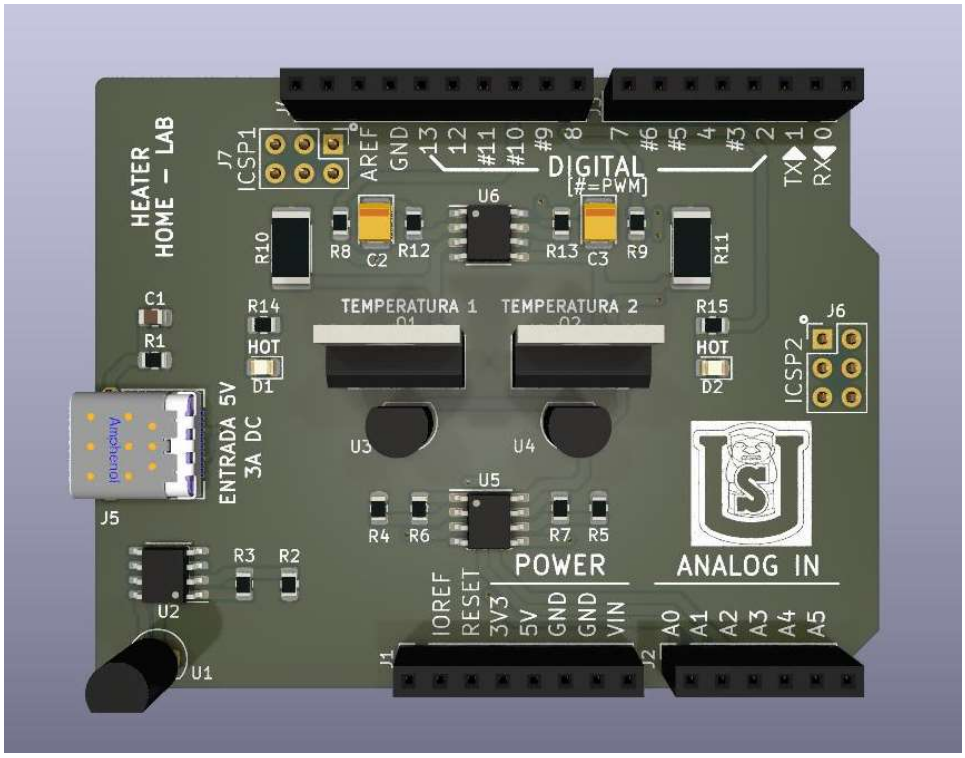
Actualizando las zonas de relleno tanto superior como inferior se tiene la PCB terminada (ver figura 14 y 15).

Figura 15. Diseño de la PCB ruteada con las zonas de tierra.



Por último, se utiliza la herramienta de vista 3D para observar cómo luce la PCB después de ser fabricada y ensamblada (ver figura 16).

Figura 16. Vista 3D de la PCB.

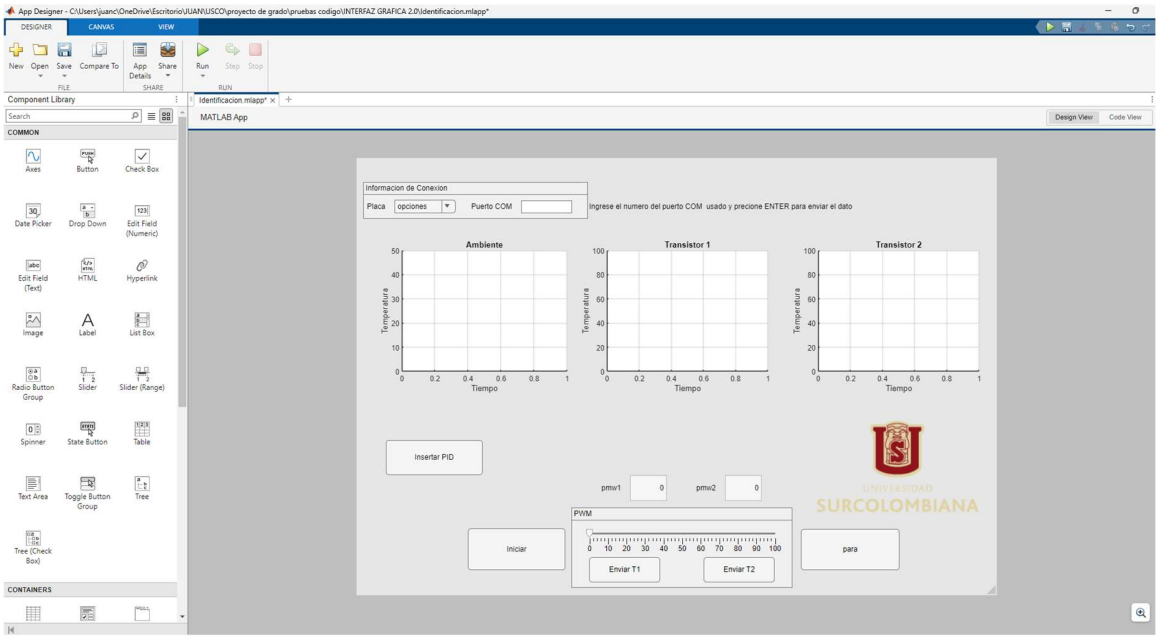


5 DESARROLLO DE LA INTERFAZ GRÁFICA

La interfaz gráfica se desarrolla en el software de MATLAB, dando uso a la licencia con la que cuenta la Universidad Surcolombiana, más específicamente se usa una herramienta llamada App Designer, la cual permite crear aplicaciones desde los componentes visuales para luego programar fácilmente su comportamiento.

En la pestaña de diseño (ver figura 17) se encuentran todas las posibles opciones para incorporar a la aplicación, teniendo en cuenta las fases que se desarrollan entre la identificación de la planta y la puesta en marcha del control definido por el estudiante, en la primera ventana se determina usar 3 gráficas para lograr observar la temperatura tomada por los 3 sensores LM35 al mismo tiempo.

Figura 17. Editor de la ventana principal de la interfaz gráfica.



Al momento de realizar la conexión de la placa de desarrollo Arduino con la aplicación se utilizaron varios códigos con el fin de implementar el reconocimiento automático del puerto y el tipo de placa de Arduino, ya que el diseño de la PCB y la interfaz está disponible para las placas de desarrollo de Arduino UNO, MEGA y LEONARDO, pero surgieron varios inconvenientes por lo que se determina que es necesario realizar esta conexión de forma manual para evitar fallas en la toma de datos. El slider funciona seleccionando el valor de PWM que se desea enviar a los transistores, además de que el valor seleccionado es mostrado en un pequeño cuadro de texto según el transistor para observar y tener presente su valor.

Esta sección bajo las gráficas en tiempo real cuenta con 5 botones, 2 se usan para enviar los datos PWM a los transistores, (Iniciar) corre la simulación y empieza a leer los datos de sensores además de mostrarlos en la gráfica, (Parar) detiene la simulación, termina la conexión con el Arduino y cierra la ventana, por último, (Insertar PID) abre la ventana diseñada para ingresar el PID cuando sea calculado. Cuando se han creado todos los elementos de la aplicación, App Designer genera automáticamente las funciones de cada elemento que se usa en el apartado de Code View, es aquí donde se programan todas las acciones que se desean realizar en la aplicación.

Figura 18. Ventana de programación de acciones de los elementos insertados.

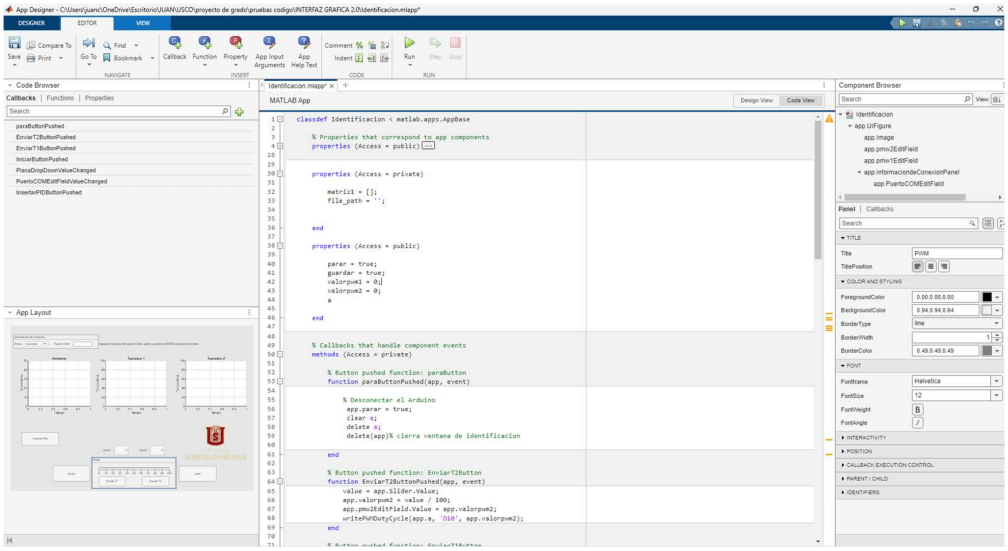
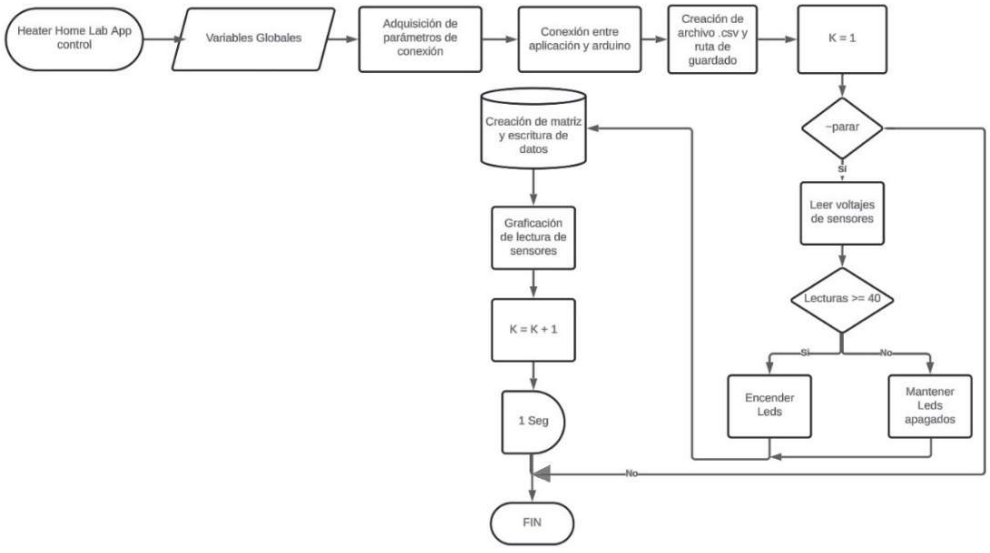


Figura 19. Diagrama de flujo botón iniciar.

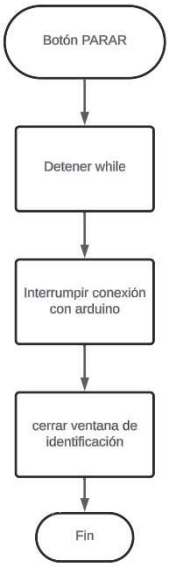


En esta primera sección de código se muestra la función principal del programa, el botón iniciar habilita la funcionalidad global de este y las demás secciones. En este se crean las variables globales de la aplicación que se usan en diferentes funciones, se guardan las variables que indican las características de la conexión entre el Arduino y el programa, una vez se establece la conexión con el Arduino, la aplicación solicita la asignación de la ruta para guardar el archivo generado .CSV con las lecturas tomadas.

Seguidamente se inicia un bucle que funciona mientras la variable parar sea falso, como primer paso en el bucle se guardan los valores de PWM usados en ambos transistores para imprimirlos en el archivo en cada instante de tiempo, luego se generan las lecturas de cada sensor, aquí se tomó un cálculo en cada sensor para establecer la temperatura, se tomaron varias muestras y se compararon con las de una termocupla calibrada con el fin de calibrar los sensores correctamente. El condicional IF controla cada uno de los leds, los cuales indican cuando los sensores captan una temperatura mayor a 40 °C y así se le advierte al estudiante que debe tener precaución al manipular el dispositivo.

A continuación, se les da formato a los datos y se crea la matriz que se guarda en el archivo final para ser manipulada con mayor facilidad, finalmente el contador de iteraciones aumenta y se hace una pausa de 1 segundo para dar tiempo al sistema de generar nuevas lecturas.

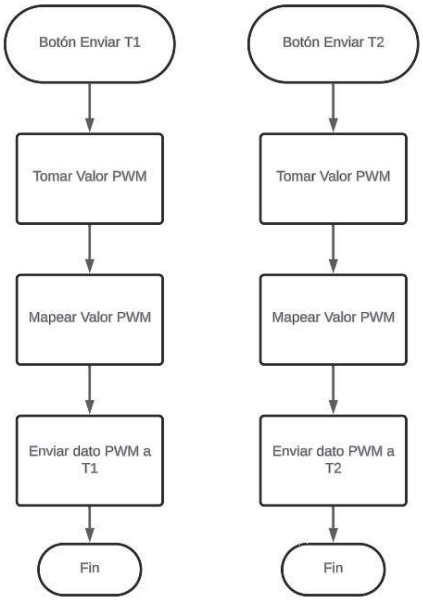
Figura 20. Diagrama de flujo botón parar.



Esta función (ver figura 20) se encarga de cambiar el estado de la variable bool parar a verdadero para luego utilizarla en el diagrama de la figura 19, adicionalmente también se encarga de limpiar la variable donde se almacenan los datos de

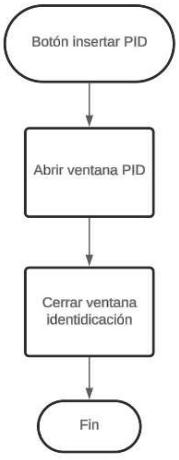
conexión con el Arduino y cortar la conexión con el mismo para finalmente cerrar la ventana de la aplicación.

Figura 21. Diagramas de flujo botones T1 y T2.



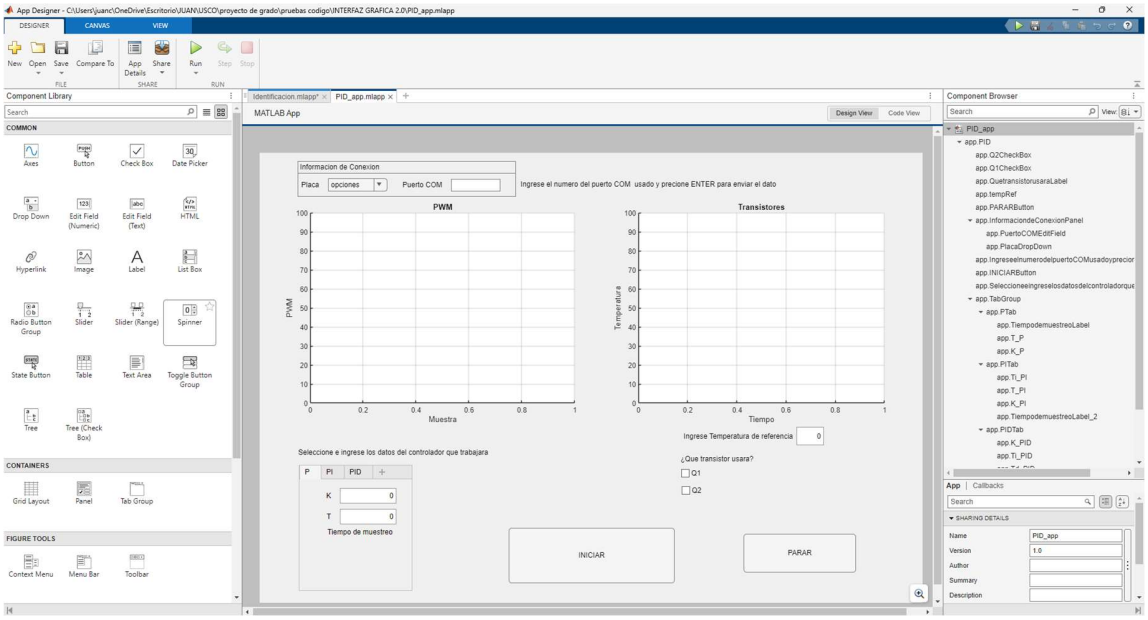
Estas dos funciones (ver figura 21) están construidas de la misma forma, se encargan de tomar los valores de PWM seleccionados con el slider y enviarlos a los pines digitales del Arduino para aplicarlos a los transistores, aquí se realiza un mapeo donde se transforma el valor tomado y lo coloca en un rango de 0 a 1, en el recuadro de texto se muestra el valor real antes de ser transformado y enviado a los pines correspondientes.

Figura 22. Diagrama de flujo botón insertar PID.



En esta última función (ver figura 22) de la ventana principal, se abre la pestaña diseñada para el controlador PID y cierra la pestaña actual para no generar conflictos con la conexión del Arduino.

Figura 23. Editor de la ventana secundaria de la interfaz gráfica.

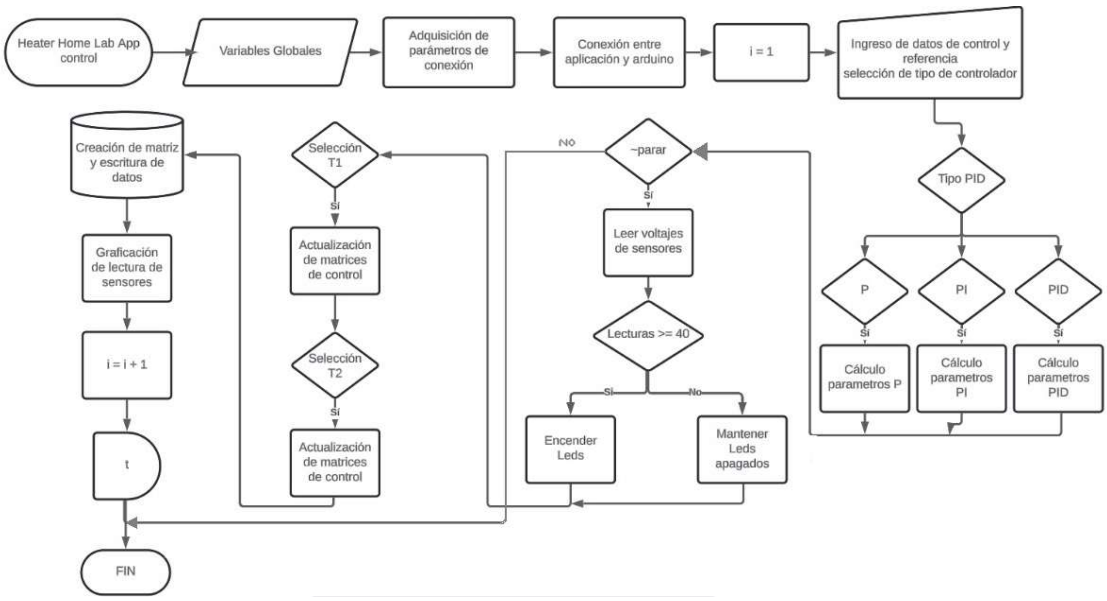


En esta segunda ventana (ver figura 23) se configuran las herramientas necesarias para que el estudiante pueda poner a prueba el diseño de su control aplicado a la planta Heater Home Lab USCO.

Nuevamente se parte del diseño y la distribución de los elementos gráficos de la aplicación, en este se determina el uso de dos gráficos donde se muestra el comportamiento del controlador en la temperatura y el PWM que se genera al ingresar los datos calculados en el control P, PI o PID según lo requiera el estudiante.

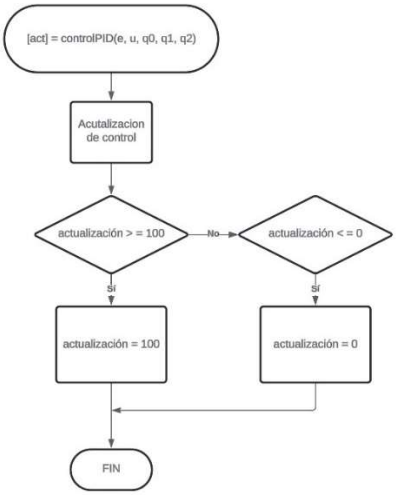
En esta primera sección de código se muestra la función principal del programa (ver figura 24), el botón iniciar habilita la funcionalidad global de este y las demás secciones siguientes. En este se establecen las variables globales, se crean dos vectores vacíos para cada transistor los cuales se usan para aplicar el control. Se crea de nuevo la matriz para guardar los datos en el archivo .CSV y una cadena de texto vacía para el nombre del archivo, adicionalmente también se establecen y se guardan los parámetros de conexión al Arduino y se define la ruta que tiene el archivo para su guardado. Para definir el tipo de control a utilizar, se crea un Tab Bar que, al ser seleccionado, guarda el dato en una variable y desarrolla un case para realizar los cálculos necesarios según el controlador determinado.

Figura 24. Diagrama de flujo botón inicio.



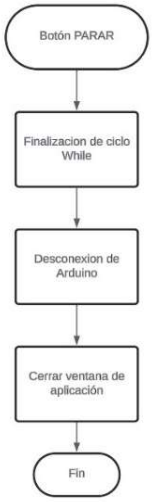
Seguidamente el ciclo while toma las lecturas de los sensores de cada transistor, además de encender el respectivo led si la temperatura del transistor es mayor o igual a 40 °C. Dependiendo de la necesidad del estudiante, se puede usar el transistor uno o dos, lo que determina que IF activarse, ambos se encargan de actualizar los vectores creados anteriormente, el error en el instante actual toma el tercer valor del vector y desplaza los valores anteriores una posición a la izquierda, de igual forma se actualizan las señales de salida del controlador, así los vectores para el cálculo del PWM se actualizan con cada iteración. Seguidamente se mapea el PWM para validar el rango a enviar al pin del Arduino.

Figura 25. Diagrama de flujo función controlador.



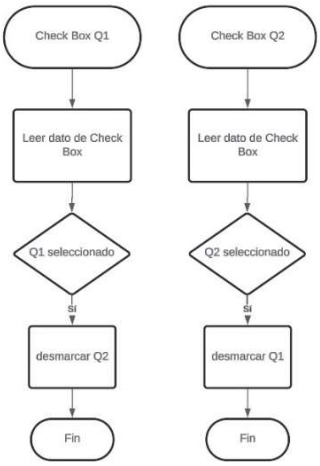
En esta sección se genera nuevamente la matriz para guardar los datos relevantes de la simulación (ver figura 25), realiza las gráficas de las señales, el contador de muestras se actualiza y se realiza una pausa de tiempo “t” que equivale al tiempo de muestreo ingresado por el estudiante en la interfaz gráfica. Finalmente, para salir del ciclo while se crea la función que usa como parámetros de entrada los valores de control y los vectores en diferentes instantes de tiempo, además de asegurarse con el método anti-windup que la acción integral del sistema no siga creciendo, definiendo el valor del PWM en 0 o 100 dependiendo del caso.

Figura 26. Diagrama de flujo botón parar.



Esta función (ver figura 26) realiza las mismas acciones que en la ventana principal, cambiando el valor de la variable a true, lo que hace que el ciclo while deje de correr, además de limpiar la conexión con el Arduino y cerrar la ventana de la aplicación.

Figura 27. Diagramas de flujo check box Q1 y Q2.

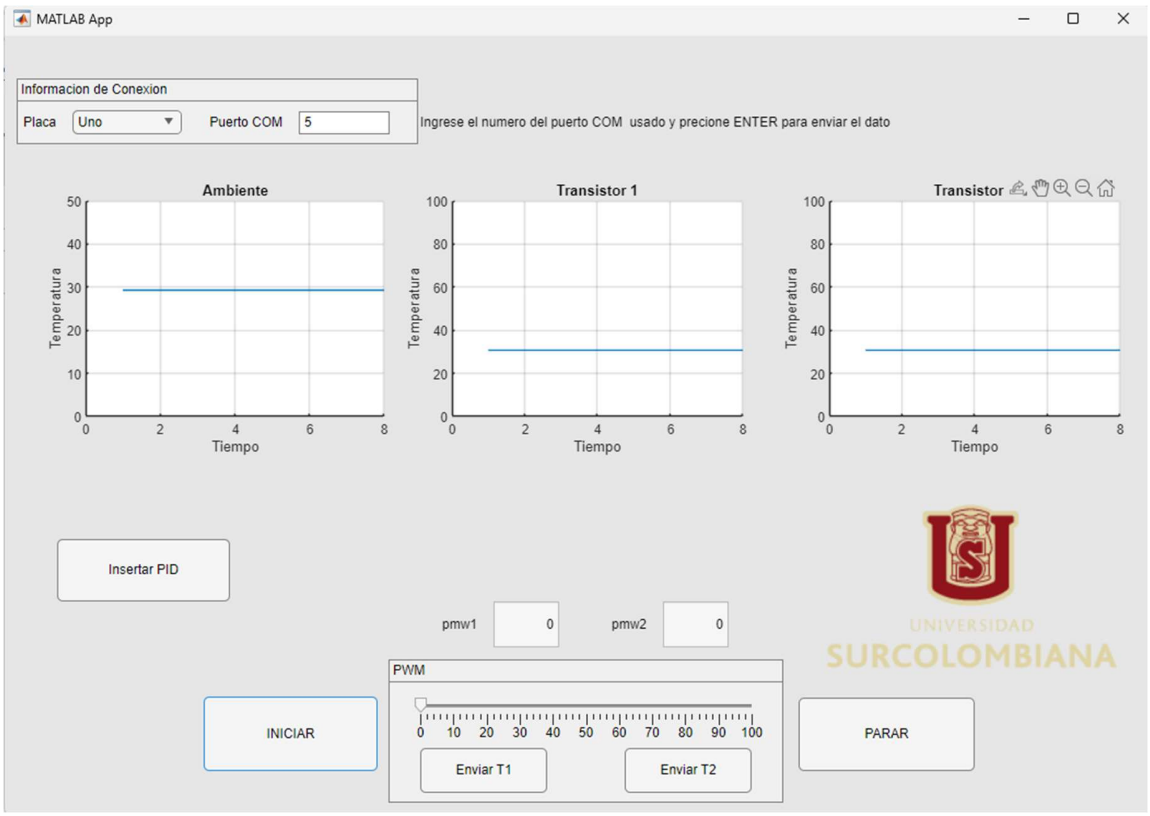


Por último, estas dos funciones (ver figura 27) se encargan de mantener el transistor actualizado al momento de presionar cualquier opción, cambia a falso el estado del otro lo que asegura que se esté trabajando con un solo transistor para evitar errores en el programa.

6 ADQUISICIÓN DE DATOS A TRAVÉS DE LA PLANTA

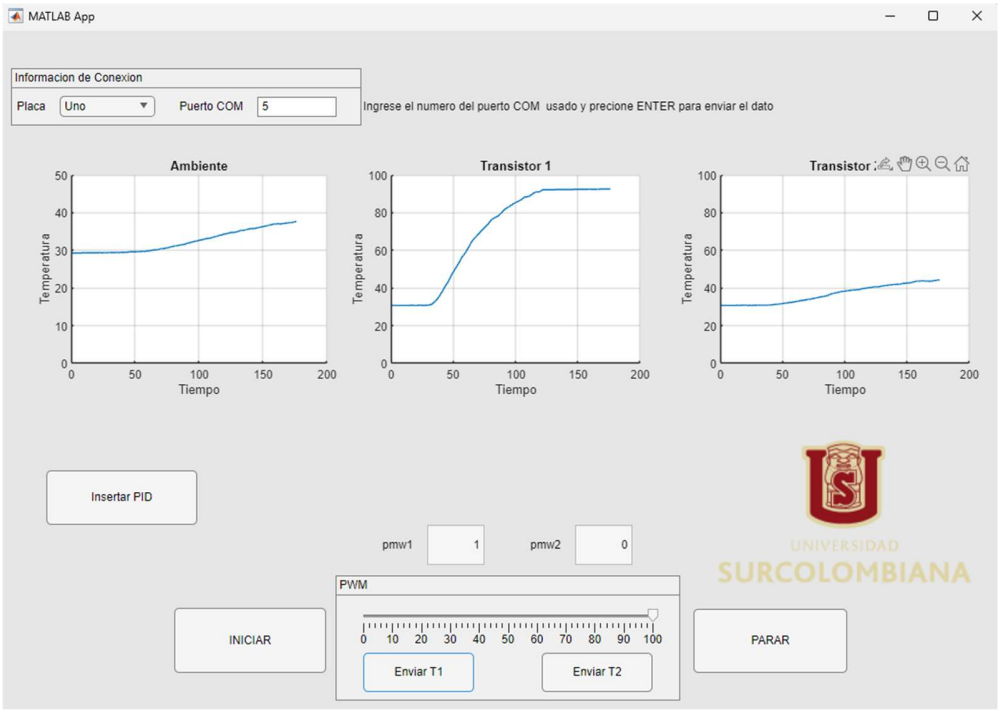
Para iniciar la interacción con la planta de temperatura, se realiza la identificación en la ventana principal de la interfaz gráfica (ver figura 28), aquí se coloca la información necesaria para realizar la conexión con el Arduino y la aplicación, para esto se selecciona el tipo de Arduino usado y el puerto COM que el computador selecciona al conectar el dispositivo, después de esto se inicia a correr el programa para empezar a guardar las lecturas.

Figura 28. Ventana principal del funcionamiento de la interfaz gráfica.



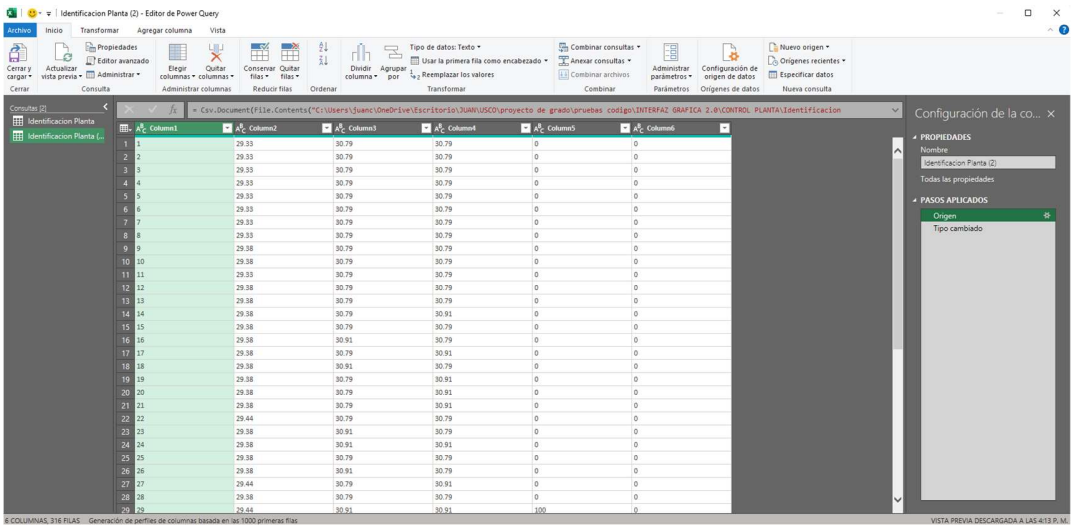
Se define la ruta donde se guarda el archivo .CSV que genera el programa y con el slide se selecciona el PWM que se usa para el proceso de identificación, en este caso se usara 100% y se presiona enviar en el transistor con el que se desee trabajar.

Figura 29. Selección de PWM enviado al transistor 1.



En este caso se trabaja con el primer transistor, se puede notar que en el cuadro de texto marca un PWM de 1(ver figura 29), la escala que admite la función de Matlab esta entre 0 y 1, el valor seleccionado es mapeado para que se ajuste a esta escala y se espera un tiempo hasta que se recolecten datos suficientes para poder adquirir la información de funcionamiento de la planta.

Figura 30. Archivo de Excel con los datos recolectados.



Al revisar los datos generados por el programa (ver figura 30) es recomendable realizar el ajuste necesario para que el formato sea el más cómodo para trabajar, con ese fin se crea un código en Matlab que organiza la tabla y recorta los datos menores al valor de PWM indicado, así se genera una nueva tabla con los datos que se usaran para la identificación y se crea una gráfica donde se puede observar el comportamiento.

Figura 31. Ventana de organización para las tablas de datos.

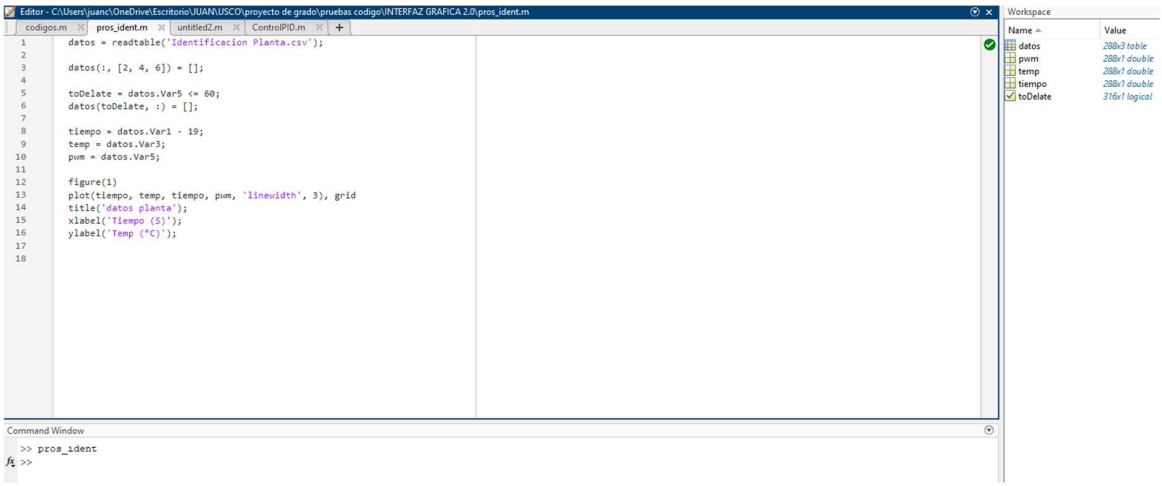
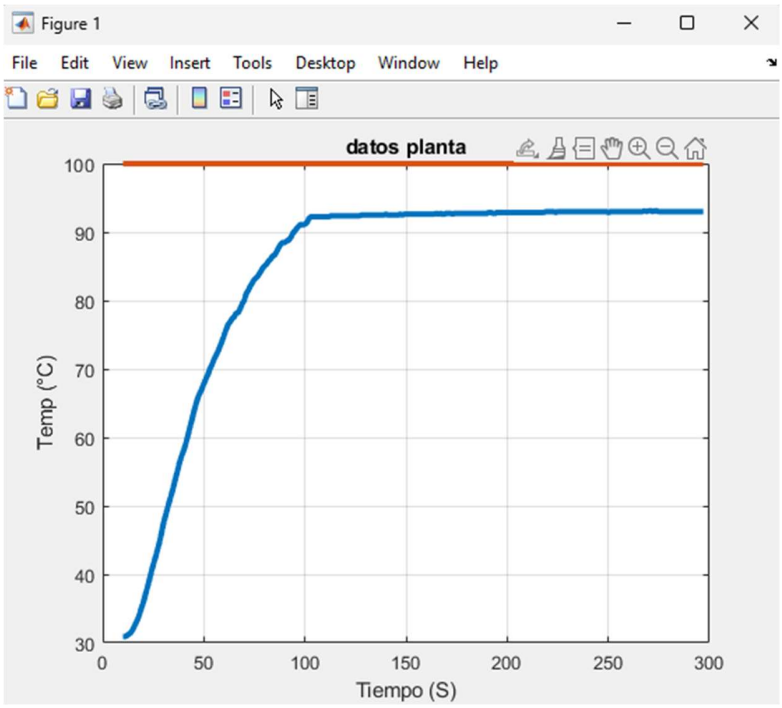
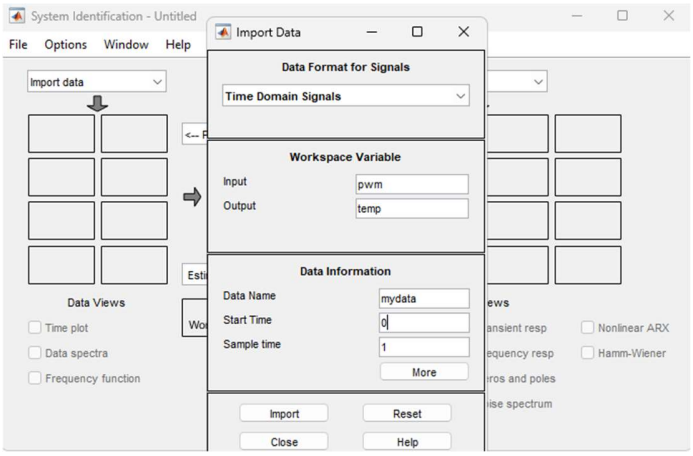


Figura 32. Gráfica generada con los datos tomados a través de la planta.



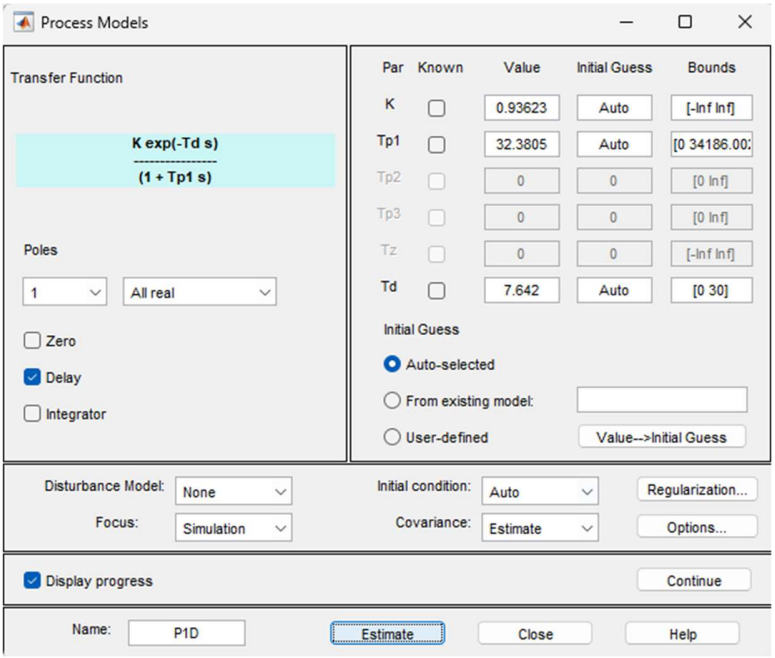
Habiendo hallado esto se tienen los recursos suficientes para realizar el proceso de identificación y desarrollo del control usando la herramienta de Matlab System Identification.

Figura 33. Pestaña de System Identification en Matlab.



Se procede con la configuración de la herramienta importando los datos obtenidos en la simulación(ver figura 34).

Figura 34. Pestaña de System Identification para importar los datos.



Se revisa que la configuración de la herramienta sea adecuada al sistema que se posee de primer orden con retraso para así poder estimar la función de transferencia. Al cerrar la herramienta se ha obtenido la función de transferencia de la planta trabajada y se puede realizar el diseño del PID.

Figura 35. Función de transferencia adquirida a través de System Identification.

```
Process model with transfer function:
                Kp
  G(s) = ---------- * exp(-Td*s)
          1+Tp1*s

         Kp = 0.93623
        Tp1 = 32.381
         Td = 7.642
```

Hay bastantes opciones para este paso, es posible apoyarse en la herramienta PID tunner para seleccionar un controlador a medida, con las especificaciones deseadas o realizar todas las operaciones matemáticas para obtener el control PID de la planta. En este caso se realizan los cálculos matemáticos según la documentación para realizar un controlador PID Ziegler-Nichols que presenta las ecuaciones para el diseño donde:

$$Kp = 1.2 * \frac{\tau}{KL}$$ Ecuación (17)

$$\tau_i = 2 * L$$ Ecuación (18)

$$\tau_d = 0.5 * L$$ Ecuación (19)

$$L = Theta + \frac{ts}{2}$$ Ecuación (20)

Reemplazando con los datos obtenidos se tiene que:

$$Kp = 1.2 * \frac{32.381}{0.9362*8.142} = 5.097$$ Ecuación (21)

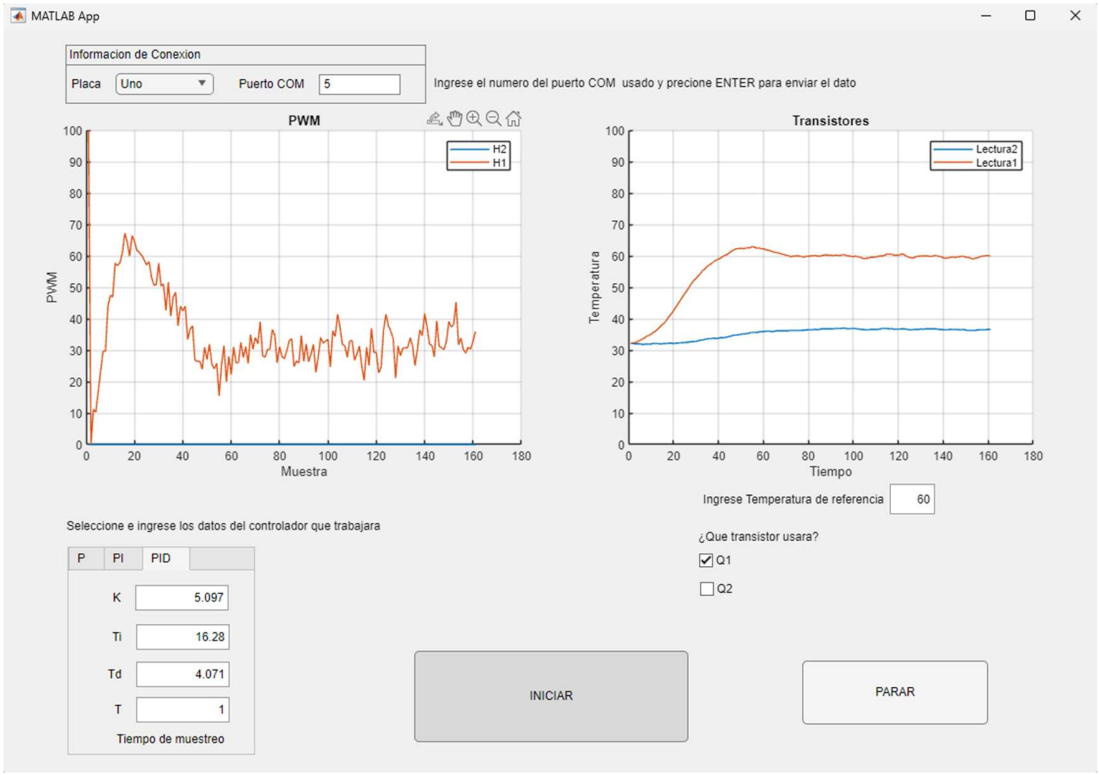
$$L = 7.642 + \frac{1}{2} = 8.142$$ Ecuación (22)

$$\tau_i = 2 * 8.142 = 16.284$$ Ecuación (23)

$$\tau_d = 0.5 * 8.142 = 4.071$$ Ecuación (24)

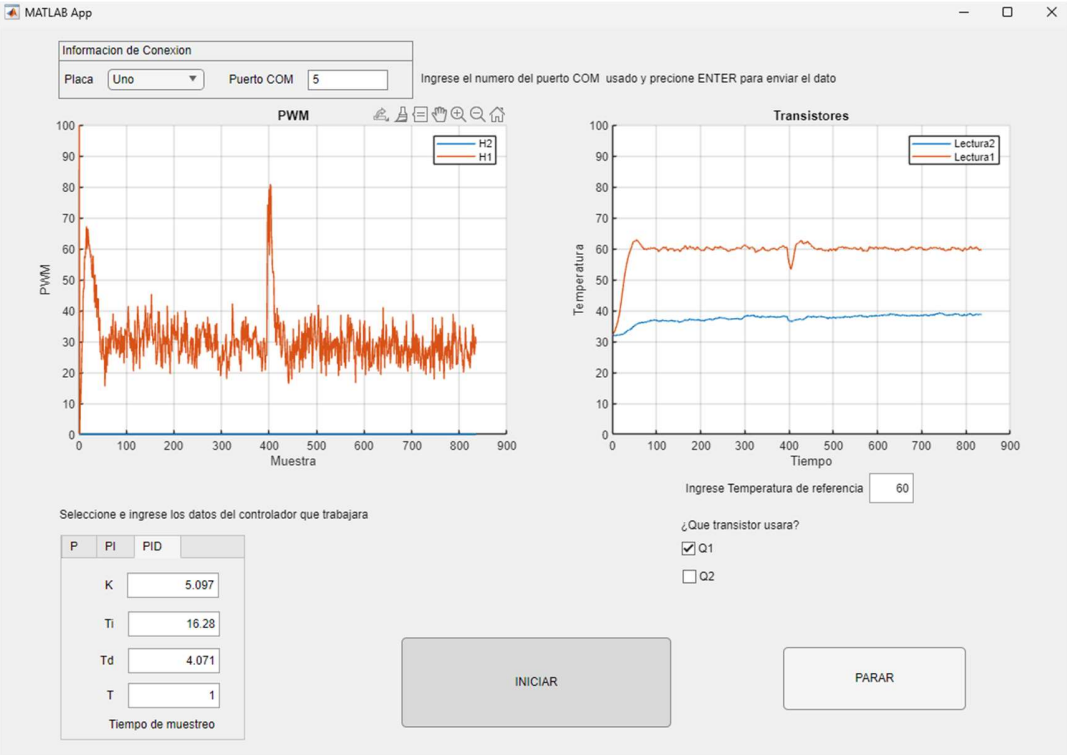
Con estos datos se puede realizar la prueba de funcionamiento del controlador para analizar su marcha y corregir el comportamiento de ser necesario.

Figura 36. Ventana secundaría de la interfaz gráfica con los datos del controlador PID.



Se ingresan los datos obtenidos (ver figura 36) y se lanza la simulación, aquí se puede observar una buena respuesta del control, un pequeño sobre impulso al llegar a la temperatura, pero se estabiliza rápidamente, además se aplica una perturbación en el sistema cerca al segundo 400 de la simulación, enfriando un poco el transistor dando como resultado la corrección del PWM para compensar la pérdida de temperatura.

Figura 37. Gráfica de la respuesta de la planta al controlador aplicado.



Se puede concluir que el controlador tiene una respuesta exitosa como practica y cumple con los parámetros deseados (ver figura 37).

7 CONCLUSIONES

El proyecto de grado expuesto en este documento muestra el proceso para el desarrollo del diseño y la implementación de una planta de temperatura y su interfaz gráfica. Las conclusiones obtenidas del presente trabajo se indican a continuación.

- Por las razones mencionadas anteriormente usando como base las características del transistor TIP31, se logra el diseño de una planta de temperatura multivariable ajustando los rangos de temperatura a las condiciones típicas de funcionamiento.
- Debido al uso de las librerías disponibles de acceso libre de Arduino, se obtiene el diseño del hardware tipo plug and play que permite la compatibilidad con los modelos de Arduino Uno, Arduino Mega y Arduino Leonardo, además de favorecer el uso del Arduino aún con la planta de temperatura acoplada.
- La evidencia presentada anteriormente demuestra que se consigue el diseño de una interfaz gráfica intuitiva y de cómodo manejo para el usuario, esto debido al uso de la herramienta App Designer de Matlab.
- Se logra diseñar e implementar el circuito impreso de la planta de temperatura, todas las pruebas, el desarrollo de la interfaz gráfica y la calibración de la planta se realizó en la PCB fabricada.
- Se ejecutó la fabricación de dos versiones, la primera versión se usó para verificar que el diseño funcionaba correctamente, con esta se logró corregir un error de conexión entre la planta de temperatura y el Arduino, seguidamente se fabricó la segunda y última versión, cuyo trabajo fue exitoso.
- Con el funcionamiento correcto del proyecto se logra crear la guía de usuario que introduce al estudiante en el funcionamiento y manejo de la planta de temperatura y la interfaz gráfica para adelantar pruebas aplicando el control.
- Este proyecto se enfoca en el control exitoso de la planta de temperatura a través del manejo del PWM, sin embargo, la placa se diseñó teniendo en cuenta que puede surgir la necesidad de aplicar otros métodos por lo que se incluyó una etapa de medición de corriente.
- Finalmente se implementa un control digital usando el método Ziegler–Nichols enseñando el proceso y las herramientas, demostrando el correcto funcionamiento de todo el proyecto.

8 RECOMENDACIONES

- Verifique que todas las partes de la planta están funcionales el mismo día que esta se le es entregada, así puede rectificar que está en buen estado.
- Trate con cuidado las partes THT de la placa (sensores y transistores) no doble ni golpee estas partes ya que puede generar un desprendimiento de la soldadura o dañar alguna vía de la PCB.
- Cualquier modificación o reemplazo de componentes debe ser consultado y autorizado por el tutor o profesor de la materia.
- Cualquier daño que sufra la planta de temperatura después de ser entregada al estudiante debe ser pagada por el estudiante, ya sea reemplazo de alguna parte por mal uso o la fabricación de una nueva placa.
- Recuerde que la PCB es solo para uso académico, no se le permite el uso de esta planta para otros fines.
- Asegure que el Arduino que se va a usar en las practicas sea reconocido por su computadora antes de correr el programa de identificación de la planta.
- Para una mejor respuesta de la planta, puede usar algún tipo de pasta térmica. Esto puede mejorar la transferencia de calor, ayudando a la precisión de lectura de los sensores.
- Utilice un cargador USB tipo C que pueda suministrar mínimo 2A.
- Una vez se realice el proceso de identificación de la planta, usar siempre el mismo cargador para trabajar con la planta.
- Cuando la planta de temperatura esté funcionando asegúrese de no tocar los transistores cuando el led de seguridad este encendido. El transistor puede llegar a temperaturas muy altas por lo que podría causar lesiones de no ser manejado con cuidado.

9 TRABAJOS FUTUROS

La interfaz gráfica tiene algunos detalles que se pueden llegar a mejorar, al usar la herramienta App Designer en Matlab se genera una facilidad en el diseño de la aplicación, pero se pueden generar retrasos en la ejecución del programa porque este debe cargar todas las características necesarias desde Matlab. Se puede realizar una interfaz gráfica usando un lenguaje de programación como Python para acelerar la respuesta de la aplicación y además para disminuir la carga en el computador.

Como mejora en la pestaña de control (ver Figura 23.) agregar una función que permita aplicar control a ambos transistores además de una función donde se pueda agregar un PWM constante a un transistor mientras se le aplica control al otro.

BIBLIOGRAFIA

ABOUT KICAD [Anónimo]. KiCad EDA - Schematic Capture & PCB Design Software [página web]. [Consultado el 19, julio, 2023]. Disponible en Internet: <https://www.kicad.org/about/kicad/>.

A COMPILAR. Comunicación serial arduino matlab en app designer| tutorial español [video]. YouTube. (17, octubre, 2020). [Consultado el 28, julio, 2024]. 22:10 min. Disponible en Internet: <https://www.youtube.com/watch?v=BK9_7io_cGE>.

AMPLIFICADOR OPERACIONAL Eléctronica [Anónimo]. Blog Arduino, LabVIEW y Electrónica [página web]. [Consultado el 19, julio, 2023]. Disponible en Internet: <https://electronicamade.com/amplificador-operacional/>.

[Consultado el 19, mayo, 2024]. Disponible en Internet: <https://hetpro-store.com/TUTORIALES/i2c/#google_vignette>.

CONTROLADOR PID - Control AutomÃ¡tico - Picuino [Anónimo]. PÃ¡gina principal - Picuino [página web]. [Consultado el 19, julio, 2023]. Disponible en Internet: <https://www.picuino.com/es/control-pid.html>.

¿EN QUÉ consiste el montaje SMD de circuitos electrónicos? - AMMi Technologies [Anónimo]. AMMi Technologies [página web]. [Consultado el 19, mayo, 2024]. Disponible en Internet: <https://ammitechnologies.com/montaje-smd-que-es/>.

GOODWIN, Graham C., et al. Emulation-Based virtual laboratories: a low-cost alternative to physical experiments in control engineering education. En: IEEE Transactions on Education [en línea]. Febrero, 2011. vol. 54, no. 1 [consultado el 19, julio, 2023], p. 48-55. Disponible en Internet: <https://doi.org/10.1109/te.2010.2043434>. ISSN 1557-9638.

HARB MECATRÓNICA. Constantes Kp, Ki, Kd y Función de transferencia de controlador PID con Ident MatLab y SimuLink [video]. YouTube. (7, septiembre, 2019). [Consultado el 19, julio, 2023]. 21:31 min. Disponible en Internet: <https://www.youtube.com/watch?v=VPC5LNaps1g>.

KALÚZ, Martin, et al. ArPi lab: a low-cost remote laboratory for control education. En: IFAC Proceedings Volumes [en línea]. 2014. vol. 47, no. 3 [consultado el 19, julio, 2023], p. 9057-9062. Disponible en Internet: <https://doi.org/10.3182/20140824-6-za-1003.00963>. ISSN 1474-6670.

PCB PROTOTYPE & PCB fabrication manufacturer - JLCPCB [Anónimo]. PCB Prototype & PCB Fabrication Manufacturer - JLCPCB [página web]. [Consultado el 19, mayo, 2023]. Disponible en Internet: <https://jlcpcb.com/>.

PWM: ¿Qué es? ¿Cómo puedo utilizarlo? DigiKey [Kohlhase, Kaleb.] [página web]. [Consultado el 19, mayo, 2024] Disponible en Internet: <https://www.digikey.com/es/blog/pulse-width-modulation>.
¿QUÉ ES una PCB o placa de circuito impreso? | altium [Anónimo]. Altium [página web]. [Consultado el 19, mayo, 2024]. Disponible en Internet: <https://resources.altium.com/es/p/what-is-a-pcb>.

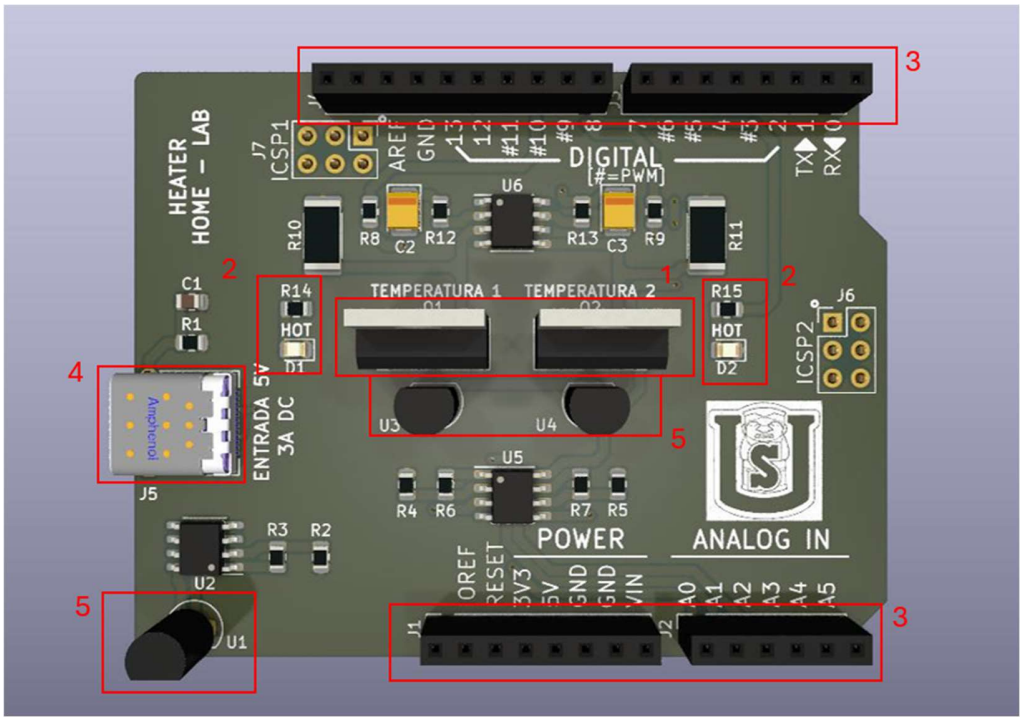
TODO sobre Ziegler Nichols - Sintonia de Control PID [Anónimo]. Control Automático Educación [página web]. [Consultado el 19, julio, 2023]. Disponible en Internet: <https://controlautomaticoeducacion.com/control-realimentado/ziegler-nichols-sintonia-de-control-pid/>.

ANEXO

Anexo A: Manual de usuario.

Inicialmente se recomienda dar un vistazo a la placa de la planta de temperatura (ver figura 38) para reconocer como interactúan los compuestos que facilitan su funcionamiento:

Figura 38. Vista preliminar de la planta de temperatura.



1. Se encuentran los transistores TIP 31 utilizados como fuente de calor y medio principal para poder interactuar con la planta y el control definido.
2. Aquí podrá observar dos leds de color rojo que pretenden indicar que los transistores han pasado los 40°C por lo que es recomendable que cuando alguno de los dos este encendido, manipule con precaución el dispositivo para que evite tocarlos directamente.
3. Los pines externos son necesarios para realizar la conexión de la planta de temperatura con el Arduino, están diseñados para que encaje perfectamente encima de cualquier placa de desarrollo que utilice, puede escoger entre el Arduino Uno, Arduino Mega y Arduino Leonardo.
4. Entrada de alimentación USB C para los transistores. Debe recordar que esta placa se alimenta únicamente con 5V, pero hay dos fuentes de alimentación dentro de ella, la primera es la alimentación 5V que brinda el Arduino, esta

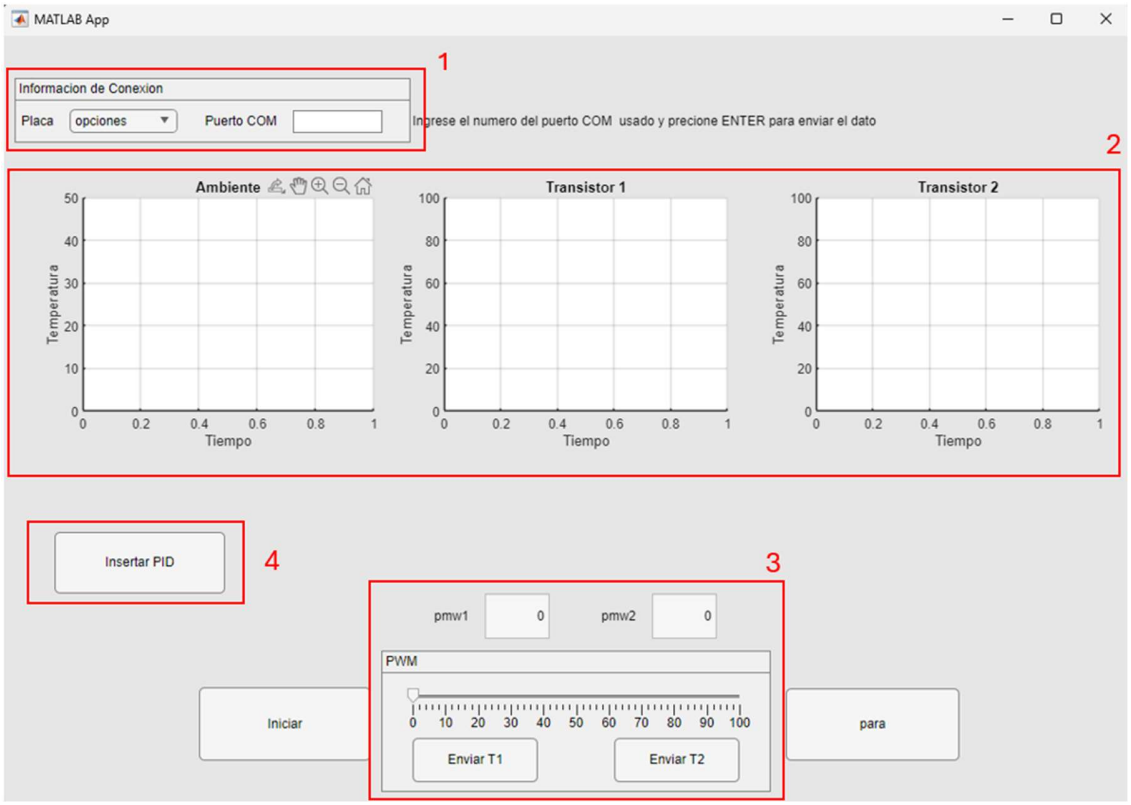
se usa para todos los componentes como sensores, resistencias, amplificadores operacionales y leds, la segunda es el USB C que ve en la imagen 27, donde puede conectar un cargador común de 5V, preferiblemente que proporcione 2A o más, para alimentar los transistores. Siempre deben conectarse ambas alimentaciones para utilizar la planta de temperatura.

5. Sensores LM35 para la lectura de la temperatura, U3 y U4 son los encargados de tomar los datos de los transistores, mientras que U1 toma la temperatura ambiente.

Es recomendable que aplique algún tipo de pasta térmica entre los sensores y los transistores con el fin de mejorar la transmisión de calor y así poder tener lecturas más precisas y confiables.

Después de tener en cuenta los componentes básicos de la planta de temperatura y las recomendaciones, puede proceder con la interacción de la interfaz gráfica, a continuación, se encuentra la ventana principal:

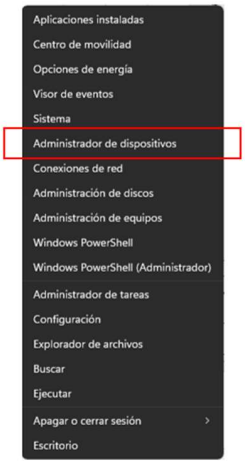
Figura 39. Ventana principal de la interfaz gráfica.



Una vez se abra la aplicación se encuentra con esta ventana (ver figura 39), la cual tiene lo necesario para llevar a cabo la identificación de su planta y se divide en secciones:

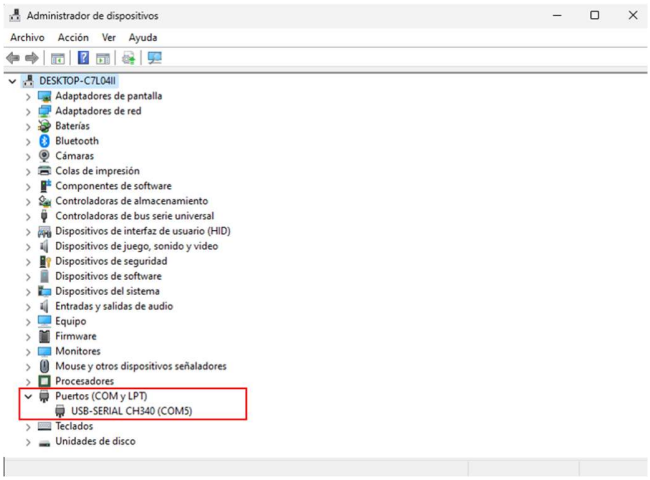
En el primer recuadro, debe tener en cuenta la información de conexión, donde tiene que seleccionar la placa que usa y el puerto COM en el que su computadora detecta el Arduino usado para poder establecer de forma correcta la conexión entre la aplicación y la placa, para esto tendrá que ir al administrador de dispositivos dando clic derecho en el icono de Windows (ver figura 40).

Figura 40. Selección de administrador de dispositivos.



Seguidamente busque el apartado de puertos (COM y LPT) (ver figura 41) donde está el puerto usado que se debe tener en cuenta para seguir la configuración de la interfaz gráfica.

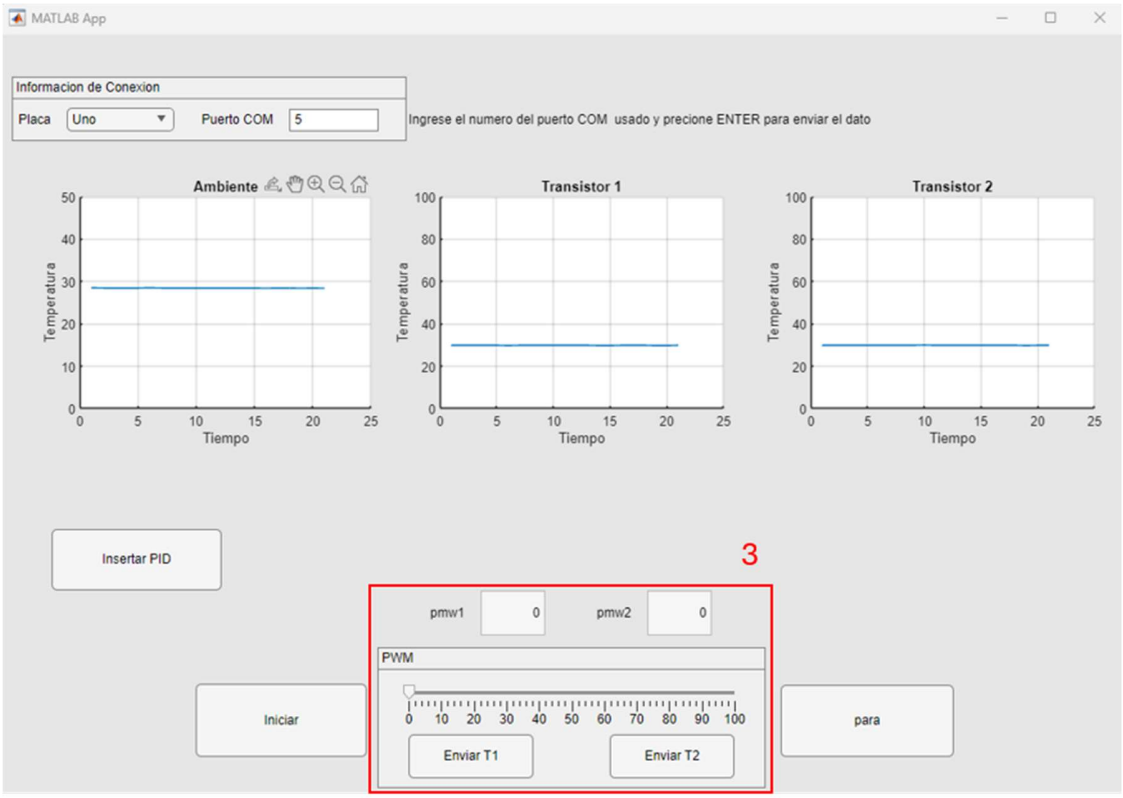
Figura 41. Puertos COM y LPT desde el dispositivo.



Teniendo esta información, asegure que la placa Heater Home Lab está conectada correctamente al Arduino y alimentada externamente al puerto tipo C en esta. Deberá iniciar la recolección de datos presionando el botón, una vez haga esto se desplegará una ventana de Windows que le pedirá seleccionar la ruta donde se va a guardar el archivo .CSV que genera la aplicación con los datos recolectados, como recomendación elija la carpeta donde tiene guardado el ejecutable de la interfaz.

Después de esto, en el recuadro 2 se muestran las 3 gráficas donde puede observar la temperatura tomada por los sensores (ver figura 42).

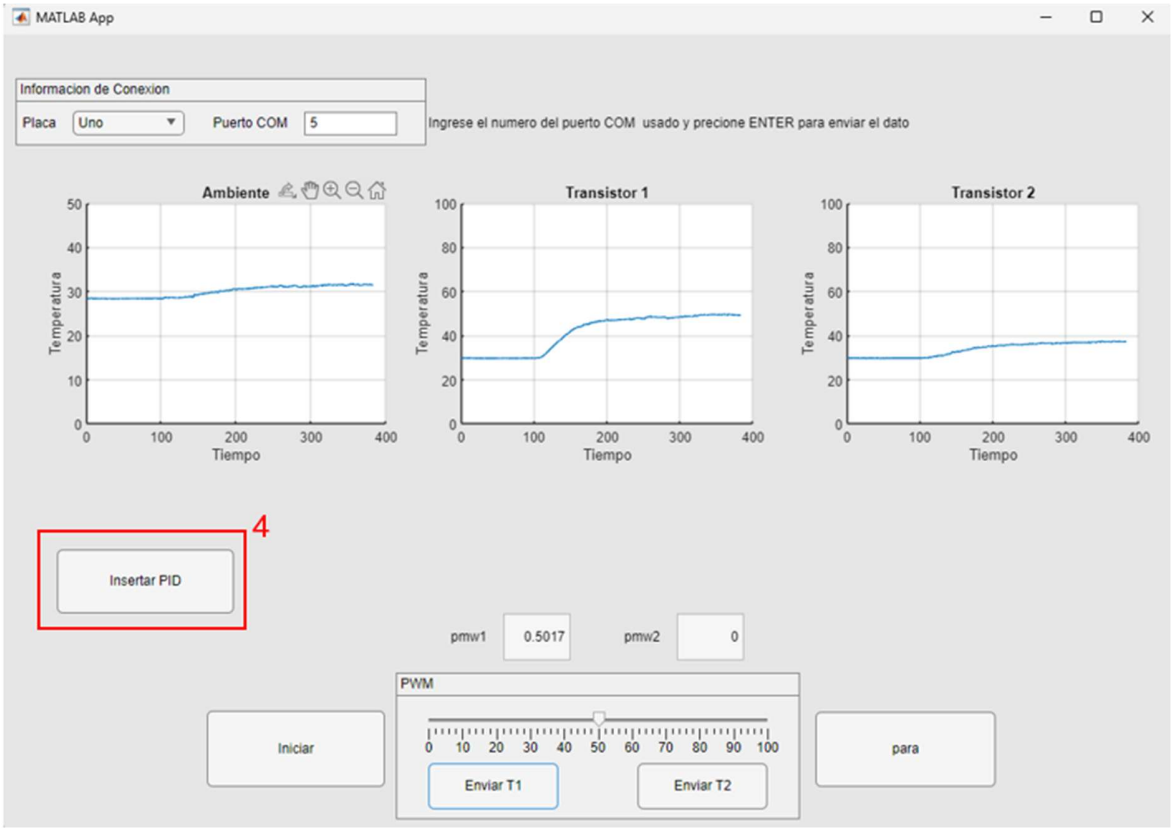
Figura 42. Graficación tomada de los sensores LM35.



En este punto la aplicación está guardando todos los datos que se muestran en las gráficas para posteriormente realizar su análisis.

El tercer recuadro muestra la barra donde puede seleccionar el PWM que quiere enviar a cualquiera de los dos transistores de la placa, necesario para el proceso de identificación, después de seleccionar el valor deseado presione enviar al transistor a trabajar, se mostrara el valor real que se le envía al transistor en la parte superior llamado pwm1 y pwm2.

Figura 43. Graficación tomada de los sensores LM35 después de enviar el PWM.

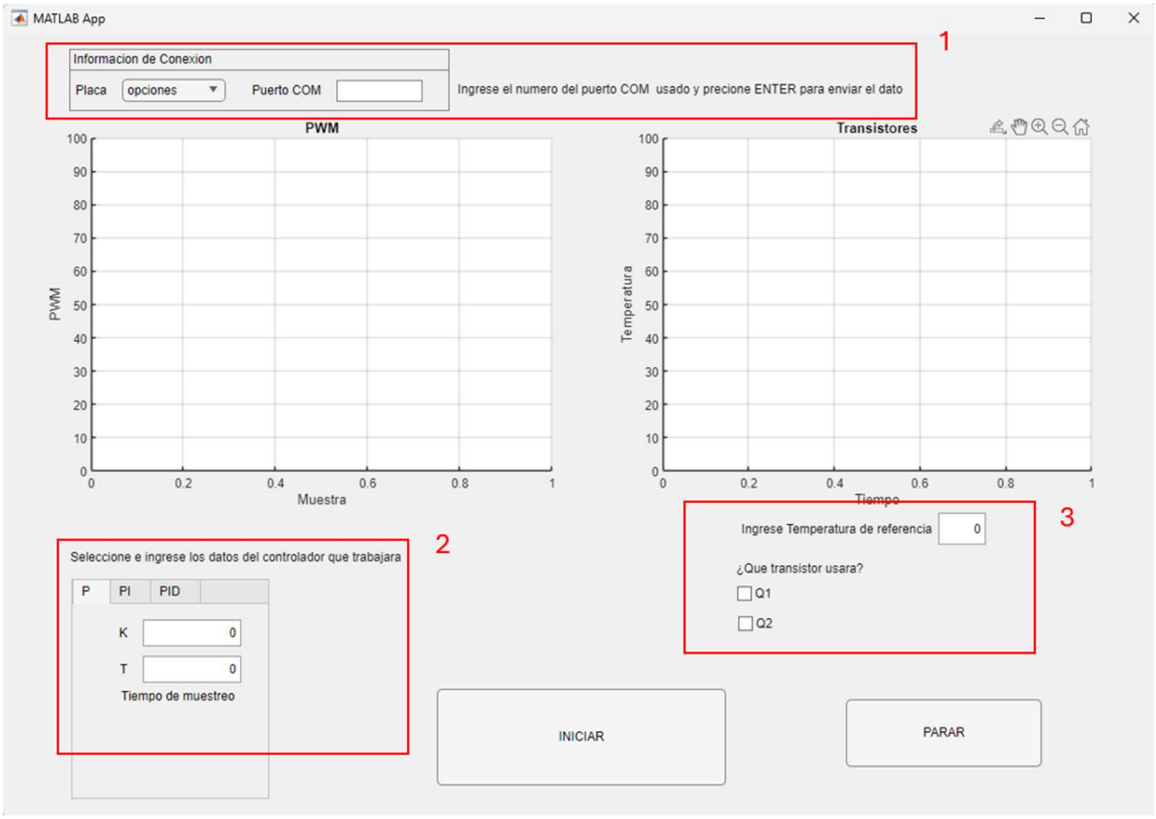


Después de enviar el PWM, puede observar cómo los sensores detectan el cambio en la temperatura. Luego de tomar los datos por el tiempo que desee, presione en parar para detener la simulación.

En este punto su archivo esta creado en la ruta seleccionada con los datos obtenidos en el proceso, por lo que puede realizar los procesos necesarios para la identificación de su planta.

Una vez realice todos los procesos de identificación y diseño del controlador podrá ver en el recuadro número 4 (ver figura 43) un botón para insertar los datos que obtuvo en una nueva pestaña.

Figura 44. Ventana secundaria de la interfaz gráfica.



Una vez aquí (ver figura 44), debe tener en cuenta que la pestaña debe establecer la conexión nuevamente con Arduino, por lo que para iniciar de forma correcta tiene que seleccionar la placa que está usando y el puerto COM ocupado, como se menciona anteriormente.

Para trabajar en esta pestaña, seleccione que tipo de controlador quiere trabajar P, PI o PID, esto se ve en el recuadro número 2 de la imagen anterior, ingrese los datos que arrojaron los cálculos que realizó para su controlador después de la identificación y seguidamente puede pasar al recuadro 3 donde debe seleccionar con que transistor está trabajando, además de la temperatura de referencia a la que quiere que llegue su controlador. Verifique que todos los datos estén ingresados de forma correcta y proceda con la simulación. Su gráfica inicia a mostrarse en los axis. Al igual que antes en esta ventana se le pide seleccionar la ruta para que los datos medidos a lo largo de toda la simulación sean guardados para poder ser analizados de ser necesario.

Anexo B: Creación de variables globales en ventana primaria.

```
    properties (Access = private)

        matriz1 = [];
        file_path = '';


    end

    properties (Access = public)

        parar = true;
        guardar = true;
        valorpwm1 = 0;
        valorpwm2 = 0;
        a

    end
```

Anexo C: Función iniciar en ventana principal.

```
      function IniciarButtonPushed(app, event)
          usada = app.PlacaDropDown.Value;
          pcom = app.PuertoCOMEditField.Value;
          elcom = strcat('COM', pcom);
          app.a = arduino(elcom, usada);
          app.parar = false;
          app.file_path = uiputfile('*.csv', 'Guardar como');
          k = 1;
          while ~app.parar
             pwm1(k) = app.valorpwm1;
             pwm2(k) = app.valorpwm2;
             muestra(k) = k;

             ambiente(k) = readVoltage(app.a,'A0')* 12.66;
             lectura1(k) = readVoltage(app.a,'A1')* 25;
             lectura2(k) = readVoltage(app.a,'A2')* 25;

             if lectura1(k) >= 40
                 writeDigitalPin(app.a, 'D7', 1);
             else
                 writeDigitalPin(app.a, 'D7', 0);
             end

             if lectura2(k) >= 40
                 writeDigitalPin(app.a,'D8',1);
             else
                 writeDigitalPin(app.a, 'D8', 0);
             end
```

```
29
30                format bank;
31
32                app.matriz1(1, 1) = muestra(k);
33                app.matriz1(1, 2) = round(ambiente(k), 2);
34                app.matriz1(1, 3) = round(lectura1(k), 2);
35                app.matriz1(1, 4) = round(lectura2(k), 2);
36                app.matriz1(1, 5) = round(pwm1(k) * 100, 2);
37                app.matriz1(1, 6) = round(pwm2(k) * 100, 2);
38                writematrix(app.matriz1, app.file_path, 'WriteMode',
39 'append');
40
41                plot(app.grafica, muestra, ambiente);
42                plot(app.grafica_2, muestra, lectura1);
43                plot(app.grafica_3, muestra, lectura2);
44
45                k = k + 1;
46
47                %set(app.inicioButton,'BackgroundColor',[0
48 0],'FontColor',[0.96 0.96 0.96],'Text','Sensando...');
49
50                pause(1);
              end
          end
```

Anexo D: Función parar en ventana principal.

```
1         function paraButtonPushed(app, event)
2
3             % Desconectar el Arduino
4              app.parar = true;
5              clear app.a;
6              delete app.a;
7              delete(app)% cierra ventana de identificación
8
9         end
```

Anexo E: Función T1 y T2 en ventana principal.

```
 1         function EnviarT2ButtonPushed(app, event)
 2             value = app.Slider.Value;
 3             app.valorpwm2 = value / 100;
 4             app.pmw2EditField.Value = app.valorpwm2;
 5             writePWMDutyCycle(app.a, 'D10', app.valorpwm2);
 6         end
 7
 8         function EnviarT1ButtonPushed(app, event)
 9             value = app.Slider.Value;
10             app.valorpwm1 = value / 100;
```

```
11              app.pmw1EditField.Value = app.valorpwm1;
12              writePWMDutyCycle(app.a, 'D11', app.valorpwm1);
13          end
```

Anexo F: Función insertar PID en ventana principal.

```
1         function InsertarPIDButtonPushed(app, event)
2             %delete(app.a) % cortar comunicacion con Arduino
3
4             PID_app % abre ventana para ingresar el PID
5
6             delete(app)% cierra ventana de identificación
7         end
```

Anexo G: Creación de variables globales en ventana secundaria.

```
 1
 2     properties (Access = private)
 3
 4         e1 = [0.0, 0.0, 0.0];
 5         e2 = [0.0, 0.0, 0.0];
 6         u1 = [0.0, 0.0];
 7         u2 = [0.0, 0.0];
 8
 9         matriz1 = [];
10         file_path = '';
11
12     end
13
14     properties (Access = public)
15
16         parar = true;
17         guardar = true;
18         a
19
20     end
21
```

Anexo H: Función iniciar en ventana secundaria.

```
           function INICIARButtonValueChanged(app, event)
1              usada = app.PlacaDropDown.Value;
2              pcom = app.PuertoCOMEditField.Value;
3              elcom = strcat('COM', pcom);
4              app.a = arduino(elcom, usada);
5              app.parar = false;
6              app.file_path = uiputfile('*.csv', 'Guardar como');
7              i = 1;
8              H1 = [];
```

```
 9              selectedTab = app.TabGroup.SelectedTab;
10              titulo = selectedTab.Title;
11              switch titulo
12                  case 'P'
13                      K = app.K_P.Value;
14                      t = app.T_P.Value;
15                      q0 = K;
16                      q1 = 0.0;
17                      q2 = 0.0;
18                  case 'PI'
19                      K = app.K_PI.Value;
20                      Ti = app.Ti_PI.Value;
21                      t = app.T_PI.Value;
22                      q0 = K * ( 1 + t/(2*Ti) );
23                      q1 = -K * ( 1 - t/(2*Ti) );
24                      q2 = 0.0;
25                  case 'PID'
26                      K = app.K_PID.Value;
27                      Ti = app.Ti_PID.Value;
28                      Td = app.Td_PID.Value;
29                      t = app.T_PID.Value;
30                      q0 = K * (1 + t/(2*Ti) + Td/t );
31                      q1 = -K * (1 - t/(2*Ti) + 2*Td/t );
32                      q2 = K*Td/t;
33              end
34
35              while ~app.parar
36
37                 muestra(i) = i;
38
39                 lectura1(i) = readVoltage(app.a,'A1')* 25;
40                 lectura2(i) = readVoltage(app.a,'A2')* 25;
41
42                 H1(i) = 0;
43                 H2(i) = 0;
44
45                 if lectura1 >= 40
46                     writeDigitalPin(app.a, 'D7', 1);
47                 else
48                     writeDigitalPin(app.a, 'D7', 0);
49                 end
50
51                 if lectura2 >= 40
52                     writeDigitalPin(app.a,'D8',1);
53                 else
54                     writeDigitalPin(app.a, 'D8', 0);
55                 end
56
57                 if app.Q1CheckBox.Value
58
```

```
59                      %Actualizacion del vector de error (etapa 1).
60                      app.e1(1) = app.e1(2);
61                      app.e1(2) = app.e1(3);
62                      app.e1(3) = app.tempRef.Value - lectura1(i);
63
64                      %Actualizacion del vector de control (etapa 1).
65                      app.u1(1) = app.u1(2);
66                      app.u1(2) = ControlPID(app.u1, app.e1, q0, q1,
   q2);
67
68                      H1(i) = app.u1(2) / 100; %Se mapea el PWM obtenico
69 para enviarlo al arduino.
                        writePWMDutyCycle(app.a, 'D11', H1(i));
70
71                  end
72
73                  if app.Q2CheckBox.Value
74
75                      %Actualizacion del vector de error (etapa 2).
76                      app.e2(1) = app.e2(2);
77                      app.e2(2) = app.e2(3);
78                      app.e2(3) = app.tempRef.Value - lectura2(i);
79
80                      %Actualizacion del vector de control (etapa 2).
81                      app.u2(1) = app.u2(2);
82                      app.u2(2) = ControlPID(app.u2, app.e2, q0, q1,
83 q2);
84
85                      H2(i) = app.u2(2) / 100; %Se mapea el PWM obtenico
   para enviarlo al arduino.
86                      writePWMDutyCycle(app.a, 'D10', H2(i));
87
88                  end
89
90                  format bank;
91
92                  app.matriz1(1, 1) = muestra(i);
93                  app.matriz1(1, 2) = round(lectura1(i), 2);
94                  app.matriz1(1, 3) = round(lectura2(i), 2);
95                  app.matriz1(1, 4) = round(H1(i) * 100, 2);
96
97                  app.matriz1(1, 5) = round(H2(i) * 100, 2);
98
99                  writematrix(app.matriz1, app.file_path, 'WriteMode',
   'append');
100
101                 plot(app.grafica_pwm, muestra, H2 * 100, muestra, H1 *
   100);
102                 legend(app.grafica_pwm, 'H2', 'H1');
103
```

```
104                 plot(app.grafica_temp, muestra, lectura2, muestra,
    lectura1);
105                 legend(app.grafica_temp, 'Lectura2', 'Lectura1');
106
107                 drawnow;
108
109                 i = i + 1;
110
111                 %set(app.inicioButton,'BackgroundColor',[0 0
    0],'FontColor',[0.96 0.96 0.96],'Text','Sensando...');
112
113                 pause(t);
114             end
```

Anexo I: Función controlador en ventana secundaria.

```
 1             function [act] = ControlPID(u, e, q0, q1, q2)
 2
 3                 act = u(1) + q0*e(3) + q1*e(2) + q2*e(1);
 4
 5                 if act >= 100
 6                     act = 100;
 7                 elseif act <= 0
 8                     act = 0;
 9                 end
10             end
```

Anexo J: Función checkbox Q1 y Q2 en ventana secundaria.

```
 1          function Q1CheckBoxValueChanged(app, event)
 2              Q1 = app.Q1CheckBox.Value;
 3
 4              if Q1
 5                  app.Q2CheckBox.Value = false;
 6              end
 7          end
 8
 9          % Value changed function: Q2CheckBox
10          function Q2CheckBoxValueChanged(app, event)
11              Q2 = app.Q2CheckBox.Value;
12              if Q2
13                  app.Q1CheckBox.Value = false;
14              end
15          end
16      end
```

Anexo K: Función parar en ventana secundaria.

```
1          function PARARButtonPushed(app, event)
```

```
2
3               % Desconectar el Arduino
4               app.parar = true;
5               clear app.a;
6               delete app.a;
7               delete app;
8
9           end
```

Anexo L: Fotografías del acople de la planta de temperatura con el Arduino Uno.

Figura 45. Vista superior de la planta de temperatura.

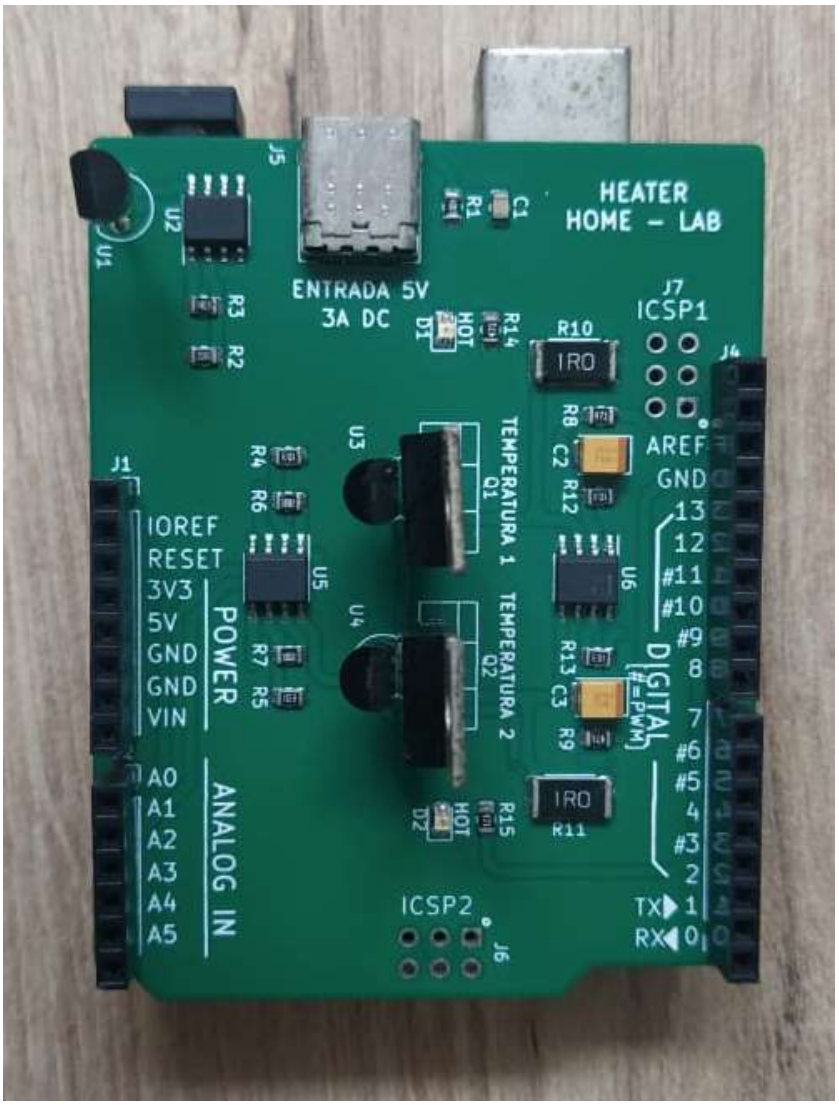


Figura 46. Vista transversal de la planta de temperatura acoplada al Arduino.