UNIVERSIDAD POLITÉCNICA SALESIANA

SEDE GUAYAQUIL

CARRERA DE ELECTRÓNICA Y AUTOMATIZACIÓN

"DISEÑO E IMPLEMENTACIÓN DE UN CONTROLADOR PID MEDIANTE SISTEMAS EMBEBIDOS PARA UN PROTOTIPO DE CÁMARA DE ESTABILIDAD ACELERADA"

Trabajo de titulación previo a la obtención del Título

de Ingeniero en Electrónica

AUTORES: POLO CARLOS GIL VERA

IAN LINZANG VALVERDE AVILÉS

TUTOR: ING. GEOVANNY XAVIER GARCÍA FLOR MSc.

GUAYAQUIL – ECUADOR

2023 - 2024

CERTIFICADO DE RESPONSABILIDAD Y AUTORÍA DEL TRABAJO DE TITULACIÓN

Nosotros, Gil Vera Polo Carlos con documento de identificación N° 0706752433 y Valverde Avilés Ian Linzang con documento de identificación N° 0931274351, manifestamos que:

Somos los autores y responsables del presente trabajo; y, autorizamos a que sin fines de lucro la Universidad Politécnica Salesiana pueda usar, difundir, reproducir o publicar de manera total o parcial el presente trabajo de titulación.

Guayaquil, 14 de febrero del año 2024.

Atentamente,

Polo Carlos Gil Vera
0706752433

Ian Linzang Valverde Avilés
0931274351

CERTIFICADO DE CESIÓN DE DERECHOS DE AUTOR DEL TRABAJO DE TITULACIÓN A LA UNIVERSIDAD POLITÉCNICA SALESIANA

Nosotros, Polo Carlos Gil Vera con documento de identificación N° 0706752433 e Ian Linzang Valverde Avilés con documento de identificación N° 0931274351, expresamos nuestra voluntad y por medio del presente documento cedemos a la Universidad Politécnica Salesiana la titularidad sobre los derechos patrimoniales en virtud de que somos autores del Proyecto Técnico: "DISEÑO E IMPLEMENTACIÓN DE UN CONTROLADOR PID MEDIANTE SISTEMAS EMBEBIDOS PARA UN PROTOTIPO DE CÁMARA DE ESTABILIDAD ACELERADA", el cual ha sido desarrollado para optar por el título de: Ingeniero en Electrónica, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

En concordancia con lo manifestado, suscribimos este documento en el momento que hacemos la entrega del trabajo final en formato digital a la Biblioteca de la Universidad Politécnica Salesiana.

Guayaquil, 14 de febrero del año 2024.

Atentamente,

Polo Carlos Gil Vera
0706752433

Ian Linzang Valverde Avilés
0931274351

CERTIFICADO DE DIRECCIÓN DEL TRABAJO DE TITULACIÓN

Yo, Geovanny Xavier García Flor con documento de identificación N° 0922357702, docente de la Universidad Politécnica Salesiana, declaro que bajo mi tutoría fue desarrollado el trabajo de titulación: DISEÑO E IMPLEMENTACIÓN DE UN CONTROLADOR PID MEDIANTE SISTEMAS EMBEBIDOS PARA UN PROTOTIPO DE CÁMARA DE ESTABILIDAD ACELERADA, realizado por Polo Carlos Gil Vera con documento de identificación N° 0706752433 e Ian Linzang Valverde con documento de identificación N° 0931274351, obteniendo como resultado final el trabajo de titulación bajo la opción de Proyecto Técnico que cumple con todos los requisitos determinados por la Universidad Politécnica Salesiana.

Guayaquil, 14 de febrero del año 2024.

Atentamente,

Ing. Geovanny Xavier García Flor, MSc.
0922357702

DEDICATORIA

A mi mamá y papá por apoyarme en todo momento.

Siempre confiaron en mí y sin ellos, nada de esto hubiese sido posible.

Polo Gil V.

A mi madre y abuela por estar siempre y darme su apoyo.

Ian Valverde A.

AGRADECIMIENTO

En primer lugar, a Dios, por cuidarme y guiarme por el buen camino. A mis padres, por confiar en mí y siempre estar cuando más los necesitaba. A mi familia, por sus buenos deseos y apoyo. A mis amigos y personas muy cercanas a mí, con quienes he compartido buenos y malos momentos a lo largo de esta travesía y que siempre estarán presentes en mi vida. Y a los docentes de la universidad, quienes han compartido su conocimiento con nosotros.

Polo Gil V.

Agradezco a Dios por permitirme llegar tan lejos, a mi madre y abuela por todo el apoyo dado a lo largo de mi vida, en especial en esta etapa tan importante.

Ian Valverde A.

Resumen

En las industrias alimenticias y farmacéuticas, el estudio para determinar el tiempo de vida útil en sus productos es algo fundamental, por lo que se requiere un equipo que permita controlar las variables de una manera muy eficiente. Esto con el fin de tener el mínimo porcentaje de error en los resultados que se desean obtener y con ello, mejorar la calidad y durabilidad en los distintos lugares de destino.

El presente proyecto plantea el desarrollo de un prototipo de una cámara de estabilidad acelerada mediante un control PID con el objetivo de comprobar si el prototipo tendrá un mejor funcionamiento con respecto a la estabilización de la temperatura y humedad relativa que se genera dentro de éste, en comparación de un control ON/OFF. La función de este controlador es mantener estables la temperatura y la humedad dentro de la cámara con las siguientes especificaciones: temperatura a 40 ºC ± 2 ºC y una humedad relativa de 75% Hr ± 5% Hr.

Además, para un mejor monitoreo, todos los datos que se generen se verán visualizados en tiempo real a través de la plataforma ThingSpeak, el cual irá mostrando los cambios de temperatura y humedad relativa cada 15 segundos.

Palabras Clave: Cámara de estabilidad acelerada, control PID, sistemas embebidos, función de transferencia, temperatura, humedad relativa.

Abstract

In the food and pharmaceutical industries, the study to determine the shelf life in their products is something fundamental, so equipment is required to control the variables in a very efficient way. This to have the minimum percentage of error in the results to be obtained and thus, improve the quality and durability in the different places of destination.

The present project proposes the development of a prototype of an accelerated stability chamber using a PID control with the objective of testing if the prototype will have a better performance with respect to the stabilization of the temperature and relative humidity generated inside it, compared to an ON/OFF control. The function of this controller is to maintain stable temperature and humidity inside the chamber with the following specifications: temperature at 40 ºC ± 2 ºC and a relative humidity of 75% Hr ± 5% Hr.

In addition, for better control and monitoring, all the data generated will be visualized in real time through the ThingSpeak platform, which will display the changes in temperature and relative humidity every 15 seconds.

Keywords: Temperature and humidity chamber, PID control, embedded systems, transfer function, temperature, relative humidity.

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

I INTRODUCCIÓN

En este Trabajo de Titulación se busca implementar un prototipo de una cámara de estabilidad acelerada con control PID, y con ello, mantener constante la temperatura y humedad relativa que se genera dentro de la cámara.

Este proyecto se originó por la ausencia de un sistema de control eficiente para estabilizar la temperatura y humedad en el interior de la cámara de estabilidad acelerada en una empresa de alimentos en la ciudad de Guayaquil. Este control no permite estabilizar el sistema de manera eficiente de temperatura y humedad, por ello, con esta iniciativa se podrán mantener controladas la temperatura y humedad acorde a lo que se necesita.

Esto se desarrolla mediante el sensor DTH22, quien captura y envía todos los datos de humedad y temperatura. A través del módulo WIFI ESP32, los datos obtenidos son enviados a la nube donde se podrán visualizar las gráficas generadas de cada variable respectivamente. Se hace uso de la plataforma ThingSpeak, donde se muestran las gráficas de la temperatura y humedad en tiempo real. Con la ayuda de la herramienta EXCEL, se guardan todos los datos obtenidos.

Además de ello, se utiliza la plataforma de programación MATLAB, con la cual se obtendrá la función de transferencia y se desarrollará el controlador PID, mediante los datos guardados en la hoja de cálculo (Excel).

II PROBLEMA

Uno de los procesos de análisis que se le realiza al producto terminado alimenticio o farmacéutico es el estudio diseñado para determinar su tiempo de vida útil, para lo que se requiere tener un equipo que permita controlar las variables. Los estudios en estas cámaras tienen las siguientes especificaciones: una temperatura de 40 ºC ± 2 ºC de tolerancia, a una humedad relativa de 75% Hr ± 5% Hr de tolerancia para sólidos. Esto es fundamental para realizar una buena simulación sobre el tiempo de vida útil de un producto y garantizar que se cumplan las normativas de la ICH (Concejo Internacional de Armonización), la FDA (Administración de Alimentos y Medicamentos) y la EMA (Agencia Europea de Alimentos) (mpcontrol.es, 2022; A/S., 2023); (Alfaro, 2006)**.**

En el mercado ecuatoriano existen Cámaras de Estabilidad Acelerada (CEA) que no cuentan con control eficiente de temperatura y humedad. Los valores de consigna de temperatura y humedad tienen períodos de tiempo largos en estabilizar y alcanzando valores máximos que salen de las tolerancias especificadas por los procedimientos establecidos por el laboratorio al momento de realizar una perturbación al sistema de control, esto está presente en los modelos de cámara de estabilidad acelerada que cuentan con un control ON/OFF, siendo la consecuencia más común la pérdida de hermeticidad del equipo al aperturar su puerta para sacar o ingresar productos lo que se realiza con frecuencia durante la jornada laboral.

Al tener este control poco eficiente en el manejo de ambas variables no permite tener una obtención de datos cercana a la realidad, ya que las perturbaciones que puede haber en este sistema de control no se van a mantener estables, por consecuencia de esto, los resultados se verán afectados.

III OBJETIVOS

3.1 Objetivo general

Diseñar e implementar un controlador PID mediante sistemas embebidos para un prototipo de cámara de estabilidad acelerada.

3.2 Objetivos específicos

1) Realizar el prototipo de la cámara de estabilidad acelerada.
2) Diseñar el control PID de temperatura y humedad mediante sistemas embebidos.
3) Implementar el sistema de control PID.
4) Registrar la obtención de datos en tiempo real.

IV FUNDAMENTOS TEÓRICOS

4.1 Control PID

Es un mecanismo de control que brinda la posibilidad de dominar el tiempo de respuesta y aumentar su precisión. Este tipo de control es aplicado en sistemas que se requiera de una alta precisión en sus mediciones o procesos y tiene 3 parámetros: (MINT FOR PEOPLE, s.f.)

Proporcional (P). – Es quien se encarga de realizar una medición del valor actual y del set-point para poder reducir el error entre ambos.

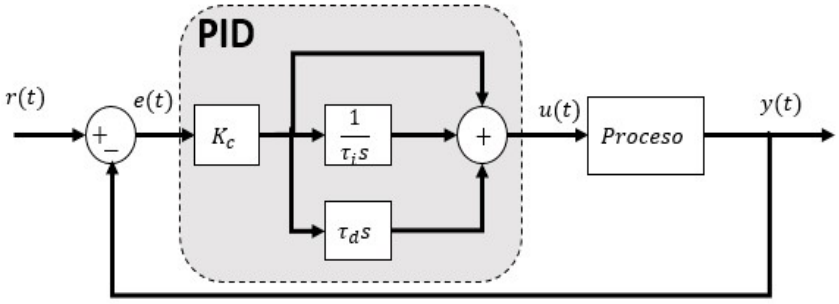
Integral (I). – Es el tiempo determinado para que la acción correctiva sea preciso para la estabilización del sistema.

Derivativo (D). – Se encarga de realizar una acción preventiva en el caso de un error para poder tomar una decisión y corregirlo.

A continuación, se muestra un ejemplo de control PID en la Figura 1.

Figura 1

Controlador PID



Nota: Representación de un Control PID, por (CASTAÑO, s.f.)

(https://controlautomaticoeducacion.com/control-realimentado/control-pid-por-asignacion-de-polos/)

4.2 Cámara de estabilidad acelerada

Es un tipo de cámara climática constante que realiza algunas funciones. Estas cámaras pueden simular la temperatura y humedad, con el fin de determinar la durabilidad de un material, identificando los posibles efectos que el ambiente pueda causar en algún material biológico, electrónico o industrial.

Las cámaras de estabilidad se distinguen del resto por ser capaces de manejar complejas condiciones ambientales con temperaturas extremas y adecuados niveles de humedad.

Son usadas en las industrias de caucho, automotriz, plástica y farmacéutica para determinar los efectos de la humedad y temperatura en sus productos o componentes sin la necesidad de ir hasta el lugar de destino, tal como se observa en la Figura 2. (Raptor Supplies BV, 2023)

Figura 2

Cámara de estabilidad acelerada

Nota: Esta imagen muestra un ejemplo de la cámara de estabilidad acelerada, extraída de (dellamarca, 2022), (https://www.dellamarca.it/es/camaras-de-prueba-de-estabilidad/)

4.3 Sistemas embebidos.

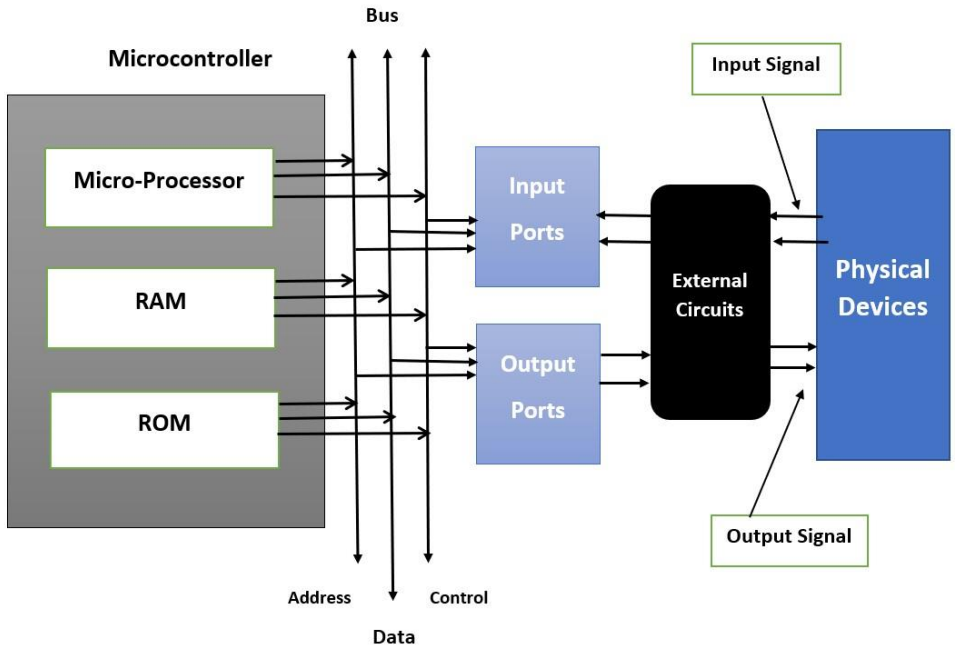
Se refiere a todo circuito electrónico digital capaz de realizar operaciones de computación. Su arquitectura contiene un microprocesador que pueden ejecutar instrucciones a una velocidad determinada con un bajo consumo de energía. (Salas Arriarán, 2015)

En un sistema embebido los componentes se encuentran integrados a una placa base, conocida también como motherboard. Todo el procesamiento central se lleva a cabo gracias a un microcontrolador. (TRBL Services, s.f.)

En la Figura 3. se muestra una referencia de un sistema embebido.

Figura 3

Sistema embebido



Nota: Referencia a los sistemas embebidos o integrados, donde se muestra el proceso y cómo funciona, de (InnovacionDigital360, 2023). (https://www.innovaciondigital360.com/iot/sistemas-embebidos-que-son-y-para-que-se-utilizan/)

4.4 Resistencia eléctrica de calentamiento

En la Figura 4 se observa una referencia, la cual consiste en un componente eléctrico que tiene como objetivo producir de calor mediante la aplicación de corriente eléctrica. Están diseñadas para que puedan soportar diferentes temperaturas, dependiendo el material con el cual fueron fabricadas y el proceso que se quiera realizar. (Electricfor, s.f.)

Figura 4.

Resistencia eléctrica de calentamiento

Nota: Imagen de referencia sobre una resistencia eléctrica de calentamiento usada en procesos industriales. Obtenido de (horecatiger, s.f.) (https://horecatiger.eu/es-es/shop/resistencia-1600w-240v-espirales-1-l-68mm-an-173mm-420090)

4.5 Humidificador ultrasónico

Es un dispositivo que funciona con vibraciones que crean gotas de agua y se dispersan en el aire en forma de niebla. (BeijinUltrasonic, 2023)

Entran en la categoría de humidificadores de vapor frío. Este dispositivo empieza a generar humedad al instante que se enciende y no consume mucha electricidad porque no utiliza calor para su funcionamiento. (H2O TEK, s.f.)

En la Figura 5. se puede observar la imagen de referencia.

Figura 5

Humidificador ultrasónico

Nota: Imagen de referencia de un humidificador ultrasónico, de (Regalogar, s.f.) (https://regalogar.es/humidificadores/14794-orbegozo-humidificador-hu-2050-52l-ultrasonico-vapor-frio-8435568400627.html)

4.6 Relé de estado sólido

Es un interruptor electrónico, con una salida de estado sólido como un SCR, TRIAC, MOSFET o un transistor, que permite el encendido o apagado de cargas cuando se aplica una señal o un voltaje externo a sus terminales de control.

Utilizan optoacopladores para aislar la entrada de la salida, por lo que no se crean picos de voltaje al apagar y se evitan arcos eléctricos. (ELECTRONICA UNIVERSAL DE MONTERREY, s.f)

Se puede observar un ejemplo de un relé de estado sólido en la Figura 6.

Figura 6

Relé de estado sólido

Nota: En la imagen se puede apreciar un ejemplo de un relé de estado sólido, extraída de (ELECTRO STORE, s.f.) (https://grupoelectrostore.com/shop/interruptores-y-switches/rele-estado-solido-ssr-25da-25a-input-3-32vdc-output-24-380vac/)

4.7 Módulo DHT22

Es un sensor que mide la temperatura en un rango de -40°C a 80°C y la humedad relativa del 0% al 100% en un mismo encapsulado (BricoGeek, s.f.).

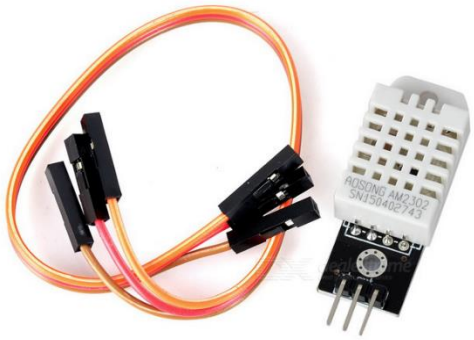
A continuación, se mostrarán las especificaciones técnicas del sensor:

- Voltaje de Operación: 3V - 6V DC
- Rango de medición de temperatura: -40°C a 80 °C
- Precisión de medición de temperatura: <±0.5 °C
- Resolución Temperatura: 0.1°C
- Rango de medición de humedad: De 0 a 100% RH
- Precisión de medición de humedad: 2% RH
- Resolución Humedad: 0.1%RH
- Modelo: AM2302

Se puede observar el sensor en la Figura 7.

Figura 7

Módulo DHT22



Nota: Imagen mostrando el sensor de temperatura y humedad, junto a sus jumpers. Extraído de (SigmaElectronica, s.f.) (https://www.sigmaelectronica.net/producto/dht22/)

4.8 Ventilador

Es un aparato que transmite energía, gracias a que genera la presión necesaria para mantener un flujo de aire continuo. Se utilizan generalmente para ventilar algún ambiente, pero

también son usados para refrigerar máquinas o hacer que los gases circulen por conductos. (hello auto, s.f.)

En la Figura 8 tenemos como referencia un ventilador.

Figura 8

Ventilador

Nota: Referencia de un ventilador, obtenido de (tiendamia, s.f.) (https://tiendamia.com/ec/producto?amz=B07VFD6VNX&pName=GDSTIME%2090mm%20Fan%2024V,%2092mm%20x%2092mm%20x%2025mm%20Brushless%20Cooling%20Fan/&gclid=CjwKCAiAtt2tBhBDEiwALZuhAKYAdhABbei6-OJ0_hB5cUQzxIyRhjEVHX9Uq7PpSf7-gsaGU03kHBoCJLgQAvD_BwE)

4.9 Pantalla Nextion Intelligent Series

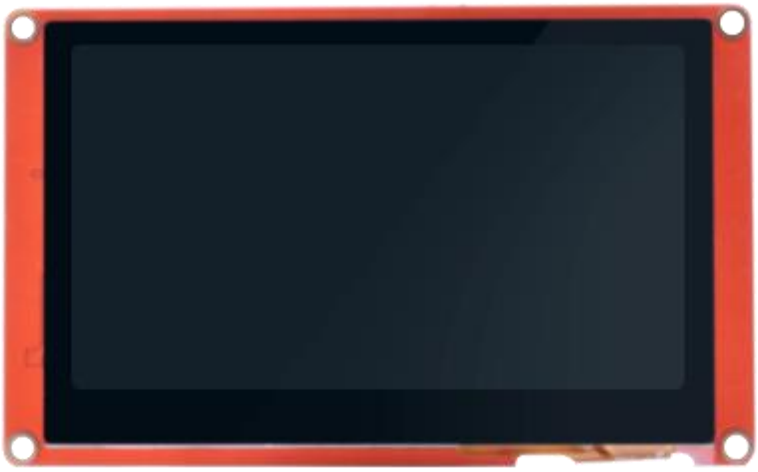
Es una serie de pantalla inteligente de la marca Nextion. Se diferencia de las Basic y Enhanced Series gracias a su potente hardware, las funciones de reproducción de audio, video y animación, las cuales hacen que la interacción HMI sea más enriquecedora.

Además, admite varias características y funciones avanzadas de software, componentes transparentes, efectos de carga de página, movimiento y arrastre de componentes. Esto hace que sea utilizada en muchas áreas. (NEXTION, s.f.)

En la Figura 9 se puede observar la pantalla inteligente.

Figura 9

Pantalla Nextion Intelligent Series.



Nota: Imagen de una pantalla de 4,3 pulgadas de la marca Nextion, Intelligent Series, modelo capacitivo NX4827P043-011C. Obtenido de (NEXTION, s.f.) (https://nextion.tech/datasheets/NX4827P043-011C/)

4.10 ESP32 Dev Kit Module

Se le denomina ESP32 a la familia de chips SoC (System on a chip / Sistema en un Chip) con tecnología Wifi y Bluetooth. Es chip fue desarrollado por Espressif Systems. Este módulo llega a realizar tareas de bajo consumo hasta tareas exigentes. (Carranza, 2021)

Cuenta con las siguientes características en sus pines:

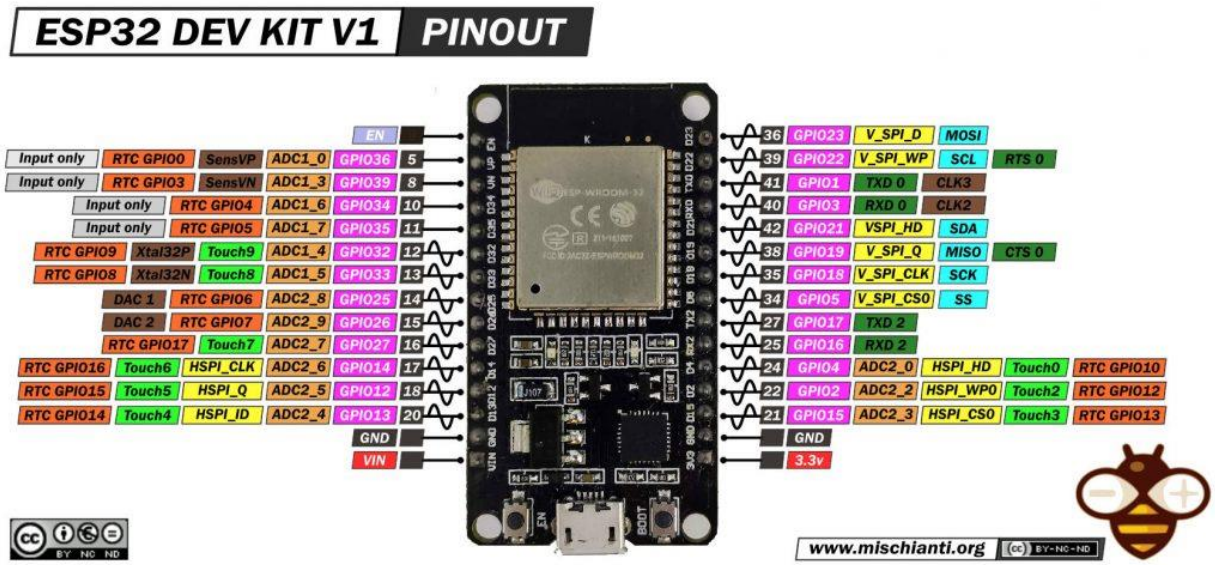
- 19 canales de convertidor analógico a digital (ADC)
- 3 interfaces SPI
- 3 interfaces UART

- 2 interfaces I2C
- 16 canales de salida PWM
- 2 convertidores de digital a analógico (DAC)
- 2 interfaces I2S
- 10 GPIO de detección capacitiva

Esta información se puede confirmar en la Figura 10.

Figura 10

Modulo ESP32 Dev Kit V1



Nota: Se muestra un módulo ESP32 Dev Kit V1 con la característica de cada pin. Obtenido de (MISCHIANTI, 2023). (https://mischianti.org/doit-esp32-dev-kit-v1-high-resolution-pinout-and-specs/)

4.11 PID Tuner (MATLAB)

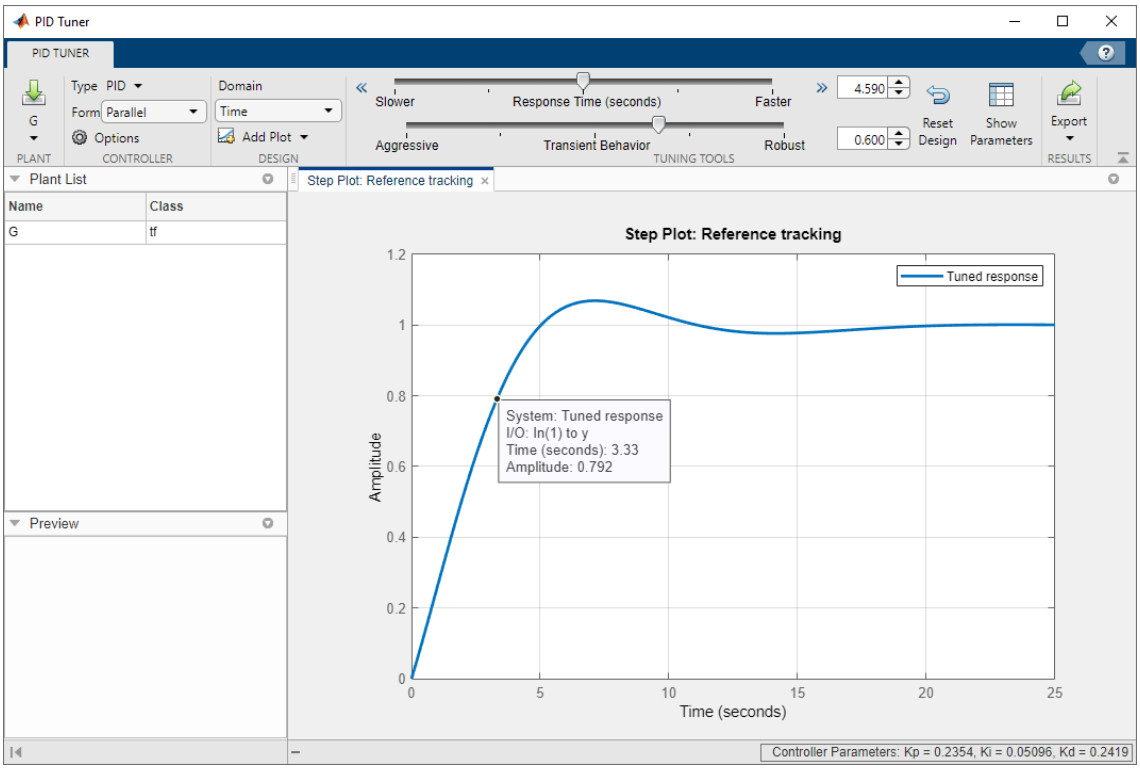
Es una aplicación del programa MATLAB que te ayuda a ajustar automáticamente las ganancias de un controlador PID. Se puede especificar el tipo de controlador, ya sea PI, PID con

filtro derivativo o con dos grados de libertad (-2DOF). También puede perfeccionar interactivamente el rendimiento del controlador para ajustar el ancho de banda del lazo y margen de fase, o para anular perturbaciones. (MathWorks, s.f.)

En la Figura 11 se muestra la aplicación.

Figura 11

PID Tuner



Nota: En la imagen se muestra una gráfica de un sistema siendo estabilizado por la aplicación PID Tuner. Obtenido de la página de MathWorks (https://la.mathworks.com/help/control/ref/pidtuner-app.html)

V MARCO METODOLÓGICO

Este estudio se llevó a cabo con el diseño de un prototipo de cámara de estabilidad acelerada, obtención de datos dentro de la cámara (humedad y temperatura), el uso de un ESP32 como control principal, Arduino IDE para la programación del sistema embebido, envío de datos a Excel y ThingSpeak para la visualización de los mismos, MATLAB para la obtención de la función de transferencia y el control PID respectivo para cada variable, por último, la implementación del control PID al sistema embebido y realizar la comparativa de éste versus un control ON/OFF.

A continuación, se detallará cada paso realizado, así como los métodos y se explicará la programación para llevar a cabo este proyecto:

5.1 Realizar el prototipo de la cámara de estabilidad acelerada.

En primer lugar, se investigó sobre las cámaras, cómo están construidas por dentro, además se buscó imágenes sobre sus modelos, y en base a ello empezar a diseñar el prototipo con el fin de hacer una representación fiel a éstas.

Hay que tomar en cuenta que la construcción de una cámara de estabilidad acelerada es muy compleja, debido al cuidado necesario en lo que se refiere a la hermeticidad y control de la humedad que es muy volátil.

Para el prototipo de la cámara se toma como referencia la Figura 12.

Figura 12

Prototipo estimado de la cámara de estabilidad acelerada.

Nota: Representación del prototipo de la cámara de estabilidad acelerada.

5.1.1 Diseño de la cámara de estabilidad acelerada

Para poder realizar el prototipo se tomaron en cuenta todos los componentes necesarios, tales como: ventilador, resistencia seca, sensor DTH, y humidificador ultrasónico con respecto a la estructura principal externa.

A continuación, se detallan las partes fundamentales de la cámara.

- **Tarjeta de control y fuerza.**

Se encargará del envío y obtención de los datos que den los sensores, así mismo se encontrara el sistema embebido haciendo la obtención de estos datos en tiempo real, estos datos serán mandados al PID para que realice el trabajo de estabilizar en caso se detecte una perturbación.

- **Pantalla Nextion TFT**

Se visualizan los parámetros dentro de la cámara. Esta pantalla permite configurar su interfaz según se requiera y mejorar la interacción Hombre - Máquina.

- **Sensor de humedad y temperatura**

Se hace uso de un sensor DTH22, el cual cumple con la función de medir la temperatura y humedad al mismo tiempo, para obtener los datos de estas variables y se conectará al sistema embebido para el tratamiento de la data obtenida.

- **Resistencia eléctrica de calentamiento**

Para poder generar la temperatura se utiliza una resistencia seca. Esta resistencia sirve para generar el calor y alcanzar la temperatura configurada por el usuario, la señal eléctrica de activación viene del sistema embebido que acciona un relé de estado sólido que está conectado a la alimentación de voltaje de la resistencia de calentamiento.

- **Ventilador**

Se hace uso de un ventilador de 5 pulgadas ubicado en el fondo de del prototipo, el cual hace circular el aire de manera que la temperatura y la humedad relativa sea homogénea en todo el interior de la cámara.

- **Humidificador**

Genera la humedad en el interior de la cámara para que se encuentre en el punto configurado por el usuario, la señal eléctrica de activación viene del sistema embebido que acciona un relé de estado sólido que estar conectado a la alimentación de voltaje del humidificador.

5.1.2 Adaptación de la estructura

Se usó una estructura hermética, ya que se requiere que la cámara no presente filtración de humedad y temperatura hacia el exterior. Se adquirieron las medidas para realizar un doble fondo con galvanizado de 1 mm, ya que el mismo es muy resistente a la humedad y la temperatura, en el diseño del doble fondo se tomó la decisión de dejar fuga en la parte superior, inferior y posterior, esto con el objetivo de colocar un aislante termino entre el fondo original y el galvanizado.

Así mismo se adquirió las medidas para realizar el un doble fondo con galvanizado de 1 mm, ya que el mismo es resistente a la humedad y la temperatura. (monterrey, s.f.)

Una base de acero con un contorno de galvanizado de 1 mm fue necesaria para la elevación de la cámara y así obtener un mejor espacio para el humidificador. A la salida del humidificador se conectó un tubo flexible para que el vapor sea dirigido dentro de la cámara sin que tenga obstrucciones o exista la posibilidad que se quede truncado.

Los cables y control se encuentran en la parte superior con un doble fondo para que el cableado quede en la parte inferior y no se exponga en la parte de control. Se colocó un Switch de control ON/OFF para toda la cámara, para el encendido y apagado de todos los componentes.

5.1.3 Tarjeta PCB

El uso de Proteus 8 Professional fue elemental para la fabricación de la tarjeta PCB. Para realizar una tarjeta PCB sin fallos se verificó si en el programa antes mencionado existe el esquemático de cada componente a usar, en caso de ser así verificó las medidas de los componentes.

Mencionado esto, en el caso del ESP32 no existía un esquemático, así que se procedió realizar uno, con las medidas 25TH para el cuerpo y la separación de los pines con 50TH, al mencionar los pines se debe verificar que los mismos estén en el sentido correcto, ya que esto a futuro nos puede ocasionar problemas en caso estar reflejados.

Una vez ya se haya realizado y verificado el esquemático del ESP32, y los demás componentes a usar, se procede a realizar las conexiones de estos, en caso de conectar algo adicional como el uso de una fuente externa de 5V como en este proyecto, se decidió representar aquello como dos pines juntos ya que la fuente iría conectada en esos pines para alimentar a los transistores y pantalla Nextion.

5.1.4 Cableado y Control

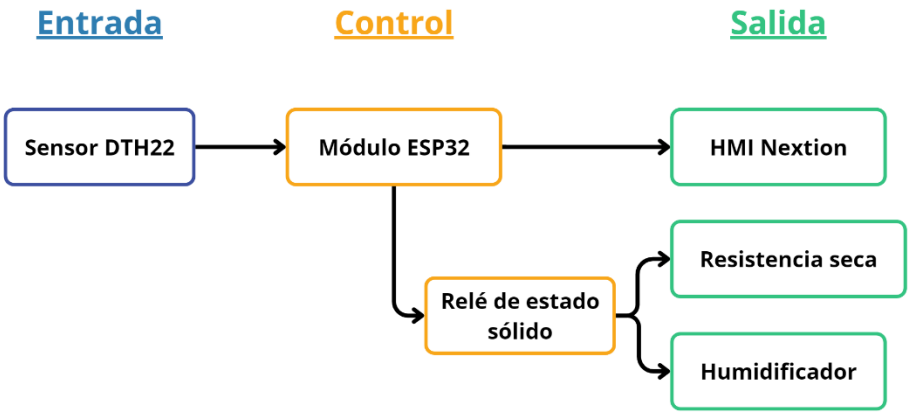
Para el control del prototipo se realizó un esquemático, representando todo el sistema. Se tomaron en cuenta las especificaciones de cada componente y en base de ello inició proceso de cableado y las conexiones necesarias.

Se utilizó el mismo cable de alimentación de la vinera para toda la cámara, desde allí se unieron los cables mediante terminales de conexión.

En la Figura 13 se explica el control del sistema.

Figura 13

Esquema del sistema

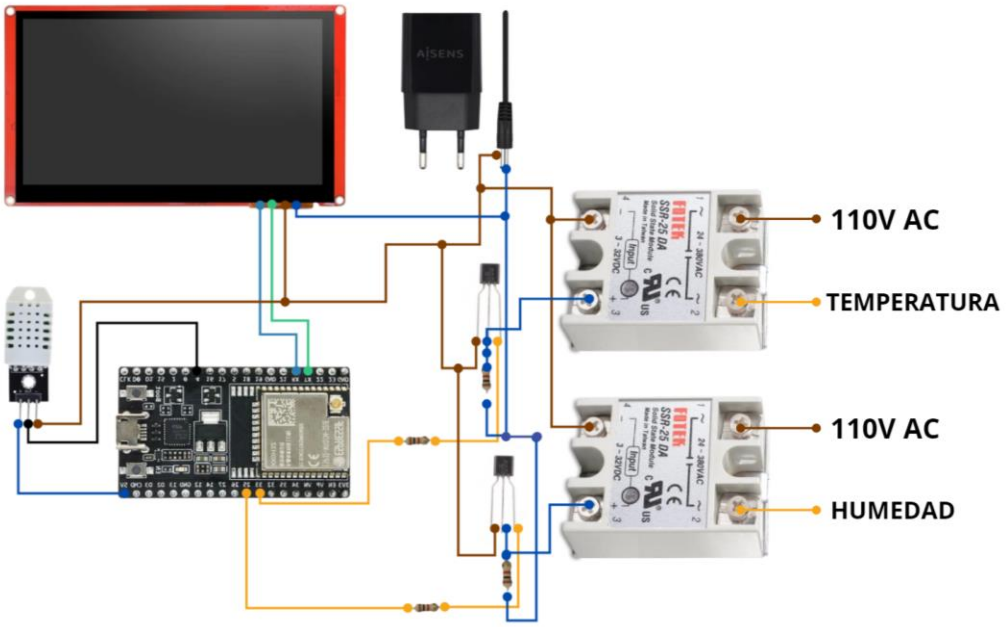


Nota: Representación del esquema explicando el control del sistema.

En la Figura 14 se visualizan las conexiones del control de manera gráfica.

Figura 14

Diagrama gráfico del sistema



Nota: Representación del esquema explicando el control del sistema de manera gráfica. Se puede observar detalladamente las conexiones y componentes utilizados.

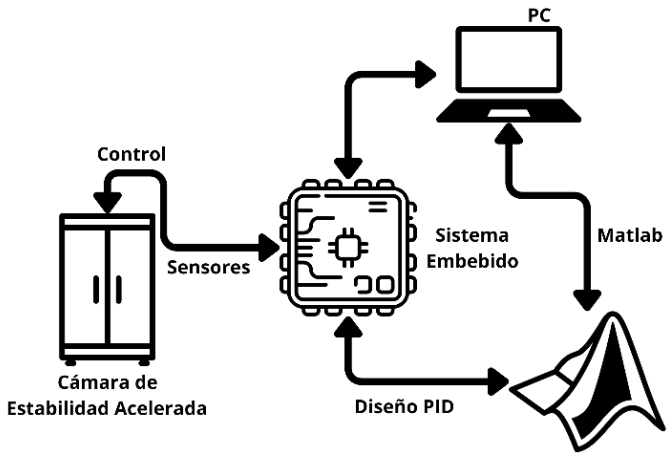
5.2 Diseñar el control PID de temperatura y humedad.

El sistema embebido obtiene los datos dentro de la cámara de estabilidad mediante los sensores que tiene implementados, estos datos se envían a un registrador, luego los mismos se dirigen a Matlab. En Matlab se ingresan estos datos obtenidos los cuales una vez sean procesados nos da como resultado la función de transferencia, que sirve para la programación del PID en el sistema embebido.

En la Figura 15 se observa el proceso antes mencionado.

Figura 15

Descripción Grafica del Diseño PID



Nota: En la gráfica se puede observar el proceso que se realiza para generar el control PID de la cámara de estabilidad acelerada.

5.2.1 Librerías, variables y procesos.

Para la obtención de datos a través de Arduino IDE se hace uso de algunas librerías, las cuales se pueden observar en la Figura 16.

Figura 16

Librerías

Nota. Se puede observar las librerías necesarias para el DTH22, el ESP32, la pantalla inteligente Nextion, la comunicación serial con la pantalla, la función MILIS y conexión con ThingSpeak. Imagen obtenida por los autores.

Se declaran las librerías para inicializar los procesos en el setup de cana una. En este caso las librerías fundamentales para la adquisición de datos son las siguientes:

- DTHesp.h: La misma está hecha para la lectura/obtención de los datos enviados desde el módulo DTH22 al ESP32.
- SoftwareSerial.h: Permite habilitar la comunicación serial del ESP32

En la Figura 17 se pueden observar las variables utilizadas.

Figura 17

Variables globales y declaración de procesos en paralelo.

```
19
20  //DECLARAMOS LAS VARIBLES QUE SE USARÁN A LO LARGO DE LA PROGRAMACIÓN
21
22  //PIN DECLARADO COMO ENTRADA PARA LA LECTURA DEL SENSOR DTH22
23  int pinDHT = 4;
24
25  //SE USARÁ 2 PINES PARA LA ACTIVACIÓN DE LOS RELÉ, UNO PARA HUMEDAD Y EL OTRO PARA TEMPERATURA
26  int releH = 32;
27  int releT = 33;
28
29
30  TaskHandle_t control_hum;
31  TaskHandle_t control_temp;
32  TaskHandle_t excel;
33  //TaskHandle_t patantalla;
34
```

Nota: Se declaran las variables a usar, y los procesos que se realizarán para adquirir datos.

Se hace uso de 3 pines, 1 para la lectura de los datos enviados por el DTH22 y los 2 restantes para el control ON/OFF realizado por los pines. Cada proceso se realiza en paralelo, el comando “TaskHandle_t” habilita multitrabajo en paralelo haciendo uso de los 2 core con los que cuenta el sistema embebido. En la Figura 18 se observan las declaraciones en el setup.

Figura 18

Declaraciones en el setup

```
37  void setup() {
38    //VELOCIDAD DE BAUDIOS POR SEGUNDO
39    Serial.begin(115200);
40
41    dht.setup(pinDHT, DHTesp::DHT22);
42
43    //CONTROL DE TAREA EN PARALELO HUMEDAD
44    xTaskCreatePinnedToCore(loop0,"control_hum",1000,NULL,1,&control_hum,1);
45
46    //CONTROL DE TAREA EN PARALELO TEMPERATURA
47    xTaskCreatePinnedToCore(loop1,"control_temp",1000,NULL,1,&control_temp,1);
48
49    //CONTROL DE TAREA EN PARALELO DE ENVIO DE DATOS A EXCEL
50    xTaskCreatePinnedToCore(loop2,"excel",1000,NULL,1,&excel,0);
51
52    pinMode (releH, OUTPUT);
53    pinMode (releT, OUTPUT);
54  }
```

Nota: Se declara en el setup la inicialización de procesos, sensores o el modo de uso para los pines y el correcto llamado de librerías o procesos.

Lo primero que se debe realizar es declaración de la velocidad de baudios por segundos, generalmente siempre de usa la mayor que pueda soportar el sistema embebido, en este caso el ESP32 puede llegar a soportar una velocidad de 115200 baudios por segundo, por consiguiente, se declara el llamado de la librería, se especifica a que sensor de humedad y temperatura ya sea DTH11 o DTH22, en este caso usaremos el segundo modulo.

Se declaran los procesos a realizar en paralelo, ambos procesos de control para la humedad y temperatura se declaran en el core 1, con un orden de prioridad de 1, el proceso para el envío de datos a Excel se lo declara de igual manera en el core 1, así mismo con un orden de prioridad de 1. El orden de prioridad define la importancia del proceso para cada core.

Se visualiza el control de humedad en la Figura 19.

Figura 19

Control de humedad

```
64  void loop0(void *parameter){
65    while (1 == 1){
66
67    TempAndHumidity data = dht.getTempAndHumidity();
68    if ( data.humidity < 73 ) {
69      digitalWrite(releH, LOW); //ENCIENDE RELÉ
70
71    }
72    else if ( 73 < data.humidity && data.humidity < 75 ) {
73      digitalWrite(releH, LOW); //ENCIENDE RELÉ
74      delay(8500); // ESPERA UN SEGUNDO Y MEDIO
75
76      digitalWrite(releH, HIGH); //APAGA RELÉ
77      delay(1500); // ESPERA UN SEGUNDO
78    }
79    else if ( 75 < data.humidity && data.humidity < 77 ) {
80      digitalWrite(releH, LOW); //ENCIENDE RELÉ
81      delay(7500); // ESPERA UN SEGUNDO Y MEDIO
82
83      digitalWrite(releH, HIGH); //APAGA RELÉ
84      delay(1500); // ESPERA UN SEGUNDO
85    }
86    else if ( data.humidity > 77) {
87      digitalWrite(releH, HIGH); //APAGA RELÉ
88    }
89    }
90    //COMANDO PARA QUE NO SALTE NINGUN PERRO GUARDIAN
91    vTaskDelay(10);
92  }
```

Nota: Se realiza el control ON/OFF por cada caso evaluando la humedad.

Se ha realizado un control evaluando continuamente la humedad en un bucle loop, esto con el objetivo de comparar constantemente la variable y dependiendo el caso se procede a mandar pulsos al relé por tiempo determinado dependiendo el porcentaje de humedad dentro de la cámara. En la figura 20, se observa el control de temperatura.

Figura 20

Control de temperatura

```
void loop1(void *parameter){
  while (1 == 1){

  TempAndHumidity data = dht.getTempAndHumidity();
  if ( data.temperature < 38 ) {
    digitalWrite(releT, LOW); //ENCIENDE RELÉ
    delay(2500); // ESPERA UN SEGUNDO Y MEDIO

    digitalWrite(releT, HIGH); //APAGA RELÉ
    delay(1500); // ESPERA UN SEGUNDO
  }
  else if ( 38 < data.temperature && data.temperature < 39 ) {
    digitalWrite(releT, LOW); //ENCIENDE RELÉ
    delay(1300); // ESPERA UN SEGUNDO Y MEDIO

    digitalWrite(releT, HIGH); //APAGA RELÉ
    delay(2500); // ESPERA UN SEGUNDO
  }
  else if ( 39 < data.temperature && data.temperature < 40 ) {
    digitalWrite(releT, LOW); //ENCIENDE RELÉ
    delay(1600); // ESPERA UN SEGUNDO Y MEDIO

    digitalWrite(releT, HIGH); //APAGA RELÉ
    delay(3600); // ESPERA UN SEGUNDO
  }
  else if ( data.temperature > 40) {
    digitalWrite(releT, HIGH); //APAGA RELÉ
  }

  //COMANDO PARA QUE NO SALTE NINGUN PERRO GUARDIAN
  vTaskDelay(10);
}
}
```

Nota: Se observa el código donde se realiza el control ON/OFF por cada caso evaluando la temperatura.

Se ha realizado un control evaluando continuamente la humedad en un bucle loop, esto con el objetivo de comparar constantemente la variable y dependiendo el caso se procede a mandar pulsos al relé por tiempo determinado dependiendo de la temperatura dentro de la cámara. Esto con el fin de recolectar los datos, previo al diseño del control PID, tal como se observa en la Figura 21.

Figura 21

Recolección de datos

```
void loop2(void *parameter){
  while (1 == 1){
    TempAndHumidity data = dht.getTempAndHumidity();
    //MOSTRAMOS LOS DATOS QUE NOS DÁ EL SENSOR DTH22
    Serial.print(String(data.humidity, 1));
    //SEPARAMOS LOS DATOS CON "," PARA REALIZAR LA SEPRACIÓN DE LOS MISMOS EN EXCEL
    Serial.print(",");
    Serial.print(String(data.temperature, 2));
    Serial.print("\n");
    //SE DEFINE EL TIEMPO DE RECOLECCIÓN DE DATOS
    delay(333);

  }
  //COMANDO PARA QUE NO SALTE NINGUN PERRO GUARDIAN
  vTaskDelay(10);
}
```

Nota: Impresión de datos por el puerto serial.

5.2.2 Adquisición de datos Excel

La comunicación serial entre el ESP32 y Excel fue fundamental para la obtención de datos, para que Excel tenga una comunicación correcta se configuro la comunicación a la misma velocidad de baudios (115200), así mismo, se configuro para que los datos se obtengan cada 333 ms. Cada dato se pone coloca en una celda, en la primera columna, primera fila se imprime la humedad, para realizar el salto a la siguiente columna, primera fila se manda a imprimir una

coma, se imprime la temperatura y por consiguiente se imprime “\n” esto con el objetivo de que regrese la impresión de los datos regrese a la primera columna, segunda fila. Tal como se puede observar en la Figura 22.

Figura 22

Datos en Excel

	A	B	C	D
1	created_at	entry_id	HUMEDAD	TEMPERATURA
2	2023-12-24T19:37:33-05:00	1	60.4	30.8
3	2023-12-24T19:37:55-05:00	2	59.1	30.9
4	2023-12-24T19:38:17-05:00	3	59.2	30.9
5	2023-12-24T19:38:33-05:00	4	58.4	30.9
6	2023-12-24T19:38:49-05:00	5	58.4	30.9
7	2023-12-24T19:39:11-05:00	6	58.5	30.8
8	2023-12-24T19:39:33-05:00	7	58.7	30.8
9	2023-12-24T19:39:55-05:00	8	59	30.7
10	2023-12-24T19:40:17-05:00	9	59.8	30.6
11	2023-12-24T19:40:39-05:00	10	72.9	30.8
12	2023-12-24T19:41:01-05:00	11	61	30.8
13	2023-12-24T19:41:22-05:00	12	60.5	30.8
14	2023-12-24T19:41:44-05:00	13	59.8	30.8
15	2023-12-24T19:42:06-05:00	14	59.8	30.8
16	2023-12-24T19:42:28-05:00	15	59.8	30.8
17	2023-12-24T19:42:50-05:00	16	60.1	30.8
18	2023-12-24T19:43:12-05:00	17	60	30.8
19	2023-12-24T19:43:34-05:00	18	59.4	30.8
20	2023-12-24T19:43:56-05:00	19	59.9	30.8
21	2023-12-24T19:44:18-05:00	20	59.7	30.7

Nota: Datos obtenidos. Se realizaron varias muestras para escoger la más estable y enviarlas a ThingSpeak.

La adquisición de datos en periodos cortos es fundamental para una correcta evaluación, en caso de ser los intervalos muy grandes se puede perder información, en cambio si los datos son muy consecutivos puede llegar a saturar la comunicación, por ello, se debe poner un tiempo prudencial para no llegar a saturar ni a perder la información.

Como se observa en la Figura 22 se coloca un intervalo de 333 ms, es decir, se toma aproximadamente 3 datos por segundo, buscando el equilibrio para transmitir la información.

5.2.3 ThingSpeak

La herramienta online de ThingSpeak es una extensión de Matlab, ya que pueden trabajar a la par, la plataforma procesa los datos de manera virtual y al mismo tiempo se envían a Matlab.

Se puede observar lo mencionado en la Figura 23.

Figura 23

Datos enviados a ThingSpeak desde Excel

Nota: Se observan los datos en la plataforma de ThingSpeak cada 15 segundos. En este caso, se tiene un número limitado de datos a enviar debido a la licencia gratuita. ThingSpeak (*https://thingspeak.com/channels/2415032*)

5.2.4 Obtención de la función de transferencia en Matlab

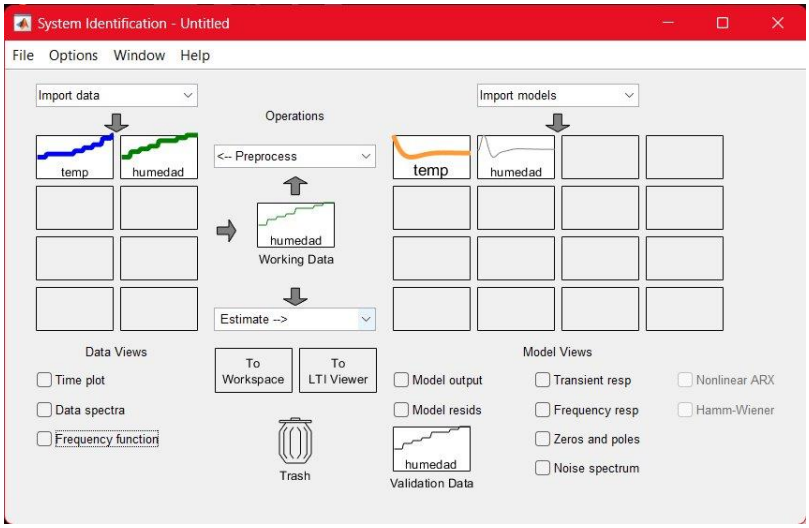
Matlab tiene varias herramientas incorporadas para la obtención de una función de transferencia por medio de datos obtenidos previamente, por ejemplo:

- PidTunner
- SystemIdentification
- Ident

En este caso, se usó SystemIdentification, se cargan las ambas columnas de datos (humedad y temperatura) en la herramienta, se debe especificar que en el eje x, este irá en función del tiempo. Una vez se hayan cargado ambos datos en el SystemIdentification, hay que especificar en el Working data que función de datos se debe usar, ya se humedad o temperatura, tal como se observa en la Figura 24.

Figura 24

Carga de datos en SystemIdentification.

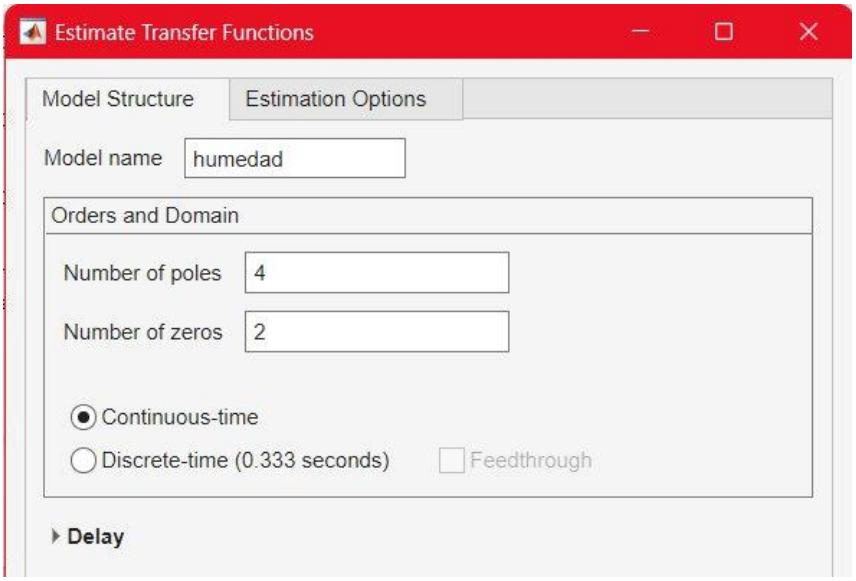


Nota: En este caso se tiene en el Working data la función obtenida de la humedad, se realiza el mismo proceso para la temperatura al momento de encontrar la función de transferencia de la temperatura.

Cuando la función de transferencia ya esté en el Working data, en la parte inferior se encuentra un menú de lo que se desea estimar, en este caso vamos a estimar la función de transferencia para la humedad, automáticamente se abrirá una nueva ventana, la misma que se observa en la Figura 25, en ella colocamos los polos y zeros que deseamos para nuestra función de trasferencia, no se puede colocar un numero al azar, ya que una vez hayamos especificado aquellos parámetros en la ventana, se debe dar click en “estimate”.

Figura 25

Definición de polos y ceros (Humedad)

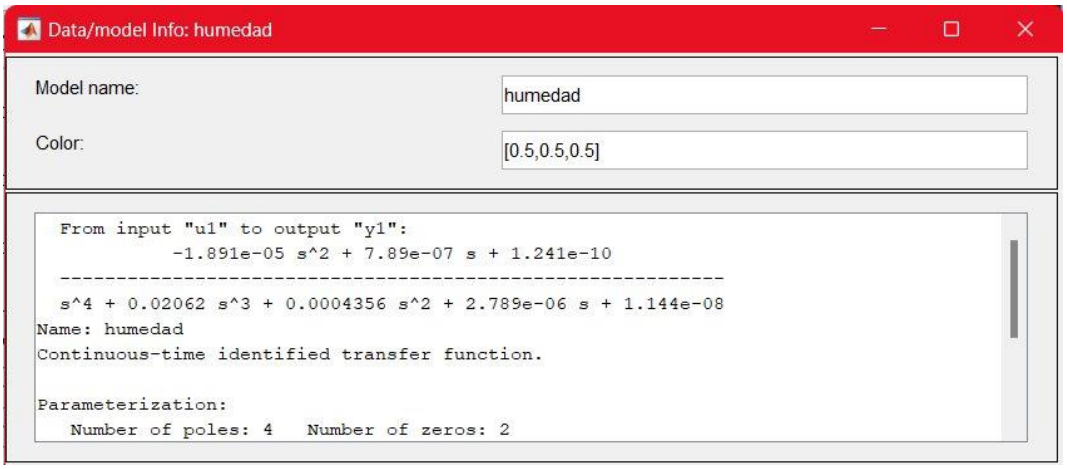


Nota: Se fijó el número de polos en 4 y de los ceros en 2.

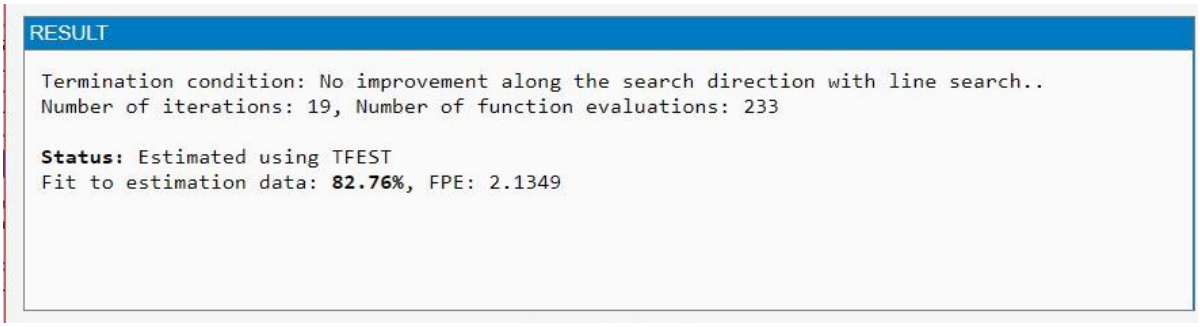
Luego de ello, Matlab abre una nueva ventana, como se observa en la Figura 26, donde mostrará la función de transferencia, en este caso debemos fijarnos en la eficiencia que nos entrega aquella función, esta eficiencia debe ser superior al 80% para que pueda generar un buen control PID.

Figura 26

Función de transferencia con eficiencia superior al 80% (Humedad)



(a)



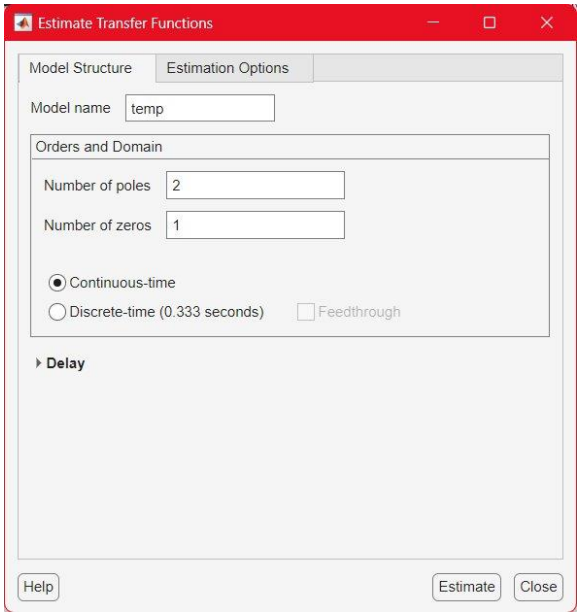
(b)

Nota: En la Figura 26 (a) se observa la función obtenida y que se debe importar al workspace, mientras en Figura 26 (b) se muestra la eficiencia estimada.

Se generó una función de transferencia superior al 80% en la humedad, se puede proseguir con el mismo proceso para la obtención de la función transferencia para la temperatura, como se observa en la Figura 27.

Figura 27

Definición de polos y ceros (Temperatura)

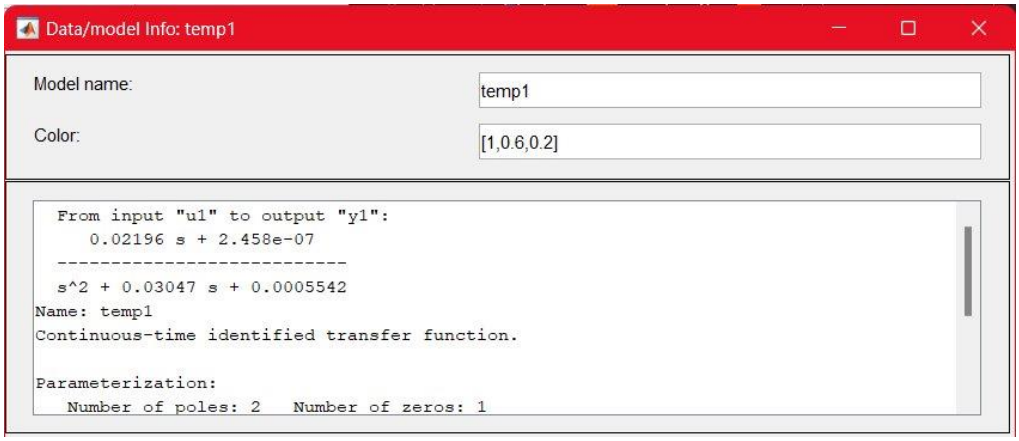


Nota: Se fijó el número de polos en 2 y de los ceros en 1.

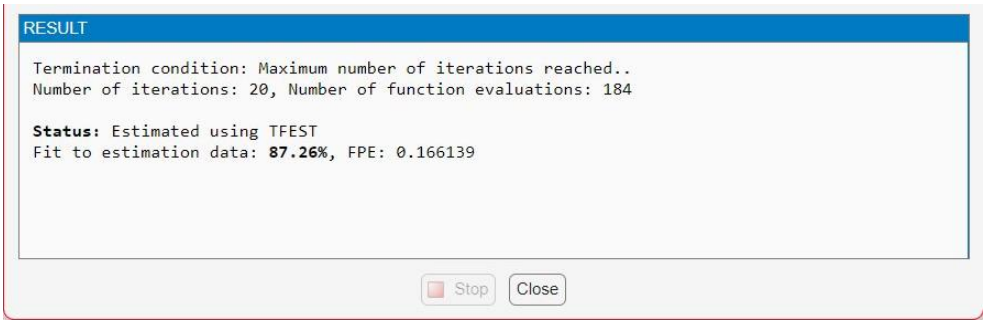
Se colocan los polos y ceros deseados sin llegar a superar los 4 polos, hasta que entregue una función de transferencia con una eficiencia superior a la del 80%, tal como se muestra en la Figura 28.

Figura 28

Función de transferencia con eficiencia superior al 80% (Temperatura)



(a)



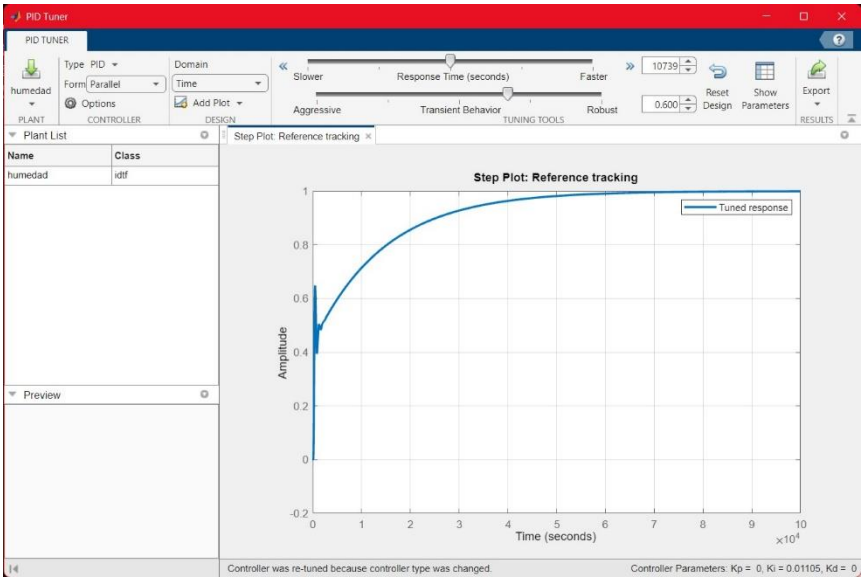
(b)

Nota: En la Figura 28 (a) se observa la función obtenida y que se debe importar al workspace, mientras en Figura 28 (b) se muestra la eficiencia estimada.

Con las funciones de transferencias para ambas variables en el workspace, se ingresa el comando de "PidTunner", se abrirá una nueva herramienta de Matlab, a la misma se le crea una nueva planta en la cual se ingresar la primera función la cual se obtuvo para la humedad como se observa en la Figura 29.

Figura 29

PID Tuner con la función de la humedad

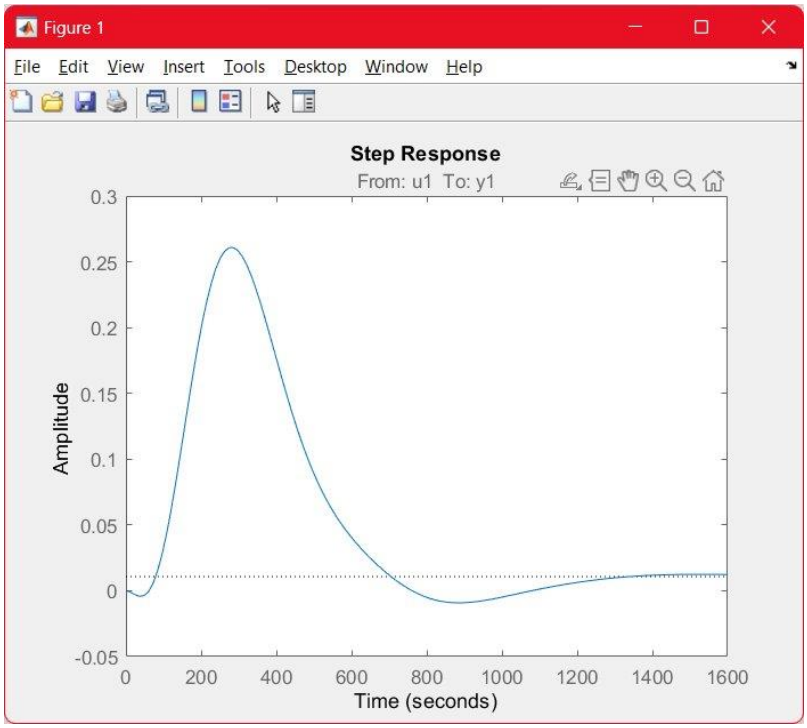


. *Nota:* Al crear una nueva planta se va a visualizar la gráfica de step plot: reference tracking para la humedad.

Luego de ello, se realiza una respuesta al impulso, se abre una nueva ventana en la cual se observa la gráfica con la respuesta al impulso de función de transferencia, la misma que se observa en la en la Figura 30.

Figura 30

Respuesta al impulso para la función de la humedad.



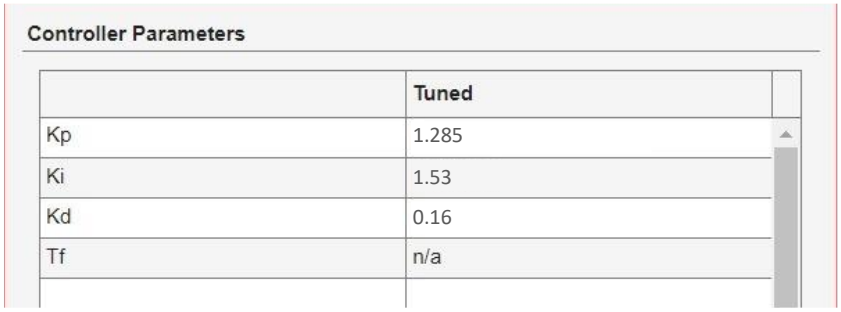
*Nota: S*e puede observar cómo reacciona el control Pid a una perturbación, en el caso de la humedad.

Se observa cómo puede llegar a reaccionar el control Pid para la humedad ante una perturbación, se observa que le cuesta un poco estabilizar, pero se logra el objetivo al final, se comprende que le cueste estabilizar ya que la humedad es volátil al momento de controlar.

Una vez se haya observado el control ante una perturbación, verifiquemos que es óptimo se puede obtener las constantes Ki, Kp y Kd, las mismas que se observan en la Figura 31.

Figura 31

Constantes para el control de humedad.



Controller Parameters

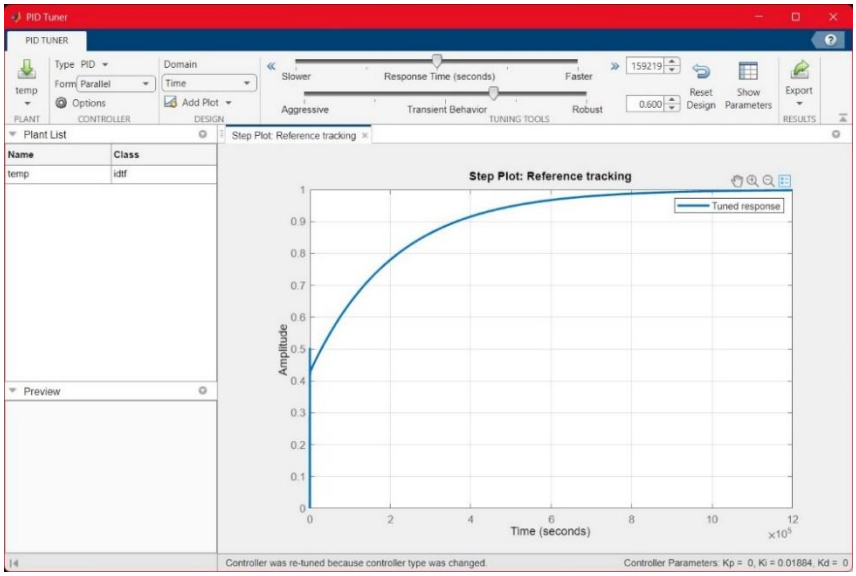
	Tuned
Kp	1.285
Ki	1.53
Kd	0.16
Tf	n/a

Nota: Las mismas constantes se deben ingresar en el código en el cual se realizará el control PID de la cámara.

Una vez se haya obtenido las constantes para el control PID de humedad, se procede a realizar el mismo procedimiento, pero esta vez con la función antes obtenida de la temperatura, como se observa en la Figura 32.

Figura 32

PID Tuner con la función de la temperatura

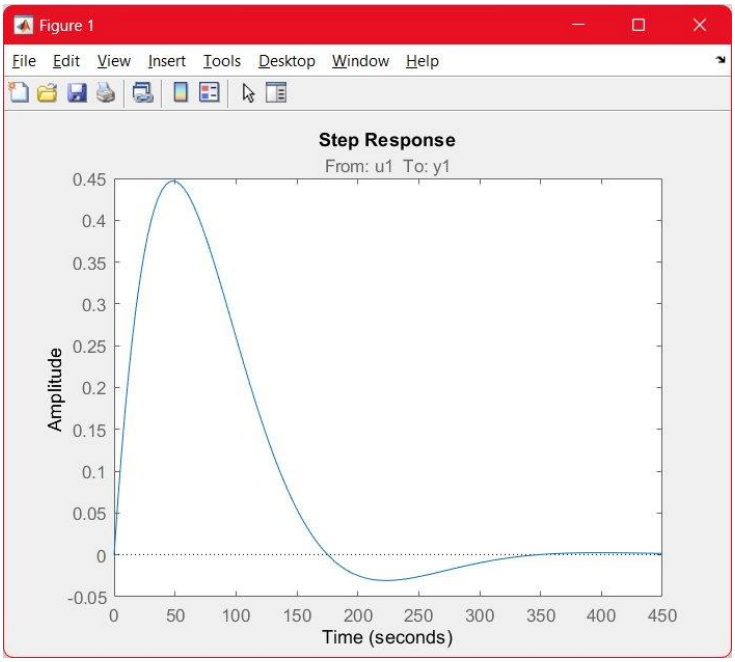


Nota: Al crear una nueva planta se va a visualizar la gráfica de step plot: reference tracking para la temperatura.

En la Figura 33 tenemos la función obtenida.

Figura 33

Respuesta al impulso para la función de la temperatura.



Nota: Se puede observar cómo reacciona el control Pid a una perturbación, en el caso de la temperatura.

Los parámetros se visualizan en la Figura 34.

Figura 34

Constantes para el control de temperatura.

	Tuned
Kp	1.075
Ki	0.914
Kd	0.154
Tf	n/a

Nota: Las mismas constantes se deben ingresar en el código en el cual se realizará el control PID de la cámara.

5.3 Implementar el sistema de control PID.

Mediante el diseño e implementación de una tarjeta electrónica basada en sistemas embebidos en la cual se colocará el algoritmo de programación, se implementará el controlador PID en el diseño del prototipo de la cámara de estabilidad acelerada, el cual contará con sus etapas de tarjeta de control y fuerza.

5.3.1 PID implementado en programación.

Se añaden las librerías a usar en la programación como se observa en la Figura 35.

Figura 35

Librerías.

```
1   //INCLUIMOS LAS LIBRERIAS QUE USARAN EN ESTE PROYECTO DE TESIS
2
3   //LIBRERIA PARA EL SENSOR DTH
4   #include "DHTesp.h"
5
6   //LIBRERIA PARA COMUNICACION SERIAL (TX - RX)
7   #include <SoftwareSerial.h>
8
9   //LIBRERIA PARA CONTROL PID
10  #include <PID_v2.h>
11
12  //LIBRERIA PARA PANTALLA NEXTION
13  #include "Nextion.h"
14
15  //LIBRERIA PARA LA CONEXCIÓN CON EL THINGSPEAK
16  #include <ThingSpeak.h>
```

Nota: Se puede observar las librerías para el correcto funcionamiento del programa.

Se declaran las librerías para inicializar los procesos en el setup de cana una. En este caso las librerías fundamentales para la adquisición de datos son las siguientes:

- DTHesp.h: La misma está hecha para la lectura/obtención de los datos enviados desde el módulo DTH22 al ESP32.
- SoftwareSerial.h: Permite habilitar la comunicación serial del ESP32
- PID_v2.h: librería para el control PID
- Nextion.h: Permite la comunicación serial en entre sistema embebido y pantalla Nextion
- ThingSpeak.h: Accede a las credenciales del usuario y permite el envío de datos.

En la Figura 36 se observan las declaraciones de las variables.

Figura 36

Declaración de objetos y variables.

```
//DECLARAMOS LAS VARIBLES QUE SE USARÁN A LO LARGO DE LA PROGRAMACIÓN

//DECLARACION DE OBJETOS PARA LOOPS EN PARALELO
TaskHandle_t OTROS;
TaskHandle_t CONTROL;

//PIN DECLARADO COMO ENTRADA PARA LA LECTURA DEL SENSOR DTH22
int pinDHT = 4;

//SE USARÁ 2 PINES PARA LA ACTIVACIÓN DE LOS RELÉ, UNO PARA HUMEDAD Y EL OTRO PARA TEMPERATURA
int releH = 32;
int releT = 33;

//CONSTANTES CHAR PARA LAS CREDENCIALES DE ACCESO WIFI
const char* ssid="CELERITY_GIL VERA";
const char* password="ISABELA7";

//CREDENCIALES PARA LA CONEXCIÓN CON EL THINGSPEAK
unsigned long channelID = 2419044;
const char* WriteAPIKey ="GRL9JNJXMSNIV2AL";

//DEFINIMOS EL CLIENTE DE WIFI QUE USAREMOS
WiFiClient cliente;

//INSTANCIA DEL SENSOR DTH22
DHTesp dht;
```

Nota: Objetos y variables necesarios para la conexión a internet, obtención de datos del sensor DTH22.

Declaración de procesos en paralelo, numero de pin para el control de los relés y sensor DTH22, credenciales de acceso para conexión wifi, credenciales de usurario para la escritura de datos en ThingSpeak, instancias para sensor y WiFi tipo cliente.

Se resalta que si no se realiza alguna instancia el momento de compilar el programa saldrá un error, o llegará a compilar, pero el ESP32 comenzará a reiniciarse, por ende, el programa no funcionará como se espera.

En la Figura 37 se visualizan las declaraciones de las constantes.

Figura 37

Constantes para control PID de humedad y temperatura.

```
46  //DECLARAMOS LOS VALORES PARA CONTROL E INICIALIZAMOS EL CONTROL PID
47  //TEMP
48  double Kp = 1.075, Ki = 0.914, Kd = 0.154;
49  PID_v2 myPID(Kp, Ki, Kd, PID::Direct);
50
51  const int WindowSize = 255;
52  unsigned long windowStartTime;
53
54  double setpoint = 40.0;
55
56  ////////////////////////////////////////////////
57
58  //HUMEDAD
59  double KpH = 1.285, KiH = 1.53, KdH = 0.16;
60  PID_v2 myPIDH(KpH, KiH, KdH, PID::Direct);
61
62  const int WindowSizeH = 5000;
63  unsigned long windowStartTimeH;
64
65  double setpointH = 75.0;
66
```

Nota: Declaración de las contantes y ventanas de trabajo para cada control.

La definición de cada constante es la clave para el correcto funcionamiento del control PID. Existen casos de errores para cada constante y esto puede dificultar el proceso de control. Por ejemplo:

Kp (Proporcional):

- Elevado: va a generar oscilaciones alrededor del setpoint
- Bajo: el control puede no ser sensible a las desviaciones del setpoint.

Ki (Integral):

- Elevado: puede causar una acción de control elevada, lo que por consecuencia causara que el sistema no logre llegar al setpoint.
- Bajo: causara que no se pueda eliminar el estado estacionario del control, lo que lleva a que el sistema se detenga antes del punto establecido.

Kd (Derivativo):

- Elevado: el control podría llegar a ser demasiado sensible a las variaciones rápidas de error.
- Bajo: puede no llegar a ser muy eficaz al momento de prevenir oscilaciones o al suavizar la respuesta del sistema.

Información obtenida de Rethinking The Future (S.L., 2024).

Para la pantalla HMI, se necesitan declarar los objetos, tal como se observa en la Figura 38.

Figura 38

Declaración de objetos para pantalla Nextion.

```
68  //CREAMOS EL OBEJTO PARA ENVIO DE DATOS A LA GRFICA
69    NexWaveform ghum= NexWaveform (0,2,"ghum");
70    NexWaveform gtemp= NexWaveform (0,1,"gtemp");
71
72  //CREAMOS OBJETOS PARA EL ENVIO DE DATOS
73    NexGauge generalhum= NexGauge (0, 1, "generalhum");
74    NexGauge generaltemp= NexGauge (0, 2, "generaltemp");
75    NexGauge humvalor= NexGauge (0, 4, "humvalor");
76    NexGauge tempvalor= NexGauge (0, 4, "tempvalor");
```

Nota: Creación de objetos para cada dato a enviar.

Es importante declarar un objeto para cada variable que llegue a tener la pantalla Nextion, identificar el id y nombre para cada variable, los mismos se debe declarar en cada objeto. Se debe realizar lo mismo para las variables que se envían a graficar en la pantalla.

Para la declaración de la velocidad de comunicación, se realiza una comunicación de 115200 baudios por segundos, inicialización de pantalla Nextion, sensor DTH22 y procesos en paralelo, declaración de los pines como salida, ya que serán los encargados de enviar los pulsos para el control, declaración para encender el control PID para cada variable a controlar, como se observa en la Figura 39.

Figura 39

Setup de la programación.

```
void setup() {

  //VELOCIDAD DE BAUDIOS POR SEGUNDO
  Serial.begin(115200);

  //INICIALIZAMOS PANTALLA NEXTION
  nexInit();

  //INICIALIZAMOS EL DTH
  dht.setup(pinDHT, DHTesp::DHT22);

  //DECLARA AL PIN COMO SALIDA
  pinMode (releH, OUTPUT);
  pinMode (releT, OUTPUT);

  TempAndHumidity data = dht.getTempAndHumidity();

  //CONTROL DE TAREA EN PARALELO TEMPERATURA
  xTaskCreatePinnedToCore(loop0,"CONTROL",1500,NULL,1,&CONTROL,1);
  xTaskCreatePinnedToCore(loop1,"OTROS",1000,NULL,2,&OTROS,0);

  windowStartTime = millis();
  myPID.SetOutputLimits(0, WindowSize);

  // turn the PID on
  myPID.Start(analogRead(data.temperature), setpoint, setpoint);

////////////////////////////////////////////////////////////////
  windowStartTimeH = millis();
  myPIDH.SetOutputLimits(0, WindowSizeH);

  // turn the PID on
  myPIDH.Start(analogRead(data.humidity), setpointH, setpointH);

}
```

Nota: En el setup de debe inicializar cada proceso a realizar.

En la Figura 40 se puede ver el proceso de loop para el caso de la temperatura.

Figura 40

loop para control PID de temperatura.

```
void loop() {
  while (1 == 1){

    TempAndHumidity data = dht.getTempAndHumidity();

    const double input = analogRead(data.temperature);
    const double output = myPID.Run(input);

  while (millis() - windowStartTime > WindowSize) {
    // time to shift the Relay Window
    windowStartTime += WindowSize;
  }
  if (output < millis() - windowStartTime)
    digitalWrite(releT, LOW);
  else
    digitalWrite(releT, HIGH);


  }
  //COMANDO PARA QUE NO SALTE NINGUN PERRO GUARDIAN
  vTaskDelay(100);
}
```

Nota: El proceso se realizará continuamente mientras el ESP32 permanezca encendido.

El proceso de loop0 para la humedad se visualiza en la Figura 41.

Figura 41

loop0 para control PID de humedad.

```
void loop0(void *parameter){
  while (1 == 1){

  TempAndHumidity data = dht.getTempAndHumidity();

    const double inputH = analogRead(data.humidity);
    const double outputH = myPIDH.Run(inputH);

  while (millis() - windowStartTimeH > WindowSizeH) {
    // time to shift the Relay Window
    windowStartTimeH += WindowSizeH;
  }
  if (outputH < millis() - windowStartTimeH)
    digitalWrite(releH, LOW);
  else
    digitalWrite(releH, HIGH);

  }
  //COMANDO PARA QUE NO SALTE NINGUN PERRO GUARDIAN
  vTaskDelay(100);
}
```

Nota: El proceso se realizará continuamente mientras el ESP32 permanezca encendido.

Se debe declarar la función para la lectura de los datos a recibir del sensor DTH22, esta variable de obtención será nuestra lectura analógica para obtener la diferencia entre el sensor DTH22 y el *setpoint,* así mismo se realiza un control *while* en el cual compara los tiempos de inicialización del proceso y la ventana de trabajo que anteriormente ya se definió.

De la misma manera se realiza un control *if*, en el cual se compara el tiempo de inicio del ESP32 menos la ventana de inicio superior a la salida, dependiendo de la comparación, se envía a encender o apagar el relé, por último, se declara el comando para que el perro guardián del ESP32 no imprima información de reinicio del módulo.

Cada proceso de control se realiza en paralelo para que haya un mejor control y no se lleguen a retrasar los otros procesos.

En la Figura 42 se puede observar el proceso de loop1 para el envío de datos.

Figura 42

loop1 para el envío de datos a la nube.

```
159  void loop1(void *parameter){
160    while (1 == 1){
161
162    TempAndHumidity data = dht.getTempAndHumidity();
163    generalhum.setValue(data.humidity);
164    generaltemp.setValue(data.temperature);
165
166    humvalor.setValue(data.humidity);
167    tempvalor.setValue(data.temperature);
168
169    ghum.addValue(0, data.humidity);
170    gtemp.addValue(0, data.temperature);
171
172    delay(1000);
173
174    }
175    //COMANDO PARA QUE NO SALTE NINGUN PERRO GUARDIAN
176    vTaskDelay(100);
177  }
```

Nota: El proceso se realiza continuamente mientras el ESP32 permanezca encendido.

En este proceso se declara en envío de datos a la nube de ThingSpeak y a la pantalla Nextion, la información se envía a la pantalla cada segundo, ya que se declara un “delay” cada mil milisegundos. De igual manera que en los loops mencionado antes se debe declarar un comando para que el perro guardián no se active y el ESP32 no comience a enviar errores por el puerto serial o llegar a reiniciarse.

Una vez se verifica la correcta compilación del código, ya se puede cargar sin errores al módulo ESP32. Este caso se resalta que al tener una cuenta gratuita en la plataforma de ThingSpeak solo recepta información cada 15 segundos.

5.3.2 Programación de pantalla Nextion.

Se hizo uso del programa Nextion Editor para la programación de la pantalla táctil la cual se usó en este proyecto, como se muestra a continuación en la Figura 43.

Figura 43

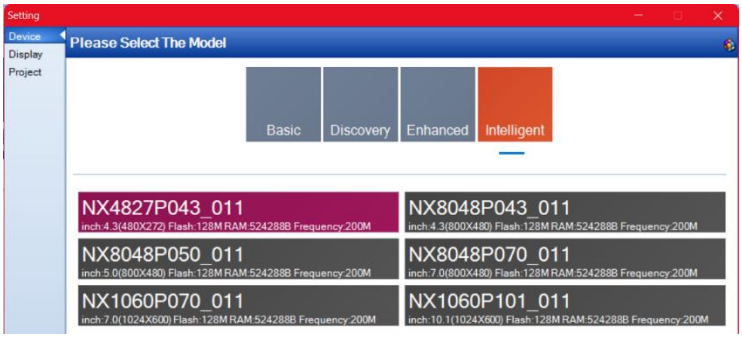
Nextion Editor.

Nota: Programa específico para la programación de pantallas Nextion.

Se abre el programa creamos un nuevo proyecto e identificamos el modelo de la pantalla la cual usaremos y su orientación, como se observa en la Figura 44.

Figura 44

Selección del modelo de pantalla



Nota: Se observa que existes varios modelos, en este caso se usó el modelo NX4827P043_011.

Es importante seleccionar el modelo adecuado, ya que al momento de cargar el archivo tft el cual se deberá cargar en la pantalla no lo hará de manera correcta si en un caso se ha seleccionado un modelo el cual no concuerde con el que se posee.

Seleccionado el modelo se procede a cargar en el programa todo material multimedia a usar, ya sea: fotos, videos, tipografía, etc. Se puede observar los archivos multimedia cargados en este proyecto en la Figura 45.

Figura 45

Carga de datos multimedia

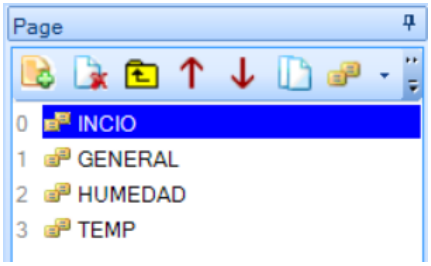
Nota: Se debe cargar cada tipo de archivo de manera individual.

Una vez se hayan cargado todos los datos, el programa le va a asignar un ID (se puede observar en la Figura 45, este ID es la dirección a la cual se llega información o se sabrá en que página de la programación estará.

Se añaden las páginas que se desea tener, en este proyecto se usaron 4, 2 para animación de entrada y vista general, 2 para vista de datos y gráfica por cada variable que se controló, se lo puede observar en la Figura 46.

Figura 46

Páginas de pantalla Nextion



Nota: Se pueden añadir las páginas que uno desee, pero siempre tener presente la memoria que se ocupará.

Se observa en la Figura 47 la animación principal al iniciar la pantalla Nextion.

Figura 47

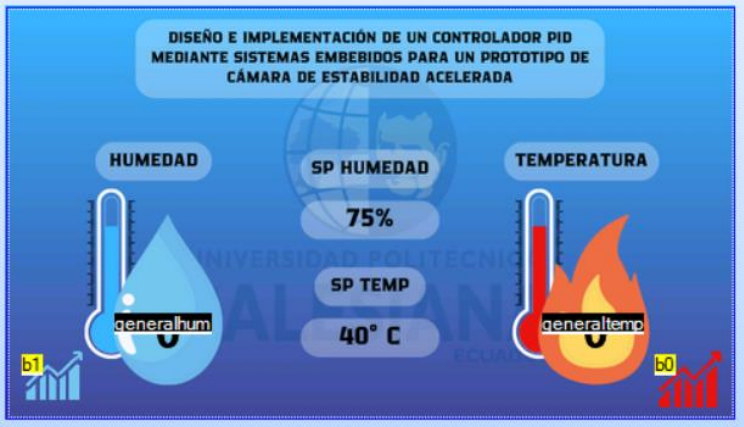
Animación de entrada

Nota: En esta imagen se menciona a los autores y docente tutor de este proyecto de titulación.

Se observa en la Figura 48 la vista general en la cual se encuentran los datos generales dentro de la cámara.

Figura 48

Vista general.



Nota: Se muestra el setpoint para la humedad y temperatura en tiempo real.

Una vista más detallada del control que se realiza a la humedad se muestra en la Figura 49.

Figura 49

Vista de humedad con gráfica.

Nota: Se muestra los setpoint para la humedad y grafica en tiempo real.

En la Figura 50 se visualiza el control que se realiza a la temperatura.

Figura 50

Vista de temperatura con gráfica.

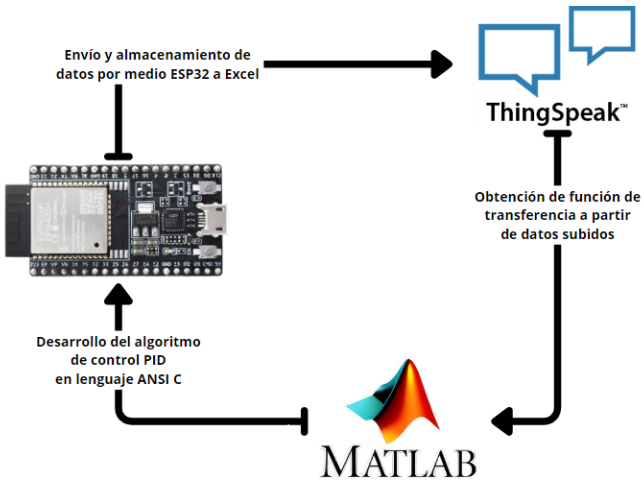
Nota: Se muestran los setpoint para la temperatura y gráfica en tiempo real.

5.4 Registrar la obtención de datos en tiempo real.

Los datos obtenidos llevarán un registro en la nube, a los cuales se podrán ingresar a visualizar en cualquier momento. La Figura 51 ayudará a visualizar mejor.

Figura 51

Esquema del control PID y la nube

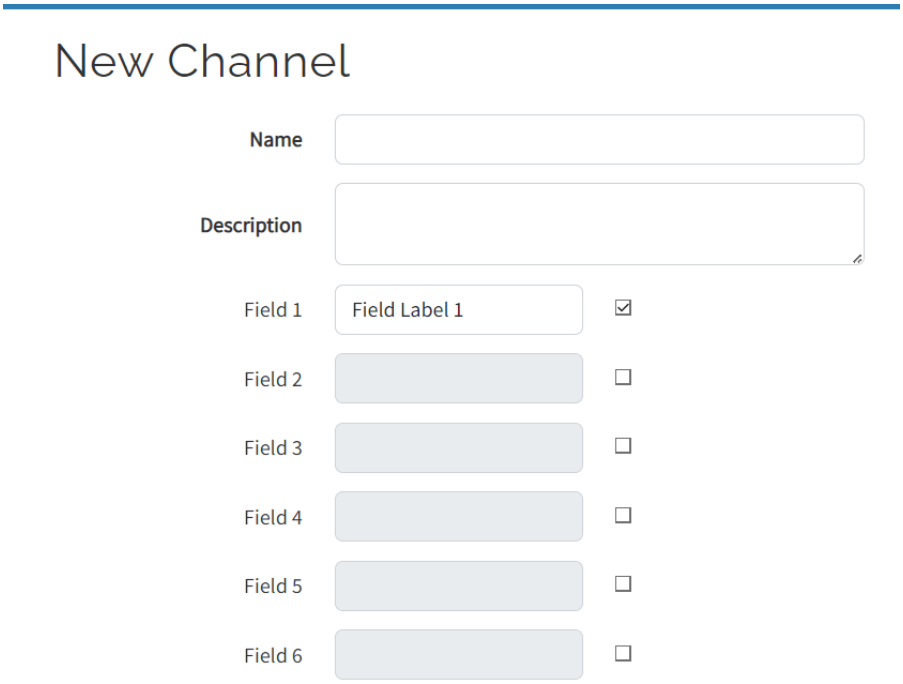


Nota: En la gráfica se puede observar el todo el proceso del control PID, con los datos enviándose a la nube.

Una vez creada la cuenta de ThingSpeak procedemos a crear un nuevo canal, para la adquisición de datos, se pude ver la creación del canal en la Figura 52.

Figura 52

Creación de canal



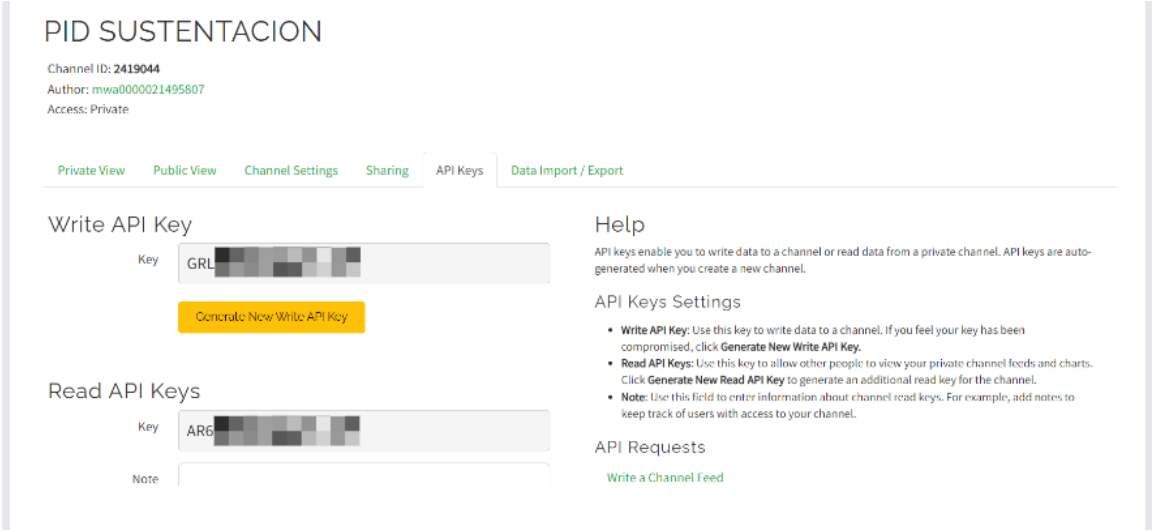
Nota: Se añade un field, ya que por defecto solo se crea uno.

Creado el nuevo canal, en la parte superior se encuentra la pestaña de API keys en la cual nos da unas credenciales alfanuméricas, ya sea para escritura o lectura de datos en ese canal, así mismo en la parte superior se encuentra un Channel ID, el cual es nuestra identificación, sin ella no se podría realizar el envío o lectura de datos, así se posea algún API keys.

Se puede observar mejor en la Figura 53.

Figura 53

Credenciales del channel



Nota: No se muestran las credenciales por seguridad.

Una vez se obtengan ambas credenciales, se colocan las mismas en el código antes mostrado, con ello en el código se puede enviar datos sin ninguna dificultad.

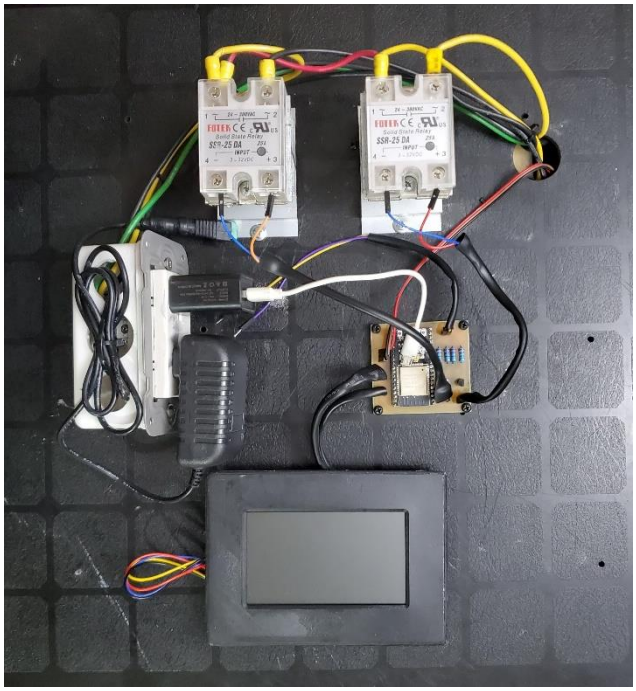
VI RESULTADOS

6.1 Prototipo de la cámara de estabilidad acelerada.

Con el cableado y conexiones finalizados, el control del sistema queda fijado en el prototipo, el cual se encuentra en la parte superior de éste. Se puede observar en la Figura 54:

Figura 54

Control del sistema

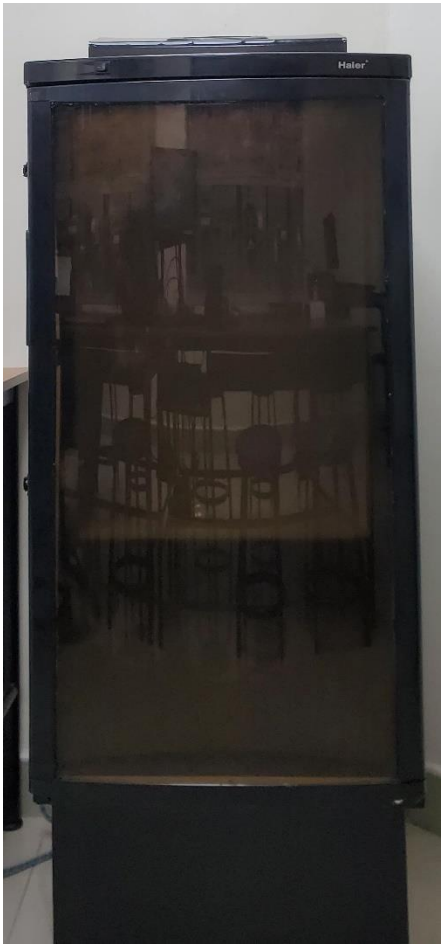


Nota: Imagen del control finalizado, donde se pueden observar las conexiones de los relés, tarjeta PCB y pantalla HMI.

Concluido con el diseño e implementación del prototipo, se procede a encender y poner en funcionamiento la cámara, conectando el cable de alimentación principal al tomacorriente y colocando en ON el switch. Se observa el diseño en la Figura 55.

Figura 55

Prototipo cámara de estabilidad acelerada



Nota: Finalizado el prototipo de la cámara de estabilidad acelerada.

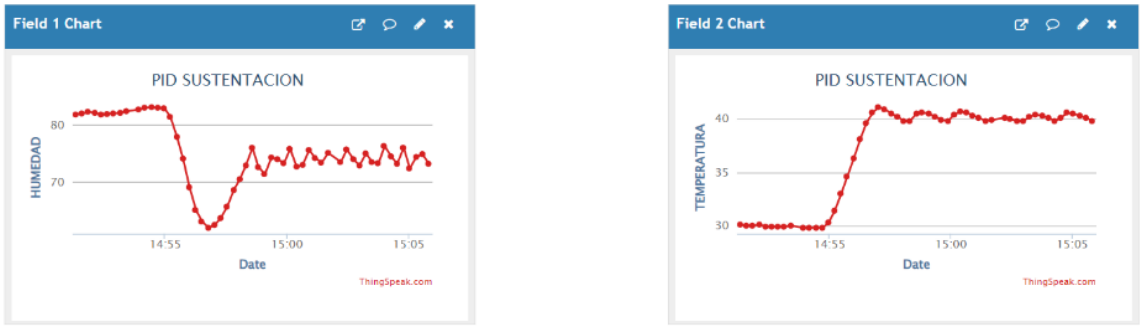
6.2 Datos obtenidos mediante el Control PID.

Se observa que se logra estabilizar a los puntos deseados, por ello se puede concluir que las constantes obtenidas de la función de transferencia para cada variable a controlar, encontrada por medio de la adquisición de datos en un control ON/OFF fueron correcta

Esto se puede confirmar en la Figura 56

Figura 56

Control PID



Nota: Grafica del control PID de humedad relativa y temperatura.

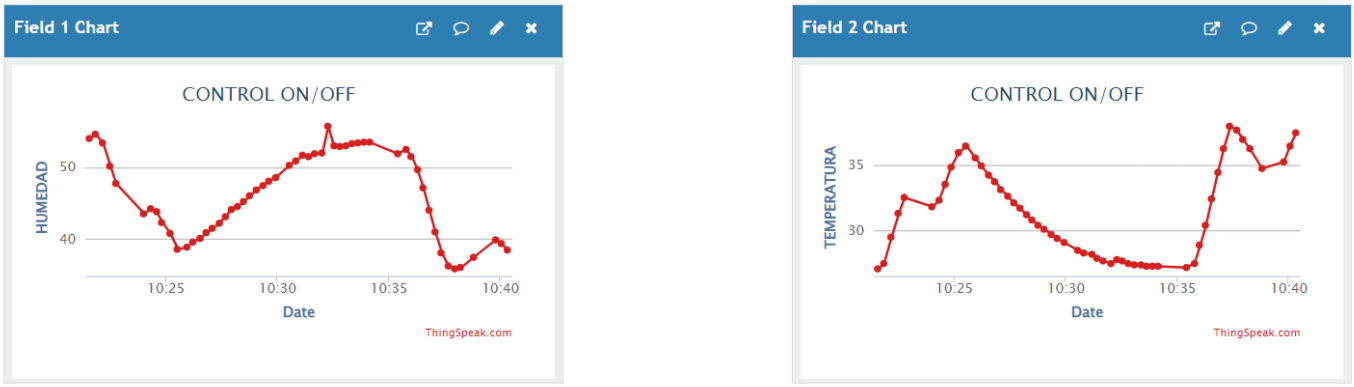
6.3 Comparación del sistema de control PID versus Control ON/OFF.

Se observa que se logra estabilizar a los puntos deseados, por ello se puede concluir que las constantes obtenidas de la función de transferencia para cada variable a controlar, encontrada por medio de la adquisición de datos en un control ON/OFF fueron correcta

Esto se puede confirmar en la Figura 57:

Figura 57

Primer diseño de software para control ON/OFF



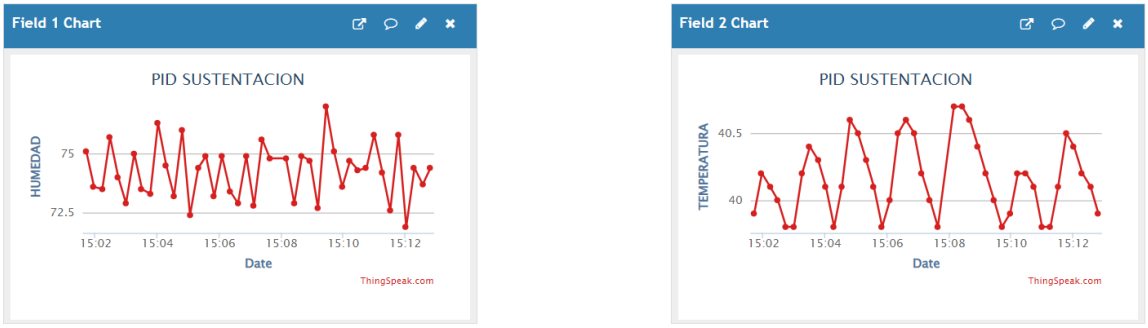
Nota: Gráfica de control ON/OFF.

Con los parámetros correspondientes a la temperatura y humedad relativa establecidos anteriormente: temperatura de 40 ºC ±2 ºC de error y una humedad relativa de 75% Hr ± 5% Hr de error. Se puede contrastar la diferencia entre un control ON/OFF y un control PID para cada variable. La estabilización es más eficiente, incluso al momento de una perturbación, ya que el control busca estabilizar el sistema en el menor tiempo posible y mantener un margen de error mínimo durante todo el funcionamiento del sistema.

Se pueden comparar ambas gráficas en la Figura 58.

Figura 58

Control PID para humedad y temperatura

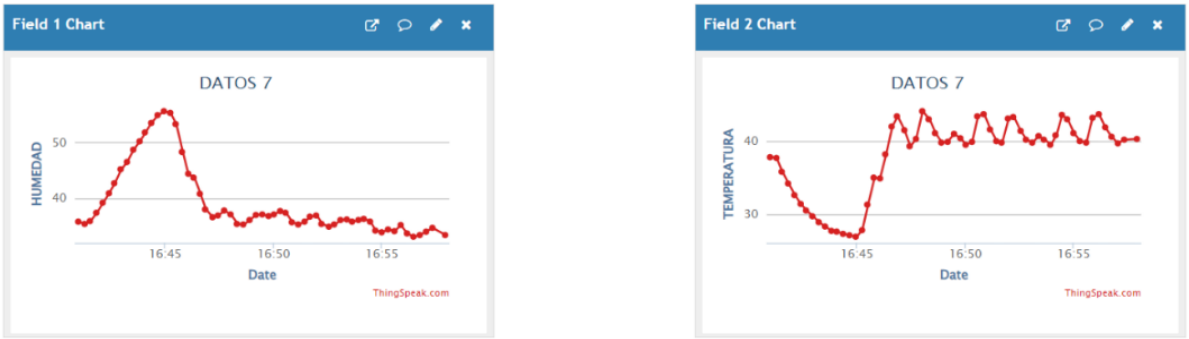


Nota: Gráfica de datos obtenidos con control PID.

Se visualiza la gráfica obtenida del control ON/OFF en la Figura 59.

Figura 59

Control ON/OFF



Nota: Gráfica obtenida con datos de control ON/OFF

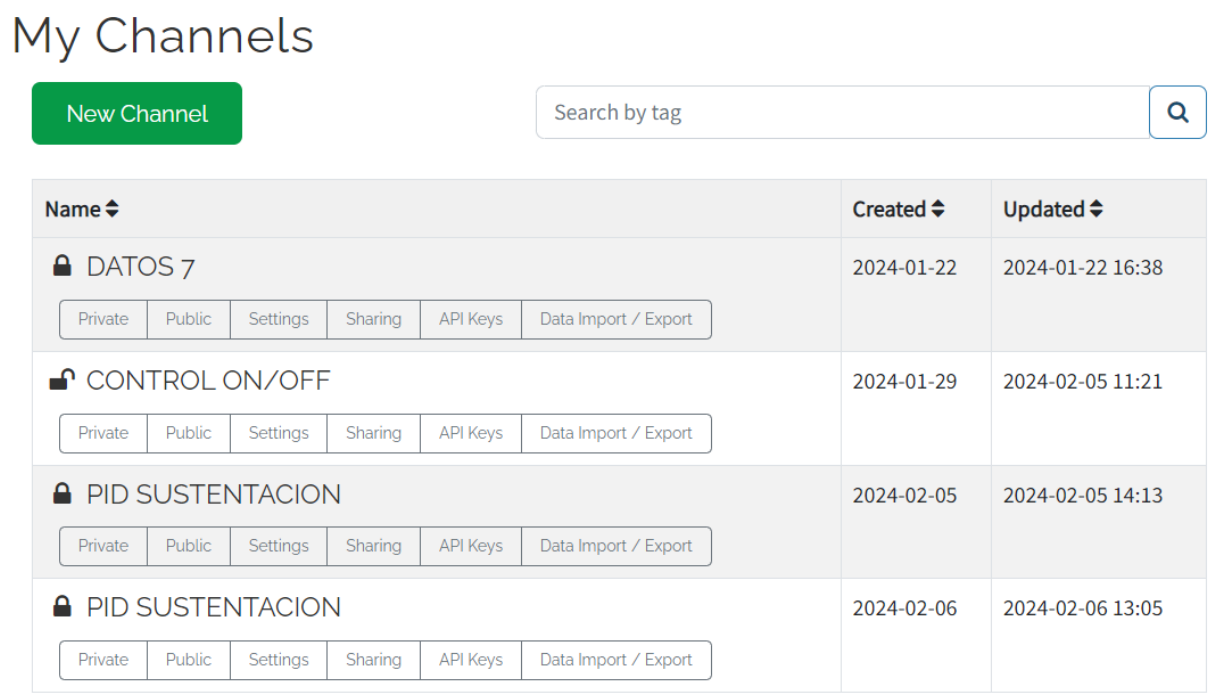
6.4 Datos mostrados en tiempo real.

El envío de datos se realiza con normalidad cada 15 segundos gracias a la plataforma de ThingSpeak, ya que la misma es muy estable y permite visualizar datos de manera constante, así mismo permite compartir con otras plataformas o páginas web para su visualización.

Los datos se pueden observar en la Figura 60.

Figura 60

Canales para recepción de datos.



Nota: Se observa la creación de diferentes canales para la recepción de datos.

VII CRONOGRAMA

En la tabla 1 se muestra el cronograma que tendrá 3 fases los cuales se basan en los puntos específicos.

Las fases inician desde el 27 de noviembre del presente año hasta el 29 de enero del 2024.

Tabla 1

Cronograma de actividades para el proyecto de tesis

ACTIVIDADES	DICIEMBRE				ENERO				FEBRERO	
	1	2	3	4	1	2	3	4	1	2
Diseñar el control PID de temperatura y humedad mediante sistemas embebidos.										
Realizar el prototipo de la cámara de estabilidad acelerada.										
Implementar el sistema de control.										
Registrar la obtención de datos en tiempo real.										
Comprobación de todo el sistema.										

VIII PRESUPUESTO

En la tabla 2 se muestra el presupuesto estimado para el proyecto.

Se estima que todos los precios mencionados son en base al costo del mercado ecuatoriano.

Tabla 2

Presupuesto referencial del diseño del proyecto de titulación

Cant. (uni)	Detalle	Valor unit. ($)	Valor total ($)
1	Prototipo de Cámara de Estabilidad Acelerada	60,00	60,00
1	Sistema Embebido	40,00	40,00
1	Humidificador	35,00	35,00
1	Sensor DTH22	10,00	10,00
1	Resistencia Seca	30,00	30,00
1	Pantalla Nextion	50,00	50,00
100	Horas de ingeniería	3,00	300,00
	Valor Total		525,00

IX CONCLUSIONES

Durante el desarrollo del prototipo se observó mediante varias pruebas que la ubicación recomendada de los elementos de generación de humedad y temperatura es en la parte inferior, porque tanto el calor como la humedad tienden a ascender, por lo cual el sensor de humedad y temperatura DTH22 fue colocado en la parte superior para garantizar la homogeneidad interna de la cámara, de tal forma que el sistema se mantenga estable a lo largo del tiempo con el control PID programado en el módulo ESP32.

El control PID para la temperatura tiene un tiempo de estabilización de menos de 3 minutos aproximadamente. Por otro lado, la estabilización del control PID para la humedad tiene un mayor tiempo de estabilización, alrededor de 7 a 9 minutos. Esto se debe a la sensibilidad que tiene la humedad respecto a la temperatura, ya que es muy volátil e inestable, a lo que lleva su oscilación dentro del margen de tolerancia de ± 5% Hr.

La adquisición de datos en tiempo real se visualiza correctamente en la plataforma de ThingSpeak, lo que permite que se pueda tener acceso a la información obtenida de la cámara en cualquier momento y cualquier lugar con acceso a internet, facilitando en gran medida el monitoreo y verificación de las variables de temperatura y humedad establecidas.

X RECOMENDACIONES

En caso de existir una repotenciación de este equipo se debe comprobar las conexiones realizadas previamente, ya que hay componentes que están en serie.

Comprobar que todos los componentes funcionen, tales como, el sensor DTH22, el módulo WiFi ESP32, la tarjeta electrónica PCB, pantalla HMI y los relés de estado sólido (SSR) y los cables.

Mantener una buena hermeticidad en el prototipo de la cámara de estabilidad acelerada para que no haya fugas de temperatura y humedad.

La recolección de datos con un periodo corto es muy importante, si se deseara realizar varios puntos de setpoint habría que generar una programación para cada setpoint, juntar esta data y mandar todo esto a Matlab.

Investigar y revisar que las librerías sean compatibles con los componentes a utilizar, debido a que algunas sí lo son para Arduino, pero no para el módulo WiFi ESP32.

XI BIBLIOGRAFÍA

A/S., E. (2023). *ellab.com*. Obtenido de https://www.ellab.com/es/sectores/farma/camara-de-estabilidad/

Alfaro, L. d. (20 de junio de 2006). *repositorio.uvg.edu.gt.* Obtenido de https://repositorio.uvg.edu.gt/xmlui/bitstream/handle/123456789/2722/Estudio%20de%20estabilidad%20acelerada%20de%20Melatonina%20-%20Leonel%20de%20Gandarias%2000564%20QF.pdf?sequence=1

ARDUINO.cl. (s.f.). Obtenido de ARDUINO.cl: https://arduino.cl/programacion/

BeijinUltrasonic. (21 de Abril de 2023). Obtenido de https://www.bjultrasonic.com/es/how-does-an-ultrasonic-humidifier-work/

BricoGeek. (s.f.). *tienda.bricogeek.com.* Obtenido de https://tienda.bricogeek.com/sensores-temperatura/1839-modulo-sensor-temperatura-y-humedad-dht22-am2302.html

Carranza, S. (27 de Octubre de 2021). *TodoMaker*. Obtenido de https://todomaker.com/blog/conociendo-al-esp32/

CASTAÑO, S. A. (s.f.). *CONTROL AUTOMATICO EDUCACION.* Obtenido de https://controlautomaticoeducacion.com/control-realimentado/control-pid-por-asignacion-de-polos/

Choez, R. S., & Franco, J. M. (2023). *Repositorio Institucional de la Universidad Politécnica Salesiana*. Obtenido de https://dspace.ups.edu.ec/bitstream/123456789/25005/1/UPS-GT004389.pdf

Company, L. (02 de Febrero de 2017). *laboratoriosentema.company.* Obtenido de https://laboratoriosentema.company/importancia-de-los-estudios-de-estabilidad/

controltecnica. (20 de mayo de 2020). *https://www.controltecnica.com*. Obtenido de https://www.controltecnica.com/test/producto/camaras-de-estabilidad-farmaceutica-serie-kbf/

dellamarca. (3 de junio de 2022). *FDM Environment Makers*. Obtenido de https://www.dellamarca.it/es/camaras-de-prueba-de-estabilidad/

Electricfor. (s.f.). *www.electricfor.es.* Obtenido de https://www.electricfor.es/es/18061/diccionario/Resistencia-electrica-calefactora.htm#:~:text=Dispositivo%20el%C3%A9ctrico%20cuyo%20objetivo%20es,por%20convecci%C3%B3n%2C%20conducci%C3%B3n%20o%20radiaci%C3%B3n.

ELECTRO STORE. (s.f.). *grupoelectrostore.com*. Obtenido de https://grupoelectrostore.com/shop/interruptores-y-switches/rele-estado-solido-ssr-25da-25a-input-3-32vdc-output-24-380vac/

ELECTRONICA UNIVERSAL DE MONTERREY, S. D. (s.f). *www.electronicauniversal.com.mx.* Obtenido de https://www.electronicauniversal.com.mx/2021/10/19/que-es-un-rele-de-estado-solido/

Ferndández, J. L. (s.f.). Obtenido de FISICALAB: https://www.fisicalab.com/apartado/resistencia-electrica-conductor

Group, P. (2023). *PQE Group*. Obtenido de https://blog.pqegroup.com/es-es/cumplimiento-gxp/importancia-de-un-estudio-de-estabilidad#:~:text=Los%20estudios%20de%20estabilidad%20acelerada,la%20calidad%20del%20producto%20era

H2O TEK. (s.f.). *humidificadores.mx*. Obtenido de https://humidificadores.mx/noticias/para-que-sirve-un-humidificador-ultrasonico/

hello auto. (s.f.). Obtenido de https://helloauto.com/: https://helloauto.com/glosario/ventilador

horecatiger. (s.f.). *horecatiger.eu.* Obtenido de https://horecatiger.eu/es-es/shop/resistencia-1600w-240v-espirales-1-l-68mm-an-173mm-420090

InnovacionDigital360. (22 de Septiembre de 2023). *Innovacion Digital 360.* Obtenido de https://www.innovaciondigital360.com/iot/sistemas-embebidos-que-son-y-para-que-se-utilizan/

leDuc, J. (21 de 12 de 2017). Obtenido de DigiKey: https://www.digikey.com/es/articles/transistor-basics

MathWorks. (s.f.). *MathWorks*. Obtenido de https://la.mathworks.com/help/control/ref/pidtuner-app.html

MINT FOR PEOPLE. (s.f.). *mint for people*. Obtenido de mintforpeople.com: https://mintforpeople.com/noticias/que-es-control-pid/

MISCHIANTI, R. (12 de diciembre de 2023). *Renzo Mischiati*. Obtenido de https://mischianti.org/doit-esp32-dev-kit-v1-high-resolution-pinout-and-specs/

monterrey, p. y. (s.f.). *panelyacanalados.com*. Obtenido de https://panelyacanalados.com/blog/acero-galvanizado-ventajas-y-usos/#:~:text=El%20acero%20galvanizado%20es%20un,material%20más%20duradero%20y%20resistente.

mpcontrol.es. (2022). *mpcontrol.es*. Obtenido de https://www.mpcontrol.es/equipos/camaras-climaticas-de-estabilidad-visitables-modulares-ich-fda/

NEXTION. (s.f.). Obtenido de nextion.tech/: https://nextion.tech/datasheets/NX4827P043-011C/

Nextion. (s.f.). *nextion.tech*. Obtenido de https://nextion.tech/intelligent-series-introduction/

Regalogar. (s.f.). *regalogar.es.* Obtenido de https://regalogar.es/humidificadores/14794-orbegozo-humidificador-hu-2050-52l-ultrasonico-vapor-frio-8435568400627.html

Robotics. (s.f.). *roboticsec.com.* Obtenido de https://roboticsec.com/producto/rele-de-estado-solido-ssr-60da-3-32vdc-60a/

S.L., M. d. (2024). *Rethinking The Future*. Obtenido de https://mintforpeople.com/noticias/que-es-control-pid/

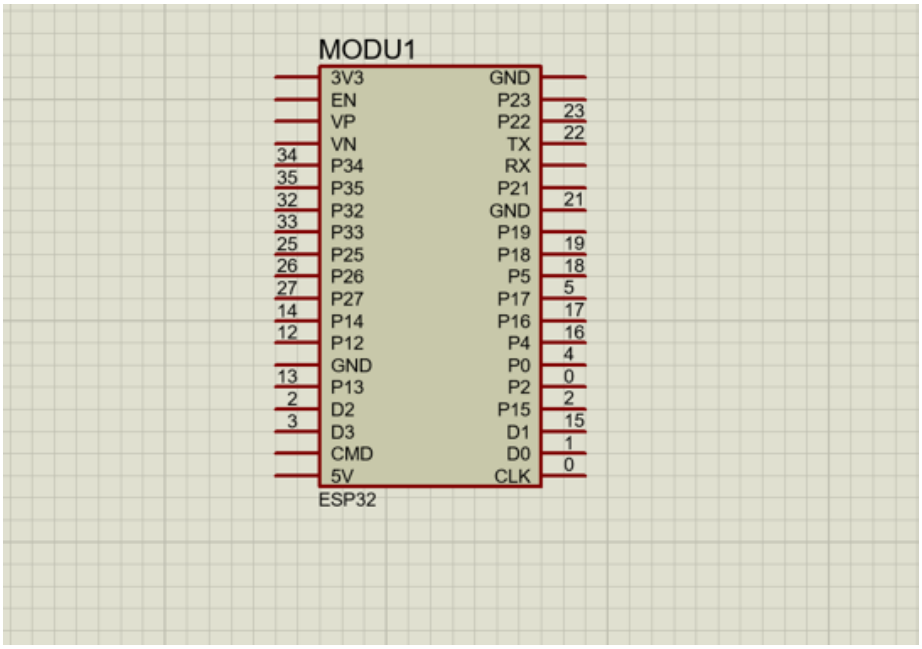
SigmaElectronica. (s.f.). *www.sigmaelectronica.net/.* Obtenido de https://www.sigmaelectronica.net/producto/dht22/

tiendamia. (s.f.). Obtenido de https://tiendamia.com/: https://tiendamia.com/ec/producto?amz=B07VFD6VNX&pName=GDSTIME%2090mm%20Fan%2024V,%2092mm%20x%2092mm%20x%2025mm%20Brushless%20Cooling%20Fan/&gclid=CjwKCAiAtt2tBhBDEiwALZuhAKYAdhABbei6-OJ0_hB5cUQzxIyRhjEVHX9Uq7PpSf7-gsaGU03kHBoCJLgQAvD_BwE

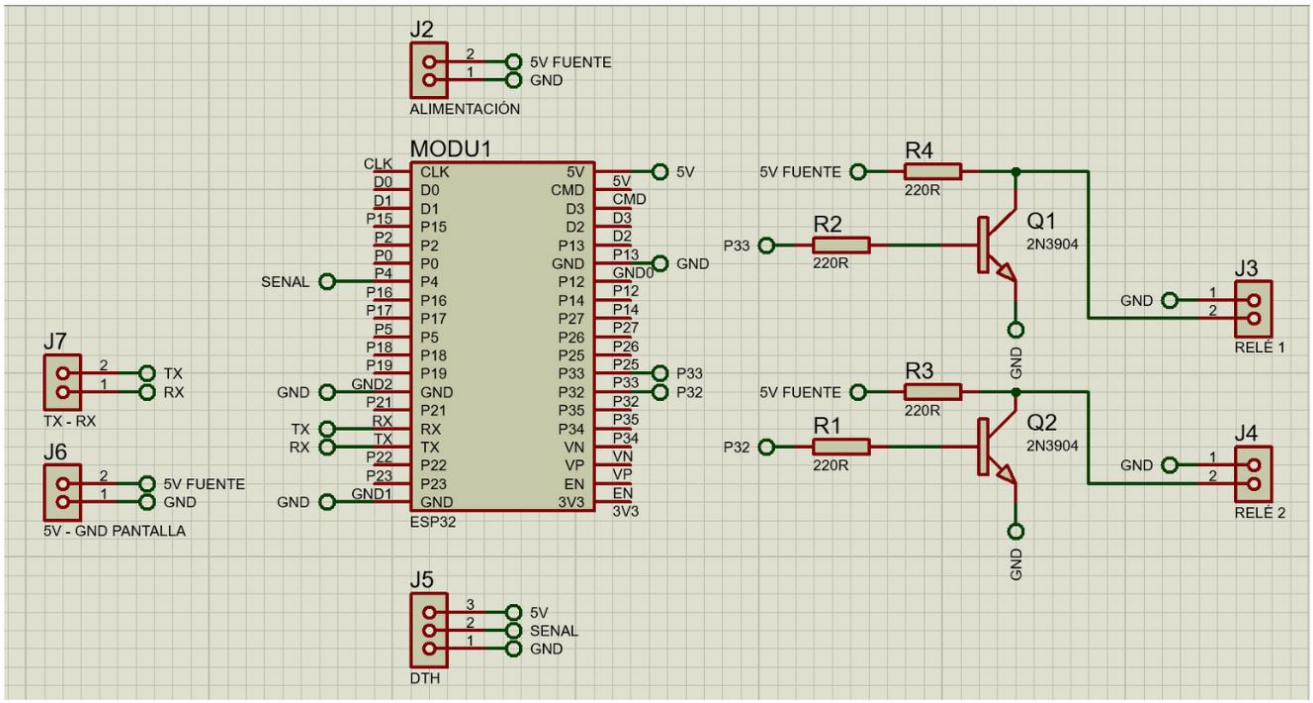
TRBL Services. (s.f.). Obtenido de trbl-services.eu: https://trbl-services.eu/blog-sistema-embebido-caracteristicas/

XII ANEXOS

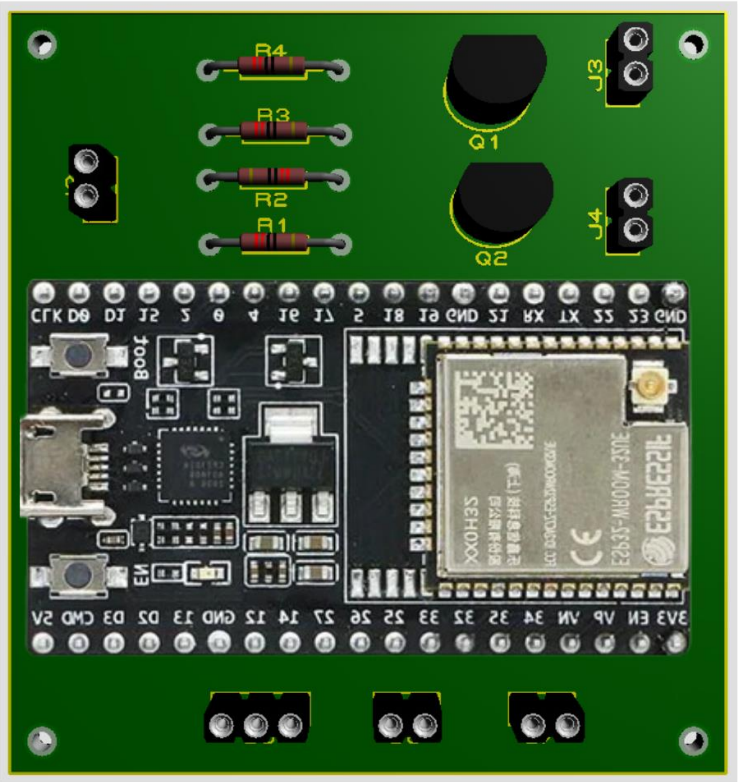
Creación del esquemático del módulo WiFi ESP32



Esquemático del control del sistema embebido



Modelado de tarjeta en 3D



Adquisición de datos de temperatura y humedad realizado en Arduino IDE

```
//INCLUIMOS LAS LIBRERIAS QUE USARAN EN ESTE PROYECTO DE TESIS

//LIBRERIA PARA EL SENSOR DTH
#include "DHTesp.h"

//LIBRERIA PARA COMUNICACION SERIAL (TX - RX)
#include <SoftwareSerial.h>

//DECLARAMOS LAS VARIBLES QUE SE USARÁN A LO LARGO DE LA PROGRAMACIÓN

//PIN DECLARADO COMO ENTRADA PARA LA LECTURA DEL SENSOR DTH22
int pinDHT = 4;

//SE USARÁ 2 PINES PARA LA ACTIVACIÓN DE LOS RELÉ, UNO PARA HUMEDAD Y EL OTRO
PARA TEMPERATURA
int releH = 32;
int releT = 33;
```

```
TaskHandle_t control_hum;
TaskHandle_t control_temp;
TaskHandle_t excel;
//TaskHandle_t patantalla;

DHTesp dht;

void setup() {
  //VELOCIDAD DE BAUDIOS POR SEGUNDO
  Serial.begin(115200);

  dht.setup(pinDHT, DHTesp::DHT22);

  //CONTROL DE TAREA EN PARALELO HUMEDAD
  xTaskCreatePinnedToCore(loop0,"control_hum",1000,NULL,1,&control_hum,1);

  //CONTROL DE TAREA EN PARALELO TEMPERATURA
  xTaskCreatePinnedToCore(loop1,"control_temp",1000,NULL,1,&control_temp,1);

  //CONTROL DE TAREA EN PARALELO DE ENVIO DE DATOS A EXCEL
  xTaskCreatePinnedToCore(loop2,"excel",1000,NULL,1,&excel,0);

  //CONTROL DE TAREA EN PARALELO DATOS A PANTALLA NEXTION
  //TaskCreatePinnedToCore(loop0,"patantalla",1000,NULL,1,&control_temp,0);

  pinMode (releH, OUTPUT);
  pinMode (releT, OUTPUT);
}


void loop() {
  //COMANDO PARA VISUALIZAR POR EL PUERTO SERIAL EN QUE NUMERO DE CORE ESTA
REALIZANDO LA TAREA
  //Serial.print("Proceso en Core #:" + String(xPortGetCoreID()));
  vTaskDelay(10);
}


void loop0(void *parameter){
  while (1 == 1){

  TempAndHumidity data = dht.getTempAndHumidity();
  if ( data.humidity < 73 ) {
```

```
    digitalWrite(releH, LOW); //ENCIENDE RELÉ

  }
  else if ( 73 < data.humidity && data.humidity < 75 ) {
    digitalWrite(releH, LOW); //ENCIENDE RELÉ
    delay(8500); // ESPERA UN SEGUNDO Y MEDIO

    digitalWrite(releH, HIGH); //APAGA RELÉ
    delay(1500); // ESPERA UN SEGUNDO
  }
  }
}

void loop1(void *parameter){
  while (1 == 1){

  TempAndHumidity data = dht.getTempAndHumidity();
  if ( data.temperature < 38 ) {
    digitalWrite(releT, LOW); //ENCIENDE RELÉ
    delay(2500); // ESPERA UN SEGUNDO Y MEDIO

    digitalWrite(releT, HIGH); //APAGA RELÉ
    delay(1500); // ESPERA UN SEGUNDO
  }
  else if ( 38 < data.temperature && data.temperature < 39 ) {
    digitalWrite(releT, LOW); //ENCIENDE RELÉ
    delay(1300); // ESPERA UN SEGUNDO Y MEDIO

    digitalWrite(releT, HIGH); //APAGA RELÉ
    delay(2500); // ESPERA UN SEGUNDO
  }

}
}

void loop2(void *parameter){
  while (1 == 1){
    //MOSTRAMOS LOS DATOS QUE NOS DÁ EL SENSOR DTH22
    Serial.print(String(data.humidity, 1));
    //SEPARAMOS LOS DATOS CON "," PARA REALIZAR LA SEPRACIÓN DE LOS MISMOS EN
EXCEL
    Serial.print(",");
    Serial.print(String(data.temperature, 2));
    Serial.print("\n");
    //SE DEFINE EL TIEMPO DE RECOLECCIÓN DE DATOS
```

```
    delay(333);

  }
}
```

Controlador PID en Arduino IDE

```
//INCLUIMOS LAS LIBRERIAS QUE USARAN EN ESTE PROYECTO DE TESIS

//LIBRERIA PARA EL SENSOR DTH
#include "DHTesp.h"

//LIBRERIA PARA COMUNICACION SERIAL (TX - RX)
#include <SoftwareSerial.h>

//LIBRERIA PARA CONTROL PID
#include <PID_v2.h>

//LIBRERIA PARA PANTALLA NEXTION
#include "Nextion.h"

//LIBRERIA PARA LA CONEXCIÓN CON EL THINGSPEAK
#include <ThingSpeak.h>


//DECLARAMOS LAS VARIBLES QUE SE USARÁN A LO LARGO DE LA PROGRAMACIÓN

//PIN DECLARADO COMO ENTRADA PARA LA LECTURA DEL SENSOR DTH22
int pinDHT = 4;

//SE USARÁ 2 PINES PARA LA ACTIVACIÓN DE LOS RELÉ, UNO PARA HUMEDAD Y EL OTRO
PARA TEMPERATURA
int releH = 32;
int releT = 33;

//CONSTANTES CHAR PARA LAS CREDENCIALES DE ACCESO WIFI
const char* ssid = "CELERITY_GIL VERA";
const char* password = "ISABELA7";

//CREDENCIALES PARA LA CONEXCIÓN CON EL THINGSPEAK
unsigned long channelID = 2419044;
const char* WriteAPIKey = "GRL9JNJXMSNIV2AL";
```

```
//DEFINIMOS EL CLIENTE DE WIFI QUE USAREMOS
WiFiClient cliente;

//INSTANCIA DEL SENSOR DTH22
DHTesp dht;

//DECLARAMOS LOS VALORES PARA CONTROL E INICIALIZAMOS EL CONTROL PID
//TEMP
double Kp = 1.075, Ki = 0.914, Kd = 0.154;
PID_v2 myPID(Kp, Ki, Kd, PID::Direct);

const int WindowSize = 255;
unsigned long windowStartTime;

double setpoint = 40.0;

////////////////////////////////////////////////

//HUMEDAD
double KpH = 1.285, KiH = 1.53, KdH = 0.16;
PID_v2 myPIDH(KpH, KiH, KdH, PID::Direct);

const int WindowSizeH = 5000;
unsigned long windowStartTimeH;

double setpointH = 75.0;


//CREAMOS EL OBEJTO PARA ENVIO DE DATOS A LA GRFICA
NexWaveform ghum = NexWaveform(0, 2, "ghum");
NexWaveform gtemp = NexWaveform(0, 1, "gtemp");

//CREAMOS OBJETOS PARA EL ENVIO DE DATOS
NexGauge generalhum = NexGauge(0, 1, "generalhum");
NexGauge generaltemp = NexGauge(0, 2, "generaltemp");
NexGauge humvalor = NexGauge(0, 4, "humvalor");
NexGauge tempvalor = NexGauge(0, 4, "tempvalor");

void setup() {

  //VELOCIDAD DE BAUDIOS POR SEGUNDO
  Serial.begin(115200);

  //INICIALIZAMOS PANTALLA NEXTION
  nexInit();
```

```
  //INICIALIZAMOS EL DTH
  dht.setup(pinDHT, DHTesp::DHT22);

  //DECLARA AL PIN COMO SALIDA
  pinMode(releH, OUTPUT);
  pinMode(releT, OUTPUT);

  TempAndHumidity data = dht.getTempAndHumidity();

  windowStartTime = millis();
  myPID.SetOutputLimits(0, WindowSize);

  // turn the PID on
  myPID.Start(analogRead(data.temperature), setpoint, setpoint);

  ////////////////////////////////////////////////////////////////
  windowStartTimeH = millis();
  myPIDH.SetOutputLimits(0, WindowSizeH);

  // turn the PID on
  myPIDH.Start(analogRead(data.humidity), setpointH, setpointH);
}

void loop() {
  while (1 == 1) {

    TempAndHumidity data = dht.getTempAndHumidity();

    const double input = analogRead(data.temperature);
    const double output = myPID.Run(input);

    while (millis() - windowStartTime > WindowSize) {
      // time to shift the Relay Window
      windowStartTime += WindowSize;
    }
    if (output < millis() - windowStartTime)
      digitalWrite(releT, LOW);
    else
      digitalWrite(releT, HIGH);
  }
  //COMANDO PARA QUE NO SALTE NINGUN PERRO GUARDIAN
  vTaskDelay(100);
}
```

```
void loop0(void* parameter) {
  while (1 == 1) {

    TempAndHumidity data = dht.getTempAndHumidity();

    const double inputH = analogRead(data.humidity);
    const double outputH = myPIDH.Run(inputH);

    while (millis() - windowStartTimeH > WindowSizeH) {
      // time to shift the Relay Window
      windowStartTimeH += WindowSizeH;
    }
    if (outputH < millis() - windowStartTimeH)
      digitalWrite(releH, LOW);
    else
      digitalWrite(releH, HIGH);
  }
}

void loop1(void* parameter) {
  while (1 == 1) {

    TempAndHumidity data = dht.getTempAndHumidity();
    generalhum.setValue(data.humidity);
    generaltemp.setValue(data.temperature);

    humvalor.setValue(data.humidity);
    tempvalor.setValue(data.temperature);

    ghum.addValue(0, data.humidity);
    gtemp.addValue(0, data.temperature);

    delay(1000);
  }
}
```