Universidad de Jaén
Escuela Politécnica Superior de Jaén

CONTROL DE TEMPERATURA DE UNA VIVIENDA MEDIANTE SUELO RADIANTE

Autor: Facundo Vega Montoya

Grado: Ingeniería Electrónica Industrial

Director: Elisabet Estévez Estévez

Departamento del director: Departamento de Ingeniería Electrónica y Automática

Fecha: 02/07/2024

Universidad de Jaén
Escuela Politécnica Superior de Jaén
Departamento de Informática

Don Elisabet Estévez Estévez, tutor del Proyecto Fin de Carrera titulado: Control de temperatura de una vivienda mediante suelo radiante, que presenta Facundo Vega Montoya, autoriza su presentación para defensa y evaluación en la Escuela Politécnica Superior de Jaén.

Jaén, Julio de 2024

El alumno: Los tutores:

ESTEVEZ ESTEVEZ ELISABET - 72578245C
Firmado digitalmente por ESTEVEZ ESTEVEZ ELISABET - 72578245C
Fecha: 2024.07.02 17:39:01 +02'00'

SANCHEZ GARCIA ALEJANDRO - 77321466C
Firmado digitalmente por SANCHEZ GARCIA ALEJANDRO - 77321466C
Fecha: 2024.07.02 23:19:07 +02'00'

FACUNDO VEGA MONTOYA Elisabet Estévez Estévez,
Alejando Sánchez García

Contenido

Índice de Figuras

1. Introducción.

1.1. Contexto y Motivación del proyecto.

El control de temperatura es crucial en diversos procesos industriales, ya que afecta directamente la calidad, seguridad y eficiencia de la producción. En sectores como la manufactura química, la industria alimentaria, la biotecnología y la gestión de sistemas HVAC (calefacción, ventilación y aire acondicionado), mantener una temperatura constante y precisa es esencial.

En la manufactura química, las reacciones deben realizarse a temperaturas específicas para obtener productos de alta calidad y rendimientos óptimos. La precisión en el control de temperatura influye en la velocidad de reacción, la selectividad de los productos y la seguridad del proceso.

En la industria alimentaria, el control de temperatura garantiza la calidad y seguridad de los productos durante la producción, almacenamiento y transporte. Procesos como la pasteurización, esterilización y refrigeración son fundamentales para preservar la integridad de los alimentos y prevenir el crecimiento de microorganismos patógenos.

En sistemas HVAC, el control de temperatura es esencial para mantener ambientes confortables y seguros en edificios y vehículos. Los sistemas de control aseguran que las temperaturas interiores se mantengan dentro de los rangos deseados, optimizando el consumo de energía y mejorando la eficiencia operativa. La capacidad de controlar la temperatura de manera precisa y eficiente contribuye a la sostenibilidad ambiental al reducir el consumo de energía y las emisiones de gases de efecto invernadero.

En el ámbito educativo, disponer de sistemas de entrenamiento que simulan entornos de control industrial permite a los estudiantes adquirir habilidades prácticas y comprender mejor los principios teóricos.

Este proyecto tiene una doble motivación: primero, proporcionar una herramienta didáctica avanzada para la formación en control de procesos; y segundo, ofrecer una solución eficiente y económica para aplicaciones industriales que requieren control de temperatura. En un mundo donde la precisión y la eficiencia son cada vez más valoradas, este sistema puede ser una pieza clave para diversos sectores.

El enfoque se centra en la integración de un microcontrolador con el equipo de control de procesos de temperatura ROCON 38.002(véase Figura 1). Se eligió el ESP32 por su versatilidad y capacidad de conectividad, lo que lo convierte en la plataforma ideal para desarrollar un sistema de control accesible y potente. Este microcontrolador no solo es conocido por su capacidad de procesamiento y flexibilidad, sino también por su capacidad de conectarse a redes Wi-Fi y Bluetooth, lo que facilita la implementación de soluciones IoT.

Figura 1 : Estación ROCON 38.002 original.

La posibilidad de controlar y monitorizar el sistema a través de una interfaz web añade un valor significativo. Esta característica permite la supervisión remota y ajustes en tiempo real, alineándose perfectamente con las demandas modernas de conectividad y automatización. Imagina poder ajustar y controlar los procesos de temperatura desde cualquier parte del mundo, asegurando así la máxima eficiencia y respuesta inmediata a cualquier fluctuación o necesidad de ajuste.

Además, la implementación de este sistema tiene un impacto positivo en el ámbito educativo. Los estudiantes tendrán la oportunidad de interactuar con tecnologías de vanguardia, preparándose mejor para los desafíos del mercado laboral actual. La experiencia práctica adquirida a través de este sistema de control de temperatura no solo fortalecerá sus conocimientos teóricos, sino que también les

permitirá desarrollar competencias críticas en resolución de problemas y adaptación a nuevas tecnologías.

Desde el punto de vista industrial, este sistema ofrece una solución rentable y eficiente. La capacidad de integrar el control de temperatura en diversos procesos industriales puede llevar a una reducción significativa de costos operativos y a una mejora en la calidad del producto final. Las empresas podrán beneficiarse de una mayor precisión en sus procesos, lo que se traduce en una mayor competitividad en el mercado.

En resumen, este proyecto no solo aborda una necesidad técnica, sino que también ofrece un enfoque innovador y práctico que beneficia tanto a estudiantes como a profesionales de la industria. Con la integración estamos creando una herramienta que es tanto educativa como aplicable en el mundo real, abriendo nuevas posibilidades para el control de temperatura en una variedad de entornos. La combinación de accesibilidad, conectividad y eficiencia convierte a este proyecto en una solución ideal para los desafíos actuales y futuros en el control de temperatura.

1.2. Objetivos.

El principal objetivo de este proyecto es desarrollar un sistema de control de temperatura utilizando un ESP32 en combinación con el equipo ROCON. Este ambicioso objetivo se desglosa en varias metas específicas que aseguren la eficiencia, precisión y utilidad del sistema tanto en entornos educativos como industriales.

- Desarrollo de un Sistema de Control de Temperatura:
  El proyecto busca implementar un sistema robusto y preciso, empleando el microcontrolador ESP32 y el equipo ROCON. La integración de un controlador PID permitirá mantener la temperatura en valores específicos, garantizando estabilidad y exactitud en diversas condiciones operativas.
- Integración Eficiente de Hardware y Software:
  Uno de los pilares del proyecto es la perfecta sincronización entre el hardware y el software. El ESP32 deberá interactuar de manera eficiente con los actuadores del equipo ROCON, optimizando el control de temperatura mediante una comunicación fluida y efectiva entre todos los componentes del sistema.

- Desarrollo de una Interfaz Web Intuitiva:
  Para potenciar la accesibilidad y el control remoto, se creará una interfaz web intuitiva. Esta plataforma permitirá a los usuarios monitorizar y ajustar el sistema de temperatura de forma sencilla. La utilización de WebSockets garantizará una comunicación bidireccional rápida y fiable entre el ESP32 y los usuarios, facilitando la supervisión remota y los ajustes en tiempo real.
- Pruebas y Validación Exhaustivas:
  Asegurar la robustez y precisión del sistema es crucial. Se realizarán pruebas exhaustivas para evaluar el rendimiento del sistema bajo diversas condiciones operativas. Esta fase es esencial para confirmar la aplicabilidad del sistema en entornos reales y garantizar que cumple con los estándares de calidad requeridos.
- Elaboración de Documentación Detallada:
  Para asegurar que el proyecto pueda ser replicado y extendido por otros estudiantes y profesionales, se desarrollarán manuales de usuario y guías detalladas. Esta documentación incluirá toda la información necesaria para la implementación, configuración y uso del sistema, facilitando su adopción y adaptación en diferentes contextos.

1.3. Estructura del proyecto

La estructura del proyecto está diseñada para abordar de manera integral cada uno de los objetivos propuestos, asegurando un enfoque sistemático y coherente en el desarrollo y presentación del sistema de control de temperatura. A continuación, se detalla cómo se organizará el contenido del proyecto:

Componentes Hardware.

Esta sección trata sobre la renovación y mejora integral de una estación de control de temperatura, transformándola con tecnología avanzada. Se detallan los cambios significativos implementados para optimizar el rendimiento del sistema y garantizar su eficiencia operativa.

Diseño y Desarrollo del Sistema de Control.

Esta sección aborda el diseño y desarrollo de un avanzado sistema de control de temperatura, centrado en la implementación de un controlador para mantener la estabilidad y precisión. Se explica cómo se adquieren los datos de temperatura mediante sensores de temperatura y cómo se procesan estos datos para regular la temperatura deseada. Además, se describe la configuración y ajuste de los parámetros PID, esenciales para el rendimiento óptimo del sistema, y la visualización de información crítica a través de una interfaz gráfica.

Interfaz web de usuario.

Esta sección trata el desarrollo de una interfaz web de usuario basada en WebSockets para el monitoreo y control en tiempo real del sistema. Se explica la inicialización del servidor WebSocket, la gestión de mensajes entrantes y salientes, y la configuración del servidor para distribuir archivos HTML, CSS y JavaScript.

Caso de Uso.

Esta sección ilustra un marco detallado para el empleo integrado de hardware y software en el control y monitoreo del sistema, adaptado a las necesidades modernas de precisión y accesibilidad. Incluye un código QR que permite observar la estación en funcionamiento, demostrando su operatividad.

Conclusiones.

En esta sección, se resume el proyecto y destacan los principales inconvenientes que se tuvieron a la hora de llevar lo a cabo, también se proponen posibles mejoras de cara al futuro.

Referencias.

Se presentará un listado completo de todas las fuentes bibliográficas y recursos utilizados.

2. Componentes Hardware.

En este apartado se detalla la completa renovación y mejora de la estación de control de temperatura, destacando cada uno de los aspectos técnicos que fueron abordados. Este proceso implicó una serie de modificaciones significativas para optimizar el rendimiento del sistema y asegurar su eficiencia operativa. A continuación, se explican las mejoras implementadas, desde la actualización de componentes clave hasta la incorporación de nuevas tecnologías avanzadas, todo con el objetivo de revitalizar esta importante estación y adaptarla a las exigencias modernas.

2.1. Proceso: ROCON 38.002.

El funcionamiento de la estación se basa en dos circuitos de agua. El circuito primario está conectado a un tanque con una resistencia que calienta el agua, y una bomba para hacer que esta circule. Por otro lado, el circuito secundario también tiene una bomba para circular el agua y un ventilador para enfriarla. Finalmente, ambos circuitos de agua se unen en un intercambiador de calor. Este dispositivo permite transferir calor entre los dos circuitos sin que se mezclen físicamente. Así, el agua caliente del circuito primario cede su calor al agua del circuito secundario sin que haya contacto directo entre ellas.

El equipo se encontraba en desuso, lo que implicaba una serie de desafíos significativos. La estructura del equipo, aunque robusta, mostraba signos de deterioro y obsolescencia en varios de sus componentes.

Se pudo conservar el tanque de agua(véase Figura 2) cuya resistencia interna resulta crucial para el calentamiento de la misma. Su estructura robusta y su capacidad de calentamiento eficiente lo hicieron un candidato ideal para ser retenido.

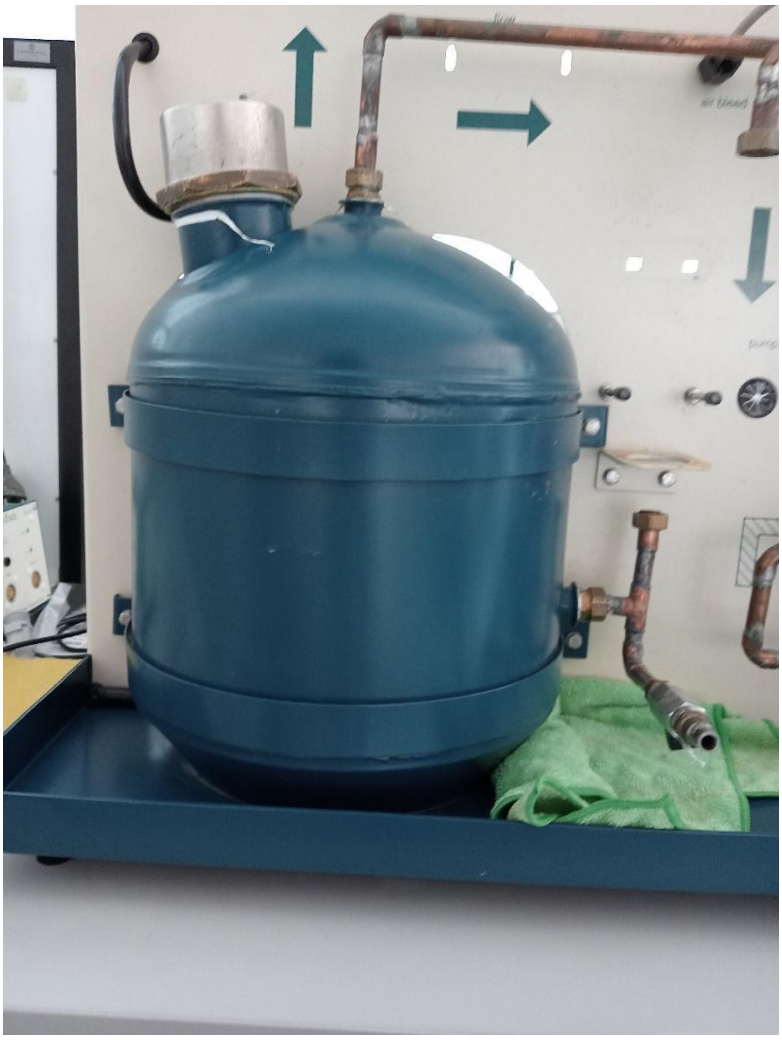

Figura 2 : Tanque de agua con resistencia.

También se pudo conservar el intercambiador de calor(véase Figura 3), elemento que conecta el circuito primario y vital para aumentar la temperatura que deseamos controlar.

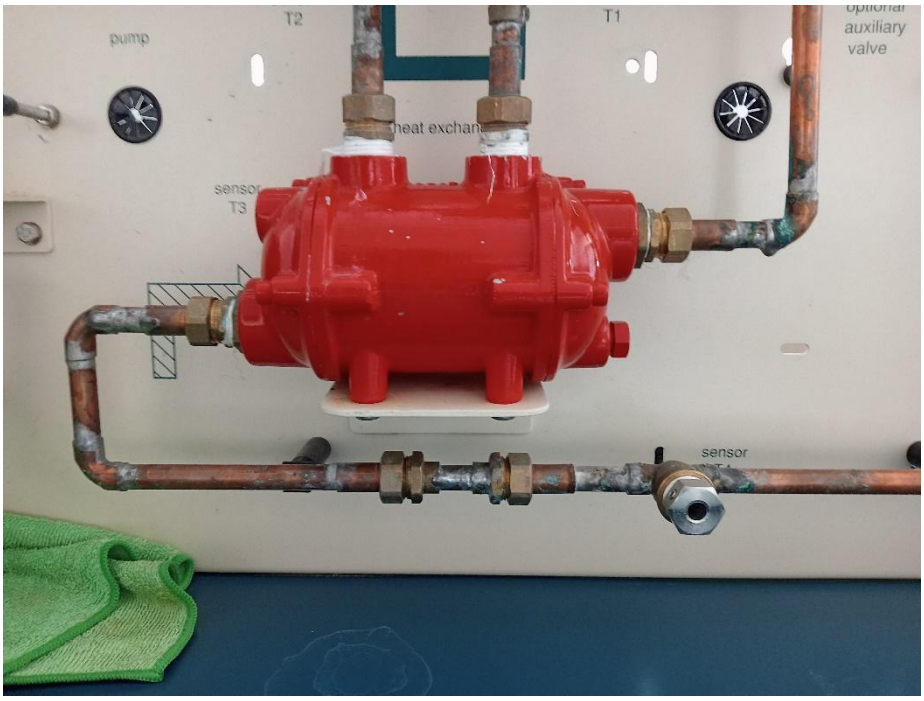

Figura 3 : Intercambiador de calor.

Las tuberías originales del equipo estaban deterioradas y no cumplían con los estándares de durabilidad y conductividad necesarios.

Se optó por reemplazarlas con tuberías de cobre debido a sus excelentes propiedades de conductividad térmica y su mayor durabilidad. Este cambio asegura un flujo de calor más eficiente y una vida útil más larga para el sistema.

El ventilador original de la estación no era lo suficientemente potente por lo que se decidió aislarlo y en su lugar colocar la estación ROCON 38.610(véase Figura 4) para un mejor enfriamiento debido al tamaño del ventilador y aprovechar la bomba para una circulación mayor por el circuito secundario.

Figura 4 : Estación ROCON 38.610 reacondicionada.

2.2. Sensor: DS18b20.

El DS18B20(véase Figura 5) es un sensor de temperatura digital que utiliza un protocolo de comunicación que permite que haya múltiples sensores conectados a un único pin del microcontrolador lo que facilita la implementación y ofrecen una medición precisa de la temperatura mejorando la exactitud y la eficiencia del sistema de monitoreo y control([3]).

Los sensores de temperatura originales eran obsoletos por lo que se optó por implementar los 18B20.

Figura 5 : Sensor de temperatura DS18B20.

2.3. Actuador: Bomba de agua.

Una bomba es un dispositivo mecánico que se utiliza para mover líquidos de un lugar a otro para crear con presión un flujo.

Las bombas originales del equipo estaban sin funcionar, lo que comprometía el flujo adecuado del agua, por lo que se instalaron nuevas bombas, mostrada en la Figura 6, para garantizar un flujo constante y adecuado, asegurando el correcto funcionamiento del sistema de control de temperatura.

Figura 6 : Bomba de agua del circuito secundario.

Se agregaron relés de estado sólido(véase Figura 7) para controlar el encendido y apagado de la bomba y ventilador con el microcontrolador.

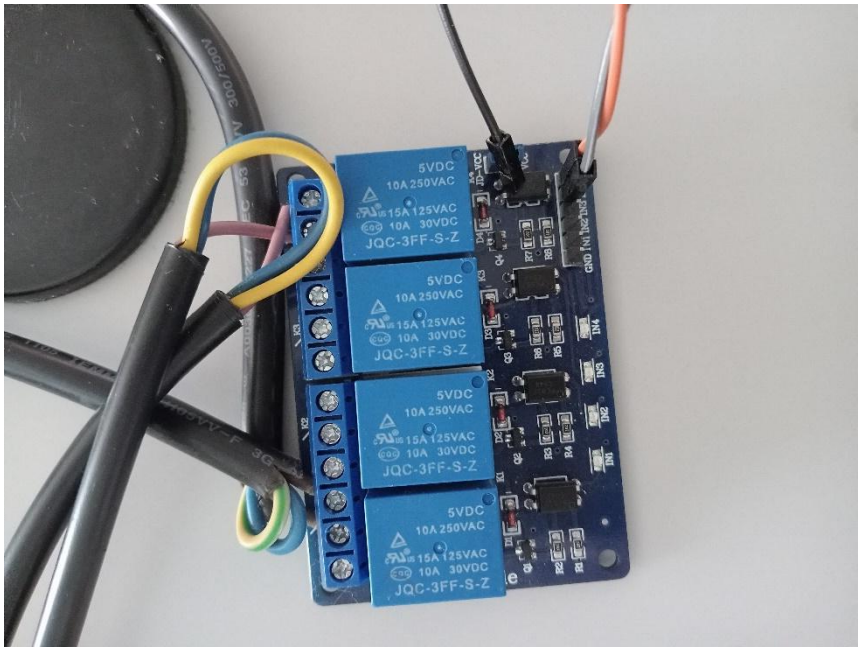

Figura 7 : Relé de estado sólido.

2.4. Microcontrolador: ESP32.

Los microcontroladores son dispositivos integrados que combinan un procesador, memoria y periféricos de entrada/salida en un solo chip, diseñados para tareas de control específicas en sistemas embebidos. Son esenciales para el desarrollo de sistemas que requieren control preciso y capacidad de procesamiento en tiempo real, como los sistemas de control de temperatura.

El ESP32 es un microcontrolador versátil y potente(véase Figura 8), ampliamente utilizado en aplicaciones de Internet de las Cosas (IoT) debido a sus características avanzadas y su integración de conectividad inalámbrica([1]):

Conectividad: El ESP32 cuenta con capacidades de conectividad Wi-Fi y Bluetooth, lo que permite una fácil integración en redes y la posibilidad de comunicación inalámbrica con otros dispositivos. Esto es crucial para aplicaciones IoT que requieren monitoreo y control remoto.

Capacidad de procesamiento: Con un procesador dual-core y velocidades de reloj de hasta 240 MHz, el ESP32 es capaz de manejar tareas complejas y exigentes, como el procesamiento de datos en tiempo real y la ejecución de algoritmos de control.

Bajo consumo de energía: Diseñado para aplicaciones IoT, el ESP32 ofrece modos de bajo consumo de energía, lo que es crucial para dispositivos que operan con baterías y requieren una operación prolongada sin intervención.

Versatilidad: El ESP32 soporta múltiples interfaces de comunicación como SPI, I2C, UART, y ADC, lo que facilita su integración con una variedad de sensores y actuadores, permitiendo una implementación flexible y adaptable a diferentes necesidades.

El ESP32 se puede programar utilizando C++ a través de entornos de desarrollo como PlatformIO, que ofrece una integración robusta con Visual Studio Code. PlatformIO facilita la gestión de bibliotecas, la configuración del entorno de desarrollo y la depuración del código.

C++ es un lenguaje de programación poderoso y flexible, ampliamente utilizado en el desarrollo de sistemas embebidos.

PlatformIO es una plataforma de desarrollo para sistemas embebidos que soporta múltiples placas y frameworks. Ofrece herramientas avanzadas para la

depuración, pruebas y monitoreo del hardware, además de una integración perfecta con entornos de desarrollo como Visual Studio Code.

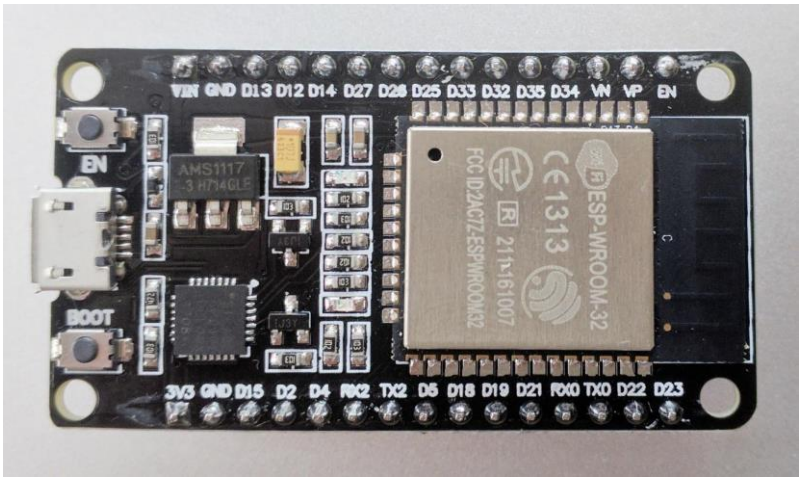

Figura 8 : Microcontrolador ESP32.

Estas características hacen del ESP32 una plataforma ideal para el desarrollo de un sistema de control de temperatura que requiere conectividad, procesamiento en tiempo real y eficiencia energética. Su capacidad para manejar múltiples tareas y comunicarse de manera eficiente con otros dispositivos permite el desarrollo de sistemas complejos y robustos.

2.5. Pantalla TFT 1.8 SPI.

Las pantallas TFT de 1.8 pulgadas con interfaz SPI(véase Figura 9) son ampliamente utilizadas en proyectos de electrónica debido a su tamaño compacto, bajo consumo de energía y facilidad de uso. Este tipo de pantalla ofrece una solución versátil y eficiente para la visualización([2]).

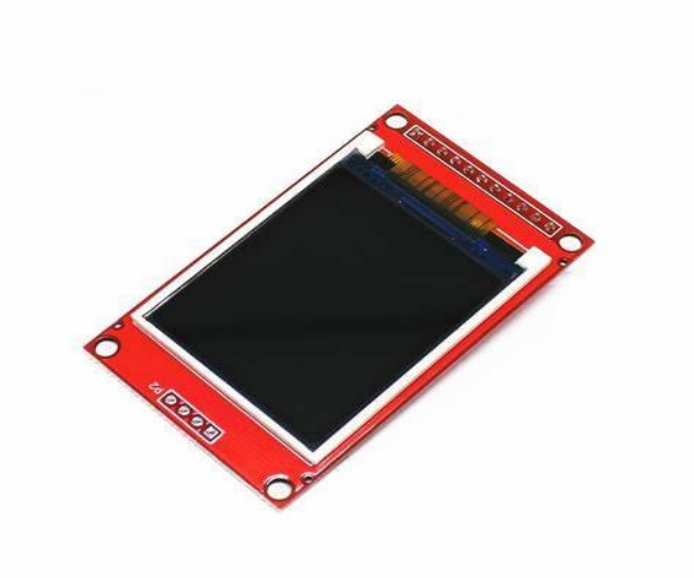

Figura 9 : Pantalla TFT 1.8" SPI

La pantalla TFT 1.8" SPI utiliza la interfaz SPI (Serial Peripheral Interface) para la comunicación con otros dispositivos como Arduino o Raspberry Pi. Esta interfaz proporciona una comunicación rápida y eficiente, permitiendo una actualización fluida de la pantalla.

3. Diseño y desarrollo del sistema de control.

El diseño y desarrollo del sistema de control se enfoca en la implementación de un controlador PID para regular la temperatura([9]). Este proceso incluye la adquisición de datos de temperatura, la implementación del controlador PID, y la visualización de información crítica a través de una interfaz gráfica siguiendo el flujograma de la Figura 10.

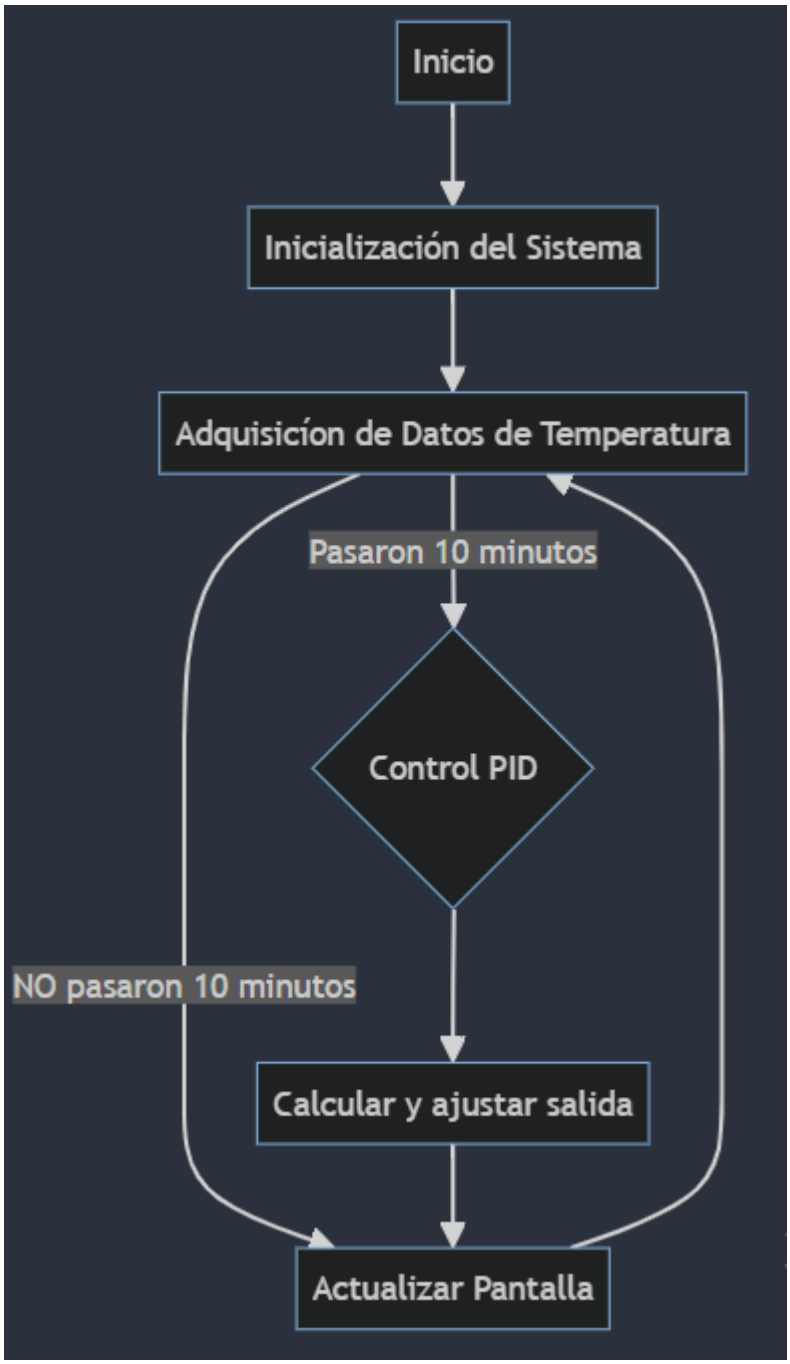

Figura 10 : Flujograma del sistema de control.

3.1. Adquisición de datos.

La adquisición de datos de temperatura es crítica para la efectividad del control PID. Se utilizan los sensores DS18B20, las bibliotecas “DallasTemperature” y “OneWire” para obtener las lecturas.

3.1.1. Inicialización del sensor de temperatura.

Primero, se deben incluir las bibliotecas y definir los pines utilizados dentro del código principal.

La biblioteca “OneWire” permite conectar con dispositivos que utilizan el protocolo OneWire([4]), mientras que la biblioteca “DallasTemperature” simplifica la interacción con los sensores de temperatura DS18B20([6]).

Todos los sensores están conectados al pin de datos D4 y cada uno tiene una dirección única (ROM code) que se define y almacena en variables de tipo “DeviceAddress” pudiendo así ser diferenciados independientemente de cómo hayan sido conectados. También se crean variables para guardar las temperaturas leídas de cada sensor.

En la función setup(), se inicia la comunicación con los sensores llamando al método begin() de la biblioteca “DallasTemperature”.

3.1.2. Lectura.

En la función loop(), se solicita a los sensores que tomen una nueva lectura de temperatura usando el método requestTemperatures(). Tras solicitar las lecturas, se obtienen los valores de temperatura de cada sensor y se guardan en las variables correspondientes. Se recorre cada sensor detectado en el bus OneWire y se compara su dirección con las direcciones conocidas.

Para comparar las direcciones de los sensores, se define la función compareAddress(), que compara byte a byte dos direcciones de sensores. Para facilitar la depuración, se crea una función que imprime la dirección de un sensor en formato hexadecimal. Otra función útil para la depuración es printAddresses(), que imprime las direcciones de todos los sensores conectados.

3.2. Control PID.

El controlador PID es fundamental para mantener la estabilidad y precisión en sistemas de control de temperatura. Se compone de tres componentes esenciales:

- **Proporcional (P):** Ayuda a reducir el error actual proporcionalmente a su tamaño, ajustando la salida en función del error medido entre el setpoint y el valor actual.
- **Integral (I):** Acumula el error a lo largo del tiempo, corrigiendo cualquier error residual que quede después de la acción proporcional, eliminando el error constante.
- **Derivativo (D):** Reacciona a la tasa de cambio del error, proporcionando una acción correctiva anticipada para reducir la sobresalida y mejorar la estabilidad del sistema.

La combinación de estos componentes permite un control preciso y eficiente, ajustando la salida del sistema para mantener la temperatura deseada. La correcta sintonización de los parámetros PID (Kp, Ki, Kd) es crucial para el rendimiento óptimo del controlador.

3.2.1. Inicialización del Control PID .

La biblioteca “QuickPID” permite implementar un controlador PID facilitando tanto la configuración de los parámetros como el cálculo del control PID([7]). Se definen los pines y variables necesarios para el control PID, incluyendo el pin de salida PWM, la configuración del temporizador, el setpoint y las variables para los parámetros PID (Kp, Ki, Kd).

En la función setup(), se inicializa el PID configurando el modo de control, el tiempo de muestreo y los límites de salida. Además, se configura el temporizador para generar interrupciones periódicas.

3.2.2. Control Activo.

Cada diez minutos se calcula el nuevo valor del ciclo del PID y se actualiza el mapeo del ciclo de trabajo, se controla el estado de la bomba en función del valor del ciclo mapeado cada minuto, es decir, si se tiene un ciclo del diez por ciento la bomba permanecerá encendida un minuto y apagada nueve.

3.3. Interfaz gráfica.

La pantalla TFT se utiliza para mostrar información crítica del sistema en tiempo real, facilitando la depuración del código como un monitor serie.

3.3.1. Inicialización de la Pantalla TFT.

Se definen los pines de conexión de la pantalla TFT al ESP32. Luego, se crea un objeto tft de la clase Adafruit_ST7735 para controlar la pantalla. Durante la configuración inicial, se utiliza “tft.initR(INITR_BLACKTAB)” para inicializar la pantalla con el esquema de colores elegido, “tft.setRotation(1)” para establecer la orientación de la pantalla, y “tft.fillScreen(ST77XX_BLACK)” para preparar la pantalla con un fondo negro([8]).

3.3.2. Actualización de la Pantalla.

En el bucle principal, se utiliza “tft.fillRect()” para limpiar la pantalla antes de actualizar los datos. Con “tft.setCursor()” y “tft.print()”, se muestra la temperatura que se quiere controlar, el setpoint actual, los valores de los parámetros PID (Kp, Ki, Kd), el estado de la salida del PID y la dirección IP, util para acceder posteriormente a la interfaz web.

4. Interfaz Web de usuario.

WebSockets son un protocolo de comunicación que proporciona una conexión bidireccional continua entre un cliente y un servidor a través de un solo socket TCP. A diferencia de las conexiones HTTP tradicionales, que son unidireccionales y requieren una nueva conexión para cada solicitud o respuesta, WebSockets permiten una interacción en tiempo real y de baja latencia([5]).

Los beneficios de WebSockets incluyen:

- Baja latencia: Al mantener una conexión abierta, WebSockets permiten la transmisión de datos con un mínimo de retraso, lo cual es esencial para aplicaciones que requieren una respuesta inmediata, como el control de temperatura en tiempo real.
- Comunicación bidireccional: Tanto el cliente como el servidor pueden enviar datos en cualquier momento, lo que facilita una interacción más dinámica y eficiente. Esto es especialmente útil para sistemas que requieren monitoreo y control continuo.
- Reducción de sobrecarga: Al evitar la necesidad de abrir y cerrar conexiones repetidamente, WebSockets reducen la sobrecarga de comunicación y mejoran el rendimiento general, optimizando el uso de recursos del sistema.

En el contexto de este proyecto, la utilización de WebSockets es crucial para la interfaz web que permitirá a los usuarios monitorizar y controlar el sistema de temperatura en tiempo real. Esta tecnología asegurará que cualquier ajuste realizado por el usuario se refleje inmediatamente en el sistema y que los datos de temperatura se actualicen constantemente en la interfaz, siguiendo el flujograma de la Figura 11, proporcionando una experiencia de usuario fluida y receptiva.

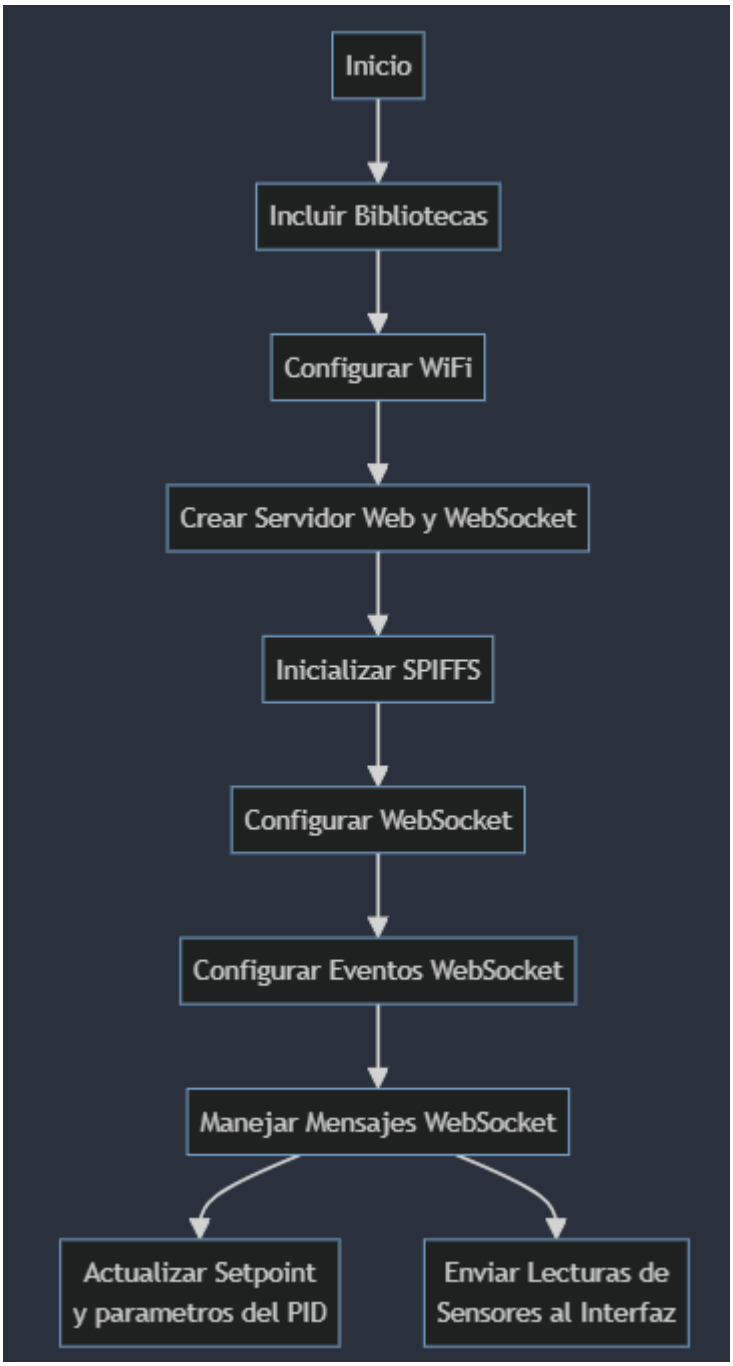

Figura 11 : Flujograma de la interfaz Web de usuario.

4.1. Inicialización.

Para comenzar, se deben incluir una serie de bibliotecas esenciales que facilitan diversas funcionalidades. "AsyncWebServer" y "AsyncWebSocket", diseñadas específicamente para la creación de un servidor web asíncrono y la comunicación en tiempo real mediante WebSockets. Mientras "AsyncWebServer" permite gestionar solicitudes HTTP de manera eficiente, "AsyncWebSocket" posibilita

la interacción bidireccional entre el servidor y los clientes, esencial para aplicaciones que requieren actualizaciones en tiempo real.

Además, se incorpora la biblioteca "SPIFFS" para gestionar el sistema de archivos en la memoria flash del ESP32. Esta funcionalidad es fundamental para almacenar archivos HTML, CSS, JavaScript y otros recursos que definen la interfaz de usuario de la aplicación web embebida en el dispositivo. Complementando estas herramientas, la librería "Arduino_JSON" simplifica la manipulación de datos en formato JSON, facilitando su formateo y transformación, aspecto crucial para la gestión y transferencia de información estructurada entre el ESP32 y los clientes conectados.

A continuación, se definen las credenciales WiFi (SSID y contraseña) necesarias para conectar el ESP32 a una red inalámbrica. Posteriormente, se crean instancias de objetos para el servidor web y el WebSocket. La función "initWiFi" se encarga de establecer la conexión del ESP32 a la red WiFi especificada, garantizando así la conectividad del dispositivo con el entorno inalámbrico.

Por otra parte, la función "initSPIFFS" monta el sistema de archivos SPIFFS durante el arranque, permitiendo al ESP32 acceder a los archivos almacenados en la memoria flash de manera eficiente. Este paso es esencial para la carga de recursos estáticos como la página HTML, la hoja de estilo CSS y scripts JavaScript que componen la interfaz de usuario

Además, se implementa la función "initWebSocket" para configurar el WebSocket, preparándolo para gestionar eventos como la recepción de datos, conexiones y desconexiones de clientes. Este componente es vital para aplicaciones que requieren interacción en tiempo real, proporcionando una vía eficaz para la comunicación bidireccional entre el ESP32 y los clientes conectados..

En el aspecto funcional, la función "handleWebSocketMessage" juega un papel crucial al manejar los mensajes entrantes a través del WebSocket. Entre sus tareas se incluye la gestión de actualizaciones de parámetros como el setpoint y los coeficientes PID, permitiendo ajustes dinámicos desde los clientes conectados. Por otro lado, la función "onEvent" se encarga de manejar diferentes eventos del

WebSocket, como conexiones establecidas, desconexiones y la recepción de datos específicos desde los clientes.

Finalmente, en el bucle principal del programa, se implementa la función "notifyClients" para enviar periódicamente las lecturas de los sensores a todos los clientes conectados mediante WebSocket. Esta funcionalidad asegura que los datos sean actualizados en tiempo real en la interfaz de usuario, proporcionando una experiencia dinámica y actualizada para los usuarios finales que interactúan con la aplicación implementada en el ESP32.

4.2. Archivos SPIFFS.

Para la interfaz de usuario, el servidor web distribuye archivos estáticos desde SPIFFS, los cuales definen la experiencia del usuario en el navegador. Estos archivos, alojados en la carpeta "data" dentro del directorio del proyecto, incluyen HTML para la estructura de la página, CSS para el diseño visual y JavaScript para la interactividad dinámica, formando así la base de la interfaz de usuario accesible desde cualquier navegador web (véase Figura12).

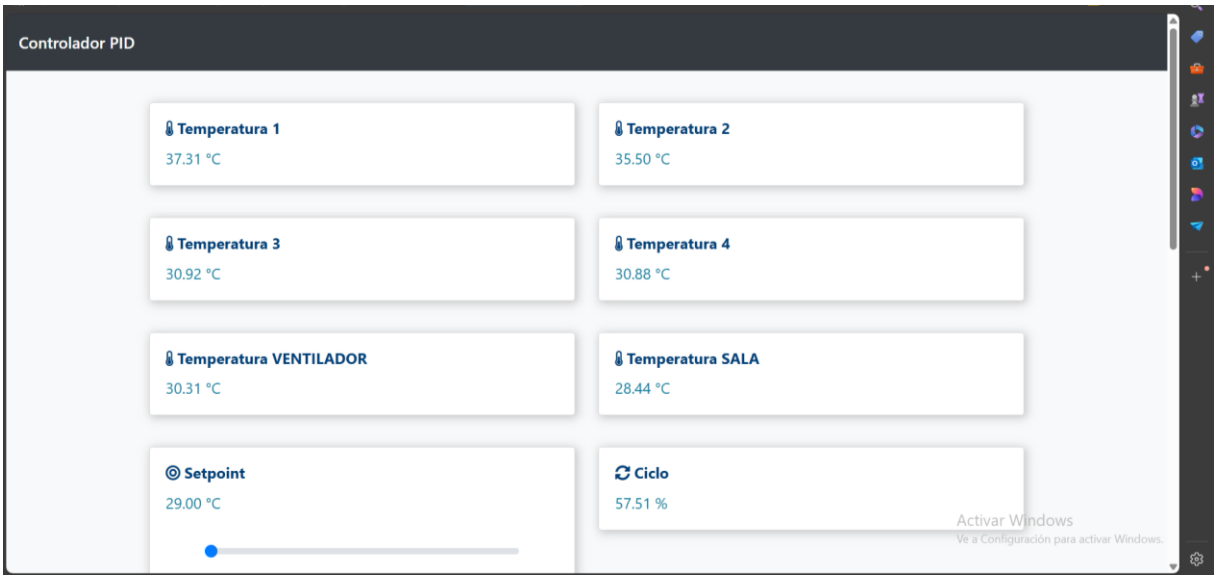

Figura 12 : Interfaz web de usuario.

4.2.1. Archivo HTML.

El HTML (HyperText Markup Language) es el lenguaje estándar utilizado para crear páginas web, definiendo la estructura y el contenido mediante una serie de elementos y etiquetas. El archivo "index.html" establece la base de la interfaz de usuario que se carga en el navegador.

Se comienza con la declaración "<!DOCTYPE html>" para especificar que se trata de un documento HTML5, seguido de "<html>" que marca la raíz del documento. Se utiliza "<meta charset="UTF-8">" para definir la codificación de caracteres asegurando la correcta interpretación de texto especial. Además, "<meta name="viewport" content="width=device-width, initial-scale=1.0">" configura la visualización adaptable en dispositivos móviles.

El título de la página se establece con "<title>Controlador PID</title>", visible en la pestaña del navegador. Las hojas de estilo se incluyen con etiquetas "<link>", vinculando:

- "<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">" para la hoja de estilos de Bootstrap.
- "<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.15.3/css/all.min.css">" para los iconos de FontAwesome.
- "<link rel="stylesheet" href="style.css">" para estilos personalizados adicionales.

En el cuerpo del documento ("<body>"), se estructura la interfaz con:

- Una barra de navegación ("<nav class="navbar navbar-dark bg-dark">") que utiliza las clases de Bootstrap para un diseño oscuro, con un título marcado por "".
- Un contenedor principal (" ") con un margen superior definido. Dentro de este contenedor:
    - Se crea una fila (" ") en la cuadrícula de Bootstrap.
    - Una columna (" ") que ocupa la mitad del espacio en pantallas medianas y más grandes.

- Las tarjetas de información (" ") se utilizan para presentar datos en tiempo real, cada una con:
    - Cuerpo de tarjeta (" ").
    - Título ("<h5 class="card-title">") que incluye un icono de FontAwesome.
    - Texto ("<p class="card-text">") para mostrar información como temperatura, identificada con un id para actualización dinámica.
- JavaScript se integra con las bibliotecas:
    - jQuery para manipulación del DOM y gestión de eventos.
    - Popper.js, empleado por Bootstrap para posicionamiento de tooltips y popovers.
    - Bootstrap JavaScript, que proporciona funcionalidades interactivas.

4.2.2. Archivo CSS.

CSS (Cascading Style Sheets) es un lenguaje fundamental en el diseño web, utilizado para definir cómo se presenta visualmente un documento HTML.

Para establecer la fuente global del documento HTML, se utiliza "font-family: Arial, Helvetica, sans-serif;", donde Arial es la primera opción, seguida de Helvetica y sans-serif como alternativas genéricas para garantizar la legibilidad en diferentes sistemas.

El estilo "display: inline-block;" permite que el contenido se comporte como un elemento en línea con un tamaño específico, mientras que "text-align: center;" centraliza el texto dentro de los elementos HTML, mejorando la presentación visual y la legibilidad.

Para el diseño del cuerpo del documento, se emplea "margin: 0;" para eliminar márgenes predeterminados del navegador, y "background-color: #f8f9fa;" para establecer un fondo claro y limpio, mejorando la legibilidad del contenido.

En cuanto a la barra de navegación y las tarjetas ("cards"), se añade "padding: 1rem;" para proporcionar un espacio uniforme alrededor del contenido, y "box-shadow: 2px 2px 12px 1px rgba(140, 140, 140, 0.5);" para crear un efecto de elevación sutil que hace que las tarjetas resalten visualmente en la página.

Las tarjetas y sus títulos se estilizan con "font-size: 1.2rem;" para definir un tamaño de fuente legible y coherente en los títulos, "font-weight: bold;" para destacar el texto en negrita y "color: #034078;" para establecer un color azul oscuro que mejora la visibilidad y la estética del texto.

Para el control deslizante personalizado, se utiliza "width: 80%;" para definir su ancho al 80% del contenedor disponible, y "margin: 20px 10%;" para añadir un margen de 20px arriba y abajo, y del 10% a los lados, centrando el control deslizante horizontalmente en la interfaz de usuario.

4.2.3. Archivo JavaScript.

JavaScript (JS) es un lenguaje de programación crucial para dotar a las páginas web de funcionalidades interactivas y dinámicas, permitiendo que respondan en tiempo real a las acciones del usuario sin necesidad de recargar la página.

Para comenzar, se declaran varias variables clave: "gateway" (URL del servidor WebSocket), "websocket" (instancia del objeto WebSocket), "canvas" (elemento del lienzo HTML5) y "ctx" (contexto de dibujo en el lienzo). También se definen "chartData" y "csvData" para almacenar los datos de la gráfica y para exportar a CSV, respectivamente, junto con "maxPoints", que determina el número máximo de puntos que se mostrarán en la gráfica.

La función "onload" se ejecuta al cargar la página, inicializando el lienzo, estableciendo la conexión WebSocket y dibujando la gráfica inicial.

La inicialización de la conexión WebSocket se realiza mediante "initWebSocket", que define los manejadores de eventos "onopen", "onclose" y "onmessage" para gestionar la apertura, el cierre y la recepción de mensajes en la conexión.

Cuando se abre la conexión WebSocket, la función "onOpen" solicita inicialmente las lecturas al servidor. En caso de que la conexión se cierre, la función

"onClose" intenta reconectar automáticamente después de un intervalo de 2 segundos.

La función "onMessage" se activa al recibir mensajes del servidor, actualizando dinámicamente los elementos HTML y las gráficas con los datos recibidos, proporcionando una retroalimentación inmediata al usuario.

Para solicitar nuevas lecturas al servidor, la función "getReadings" envía un mensaje correspondiente.

Para ajustar el setpoint o los coeficientes PID (véase Figura13), las funciones "updateSetpoint" y "updatePID" envían los nuevos valores respectivos al servidor para su procesamiento.

El dibujo y la actualización de la gráfica se gestionan mediante las funciones "drawChart", que prepara el lienzo y dibuja los ejes y la leyenda de la gráfica(véase Figura14), y "drawLine", que añade líneas con los datos proporcionados a la gráfica.

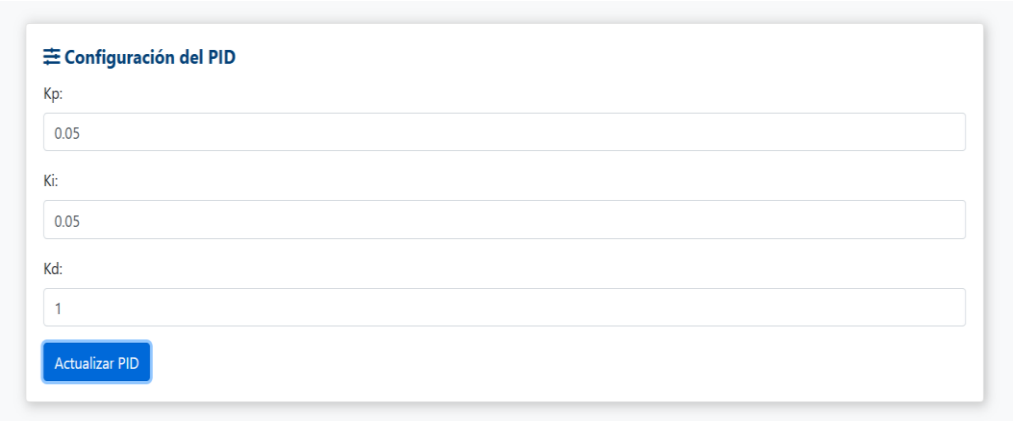

Figura 13 : Sección para modificar los parámetros de control.

Las funciones "updateChart" y "updateCSV" se encargan de actualizar dinámicamente los datos de la gráfica y del archivo CSV, respectivamente, asegurando que reflejen con precisión los últimos datos recolectados.

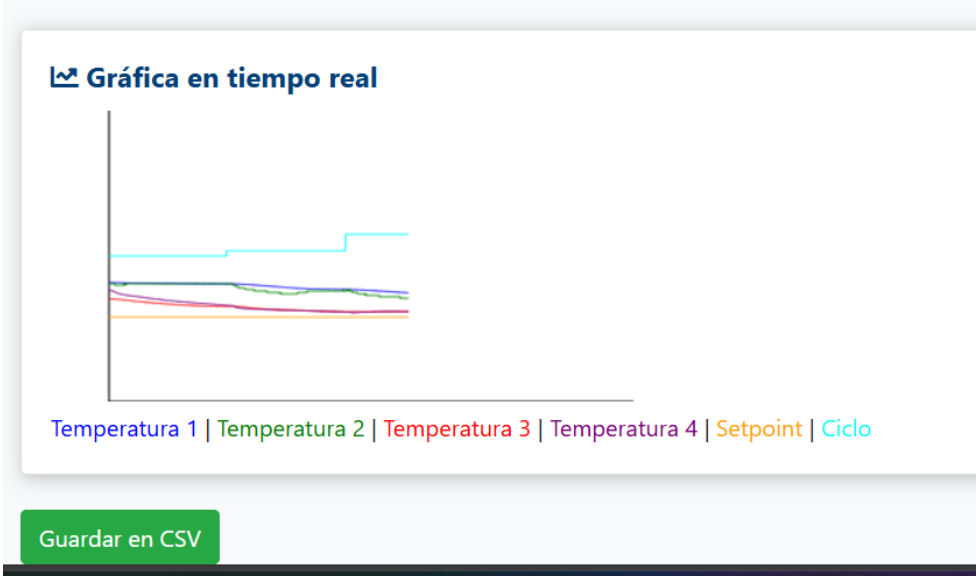

Figura 14 : Sección de graficas en tiempo real y botón para guardar en CSV.

Finalmente, la función "saveCSV" genera un archivo CSV con los datos acumulados y lo descarga en el navegador del usuario, facilitando la exportación y el análisis de datos.

5. Caso de uso.

El objetivo es controlar y monitorizar la temperatura de un sistema utilizando un controlador PID implementado en un microcontrolador ESP32, con visualización en tiempo real a través de una pantalla TFT y una interfaz web.

Este caso de uso proporciona un marco claro para cómo se utilizan los componentes hardware (sensores, microcontrolador, pantalla TFT) y software (control PID, interfaz web) para controlar eficazmente y monitorear un sistema de control de temperatura avanzado y adaptado a las exigencias modernas. Escaneando el código QR a continuación, se puede observar la estación en funcionamiento, demostrando su plena operatividad y eficacia.

Figura 15 : Código QR de la estación en funcionamiento.

Las múltiples pruebas realizadas permiten extraer conclusiones detalladas sobre el comportamiento y la eficacia del sistema([10]).

Prueba Inicial con Setpoint de 30°C y Parámetros KP = 0.05, KI = 0.05, KD = 1:

- **Eficiencia de Intercambio de Calor:** El sistema de control de temperatura operó de manera efectiva, evidenciando un buen intercambio de calor. La respuesta del sistema a los ciclos de encendido y apagado de la bomba fue adecuada, manteniendo las temperaturas dentro de los límites esperados.

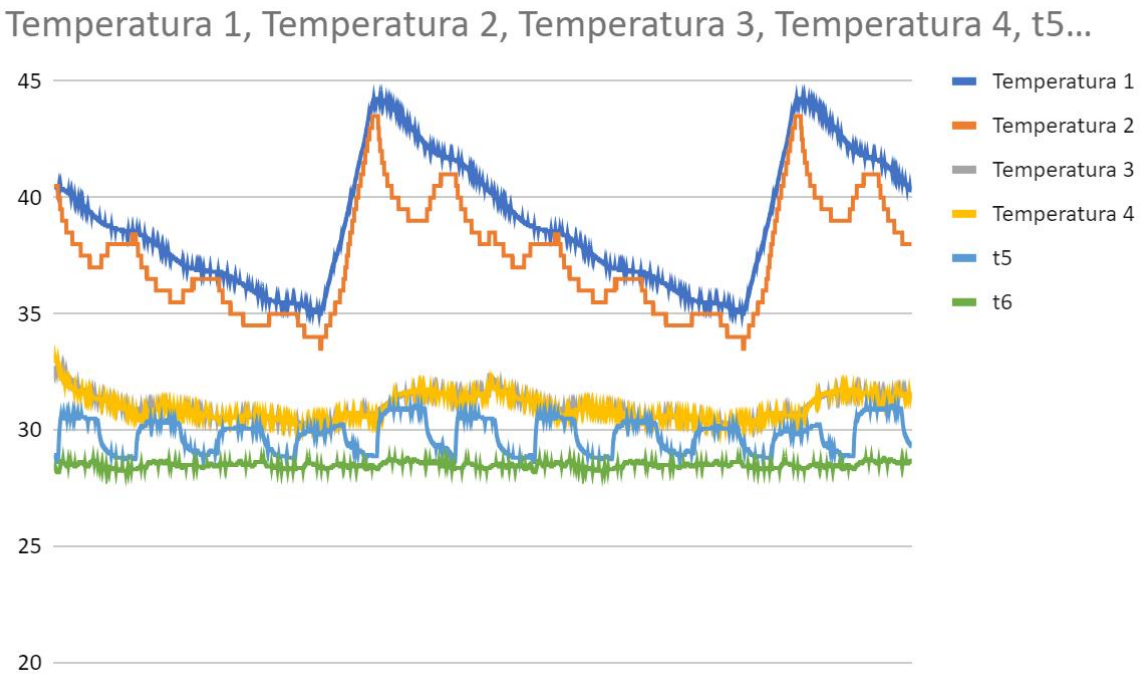

Figura 16 : Grafica completa con parámetros KP = 0.05, KI = 0.05, KD = 1.

- **Control de la Temperatura:** La temperatura 1 mostró una diferencia de 10 grados entre su máximo y mínimo(véase figura 16), debido al control interno de la resistencia del tanque de agua. La temperatura 2 reflejó una pérdida de temperatura, indicando un intercambio de calor efectivo durante los ciclos de encendido de la bomba. Este proceso enfría rápidamente la temperatura del tanque, activando la resistencia para calentar el circuito durante 13 minutos.
- **Impacto de la Bomba:** El encendido de la bomba (representada a nivel alto y bajo en las gráficas) causó un aumento en las temperaturas 3 y 4(véase figura 17). La temperatura controlada T5 aumentó cuando la bomba estaba encendida y disminuyó cuando estaba apagada, ajustándose hacia la temperatura ambiente. El ciclo de trabajo de la bomba osciló entre el 50% y el 65%, funcionando aproximadamente 5 de cada 10 minutos. La temperatura controlada máxima alcanzada fue de 31.06°C, la mínima de 28.6°C y la media de 29.78°C, muy próxima al setpoint de 30°C. El consumo energético fue de 37W/h.

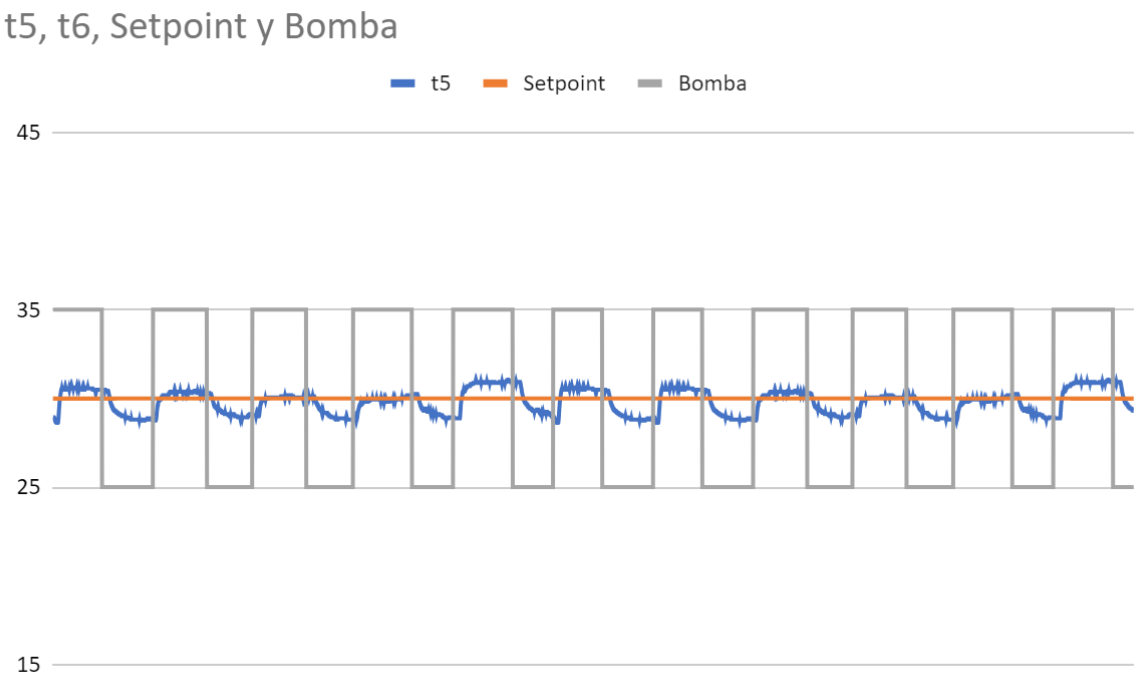

Figura 17 : Grafica de la evolución de la temperatura, Setpoint y estado de la bomba.

- **Condiciones Ambientales:** La temperatura de la sala se mantuvo casi constante a 28.5°C debido al gran tamaño del entorno y las condiciones de verano, estableciendo así un setpoint de 30°C.

Prueba con Menor Temperatura Ambiente y Setpoint Diferente:

- **Eficiencia del Sistema:** Con un setpoint ajustado y una temperatura ambiente de 27.7°C, el sistema demostró una relación más clara entre las temperaturas del sistema. Los picos iniciales en la temperatura 3 se debieron al bajo ciclo de la bomba, que acumulaba calor en el intercambiador. A medida que los ciclos de la bomba aumentaron, estos picos se estabilizaron como se ve en la siguiente figura.

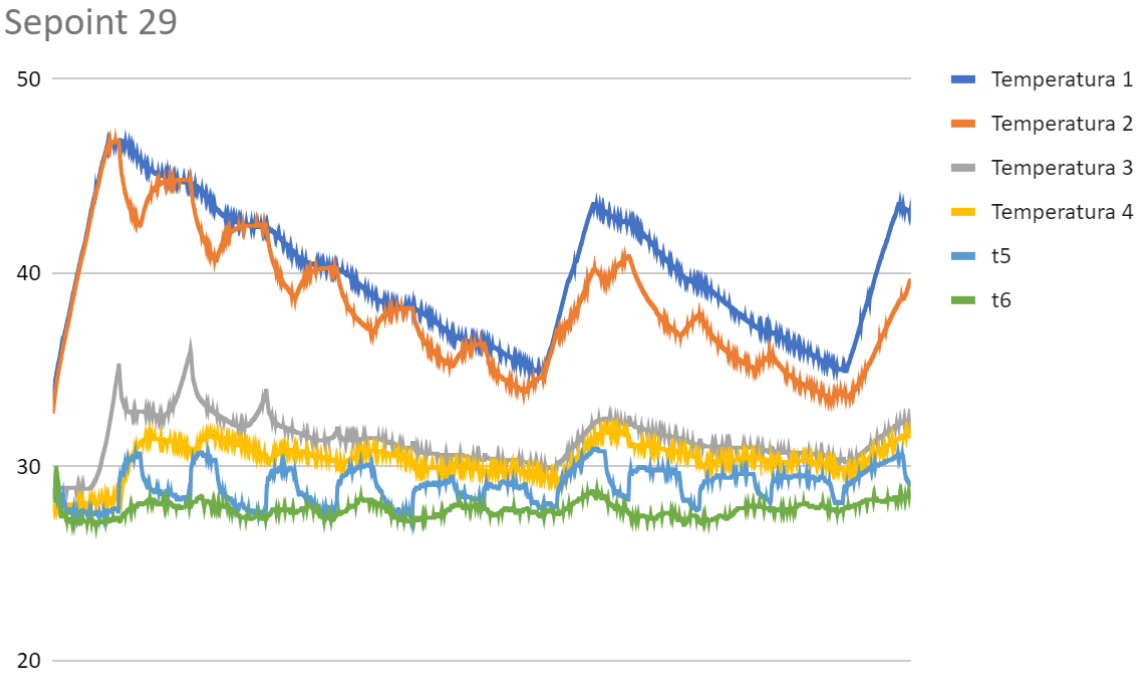

Figura 18 : Grafica completa en condiciones distintas.

- **Control de Temperatura:** La prueba mostró una temperatura controlada máxima de 30.9°C, una mínima de 27.3°C y una media de 29.01°C, cercana al setpoint de 29°C. El consumo energético fue de 43W/h, ligeramente mayor debido a la necesidad de ajuste como se ve en la figura a continuación.

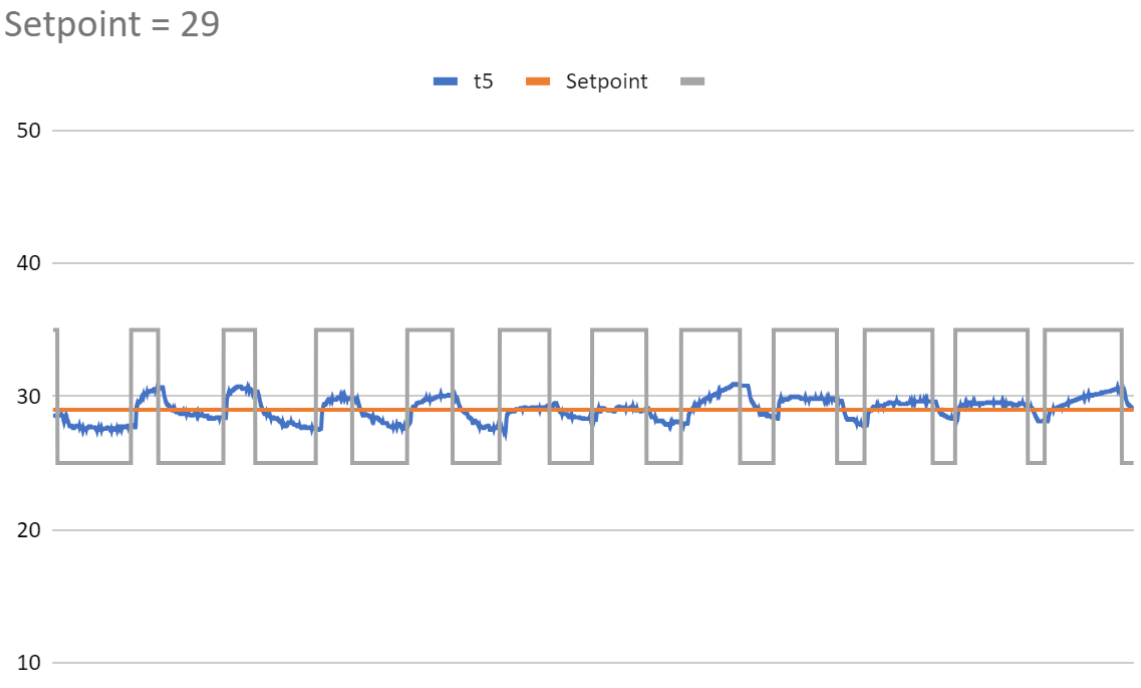

Figura 19 : Grafica de la evolución de la temperatura, Setpoint y estado de la bomba en condiciones distintas.

Prueba con PID Ajustado (KD = 0.2):

- **Ajuste del Setpoint:** Con un setpoint de 30°C, el sistema necesitó que la bomba funcionara al 100% constantemente, lo que resultó insostenible(véase figura 20). Se optó por bajar el setpoint, observando un consumo energético de 60W/h(véase figura 21).

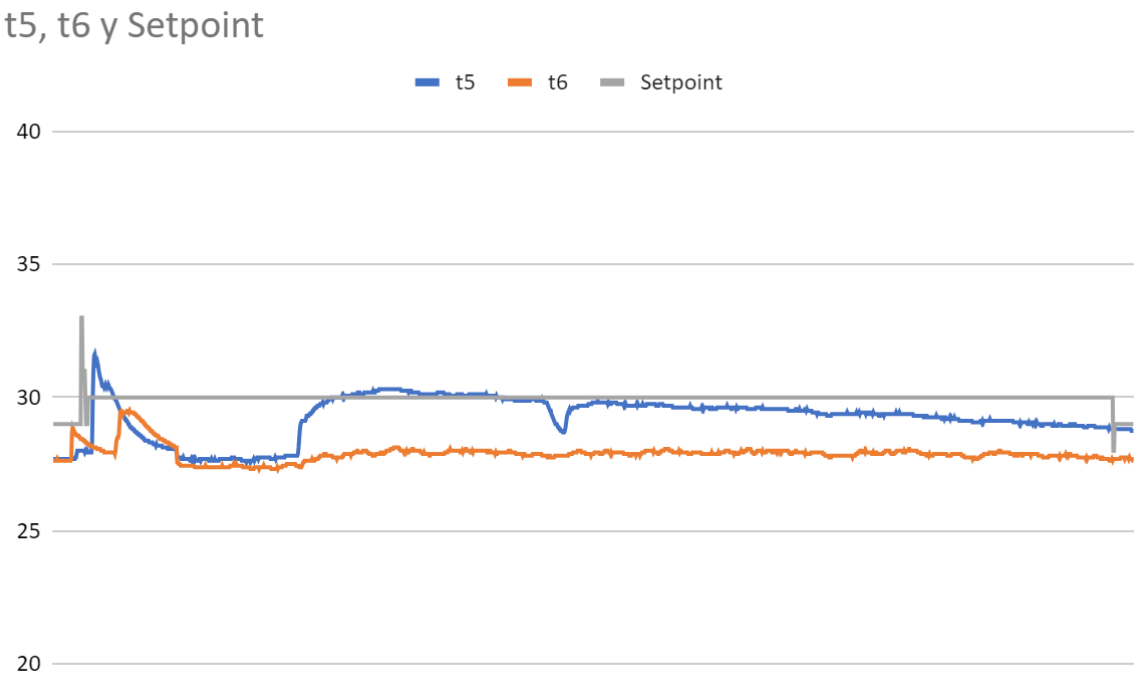

Figura 20 : Grafica de la evolución de la temperatura y Setpoint = 30.

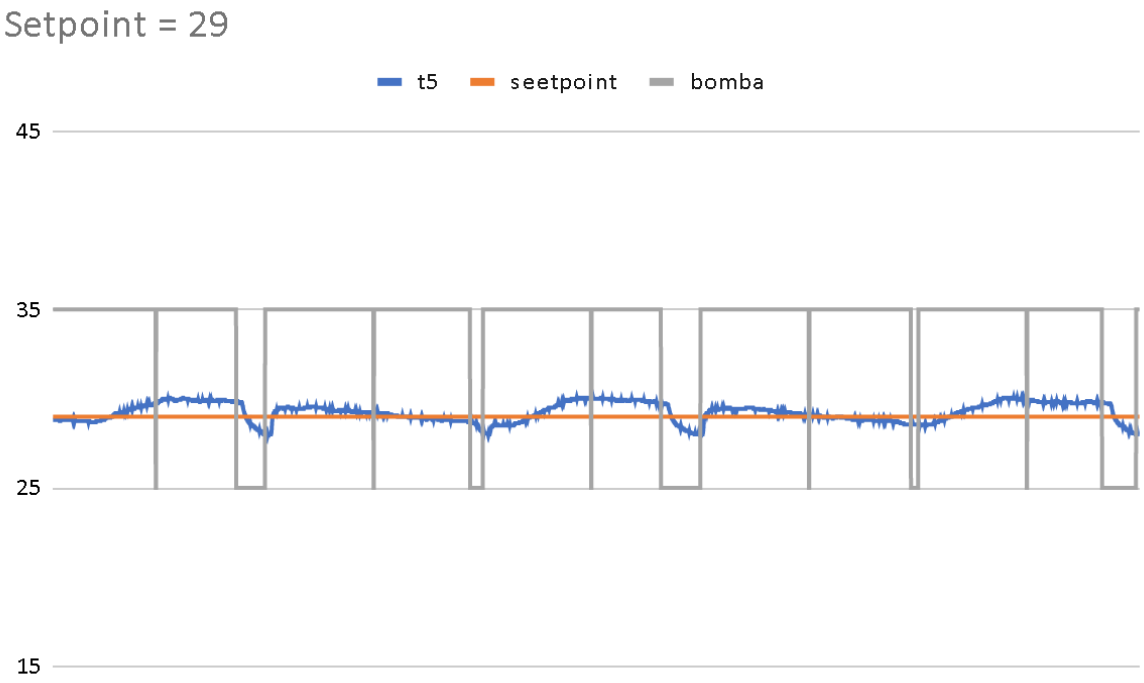

Figura 21 : Grafica de la evolución de la temperatura, Setpoint = 29 y estado de la bomba.

Prueba de Perturbación (Apagado del Ventilador):

- **Impacto del Ventilador:** Al apagar el ventilador, la temperatura aumentó significativamente, ya que el sistema no se enfriaba adecuadamente. La bomba redujo su ciclo y eventualmente se apagó. Para acelerar el enfriado, el ventilador se encendió nuevamente, observando ciclos cortos para intentar mantener la temperatura como se ve en la siguiente figura. Este proceso duró una hora y 20 minutos.

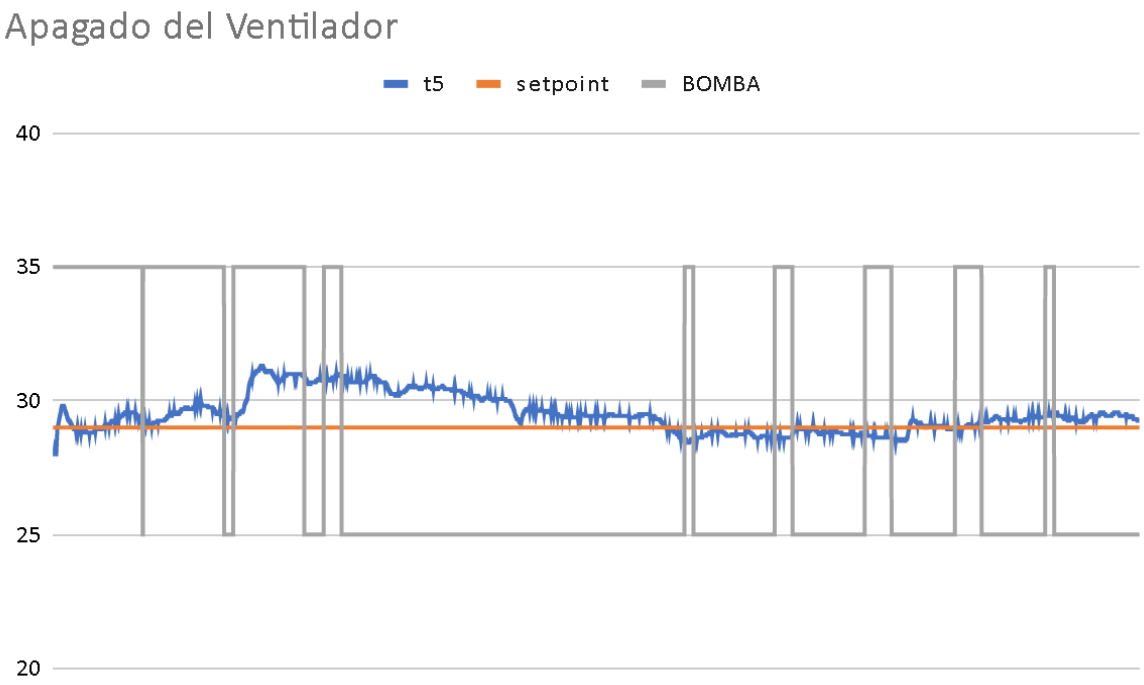

Figura 22 : Grafica de la evolución de la temperatura, Setpoint y estado de la bomba al apagarse el ventilador.

Prueba de Perturbación (Encendido del Ventilador):

- **Respuesta a la Perturbación:** Al encender el ventilador, la temperatura disminuyó y el ciclo de la bomba aumentó, demostrando la capacidad del sistema para adaptarse rápidamente a cambios bruscos en las condiciones ambientales.

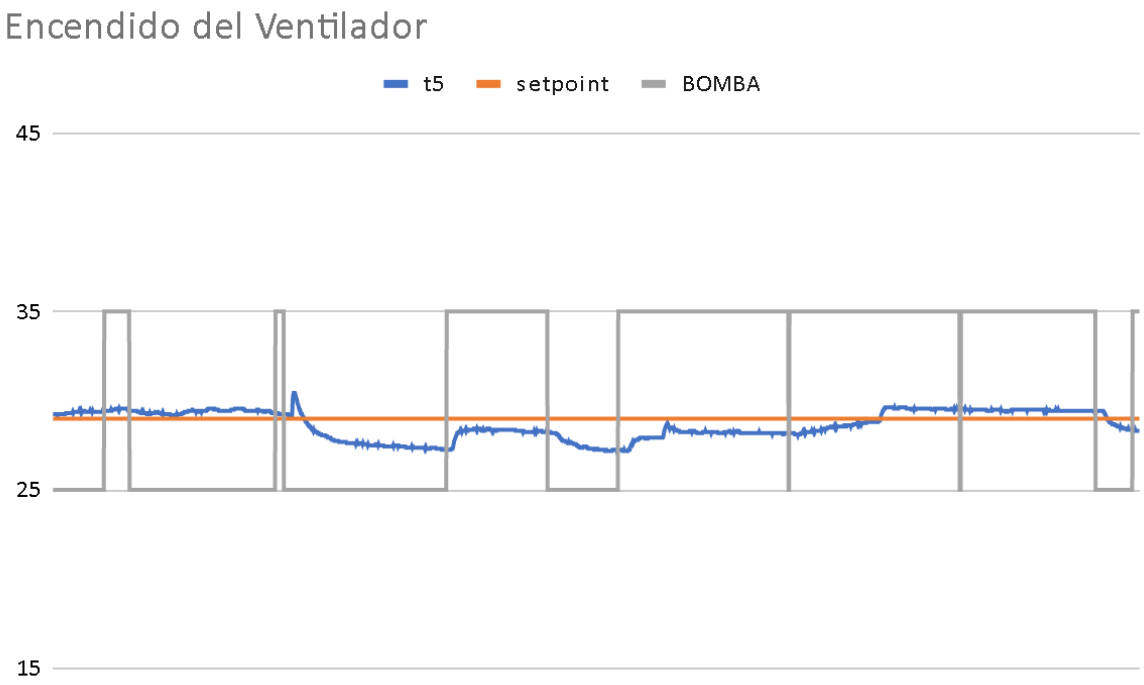

Figura 23 : Grafica de la evolución de la temperatura, Setpoint y estado de la bomba al encenderse el ventilador.

Desempeño General:

- **Estabilidad y Control Preciso:** A lo largo de todas las pruebas, el sistema mantuvo la temperatura deseada de manera eficaz, mostrando estabilidad y precisión en el control térmico.
- **Eficiencia Energética:** El consumo energético varió según las condiciones de prueba y los ajustes de setpoint, pero se mantuvo en rangos aceptables, demostrando la eficiencia del sistema.

Gráfica Completa:

- La gráfica completa final, que abarca las pruebas realizadas durante 3 horas y cuarto, proporciona una visión detallada del rendimiento del sistema. Se pueden observar las variaciones en las temperaturas medidas y el comportamiento del sistema ante diferentes condiciones y perturbaciones.

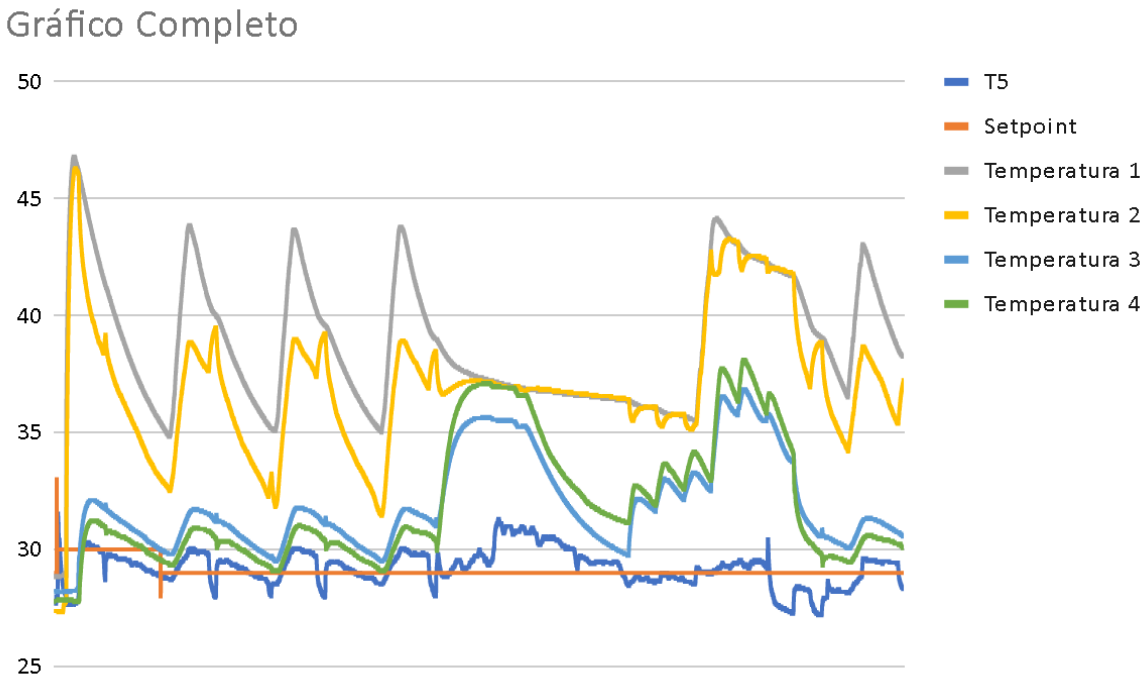

Figura 24 : Grafica de la prueba completa.

5.1. Flujo del caso de uso.

Inicio del sistema:

El sistema de control de temperatura se inicia, arrancando todos los componentes hardware al enchufarlos a la toma de corriente, en caso del microcontrolador puede ser a un ordenador también se ha insertar los sensores de temperatura dentro de las cánulas adaptadas para ellos (véase Figura 25) y conectar la salida PWM al relé para activar la bomba.

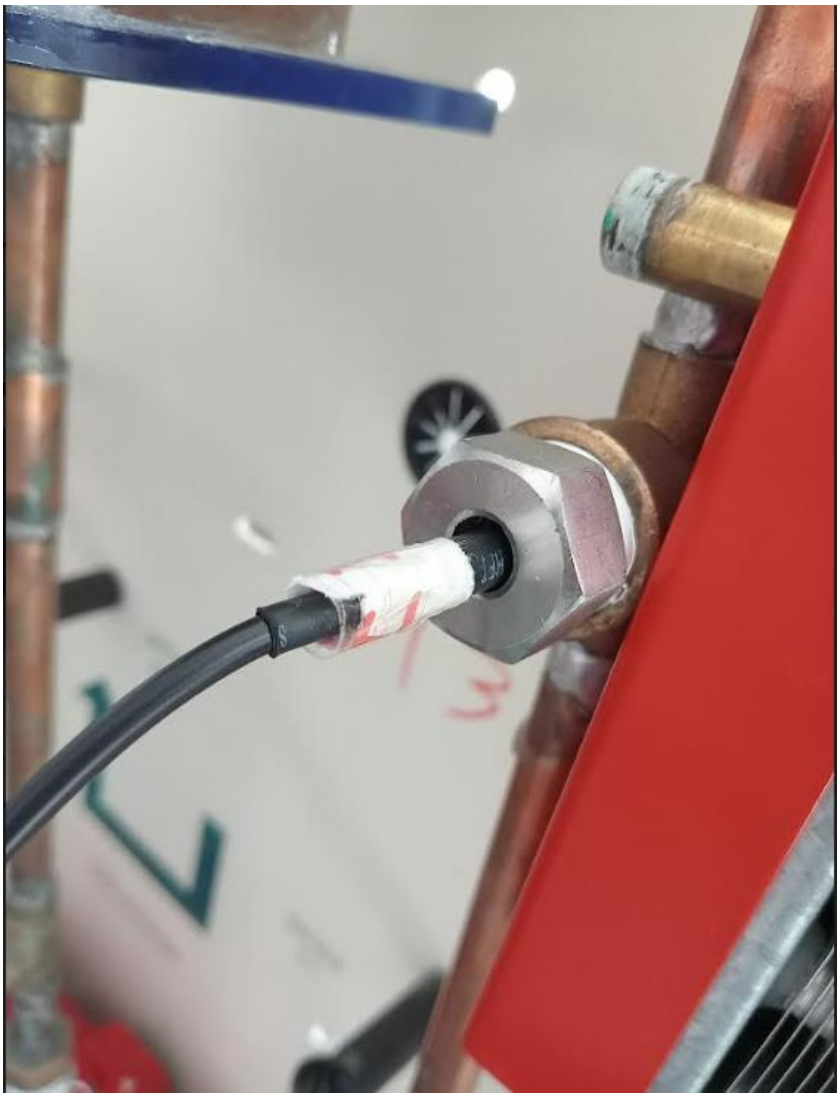

Figura 25 : Sensor de temperatura insertado en la cánula.

Monitorización de sensores:

Los sensores DS18B20 leen la temperatura actual de los circuitos primario, secundario de agua y las temperaturas del ventilador y sala.

Estos datos son enviados al microcontrolador ESP32 para su procesamiento.

Procesamiento y control PID:

El microcontrolador ESP32 utiliza un controlador PID (Proporcional-Integral-Derivativo) para comparar la temperatura medida con el setpoint deseado.

Calcula el ciclo de trabajo PWM para controlar la bomba de agua según sea necesario para mantener la temperatura establecida.

Visualización en pantalla TFT:

La pantalla TFT 1.8" SPI muestra información crítica en tiempo real, incluyendo la temperatura actual medida por los sensores, el setpoint de temperatura, el estado del controlador PID (activado/desactivado), y otros parámetros relevantes como los valores de los coeficientes PID (Kp, Ki, Kd) y la IP para establecer coneccion la interfaz web, especialmente útil en caso de no disponer un ordenador a la hora de realizar el experimento.

Interacción con la interfaz web:

Se establece una conexión WebSocket entre el ESP32 y la interfaz web del usuario permitiendo al operador visualizar y ajustar el setpoint de temperatura y los parámetros del controlador PID (Kp, Ki, Kd) en tiempo real.

Los ajustes realizados por el operador son enviados al ESP32 a través de WebSockets para modificar el comportamiento del controlador PID.

Actualización continua:

Los datos de temperatura y estado del sistema se actualizan continuamente tanto en la pantalla TFT como en la interfaz web.

Se implementan gráficas en la interfaz web para mostrar históricos y tendencias de temperatura, proporcionando al operador una visión clara del rendimiento del sistema.

5.2. Extensiones del caso de uso.

Las extensiones del caso de uso son funcionalidades adicionales que amplían las capacidades básicas del sistema. Estas extensiones permiten mejorar el rendimiento, la flexibilidad y la utilidad del sistema de control de temperatura. En este

contexto, las extensiones se centran en la captura de datos históricos y la capacidad de realizar ajustes y monitoreo remoto. A continuación, se detallan estas extensiones:

- Registro de datos: Captura y almacenamiento de datos históricos de temperatura en un archivo CSV para análisis posterior.
- Mantenimiento remoto: Posibilidad de realizar ajustes y monitoreo remoto a través de la interfaz web desde cualquier ubicación con conexión a Internet.

Estas extensiones no solo mejoran la funcionalidad básica del sistema, sino que también aumentan su eficiencia y utilidad, ofreciendo una solución más completa y adaptable a las necesidades de los usuarios.

6. Conclusión.

En la renovación de la estación de control de temperatura se realizaron mejoras significativas, incluyendo componentes modernos como el sensor de temperatura DS18B20, una bomba de agua moderna, y el microcontrolador ESP32 con capacidades IoT. Estas mejoras fueron clave para optimizar el rendimiento y eficiencia del sistema, haciendo que el proyecto implemente con éxito un control de temperatura avanzado para alcanzar y mantener condiciones específicas de manera eficiente.

El proyecto enfrentó desafíos significativos, como el reacondicionamiento de las estaciones debido a su estado inicial de desuso, lo cual requirió un mantenimiento extensivo. Además, la integración de la interfaz web implicó cambios importantes en la estrategia inicial, desde la consideración de una interfaz con Processing y comunicación MQTT hasta la decisión final de usar el microcontrolador como broker mediante PlatformIO, en lugar de ArduinoIDE, debido a las capacidades necesarias para implementar WebSockets.

A pesar de obtener unos resultados satisfactorios se identificaron áreas para posibles mejoras para trabajos futuros.

- **Ventilador Controlado:** Automatizar el control del ventilador para mejorar el intercambio de calor y el control de la temperatura.
- **Inclusión de memoria ssd :** Incluir una tarjeta ssd permite registrar los datos obtenidos sin necesidad de estar dentro de la página web lo que permitiría poder registrar los datos durante la noche sin que esté el operario con su pc o móvil conectado a la red.
- **Tanque Controlado:** Automatizar el control de la resistencia del tanque de agua para reducir las pérdidas de calor y mantener una temperatura más estable dentro del intecambiador y reducir así perturbaciones en la temperatura controlada.
- **Inclusión de sensores de flujo:** Añadir sensores de flujo para monitorear el caudal de agua puede contribuir a un control más preciso y facilita saber el estado de las bombas.

- **Mejorar el estado de la estación:** Debido al desuso las partes no reemplazadas acumularon lodo al fondo lo que perjudicaba el flujo de agua provocando que en ocasiones la bomba no circule el agua correctamente, haciendo una limpieza este problema se solucionaría.
- **inclusión de una PCB:** Con una PCB, puedes eliminar gran parte del cableado desordenado, lo que hará que sea más limpio y organizado, las conexiones serán más seguras y menos propensas a fallos que las conexiones de cables sueltos. Puede diseñarse para incluir módulos de alimentación por batería, lo que permitiría que funcione sin necesidad de estar enchufado a la red eléctrica.

7. Referencias

[1] Página de la tienda oficial del fabricante de ESP32:

https://mkelectronica.com/producto/esp32-devkit-c/

(Último acceso: 01-Julio-2024)

[2] Página del hardware TFT 1.8 SPI:

https://www.sparkfun.com/products/15143

(Último acceso: 02-Julio-2024)

[3] Página del hardware DS18B20:

https://www.sparkfun.com/products/11050

(Último acceso: 02-Julio-2024)

[4] Especificaciones ONE WIRE:

https://www.arduino.cc/reference/en/libraries/onewire/

(Último acceso: 02-Julio-2024)

[5] Especificaciones WebSockets:

https://registry.platformio.org/libraries/links2004/WebSockets

https://randomnerdtutorials.com/esp32-websocket-server-sensor/

(Último acceso: 02-Julio-2024)

[6] Especificaciones DallasTemperature:

https://www.arduino.cc/reference/en/dallastemperature/

(Último acceso: 02-Julio-2024)

[7] Especificaciones Quick PID:

https://www.arduino.cc/reference/en/quickpid/

(Último acceso: 02-Julio-2024)

[8] Especificaciones pantalla TFT:

https://www.arduino.cc/reference/en/libraries/tft/

(Último acceso: 02-Julio-2024)

[9] Repositorio del Código fuente:

https://github.com/asgarcia/ControlTemperatura

(Último acceso: 02-Julio-2024)

[10] Repositorio de los datos experimentales:

https://github.com/asgarcia/ControlTemperatura/tree/main/Datos_Experimentales

(Último acceso: 02-Julio-2024)