INSTITUTO POLITÉCNICO NACIONAL

Unidad Interdisciplinaria de Ingeniería Campus Zacatecas

UPIIZ

Título del reporte:

Reporte final

Materia:

RTOS

Nombre del maestro:

Ramon Jaramillo Martínez

Nombre de los alumnos:

Abner Kalid Verde Padilla

Diego Pérez Domínguez

Fecha y lugar de entrega:

Lunes 12 de enero del 2026

Zacatecas, Zacatecas, México

Objetivos

- Hacer uso de FreeRTOS para poder hacer el cumplimiento de distintas tareas, que involucren mostrar información, enviar datos, recibir datos, procesar estos datos.

- Hacer el control de un sistema térmico a un secador de filamento en donde se controla la humedad y la temperatura

Introducción

En el presente proyecto se desarrolla un sistema embebido basado en ESP32 que integra la medición de temperatura y humedad ambiental mediante un sensor AHT20, con la visualización y control de datos a través de una pantalla táctil TFT. El sistema utiliza FreeRTOS para gestionar múltiples tareas concurrentes, permitiendo una lectura eficiente del sensor, la interacción con el usuario por medio de la interfaz táctil, y la comunicación serial para el envío y recepción de datos. Esto facilita la creación de un dispositivo interactivo y responsivo, adecuado para aplicaciones en control ambiental y monitoreo en tiempo real.

El código implementa diferentes menús en la pantalla táctil que permiten al usuario configurar parámetros como la temperatura deseada y el tiempo de funcionamiento. Además, el sistema calcula el error de temperatura y transmite esta información para su posible uso en controladores externos. La integración de FreeRTOS garantiza una gestión óptima de las tareas, evitando bloqueos y mejorando la estabilidad del dispositivo. Este proyecto representa un ejemplo práctico de cómo combinar sensores, pantallas y sistemas operativos en tiempo real para el desarrollo de soluciones inteligentes y personalizables.

Marco teórico

FreeRTOS se integra en el ESP32 como un sistema operativo de tiempo real diseñado para gestionar la complejidad de las aplicaciones modernas de IoT. A diferencia de los sistemas operativos convencionales de escritorio, su núcleo está optimizado para ejecutarse con muy pocos recursos de memoria, permitiendo que el ESP32 administre múltiples hilos de ejecución de manera concurrente. En el ecosistema de Espressif, FreeRTOS actúa como la capa intermedia que organiza cómo y cuándo se ejecutan las instrucciones, permitiendo que el desarrollador deje de preocuparse por el flujo secuencial y se enfoque en el diseño modular por bloques independientes. [1]

El funcionamiento principal se basa en un planificador o "scheduler" que utiliza un algoritmo de prioridad fija y apropiativa. Esto significa que el sistema siempre otorga el control del procesador a la tarea de mayor importancia que esté lista para ejecutarse. En el contexto específico del ESP32, FreeRTOS destaca por su capacidad multinúcleo (Symmetric Multiprocessing), lo que le permite distribuir la carga de trabajo entre los dos núcleos del chip de forma simétrica o asignar tareas específicas a un núcleo concreto para evitar interferencias, como separar las comunicaciones Wi-Fi de las tareas de control crítico. [1]

Las tareas en FreeRTOS se definen como funciones independientes que poseen su propio espacio de memoria o "stack". Cada tarea puede estar en distintos estados: ejecutándose, lista para ejecutarse, bloqueada (esperando un evento) o suspendida. Esta estructura permite que procesos que consumen mucho tiempo, como la espera de una respuesta de un servidor web, no detengan la ejecución de otras funciones vitales, como la lectura de sensores de emergencia o la actualización de una interfaz de usuario, mejorando drásticamente la capacidad de respuesta del sistema. [1]

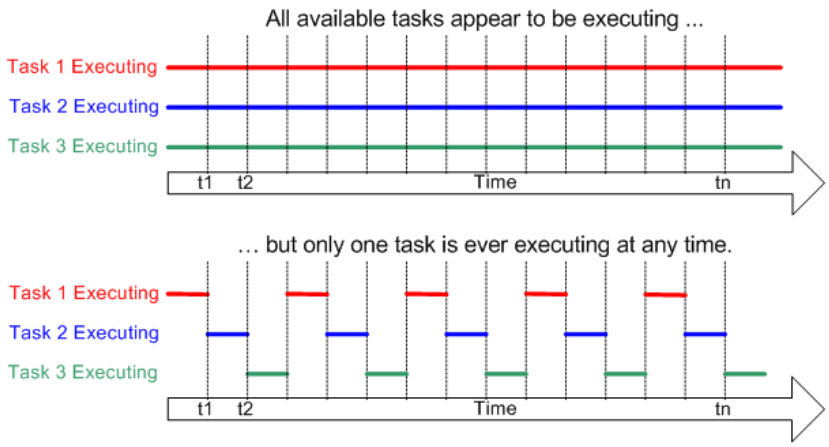

Ilustración 1. Funcionamiento de RTOS en operación de tareas

Para garantizar que estas tareas se comuniquen de forma segura, el sistema emplea mecanismos de sincronización como colas, semáforos y mutex. Estos elementos son fundamentales para evitar condiciones de carrera, situaciones donde dos tareas intentan modificar el mismo recurso físico o variable al mismo tiempo, lo que podría provocar errores impredecibles. Las colas permiten el paso de mensajes entre tareas de forma ordenada, mientras que los semáforos ayudan a gestionar el acceso compartido a periféricos como el bus I2C o el puerto serial de manera que solo una tarea los utilice a la vez. [1]

Entre las ventajas más notables de utilizar FreeRTOS en un ESP32 se encuentra la eficiencia energética y la escalabilidad del proyecto. El sistema incluye una tarea denominada "Idle" que se ejecuta cuando ninguna otra tarea requiere atención, permitiendo que el procesador entre en modos de bajo consumo de forma automática. Además, facilita enormemente el mantenimiento de proyectos complejos, ya que permite añadir o quitar funcionalidades mediante la creación de nuevas tareas sin alterar el código existente, proporcionando una robustez y un nivel de control profesional que supera ampliamente el modelo de programación tradicional. [1]

Desarrollo

Para el desarrollo de este proyecto se decido hacer uso de un sensor AHT20, una pantalla tft para el hmi, una resistencia con ventilador incluido y por último el control difuso dado por medio del envío y recepción de datos a Matlab por el serial, por lo tanto, las tareas de nuestro proyecto se realizaron por separado para posteriormente juntarlas en un solo código en RTOS.

Por lo que al probar los códigos por separado y unirlos en un solo documento resulte en el uso de 6 tareas, una exclusiva para el uso de la pantalla, otra para la lectura del sensor, otra para enviar datos por el monitor serial, otra para recibir datos e interpretarlos, otra para contar el tiempo, otra para el uso de firebase y subir datos a la nube, todas estas tareas tenían la misma prioridad.

Para el desarrollo de la interfaz hombre maquina se utilizó una pantalla táctil con 4 recuadros en el menú principal, los cuales eran: Temperatura, tiempo, estado y reinicio, en temperatura dada en grados Celsius y en tiempo dado en minutos se podía configurar desde 0 hasta 99, mientras que en el menú de estado mostraba el estado en el que se presionó el botón de estado, no es en tiempo real, por último en el botón de reinicio, regresaba las variables de temperatura y tiempo a 0 para poder volver a ser configuradas.

Por otro lado se utilizo un puente h l298n para la etapa de potencia del ventilador, y se hizo uso de un modulo relee para poder controlar la resistencia, en este caso como el voltaje era bajo para la resistencia y no se conto con una fuente de mayor capacidad, se decidió que el control seria un on-off por medio de un relee esto porque al ser de 12 volts entregaba poco calor y si aun así se controlaba con un pwm se tardaría mucho mas nuestro sistema de control, pero esto es una característica propia de los sistemas de control térmico.

Y aunque se haya considerado un control on-off para la resistencia aun así se añadió al control difuso desarrollado en Matlab como se puede ver en las siguientes figuras están las funciones de membrecía de nuestras entradas y salidas, así como las reglas:

Entradas:

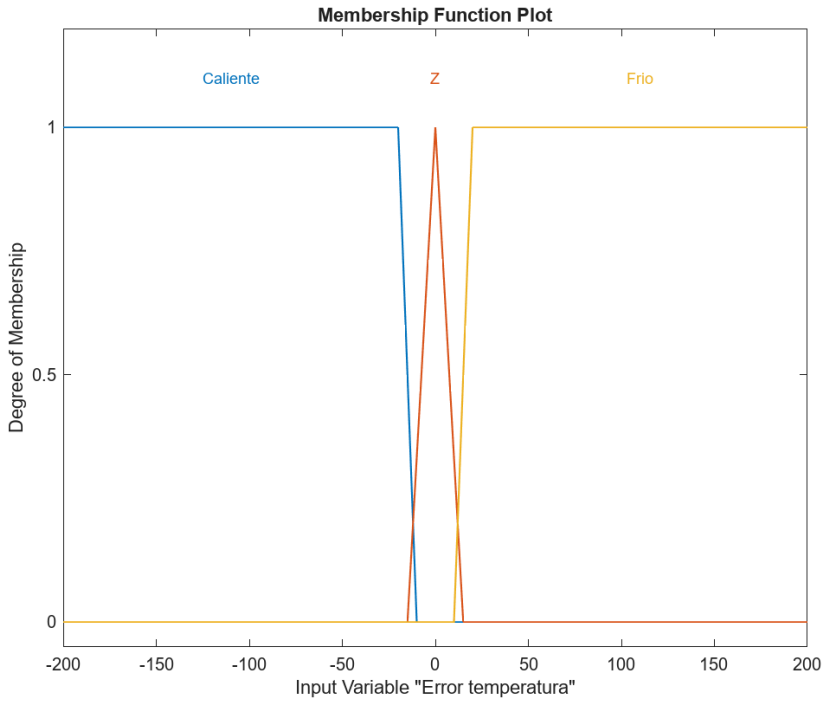

Ilustración 2.Funciones de membresía error de temperatura

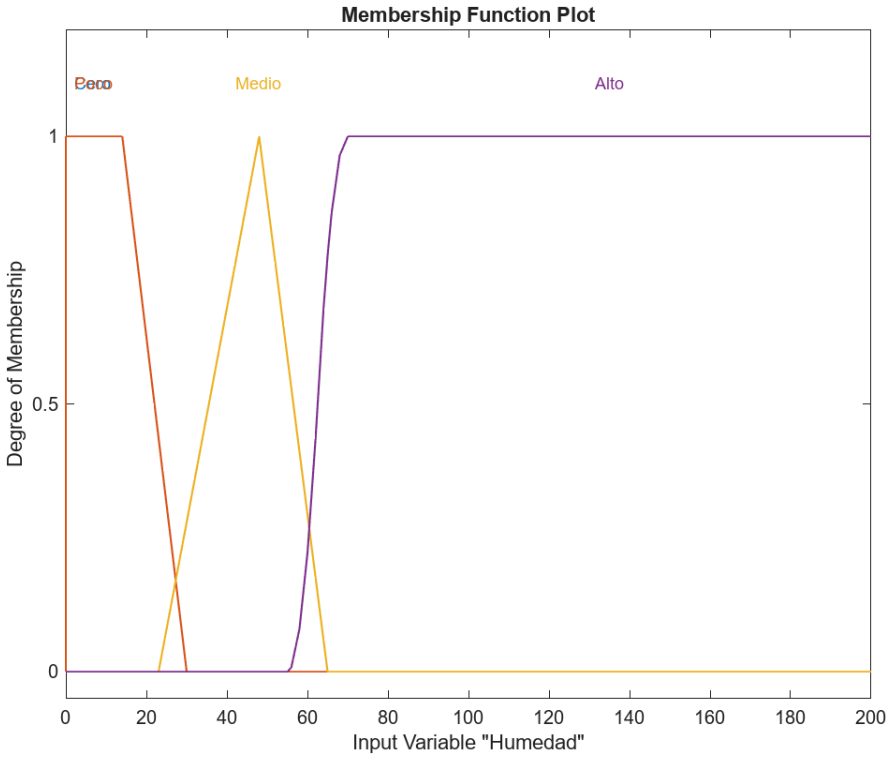

Ilustración 3.Funciones de membresía humedad

Salidas:

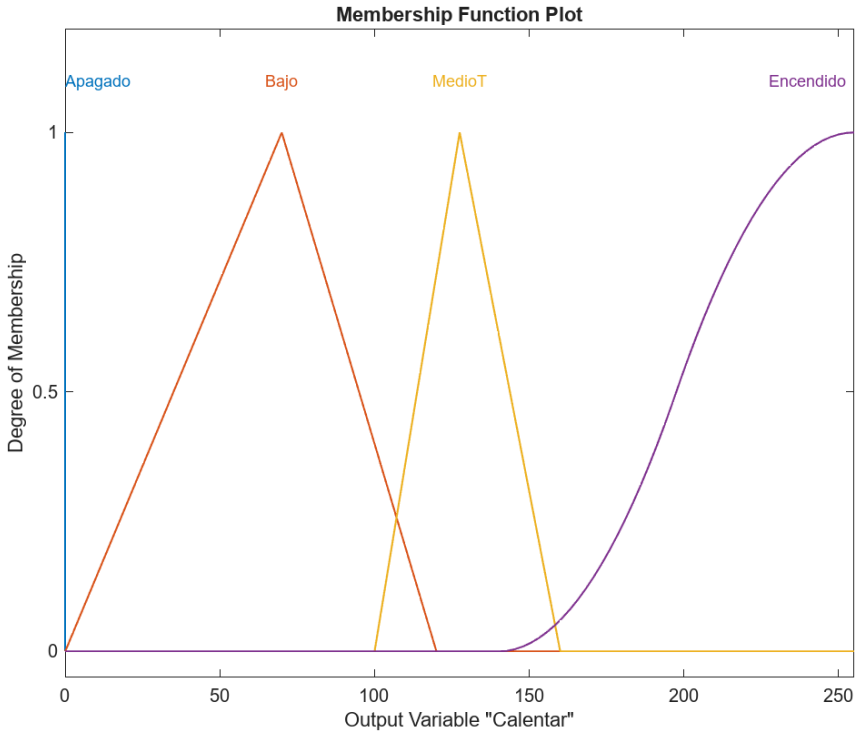

Ilustración 4Funciones de membresía resistencia

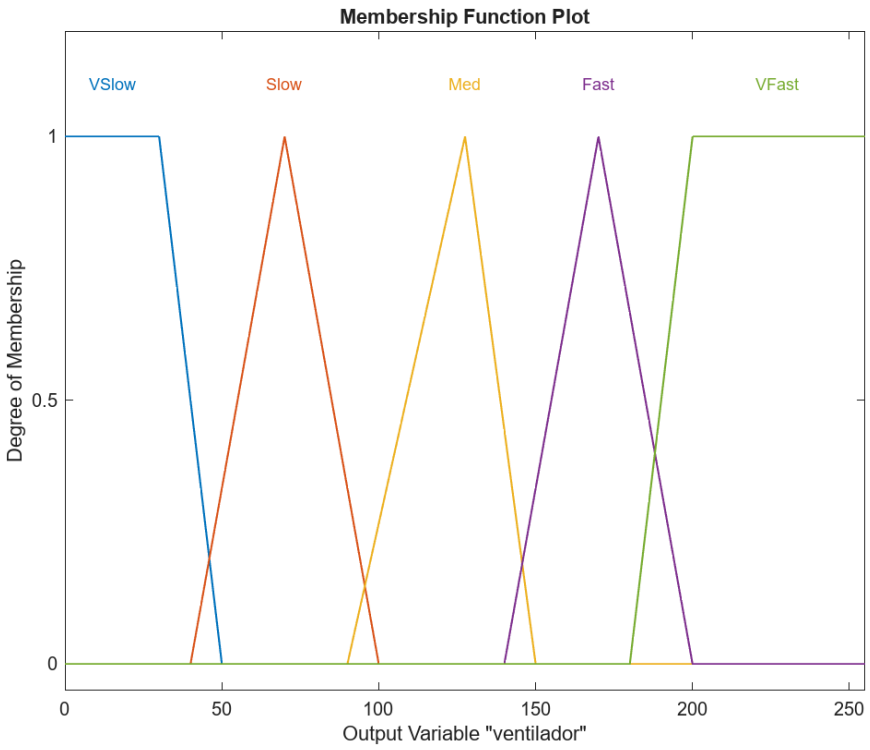

Ilustración 5. Funciones de membresía pwm del ventilador

	Rule	Weight	Name
1	If Error temperatura is Caliente and Humedad is Cero then Calentar is Apa...	1	rule1
2	If Error temperatura is Caliente and Humedad is Poco then Calentar is Apa...	1	rule2
3	If Error temperatura is Caliente and Humedad is Medio then Calentar is Ap...	1	rule3
4	If Error temperatura is Caliente and Humedad is Alto then Calentar is Apag...	1	rule4
5	If Error temperatura is Z and Humedad is Cero then Calentar is Apagado	1	rule5
6	If Error temperatura is Z and Humedad is Poco then Calentar is Apagado, ...	1	rule6
7	If Error temperatura is Z and Humedad is Medio then Calentar is Apagado,...	1	rule7
8	If Error temperatura is Z and Humedad is Alto then Calentar is Apagado, v...	1	rule8
9	If Error temperatura is Frio and Humedad is Cero then Calentar is Encendi...	1	rule9
10	If Error temperatura is Frio and Humedad is Poco then Calentar is Encendi...	1	rule10
11	If Error temperatura is Frio and Humedad is Medio then Calentar is Encen...	1	rule11
12	If Error temperatura is Frio and Humedad is Alto then Calentar is Encendid...	1	rule12

Ilustración 6. Reglas

Por otro lado, para la parte del contenedor, se opto por el uso de un contenedor de plástico hermético para que este pudiera guardar el calor generado por la resistencia, a este se le hicieron hoyos por donde entraba la resistencia y la pantalla tft para poder tener la visualización de datos en una de sus caras, este contenedor es transparente para que en todo momento se pueda ver como es que esta el filamento colocado.

Resultados:

Como resultado se presentan las siguientes imágenes en las que se puede ver cómo está constituido el menú y cada uno de sus apartados, además del monitoreo de variables.

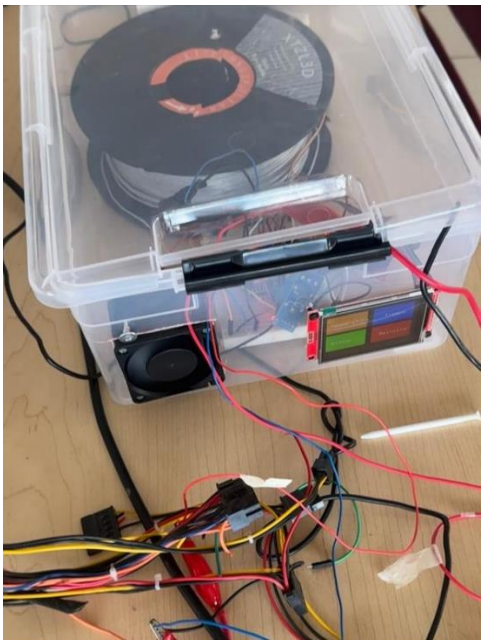

Ilustración 7.Planta de control

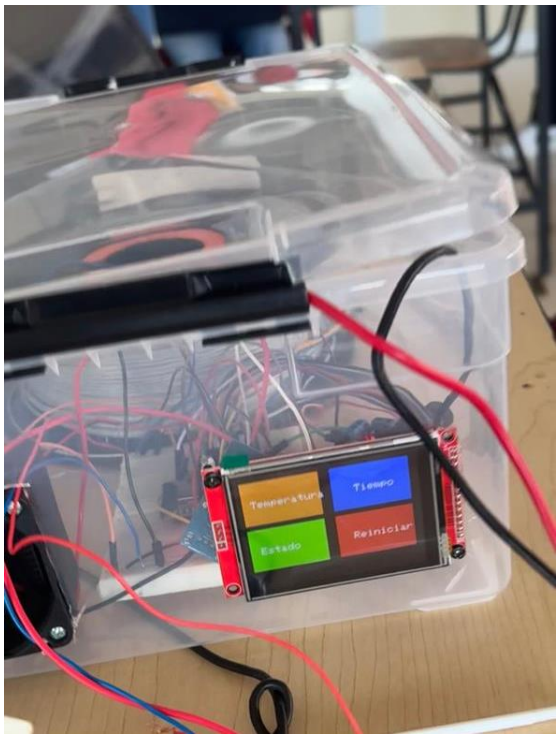

Ilustración 8. Interfaz HMI

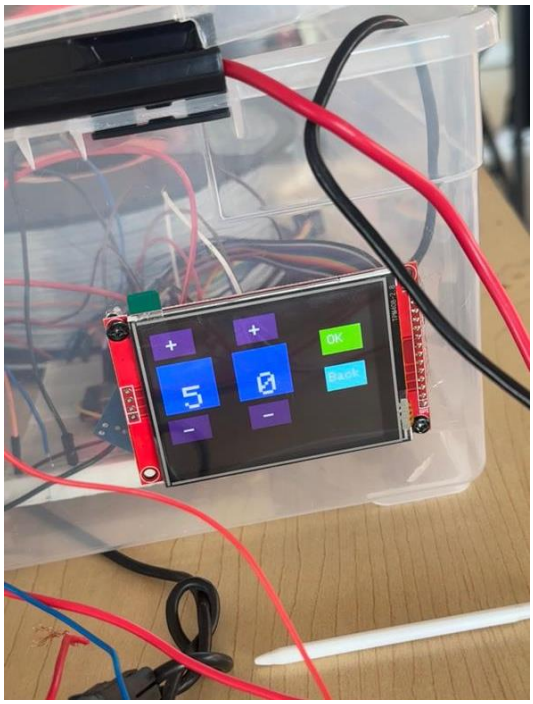

Ilustración 9. Establecimiento de setpoin en temperatura (°C)

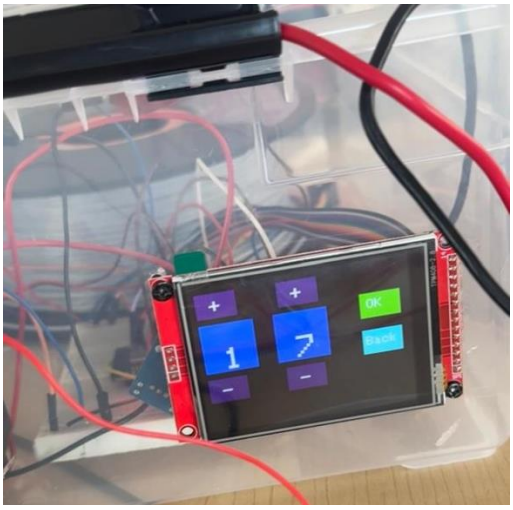

Ilustración 10. Establecimiento de un timer en minutos

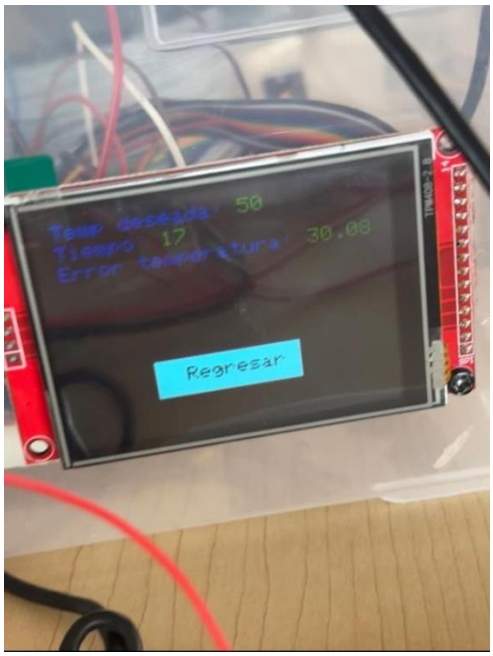

Ilustración 11. Menú de estado y variables medidas

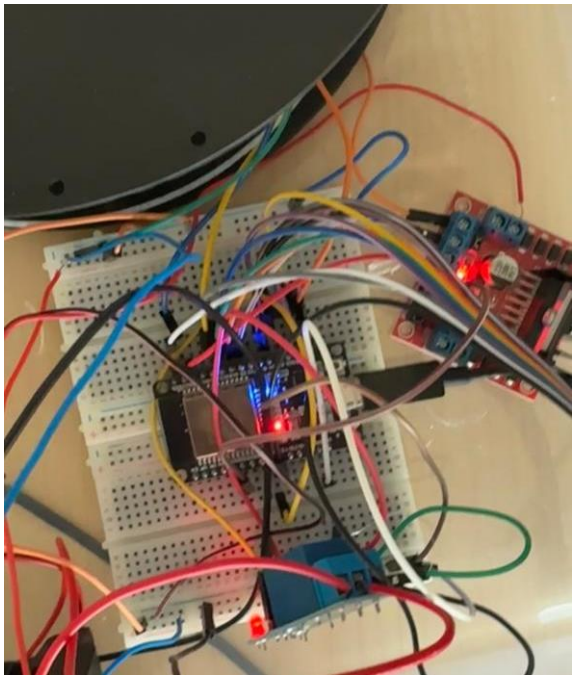

Ilustración 12. Circuiteria

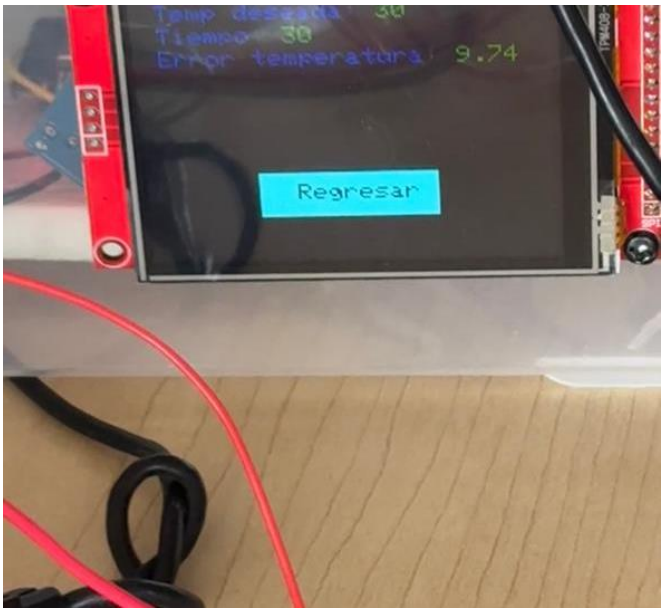

Ilustración 13 Actualización del error de temperatura

Se pudo observar que el control por medio de RTOS funciona de mejor manera y mas fluido de que de forma secuencial, sin embargo si se vio que se tardaba mucho en calentar el sistema, pero esto es debido a que la fuente seleccionada no fue la, más adecuada, sin embargo el uso de RTOS ayudo bastante a la división de tareas y eso se puede ver al momento del uso de pantalla, uso de sensor y uso de uart, usando 3 comunicaciones distintas y el micro en ningún momento fallo.

Conclusiones:

Abner Kalid Verde Padilla: En conclusión el uso de RTOS en sistemas de control optimiza mucho los recursos del microcontrolador y hace que se le pueda dar mayor importancia a una tarea relacionada al control, en este casos e tuvo un control de temperatura, el cual es un control lento , pero en controles que requieren mayor velocidad y muchas tareas, el uso de RTOS es muy útil, además es fundamental hacer comentario que la programación de códigos en RTOS es mas sencilla puesto que por medio de ella se pudieron probar los códigos por separado antes de incluirlos al código final, lo que resulto en que se redujo el error producido por ciertas tareas, por ejemplo al momento de usar la pantalla táctil, la cual ocupaba muchas funciones y si se hubiera programado el código en una sola intención había posibilidad de que no funcionara de la mejor manera.

Finalmente, el uso de RTOS es una herramienta que no solo ayuda a la organización del código, haciendo códigos mas legibles al separar las tareas en bloques específicos en donde en cada uno se describe el proceso realizado, sino que también ayudo al manejo y optimización de recursos en un microcontrolador como esp32.

Diego Pérez Domínguez: La implementación de un sistema de control de temperatura bajo un sistema operativo de tiempo real, transforma un proceso crítico en una operación altamente confiable, superando las limitaciones de los bucles de control secuenciales que maneja la programación en Arduino convencional. Gracias a la gestión por prioridades, el algoritmo de control en este caso un control difuso empleado en Matlab puede ejecutarse en una tarea de alta prioridad con intervalos de tiempo exactos, garantizando que el muestreo del sensor y el ajuste del actuador ocurran sin retardos provocados por otras funciones del sistema, como la actualización de interfaces gráficas o la

comunicación en red. Esta segregación de funciones no solo mejora la precisión en la estabilización térmica, sino que también le da al sistema de una robustez necesaria para prevenir sobrecalentamientos o fallos críticos mediante una respuesta inmediata ante eventos imprevistos.

Bibliografía

[1] FreeRTOS, «FreeRTOS Kernel Quick Start Guide,» Free RTOS, 01 2026. [En línea]. Available: https://freertos.org/Documentation/01-FreeRTOS-quick-start/01-Beginners-guide/02-Quick-start-guide. [Último acceso: 12 01 2026].