**CONTROLADORA TÉRMICA PROGRAMABLE**

**Versión 00**

**Memoria Descriptiva**

**Características funcionales:**

Esta **versión 00** es Autónoma,, la selección y activación de la rutina de control térmico requiere modificar el archivo ***Parametror.ino*** y volver a subir el archivo compilado a la placa Arduino UNO con el software Arduino IDE, incluido en el repositorio GIT.

Un grupo de rutinas térmicas junto con un reloj de tiempo real (RTC) con batería de 10 años de duración y una memoria Sram (no volátil). Permiten el funcionamiento continuo aun con cortes de energía, dado que al reiniciar lee de la memoria los parámetros seteados y los aplica a la hora de actualizada para continuar la rutina establecida .

También el controla el ***fotoperiodo simulado*** dentro de la cámara encendiendo y apagando una tira de leds blancos siguiendo el ritmo circadiano.

Una rutina tipo WatchDog para Auto Reset cada 10 minutos que previene la saturación de memoria del arduino y o inestabilidades.

**Las interfaces con el operador:**

***Interfaz Autónoma*** con un conjunto de led de colores que indican el pulso de lecturas y también la rutina térmica que está corriendo.

***Interfaz Gráfica***  El paquete de software ***Control\_Monitoreo.pde*** corriendo en processing y conectado por puerto serie, muestra la curva temporal de temperatura dela rutina activa, arroja el estado instantáneo del software y la cámara Hora y temperatura instantánea y calculada

Asímismo graba un archivo txt histórico a modo de ***datalogger*** que guarda en el disco duro de la PC.

Si el equipo se conecta a un sketch de processing también provisto se puede monitorear la curva circadiana la temperatura instantánea las temperaturas seteadas y demás parámetros

Este paquete adicional tiene incorporada las funciones para cambio de rutinas para ser usadas en la ***versión 01 de la CTP.***

**Consideraciones**

Calibración de sensores: El ajuste de la lectura de los sensores con adjSensor es importante para obtener datos precisos.

Manejo de eventos: El uso de pulsadores físicos y comandos serie permite un control flexible del sistema.

Robustez: La inclusión de un reset automático cada 15 minutos ayuda a prevenir bloqueos del sistema.

**Bloques funcionales del Harware::**

Un microcontrolador arduino UnO programado en C++ (arduino IDE) orientado a objetos

Un grupo de librerías para las funciones más complejas.

Un display Oled B&N junto con un encoder rotativo con boton pulsador y 2 pulsadores adicionales componen la Interfase Humana

Un puente H integrado de grado automotriz para modulación de poder.

Un sensor térmico integrado DS18b20 conectado por I2C de 0,20ºC de precisión

Un sensor térmico para estabilización del RTC y lectura de la temperatura ambiental

Una fuente de alimentación de PC, reemplazable rápidamente con enchufes Molex estándar.

**ANALIZSIS del PROGRAMA CTP**

Este código combina múltiples tareas de monitoreo, control y comunicación para gestionar un sistema de temperatura, ajustando configuraciones y reaccionando a eventos en tiempo real.

El programa de Arduino se encarga de monitorear y controlar sensores de temperatura y un dispositivo de calefacción y/o refrigeración con celdas Peltier.

## Estructura General

1. \*\*Inicialización y Lectura de Sensores\*\*:

- La función `loop()` se repite continuamente y comienza por llamar a `avisala()`y activa un pin (`pinTCrict`) por un breve período para realizar una señalización.

- Se desactiva un flag (`test = false`) y se inicializan variables (`Suma` y `SumaDisip`) para almacenar sumas acumuladas de lecturas de temperatura.

2. \*\*Medición de Temperaturas\*\*:

- Se toman 5 mediciones de temperatura usando 2 sensores DS18B distintos (referidos como `tempActual` y `TempDisip`). - Se acumulan las lecturas y luego se promedian.

- `tempActual` es la temperatura de una cámara específica y `TempDisip` la temperatura del disipador para evaluar el salto térmico del Peltier.

- Se realizan ajustes a las lecturas de temperatura (`adjSensor` se suma a las lecturas). Para corregir el desvío de lectura obtenido en la calibración de las sondas sensoras.

3. \*\*Manejo de Datos de Tiempo\*\*:

- Se obtiene la hora actual usando un módulo RTC (`rtc.now()`).

- Se lee la temperatura del RTC y se realiza un seguimiento del tiempo para determinar el reinicio periódico del sistema. - \*\*Reinicio del Sistema\*\*: Realiza un reinicio automático si se detecta un bloqueo o pasa un tiempo específico.

4. \*\*Comparación y Almacenamiento de Temperaturas\*\*:

- Si las temperaturas actuales difieren de las anteriores (`tempAnt` y `tempDisAnt`), se actualizan las variables y se incrementa el contador de lecturas (`nLectu`).

5. \*\*Lectura de Entradas y Gestión de Eventos\*\*:

- Se leen estados de varios pines (botones o eventos).

- Si se presiona un botón de evento (`E`), se activa un buzzer y se alterna el estado de `test`.para esteo durante el desroollo del programa.

6. \*\*Recepción de Datos por Serial\*\*:

- El sistema escucha en el puerto serie (`Serial.available() > 0`) y responde a comandos específicos recibidos:

- \*\*Comando '4'\*\*: Realiza una serie de señales y lee varios valores desde el puerto serie (`T1`, `T2`, `nModo`, `nEvento`, `s3`, `s4`, `s5`).

- \*\*Comando '2'\*\*: Inicia el sistema, activando un optoacoplador y el buzzer.

- \*\*Comando '5'\*\*: Ajusta variables específicas (`T1`, `T2`, `nModo`, `nEvento`, `s3`, `s4`, `s5`).

- \*\*Comando '7'\*\*: Ajusta la hora del RTC leyendo valores específicos desde el puerto serie.

- \*\*Comando '9'\*\*: Realiza un reinicio del sistema y guarda datos críticos en memoria antes de reiniciar.

7. \*\*Control de Tareas y Estado\*\*:

- La función `rutMODO()` gestiona modos (rutinas térmicas) de operación del sistema.

- `pseudoPID\_log()` y `TERMOSTAT()` se usan para el control de temperatura basado en un PID (Proporcional-Integral-Derivativo) simplificado.

8. \*\*Reinicio Automático\*\*:

- Se deshabilita el watchdog timer (`wdt\_disable()`) y se verifica si ha pasado suficiente tiempo (15 minutos) para realizar un reinicio automático del sistema., Para prevenir rebase de memoria y otras irregularidades aleatorias.

### Funciones y Operaciones Clave

- \*\*Lectura de Temperaturas\*\*: Usa sensores DS18B20 para leer y promediar temperaturas.

- \*\*Comunicación Serie\*\*: Escucha y responde a comandos específicos recibidos por el puerto serie.

- \*\*Control de Estados y Eventos\*\*: Gestiona entradas digitales y eventos, incluyendo ajustes de tiempo y variables críticas.

- \*\*Control de Temperatura\*\*: Utiliza un sistema de control que incluye un PID para mantener la temperatura deseada.

------------------------

Descripción general

Este programa de Arduino tiene varias funcionalidades relacionadas con la lectura de sensores de temperatura, la gestión de eventos y el manejo de comandos recibidos a través del puerto serie. Se utiliza un sensor de temperatura DS18B20 y un módulo RTC (reloj en tiempo real) para obtener la hora actual. Además, el programa puede recibir y procesar comandos desde un entorno externo, como Processing.

Análisis detallado del código

1. Inicialización y Configuración

Variables y objetos: El código asume la existencia de variables globales y objetos para manejar los sensores y el RTC.

Sensores de temperatura: Se utiliza un sensor DS18B20 para medir la temperatura en diferentes puntos.

2. Función loop()

La función loop() es el núcleo del programa y se ejecuta continuamente. Aquí se realizan las siguientes acciones:

Aviso inicial y manipulación de pines:

avisala();

digitalWrite(pinTCrict, HIGH);

digitalWrite(pinTCrict, LOW);

Lectura de sensores:

Se inicializan variables para la suma de temperaturas.

Se toman cinco mediciones de temperatura y se calculan promedios.

Se realiza la compensación de las temperaturas medidas utilizando un coeficiente de calibración (adjSensor).

Se lee la temperatura actual (tempActual) y la temperatura del disipador (TempDisip).

Obtención de la hora actual del RTC:

DateTime now = rtc.now();

Lectura de temperatura del RTC:

TempAMBIENT = rtc.getTemperature();

Envío de datos:

Si hay un cambio en las temperaturas leídas, se actualizan y se envían los datos. Ia seral al programa de monitoreo grafico que corre en prossesing.

Lectura de estados de pulsadores:

Se leen los estados de varios pines que representan pulsadores (pinSTART, pinABORT, pinEVENTO).

Si se detecta un evento, se activa un buzzer y se alterna el estado de la variable test.

Procesamiento de comandos recibidos por el puerto serie:

Se verifica si hay datos disponibles en el puerto serie.

Según el comando recibido (representado por dataIN), se ejecutan diferentes acciones:

Comando '4': Realiza pruebas y emite señales con el buzzer.

Se activa y desactiva pinTCrict varias veces.

Se emiten sonidos con el buzzer.

Se leen y procesan varios valores de temperatura.

Comando '2': Inicia una operación (START).

Se activa un relé (pinOPTOCUPLER).

Se emiten sonidos con el buzzer para indicar el inicio.

Comando '5': Configura variables (DATASET).

Se leen varias configuraciones enviadas por el puerto serie y se almacenan en variables.

Se graban estos valores en memoria.

Comando '7': Ajusta la hora del RTC según los datos recibidos.

Comando '9': Realiza un reset del sistema.

Se activa pinTCrict durante un periodo prolongado.

Se emiten sonidos con el buzzer.

Se realiza un reinicio del sistema.

Lógica del termostato y control del sistema:

Se ejecuta una función para manejar modos o rutinas termicas (rutMODO()).

Se realiza un control de temperatura (TERMOSTAT(Tinstant, tempActual, Tmargen)).

Se maneja un PID lógico (pseudoPID\_log()).

**Consideraciones**

Calibración de sensores: El ajuste de la lectura de los sensores con adjSensor es importante para obtener datos precisos.

Manejo de eventos: El uso de pulsadores físicos y comandos serie permite un control flexible del sistema.

Robustez: La inclusión de un reset automático cada 15 minutos ayuda a prevenir bloqueos del sistema.

**Especificaciones Técnicas:**

Salida DC pulsante a 600 Hz, variable por ancho de pulso- PWM (pulse with modulation)

Algoritmos para estabilizar el comportamiento y evitar oscilaciones de la temperatura.

Control de histéresis, Rango de acción y Poder relativo entre frío y calor en un archivo de ajuste de comportamiento llamado  *PARAMETROR.INO*

Control electrónico de PowerOn de fuente desde el microprocesador

Salida PWM alimentación placa Peltier= 11.5 volts a 15 Amperes.

Salida 12VDC controlada por MosFet para alimentar Led de fotoperiodo en cámara.

Salida 12VDC para alimentar ventiladores (on con PowerOn)

Salida 5VDC conector usb-C, para alimentar batería de datalogger, ,etc .

Salida 5VDC permanente standBy para alimentar microprocesador.

**El archivo Parametror.ino**

**amanecer 8hs01 anochecer 20hs01**