Actividad FreeRTOS

Andrea Zarahi Rubio Quezada | A01645257

Fatima Alvarez Nuño | A01645815

Gustavo Alexander Nuño Corvera | A01644775

Design Of Advanced Embedded Systems

Termostato FreeRTOS STM32H755ZI-Q

1. Codigo Base del Termostato

Inspeccion Inicial

Arquitectura & capas

- App (tareas): `vTaskSimulador`, `vTaskControl`, `vTaskUART`, `UARTStatusTask`
- RTOS: FreeRTOS (colas, semáforos, delays)
- HAL/SDK NXP: `BOARD_Init...`, `PRINTF` (fsl_debug_console)
- Bare-metal: `UART0_Init()`, `UART0_IRQHandler()` (registros SIM/UART)

Lógica de control

- Tipo: On/Off con histéresis (no hay PID).
- Parámetros: `HISTERESIS = 0.5`, `setpoint` por comando `SET xx`.

Entradas/Salidas

- Entrada: sensor simulado (modelo térmico discreto en `vTaskSimulador`).
- Comandos: UART (`STATUS`, `SET xx`, `ON`, `OFF`).
- Salidas: flag `calefactor_on` (podrías mapearlo a LED o PWM/relay).

Temporización

- Simulador cada **500 ms** (`vTaskDelayUntil`).
- Control cada **1000 ms**.
- Telemetría cada **2000 ms**.

Supuestos

- Unidades: °C
- Límites razonables: `T_AMBIENTE=25`, `T_HEATER=80`
- Modelo: 1er orden con `ALPHA` (calentamiento) y `BETA` (pérdida)

Diagrama de capas

```
graph TB

  subgraph APP["Aplicación / Lógica"]
    SIM["vTaskSimulador (modelo termico)"]
    CTRL["vTaskControl (ON/OFF + histeresis)"]
    UART_TASK["vTaskUART + procesar_comando"]
    STAT["UARTStatusTask"]
  end

  subgraph RTOS["Servicios / RTOS"]
    FRTOS["FreeRTOS: tasks / colas / semaforos / timers"]
  end
```

```
subgraph HAL["Drivers / HAL SDK"]
  BOARD["BOARD_InitBootPins / Clocks / Debug"]
  PRINTF_NODE["PRINTF (fsl_debug_console)"]
end

subgraph BM["Bare-metal directo"]
  UINIT["UART0_Init()"]
  UIRQ["UART0_IRQHandler (registros SIM / UART)"]
end

subgraph HW["Hardware MCU"]
  UART["UART0"]
  GPIO["GPIO"]
  CLK["Clock"]
end

APP --> RTOS --> HAL --> BM --> HW
```

Statecharts (UML) y mapeo a FreeRTOS

```
stateDiagram-v2
  [*] --> Apagado

  Apagado --> Espera: power_on / setpoint_valido

  Espera --> Calefaccion: [temp < setpoint - hyst] / calefactor_on()
  Calefaccion --> Espera: [temp > setpoint + hyst] / calefactor_off()

  Espera --> Falla: sensor_timeout || limite_seguridad
  Calefaccion --> Falla: sensor_timeout || limite_seguridad

  Falla --> Apagado: reset_falla
```

```
state Espera {
  [*] --> Idle
  Idle: entry / calefactor_off()
}
state Calefaccion {
  [*] --> On
  On: entry / calefactor_on()
  On: exit  / calefactor_off()
}
```

Mapeo “statechart → FreeRTOS”

El diagrama de estados del termostato (Apagado, Espera, Calefacción y Falla) se implementa en FreeRTOS mediante una tarea periódica denominada `vTaskControl`, la cual ejecuta un ciclo de control cada 50–200 ms. Dentro de esta tarea se mantiene una variable de estado (`state_t`) y se evalúan las transiciones con base en las condiciones de histéresis y los eventos recibidos.

Los eventos del statechart se modelan con mecanismos de FreeRTOS:

- Las nuevas mediciones (sensor real o simulador) se envían a la tarea de control mediante una cola.
- Los comandos recibidos por UART (`SET`, `ON`, `OFF`, `RESET`) son procesados en la tarea `vTaskUART` y enviados también por cola a `vTaskControl`.
- El timeout de sensor se implementa con un software timer, que al expirar genera un evento hacia la tarea de control.
- Los límites de seguridad (sobretemperatura o falla de sensor) se verifican dentro de la propia tarea de control en cada iteración.

Las acciones definidas en el statechart se traducen directamente a funciones en el código:

- `calefactor_on()` y `calefactor_off()` se implementan a través de GPIO o PWM usando el HAL de STM32 para controlar el actuador.

- La acción `notify_ui()` (se utiliza `notify_ui()` ya que es una simplificacion que representan la notificacion a la parte de la interfaz/telemetria) se realiza mediante el envío de mensajes a una tarea de telemetría o interfaz de usuario, usando colas de FreeRTOS.

De esta manera, cada elemento del modelo UML encuentra su correspondencia en FreeRTOS: estados en la máquina de estados de la tarea de control, eventos manejados con colas y timers, y acciones implementadas con funciones sobre el hardware. Este mapeo asegura una relación clara entre el diseño formal (statechart) y la implementación concurrente en el sistema operativo en tiempo real.

Code:

https://github.com/Andyrubio5/Embedded-Systems/tree/main/FreeRTOS/FreeRTOS_Act