Nano-Kernel Bare-Metal para RP2040

Requerimientos

Matías Cajal

Universidad Nacional de Río Negro

Laboratorio de Sistemas Embebidos

12 de Abril de 2025

Índice

1) Presentación

Este proyecto se enmarca en la necesidad de comprender los fundamentos de los sistemas operativos y la ejecución de software directamente sobre el hardware bare-metal, sin las abstracciones proporcionadas por bibliotecas estándar o Kits de Desarrollo de Software (SDK). El objetivo es obtener un conocimiento práctico y profundo sobre cómo inicializar un microcontrolador desde cero, gestionar recursos básicos como el tiempo de CPU y la memoria, controlar periféricos mediante acceso directo a registros, manejar interrupciones e implementar mecanismos de concurrencia simples, utilizando C y Assembler como herramientas principales. Se busca una experiencia de aprendizaje intensiva en la arquitectura del procesador (ARM Cortex-M0+) y los periféricos del SoC (RP2040), dependiendo exclusivamente de la documentación técnica oficial (datasheets).

2) Descripción general

El sistema resultante será un pequeño núcleo de software experimental (nano-kernel), diseñado específicamente para ejecutarse directamente sobre el hardware del RP2040. Pretende ser una plataforma base que demuestra la viabilidad de implementar multi-tarea pre-emptive básica y una interfaz de consola serie interactiva.

2.1) Funcionalidades del Producto

2.1.1) Planificación de tareas pre-emptive.

2.1.2) Consola serie interactiva (shell) con comandos básicos.

2.1.3) Control de pines GPIO.

2.1.4) Manejo de interrupciones.

2.2) Características de los Usuarios

El sistema está destinado principalmente al desarrollador con fines educativos y de experimentación con conceptos de bajo nivel. No posee una interfaz de usuario final más allá de la consola serie.

3) Requerimientos específicos

3.1) Requerimientos de interfaz externa

3.1.1) Interfaces de usuario
Se habilitará una interfaz de línea de comandos Shell Serie vía puerto serie con las siguientes características:

3.1.1.1) Comunicación vía UART0 a 115200 baudios.

3.1.1.2) Lectura de línea de comando.

3.1.1.3) Soporte para los siguientes comandos mínimos:

3.1.1.3.1) Help
Formato: help Lista comandos disponibles.

3.1.1.3.2) Peek
Formato: peek <addr_hex> Lee e imprime 4 de memoria en la dirección dada.

3.1.1.3.3) Poke
Formato: poke <addr_hex> <val_hex> Escribe 4 bytes en la dirección dada.

3.1.1.3.4) Gpio set
Formato: gpio set <pin> <0|1> Poner un pin GPIO en bajo/alto.

3.1.1.3.5) Show tasks
Formato: show-tasks Listar tareas activas y su estado.

3.1.2) Interfaces de hardware

3.1.2.1) Microcontrolador
Se realizará interacción directa con registros RP2040.

3.1.2.2) Periféricos Internos Usados
Se utilizarán al menos GPIO, UART0, TIMER, NVIC, RESETS, PADS_BANK0, IO_BANK0, SIO.

3.1.2.3) Periféricos Externos Mínimos
Se requiere conexión PC vía USB-Serie TTL y el LED onboard (GP25).

3.1.3) Interfaces de conexión
El Puerto Serie UART0 implementado manualmente será la interfaz primaria.

3.2) Requerimientos funcionales

3.2.1) Requerimientos generales

3.2.1.1) Scheduler
Implementará planificación pre-emptive basada en tick de Timer (IRQ), política Round-Robin simple, gestión de TCBs/stacks, e invocará rutina Assembler para cambio de contexto.

3.2.1.2) Shell
Se ejecutará como tarea concurrente, leerá comandos desde UART, y parseará/ejecutará los comandos definidos.

3.2.1.3) Drivers

3.2.1.3.1) GPIO

3.2.1.3.2) UART

3.2.1.3.3) Timer

3.2.1.3.4) NVIC (Habilitación de interrupciones)

3.3) Requerimientos de desempeño

3.3.1) Tiempos de respuesta y Uso de Recursos

3.3.1.1) Latencia de cambio de contexto
Objetivo $< 200mus$.

3.3.1.2) Overhead del Scheduler
Objetivo $< 5\%$ CPU.

3.3.1.3) Tiempo de respuesta de la Shell
Interactivo (objetivo < 1s para comandos simples).

3.3.1.4) Uso de RAM
Minimizado, stacks de tamaño fijo.

3.4) Restricciones de diseño

3.4.1) Prohibido el uso de cualquier componente del SDK C/C++ de Pico.

3.4.2) Interacción con HW solo vía definiciones manuales basadas en datasheet.

3.4.3) Sin biblioteca C estándar
Funciones auxiliares implementadas manualmente si son necesarias.

3.4.4) Cambio de contexto implementado obligatoriamente en Assembler.

3.4.5) Código de arranque y linker script desarrollados manualmente.

3.5) Atributos del sistema de software

3.5.1) Modularidad
Se buscará una separación lógica entre módulos (scheduler, shell, drivers).

3.5.2) Testabilidad
La shell interactiva será la herramienta principal para pruebas funcionales.

3.6) Otros requerimientos

No aplica