Proyecto Final
Curso de Sistemas Operativos y Laboratorio

Desarrollo de un Módulo de Kernel para Control de Hardware (GPIO) en Raspberry Pi

John Edison Zapata Ramirez

Resumen

Este documento de propuesta presenta la investigación y el diseño de un módulo de kernel para Linux en Raspberry Pi que permite controlar dispositivos externos a través de los pines GPIO. Se detallan los fundamentos del subsistema GPIO en kernel space, la configuración mediante Device Tree, y el desarrollo en C de un driver que gestione un sensor DHT22 y una matriz de LEDs. Se establecen las herramientas, la metodología y el cronograma de trabajo, así como métricas de desempeño (latencia, manejo de interrupciones).

Introducción

Necesidad o problema

En la actualidad, los sistemas embebidos y dispositivos de bajo consumo se han convertido en la base de soluciones inteligentes para el hogar, la industria y el análisis ambiental. Estos sistemas requieren una comunicación confiable y precisa con sensores y actuadores externos. El sensor DHT22, por su bajo costo y facilidad de uso, se emplea ampliamente para medir temperatura y humedad. Sin embargo, la gestión de este sensor a través de bibliotecas en espacio de usuario (p. ej., RPi.GPIO) presenta latencias variables y falta de garantías de tiempo real, lo que puede traducirse en lecturas erráticas o pérdida de datos en aplicaciones críticas. Por tanto, existe la necesidad de desarrollar un controlador en espacio kernel que garantice el determinismo temporal (manejo de señal con microsegundos de resolución y mínimo jitter),

acceso directo a hardware (lectura/escritura de registros GPIO sin pasar por capas de espacio de usuario), y además la gestión eficiente de interrupciones (priorización de eventos y respuesta inmediata a cambios de estado del sensor). Esto permite asegurar la integridad de los datos y ampliar el rango de aplicaciones del sensor en entornos donde la precisión y la robustez son imprescindibles.

Importancia en el contexto tecnológico actual

La revolución del Internet de las Cosas (IoT) y la Industria 4.0 ha impulsado la adopción de plataformas embebidas como la Raspberry Pi en proyectos de monitoreo ambiental, domótica y automatización industrial. En estos escenarios, la fiabilidad y la capacidad de respuesta de los dispositivos determinan la calidad del servicio, por lo tanto implementar un controlador de kernel para GPIO en la Raspberry Pi ofrece ventajas competitivas al combinar la flexibilidad de la plataforma con la fiabilidad de un driver de nivel bajo. Además, profundiza en conocimientos críticos de sistemas operativos, como la manipulación de interrupciones, la gestión de memoria y la seguridad en núcleo, alineándose directamente con los objetivos académicos y profesionales del curso.

Antecedentes o marco teórico

Para el desarrollo de este proyecto es fundamental comprender la arquitectura y funcionamiento del subsistema GPIO en Linux, el papel del Device Tree en la descripción de hardware y las técnicas de manejo de interrupciones. Estos conceptos son la base para desarrollar un módulo de kernel que interactúe directamente con los pines de la Raspberry Pi, garantizando eficiencia y fiabilidad.

Relación con el curso de Sistemas Operativos

El desarrollo de un módulo de kernel para GPIO en Raspberry Pi refuerza varios conceptos centrales del curso de Sistemas Operativos. Al implementar un controlador en C dentro del kernel, se trabaja directamente con estructuras internas y llamadas de bajo nivel, lo que ilustra cómo el sistema operativo abstrae la complejidad del hardware y ofrece interfaces uniformes. Además registrar una rutina de servicio de interrupción permite comprender cómo el kernel prioriza y encola eventos externos, y enseña a manejar condiciones de carrera y sincronización para evitar errores al acceder a recursos compartidos así como el control de acceso a los recursos gestionados por el kernel .

Objetivos

Objetivo general
Desarrollar un módulo de kernel en C para la Raspberry Pi que permita el control fiable y eficiente de un sensor DHT22 y una matriz de LEDs a través de los pines GPIO, con soporte para operaciones de lectura y escritura.

Objetivos específicos

1. Investigar y describir la arquitectura interna del subsistema GPIO en Linux, incluyendo estructuras `gpio_chip` y llamadas al API del kernel.
2. Programar las operaciones de lectura del DHT22 y el registro de interrupciones para capturar datos con precisión.
3. Diseñar y codificar las rutinas de escritura en pines GPIO para encender, apagar y desplegar patrones en la matriz de LEDs.
4. Implementar las estructuras `file_operations` para soportar llamadas `open`, `read`, `write` y `ioctl`, documentando los códigos de operación y argumentos.
5. Realizar mediciones de latencia y jitter, validando la integridad de los datos y el tiempo de respuesta bajo carga.
6. Generar un manual de usuario y ejemplos de scripts en espacio de usuario que demuestren la instalación, configuración y consumo de datos desde el dispositivo.

Metodología

Se combinará investigación práctica y desarrollo experimental. Empleando un entorno de compilación cruzada sobre Ubuntu y Raspberry Pi. Para validar el driver antes de usar hardware real, se integrará QEMU con un kernel adaptado, complementado con pruebas posteriores en una Raspberry Pi 4 con sensor DHT22 y matriz de LEDs. Algunas de las actividades claves durante el desarrollo del proyecto serán el estudio del subsistema GPIO y Device Tree, el desarrollo de un módulo de prueba (LED blink) y la implementación de lectura de DHT22, al igual que el control de LEDs.

Cronograma

Fase	Sem 1–2	Sem 3–4	Sem 5	Sem 6	Sem 7	Sem 8
Investigación teórica	X					
Configuración toolchain		X				
Módulo LED blink		X				
Lectura DHT22			X			
Interfaz y ioctl				X		
Pruebas de latencia					X	
Documentación y entrega						X

Referencias

1. Corbet, J., Rubini, A., & Kroah-Hartman, G. (2005). *Linux Device Drivers*, 3ª ed. O'Reilly.
2. Raspberry Pi Foundation. (2024). *Device Tree and Overlays*. https://www.raspberrypi.org
3. Aosong Electronics Ltd. (2015). *DHT22 Datasheet*.
4. Brey, B. B. (2019). *The Linux Kernel Module Programming Guide*.