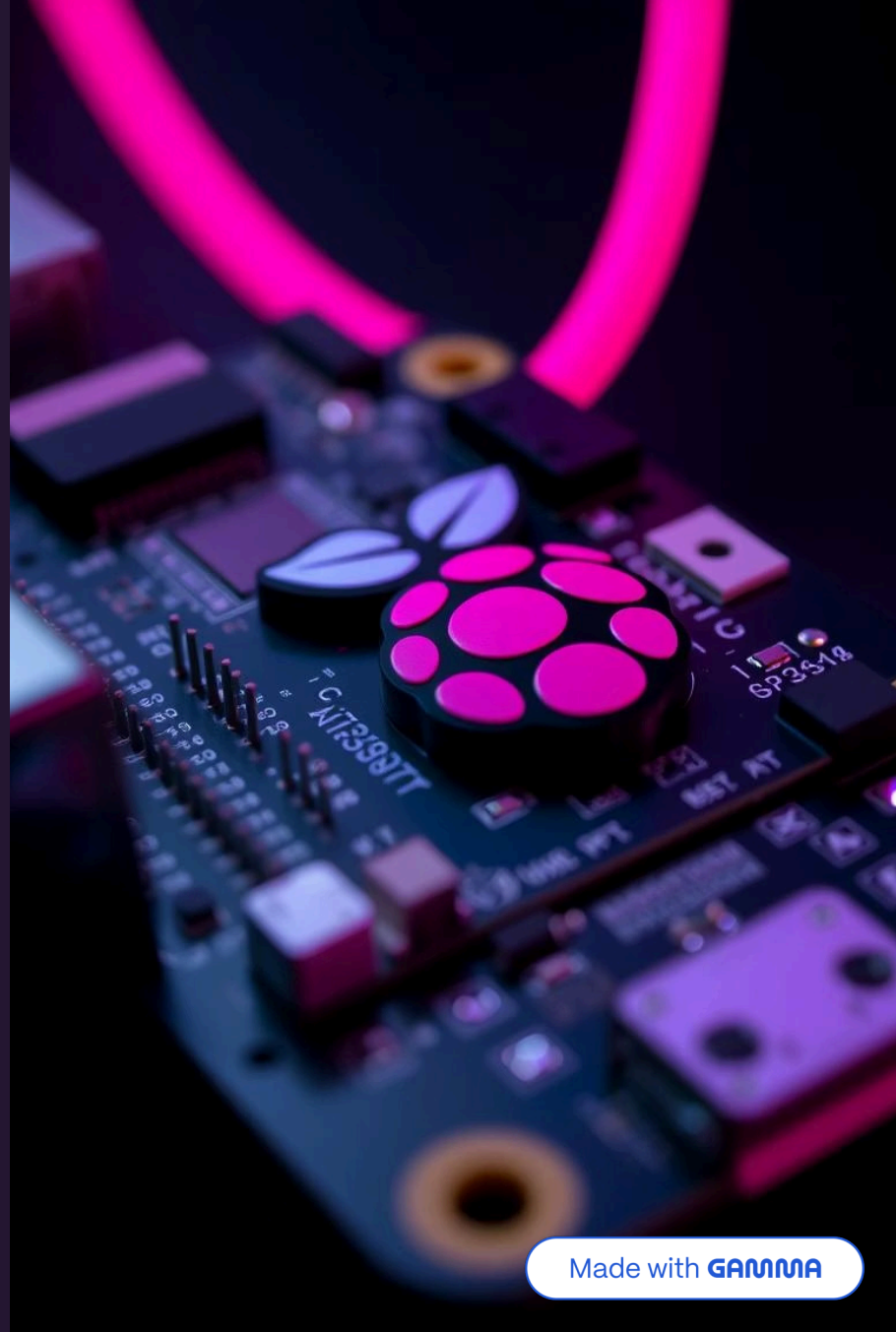
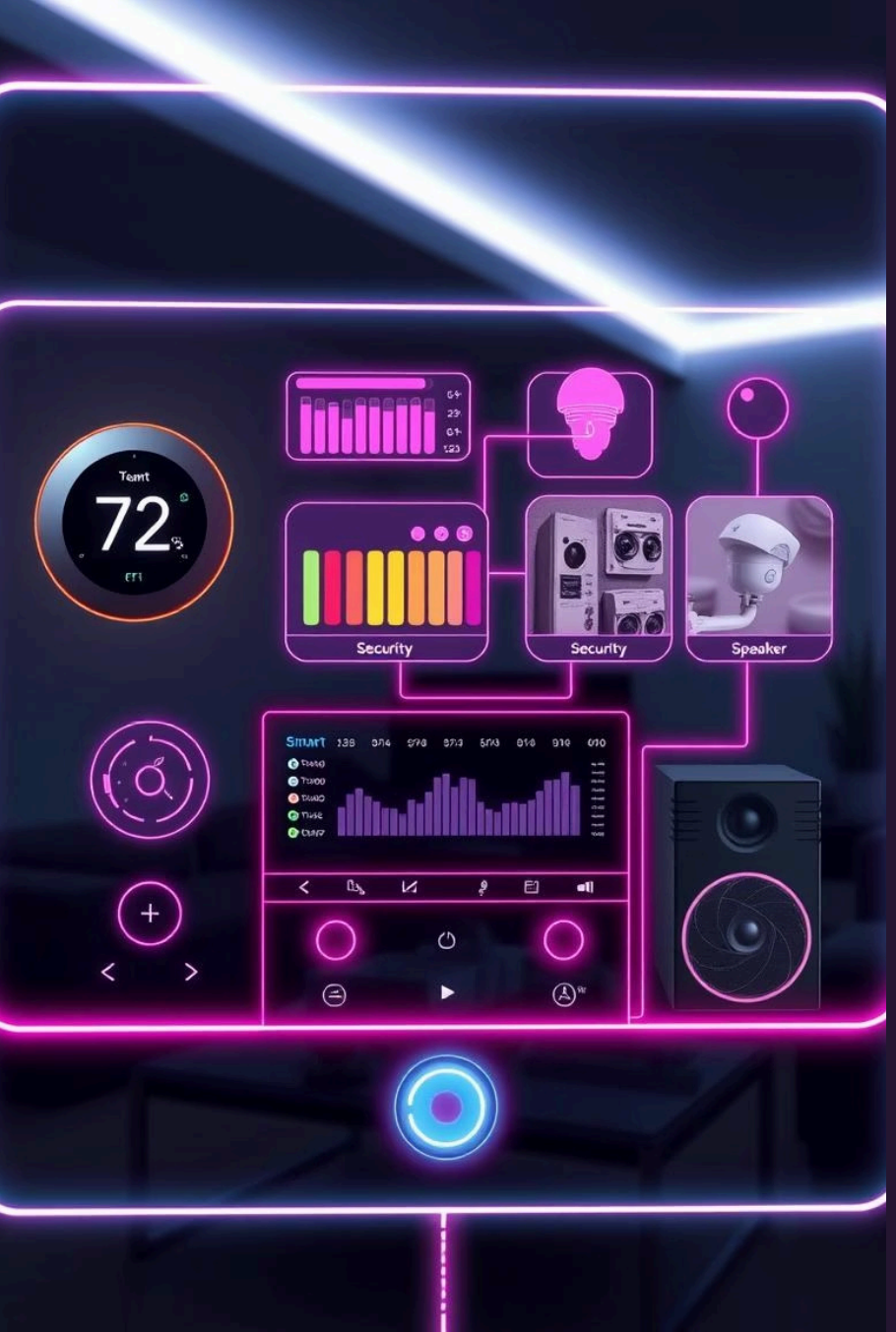
Desarrollo de un Módulo de Kernel para Control de Hardware (GPIO) en Raspberry Pi

Este proyecto final del curso de Sistemas Operativos propone el diseño de un módulo de kernel para Linux en Raspberry Pi que controla dispositivos externos mediante pines GPIO. Se aborda la configuración con Device Tree y el desarrollo en C de un driver para un sensor DHT22 y una matriz de LEDs.

El objetivo es garantizar un control fiable y eficiente con manejo de interrupciones y latencia mínima, alineado con la revolución del IoT y la Industria 4.0, aportando robustez y precisión en sistemas embebidos.

E **by Edison Zapata**





Contexto y Necesidad del Proyecto

Problema Actual

El uso de bibliotecas en espacio usuario para el sensor DHT22 presenta latencias variables y falta de garantías en tiempo real, causando lecturas erráticas.

Necesidad Técnica

Desarrollar un controlador en espacio kernel que ofrezca determinismo temporal, acceso directo a hardware y gestión eficiente de interrupciones para asegurar integridad de datos.

Importancia Tecnológica

El proyecto fortalece la fiabilidad en aplicaciones IoT y automatización, combinando flexibilidad de Raspberry Pi con la robustez de un driver de bajo nivel.

Marco Teórico y Relación con el Curso

Subsistema GPIO en Linux

Comprensión de la arquitectura y funcionamiento para interactuar directamente con pines GPIO.

Device Tree

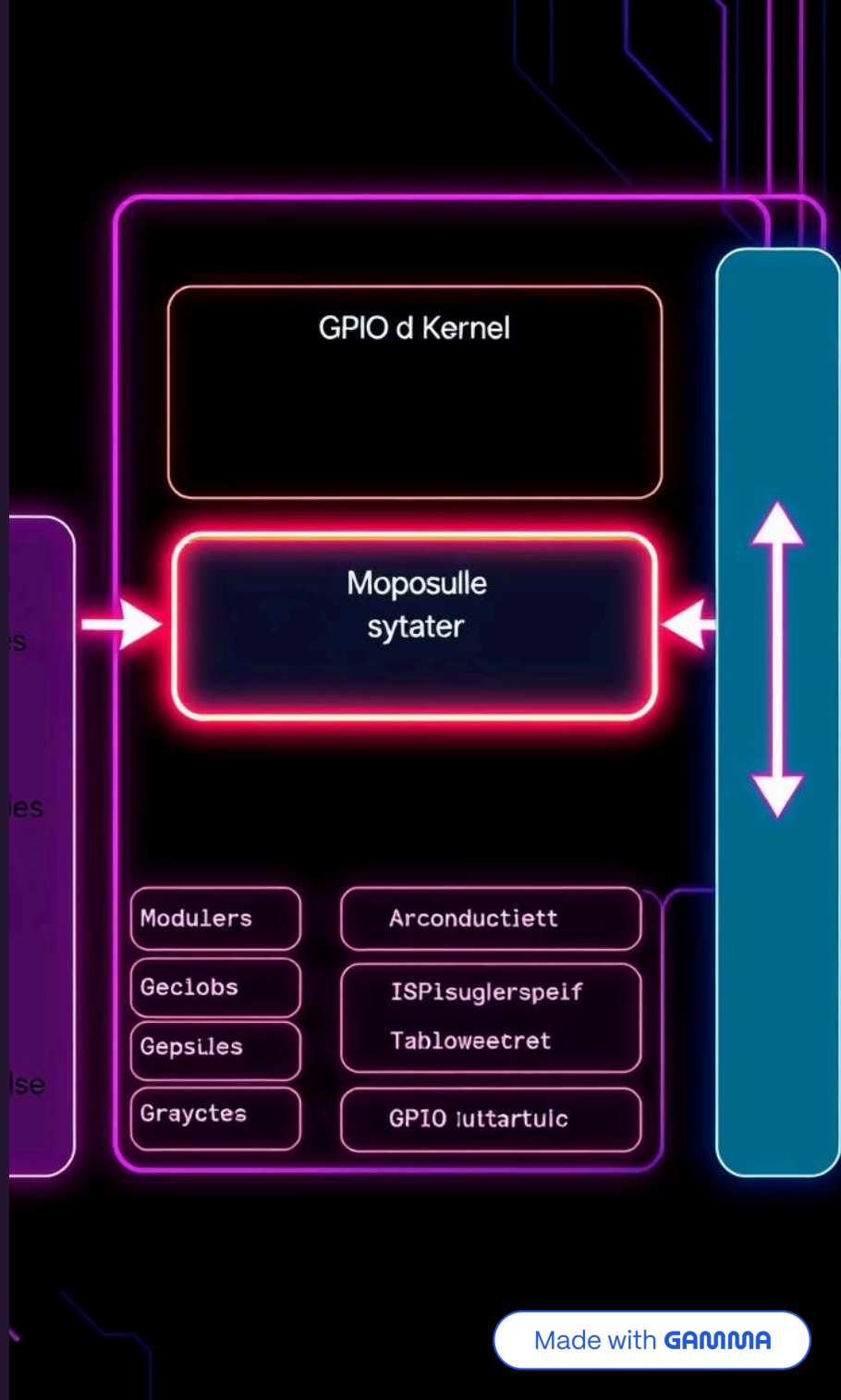
Uso del Device Tree para describir hardware y facilitar la configuración del módulo.

Manejo de Interrupciones

Registro y gestión de interrupciones para priorizar eventos y evitar condiciones de carrera.

Conceptos del Curso

Refuerza estructuras internas, llamadas de bajo nivel y sincronización en el kernel.



Objetivo General y Específicos

Objetivo General

Desarrollar un módulo de kernel en C para controlar un sensor DHT22 y una matriz de LEDs mediante GPIO en Raspberry Pi.

Objetivos Específicos

- Investigar arquitectura interna del subsistema GPIO y API del kernel.
- Programar lectura precisa y registro de interrupciones del DHT22.
- Diseñar rutinas para control de LEDs en GPIO.
- Implementar file_operations para soporte de llamadas open, read, write e ioctl.
- Medir latencia y jitter para validar integridad y tiempo de respuesta.
- Crear manual de usuario y scripts de ejemplo en espacio usuario.

Metodología de Desarrollo

1

Investigación y Estudio

Analizar subsistema GPIO y Device Tree para fundamentar el desarrollo.

2

Desarrollo Experimental

Crear módulo de prueba para parpadeo de LEDs y lectura del sensor DHT22.

3

Validación con QEMU

Usar emulación con kernel adaptado para pruebas iniciales sin hardware real.

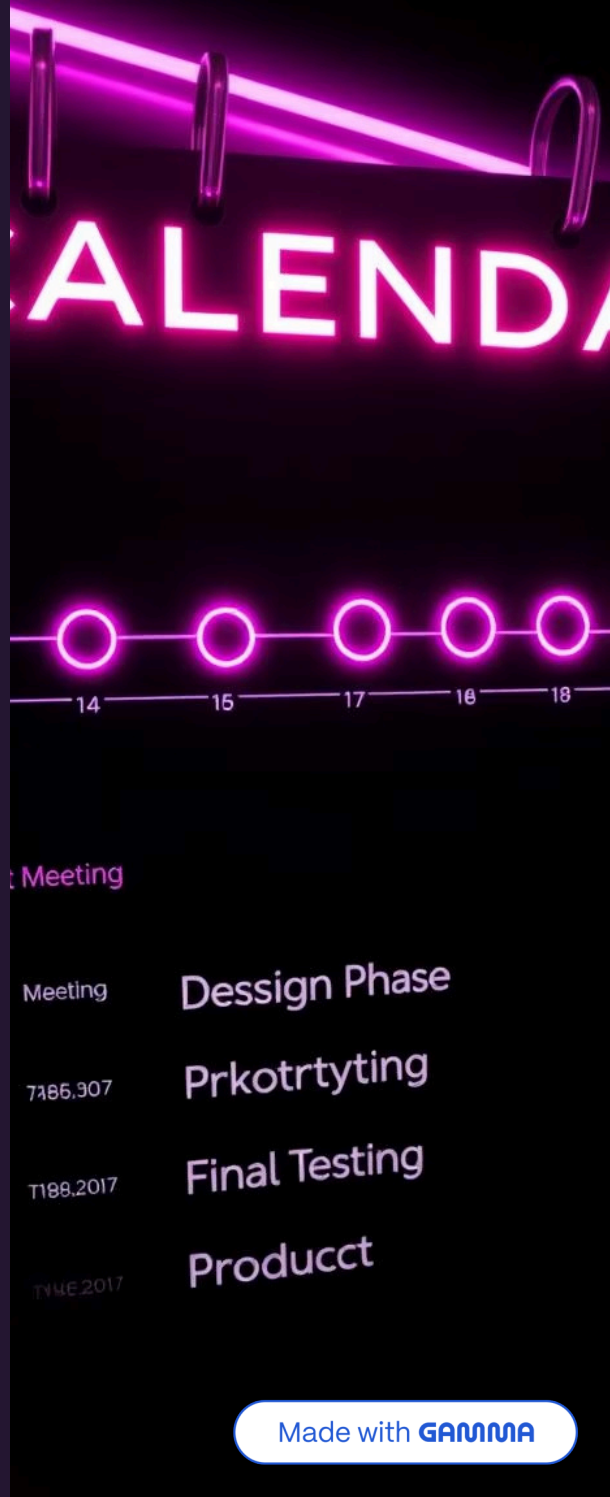
4

Pruebas en Hardware Real

Ejecutar pruebas en Raspberry Pi 4 con sensor y matriz de LEDs para validar funcionalidad.

Cronograma de Trabajo

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 Bibliográficas

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*.

Conclusiones y Próximos Pasos

Implementación del Módulo

Desarrollar y probar el driver en kernel para control preciso de GPIO y sensores.

Validación y Optimización

Medir latencia y jitter para asegurar rendimiento en tiempo real.

Documentación y Difusión

Crear manuales y ejemplos para facilitar uso y replicación del módulo.

Aplicaciones Futuras

Extender el controlador a otros sensores y actuadores en proyectos IoT e industriales.