Materia:

DISEÑO ELECTRÓNICO BASADO EN SISTEMAS EMBEBIDOS

Alumno:

Posadas Pérez Isaac Sayeg

Paniagua Rico Juan Julian

García Azzúa Jorge Roberto

Grado y grupo:

8°G

Profesor:

Garcia Ruiz Alejandro Humberto

Unidad 4 -Tarea 7:

ARQUITECTURA DE MICROSERVICIOS EN SISTEMAS EMBEBIDOS

ARQUITECTURA DE MICROSERVICIOS EN SISTEMAS EMBEBIDOS

Introducción

Tradicionalmente, los sistemas embebidos han sido diseñados como sistemas monolíticos, donde todo el software —desde la lectura de sensores hasta la lógica de control y comunicación— se implementa en una única aplicación ejecutada sobre un microcontrolador o procesador embebido. Sin embargo, con la creciente complejidad de estos sistemas, especialmente en aplicaciones como IoT, robótica, automoción y sistemas industriales inteligentes, surge la necesidad de adoptar arquitecturas más modulares y escalables. En este contexto, la **arquitectura de microservicios** comienza a tener un papel relevante incluso en entornos embebidos.

Desarrollo

Aplicar microservicios en sistemas embebidos implica **repensar la arquitectura del software** para distribuir responsabilidades entre múltiples componentes autónomos que se comunican entre sí, incluso dentro de una misma placa o red local de dispositivos. Esto puede lograrse a través de distintas estrategias:

1. **Microservicios distribuidos entre nodos embebidos:** Cada nodo (por ejemplo, una Raspberry Pi, ESP32, etc.) implementa un microservicio con una función específica: adquisición de datos, procesamiento, almacenamiento local o comunicación con la nube.

2. **Microservicios dentro de un RTOS (sistema operativo en tiempo real):** En sistemas más complejos, donde se utiliza un RTOS como FreeRTOS o Zephyr, cada tarea puede comportarse como un microservicio, ejecutándose de forma concurrente y comunicándose mediante colas de mensajes, eventos o sockets.

3. **Contenedores embebidos (como Docker en ARM):** En hardware embebido más potente (por ejemplo, en gateways IoT con Linux), es posible usar Docker o Podman para contenerizar microservicios y desplegarlos como en sistemas tradicionales de TI.

Ventajas potenciales:

- **Modularidad:** permite desarrollar, probar y mantener funciones por separado.
- **Actualización remota (OTA):** facilita actualizar servicios específicos sin interrumpir el sistema completo.
- **Escalabilidad funcional:** cada microservicio puede evolucionar de forma independiente.
- **Interoperabilidad:** mejora la integración con otros sistemas o dispositivos.

Desafíos:

- **Limitaciones de recursos:** CPU, RAM y energía limitadas dificultan la ejecución de múltiples servicios o contenedores.
- **Comunicación eficiente:** el uso de protocolos livianos (MQTT, CoAP) es clave.

- **Tiempo real:** la introducción de capas de comunicación entre servicios puede afectar el tiempo de respuesta si no se diseña cuidadosamente.

- **Seguridad distribuida:** cada microservicio necesita mecanismos propios de autenticación y validación.

Ejemplo:

Un sistema de agricultura inteligente basado en microservicios podría estar compuesto por:

- Un **microservicio de sensores** en una placa ESP32 que lee temperatura, humedad y luz.

- Un **microservicio de control** en una Raspberry Pi que toma decisiones sobre riego basadas en los datos de sensores.

- Un **microservicio de comunicación** que envía los datos a la nube mediante MQTT.

- Un **microservicio de interfaz web** que permite visualizar datos y controlar manualmente el sistema desde un navegador o app móvil.

Cada uno de estos servicios puede actualizarse individualmente y desarrollarse por distintos equipos. Además, si el nodo de sensores falla, los demás servicios continúan operando, mejorando la resiliencia del sistema.

Conclusión

La arquitectura de microservicios en sistemas embebidos representa una evolución necesaria para abordar la creciente complejidad, escalabilidad y conectividad que demandan las aplicaciones modernas. Aunque su implementación en entornos con

recursos limitados supone desafíos técnicos importantes, el avance del hardware embebido, los sistemas operativos ligeros y los protocolos eficientes permiten su adopción efectiva. Esta arquitectura mejora la modularidad, facilita el mantenimiento y permite una mayor interoperabilidad con entornos de nube, lo cual es esencial en escenarios como IoT, industria 4.0 o sistemas autónomos distribuidos.

Bibliografía

1. Desrosiers, P. (2021). *Microservices in Embedded Systems*. IEEE Internet of Things Magazine.

2. Al-Fuqaha, A. et al. (2015). *Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications*. IEEE Communications Surveys & Tutorials.

3. Espressif Systems. (2023). *ESP-IDF Programming Guide*.

4. FreeRTOS. (2024). *Real-Time Operating System Overview and Design Patterns*.

5. Docker Inc. (2022). *Running Docker on ARM Embedded Devices*.

6. Li, S., Da Xu, L., & Zhao, S. (2015). *The internet of things: a survey*. Information Systems Frontiers.