Proyecto Final
Curso de Sistemas Operativos y Laboratorio

Análisis Comparativo de Sistemas Operativos para Microcontroladores en Aplicaciones de IoT: Rendimiento y Consumo en ESP32 y Raspberry Pi Pico

Miembros del equipo

Jhon Sebastian Usuga Ferraro

Resumen

La rápida expansión de aplicaciones de IoT y sistemas embebidos ha impulsado un crecimiento significativo en el uso de microcontroladores como el ESP32 y el Raspberry Pi Pico. Estos dispositivos, reconocidos por su bajo costo y eficiencia energética, son fundamentales para tareas de automatización, monitoreo y control en entornos domésticos, industriales y urbanos. Sin embargo, la elección del sistema operativo adecuado para cada microcontrolador resulta crucial para optimizar su rendimiento, consumo energético y compatibilidad con periféricos y protocolos de comunicación.

Con el aumento de aplicaciones en tiempo real y el uso de recursos limitados, como la memoria y la potencia de procesamiento, es necesario entender cómo diferentes sistemas operativos afectan el comportamiento de estos dispositivos. Desde sistemas minimalistas, como FreeRTOS, hasta entornos más versátiles como MicroPython, la variedad de opciones puede presentar un desafío significativo para desarrolladores y empresas que buscan una solución adecuada y confiable.

Introducción

El problema principal que aborda este proyecto es la falta de información comparativa y estandarizada sobre el desempeño de diferentes sistemas operativos para microcontroladores en términos de rendimiento y eficiencia. Al no contar con estudios detallados y accesibles que comparen estos sistemas, los desarrolladores se ven obligados a elegir con base en experiencias anecdóticas o en criterios limitados. Esta situación se traduce en sistemas de IoT menos eficientes, mayor consumo energético y, en ocasiones, fallos en aplicaciones críticas donde la fiabilidad y el rendimiento son esenciales.

En el contexto tecnológico actual, donde IoT y la automatización juegan un papel fundamental, optimizar el rendimiento y eficiencia de los sistemas embebidos es más relevante que nunca. La elección de un sistema operativo adecuado no solo impacta el funcionamiento y la vida útil de los dispositivos, sino que también afecta la sostenibilidad del ecosistema IoT, al reducir el consumo energético

Desarrollar este análisis estadístico y comparativo de sistemas operativos ayuda a promover soluciones embebidas que sean tanto funcionales como sostenibles, abordando así las demandas del contexto actual de desarrollo tecnológico y contribuyendo al avance de sistemas más eficientes y optimizados

Antecedentes o marco teórico

Para abordar el desafío de comparar y analizar sistemas operativos en microcontroladores como el ESP32 y el Raspberry Pi Pico, es fundamental entender ciertos aspectos teóricos relacionados con los sistemas operativos embebidos, su arquitectura, y los conceptos específicos de administración de recursos en dispositivos de bajo consumo y con limitaciones de hardware.

1. Sistemas Operativos para Microcontroladores

Los sistemas operativos embebidos son versiones simplificadas de sistemas operativos diseñados para gestionar los recursos en microcontroladores, los cuales poseen recursos limitados (CPU, memoria, y almacenamiento) en comparación con los sistemas operativos convencionales. Existen varias opciones de sistemas operativos para dispositivos de IoT y sistemas embebidos, como:

- **FreeRTOS**: Un sistema operativo en tiempo real (RTOS) ligero, orientado a tareas y con un enfoque en la multitarea. FreeRTOS es popular en aplicaciones de IoT por su capacidad de gestionar tareas concurrentes en microcontroladores.

- **MicroPython**: Un intérprete de Python optimizado para microcontroladores. Su sencillez permite desarrollar y depurar aplicaciones rápidamente, pero puede ser menos eficiente en términos de consumo y velocidad en comparación con RTOS.
- **Arduino Core**: Proporciona un entorno minimalista y optimizado, aunque con capacidades limitadas de multitarea en comparación con un RTOS.
- **Zephyr**: Otro sistema operativo en tiempo real diseñado para IoT, que ofrece flexibilidad y soporte de arquitecturas como ESP32 y ARM

Esta teoría se relaciona con el curso de sistemas operativos en diferentes aspectos, entre ellos están:

- **Manejo de Procesos y Multitarea**: Los procesos, hilos y multitarea es central para entender cómo los sistemas operativos embebidos manejan tareas concurrentes. La implementación y medición de la eficiencia de sistemas operativos en la ejecución de tareas concurrentes reflejan los conceptos teóricos aplicados.
- **Gestión de Memoria**: La administración de memoria es fundamental para comprender cómo un sistema operativo en un microcontrolador asigna memoria a distintas tareas y libera recursos de manera eficiente. Estos conceptos se aplican al observar cómo se distribuye la memoria en sistemas embebidos.
- **Estadística y Evaluación de Desempeño**: La estadística aplicada al análisis de sistemas operativos permite medir y comparar el rendimiento de manera objetiva. La recolección y análisis de datos de rendimiento en el laboratorio complementan la teoría, demostrando cómo las métricas de desempeño ayudan a evaluar la eficacia de los sistemas operativos en entornos embebidos..

Objetivo principal

- Investigar y comparar sistemas operativos utilizados en microcontroladores (ESP32 y Raspberry Pi Pico) en términos de rendimiento, consumo de energía, compatibilidad de periféricos y facilidad de uso en aplicaciones IoT y de control embebido. A través de la estadística, analizaremos qué opciones de sistemas operativos (como FreeRTOS, MicroPython, Arduino Core) resultan más eficientes y adecuados para distintos contextos.

Objetivos específicos

- Identificar y seleccionar sistemas operativos adecuados para los microcontroladores ESP32 y Raspberry Pi Pico en función de su popularidad y compatibilidad con aplicaciones de IoT

- Implementar y ejecutar pruebas de rendimiento en cada sistema operativo seleccionado, midiendo variables como tiempos de respuesta, estabilidad y eficiencia en el manejo de tareas concurrentes.
- Evaluar el uso de CPU y memoria bajo diferentes cargas de trabajo, observando cómo cada sistema operativo maneja los recursos limitados de los microcontroladores.

Metodología

Herramientas

- Microcontroladores: ESP32 y Raspberry Pi Pico.
- Herramientas de medición: Multímetros, osciloscopios, o cualquier dispositivo que permita medir el consumo de energía y rendimiento.
- Software estadístico: Herramientas como R y Python con bibliotecas pandas, matplotlib, seaborn y scipy para el análisis y visualización de datos.

1. Selección de Sistemas Operativos

- **Sistemas operativos**: Elige sistemas populares para estos microcontroladores, como:
  - **ESP32**: FreeRTOS, Arduino Core, MicroPython, Zephyr.
  - **Raspberry Pi Pico**: FreeRTOS, MicroPython, Pico SDK, Zephyr.
- **Variables clave**: Tiempo de respuesta, consumo de energía, capacidad de procesamiento, y compatibilidad de bibliotecas.

2. Definición de Experimentos

- **Pruebas de rendimiento**: Ejecutar tareas típicas en cada sistema operativo (lectura de sensores, manejo de comunicación inalámbrica) y medir métricas como tiempo de respuesta y eficiencia de memoria.
- **Consumo energético**: Medir el consumo de energía en reposo y durante la ejecución de tareas para cada sistema operativo.
- **Uso de CPU y memoria**: Comparar la utilización de CPU y memoria en tareas idénticas para ver cómo cada sistema gestiona los recursos limitados de los microcontroladores.

3. Recopilación de Datos

- **Herramientas de monitoreo**: Utilizar herramientas como osciloscopios, multímetros, y software de monitoreo de rendimiento embebido.
- **Frecuencia de muestreo**: Registrar métricas a intervalos regulares para análisis detallados.
- **Muestreo por tareas específicas**: Realizar pruebas en cada sistema operativo con tareas específicas y repetibles para asegurar consistencia en los datos.

4. Análisis Estadístico

- **Estadísticas descriptivas**: Calcular promedios, medianas y desviaciones estándar para cada métrica y sistema operativo.
- **Comparación de rendimiento**: Usar gráficos de barras y diagramas de cajas para mostrar diferencias entre los sistemas.
- **Análisis de varianza (ANOVA)**: Determinar si las diferencias en rendimiento y consumo de energía entre sistemas son estadísticamente significativas.
- **Regresión**: Si se tiene en cuenta la carga del sistema, ver cómo afecta la carga a las métricas de rendimiento y consumo.

5. Visualización de Resultados

- **Tablas y gráficos**: Mostrar comparativas de cada métrica por sistema operativo en tablas o gráficos de barras para facilitar la interpretación.
- **Diagramas de dispersión**: Relacionar el consumo de recursos con el tipo de tarea ejecutada en cada sistema.

Diseño de Experimentos

Para validar la funcionalidad y el rendimiento de los sistemas operativos en microcontroladores como el ESP32 y el Raspberry Pi Pico, se propone un diseño de experimentos que permitirá evaluar, bajo condiciones controladas, las principales métricas de rendimiento y consumo energético. Estos experimentos seguirán un enfoque comparativo y estadístico que garantice la validez y fiabilidad de los resultados.

1. Definición de Variables de Interés

Las variables que se evaluarán en los experimentos son:

- **Tiempo de respuesta (latencia)**: Tiempo que tarda el sistema operativo en responder a una interrupción o evento externo.
- **Consumo de memoria (RAM)**: Cantidad de memoria utilizada por el sistema operativo al ejecutar tareas.
- **Consumo de energía**: Energía consumida durante la ejecución de tareas en diferentes modos de operación (activo, inactivo, suspensión).
- **Uso de CPU**: Carga de procesamiento durante la ejecución de tareas concurrentes.
- **Estabilidad del sistema**: Frecuencia de fallos o errores bajo condiciones de carga

2. Selección de Sistemas Operativos y Configuración Experimental

Para realizar una comparación detallada, se seleccionarán los siguientes sistemas operativos:

- **FreeRTOS**
- **MicroPython**

- **Zephyr**

Cada sistema operativo será evaluado en ambos dispositivos, ESP32 y Raspberry Pi Pico, en condiciones controladas de hardware y software.

3. Pruebas y Escenarios de Evaluación

Para capturar un conjunto de datos confiable, se definirán distintos escenarios para simular aplicaciones comunes en IoT:

- **Escenario de Latencia en Interrupciones**: El sistema operativo responderá a una señal externa periódica para evaluar el tiempo de respuesta.
- **Escenario de Multitarea y Concurrencia**: Se ejecutarán varias tareas concurrentes, midiendo el uso de CPU, la eficiencia en la multitarea y el consumo de memoria.
- **Escenario de Bajo Consumo**: Se pondrá a prueba el sistema operativo en distintos modos de ahorro de energía, como suspensión y reposo, midiendo el consumo energético en cada estado.
- **Escenario de Estabilidad**: Ejecutar tareas repetitivas y de larga duración para evaluar la estabilidad y la gestión de memoria a lo largo del tiempo, observando la frecuencia de errores o fallos.

4. Instrumentación y Recolección de Datos

- **Medición de Consumo Energético**: Usar un medidor de potencia para monitorear el consumo energético del microcontrolador en cada escenario.
- **Monitoreo de Recursos**: Utilizar herramientas de monitoreo interno del sistema operativo o instrumentos de depuración para registrar el uso de CPU y memoria en tiempo real.
- **Registro de Latencias**: Para medir el tiempo de respuesta, se empleará un osciloscopio o un temporizador de alta precisión que capture el tiempo entre la señal de interrupción y la respuesta del sistema operativo.

5. Análisis Estadístico

- **Estadísticas Descriptivas**: Se calcularán métricas como la media, desviación estándar y percentiles para resumir el rendimiento y el consumo de cada sistema operativo en cada escenario.
- **Pruebas de Hipótesis**: Se realizará un análisis ANOVA para comparar las medias entre sistemas operativos y determinar si existen diferencias significativas en rendimiento y consumo.
- **Gráficos Comparativos**: Se utilizarán gráficos de barras y diagramas de cajas para visualizar los resultados de cada sistema operativo en las métricas de interés, facilitando la interpretación de las diferencias.

6. Validación de Resultados y Conclusiones

Los resultados obtenidos en los experimentos serán interpretados en función de los objetivos del proyecto y comparados con las especificaciones de rendimiento esperadas en aplicaciones IoT. A partir de los análisis estadísticos, se validarán los sistemas operativos en términos de rendimiento, eficiencia energética y estabilidad, y se proporcionarán recomendaciones claras sobre la idoneidad de cada sistema operativo para microcontroladores en escenarios específicos de IoT y sistemas embebidos.

Cronograma

Actividades	**Octubre**	**Noviembre**	**Diciembre**
1. Planificación y Definición de Objetivos	●		
Definir objetivos generales y específicos	●		
Delimitación del marco teórico	●		
2. Investigación de Sistemas Operativos	●	●	
Estudio de FreeRTOS, MicroPython, Zephyr	●		
Evaluación de herramientas de medición		●	
3. Diseño de Experimentos		●	
Definición de variables y métricas		●	
Preparación de los escenarios de prueba		●	
4. Preparación y Configuración del Hardware		●	
Instalación y configuración en ESP32 y RP Pico		●	
Instrumentación de herramientas de medición		●	
5. Ejecución de Experimentos		●	
Pruebas de latencia y respuesta a interrupciones		●	
Evaluación de multitarea y consumo energético		●	
Recopilación de datos de estabilidad		●	
6. Análisis de Resultados			●
Análisis estadístico de los resultados			●

Visualización de resultados (gráficos y tablas)			●
7. Redacción del Informe Final			●
Elaboración de conclusiones y recomendaciones			●
Presentación del informe y revisión			●

Desglose Temporal de Actividades

- **Octubre**:
  - Planificación inicial, definición de objetivos, y delimitación del marco teórico.
  - Comienzo de la investigación sobre sistemas operativos y selección de herramientas de medición.
- **Noviembre**:
  - Profundización en el diseño de experimentos, incluyendo la definición de variables, métricas y preparación de escenarios.
  - Configuración del hardware y pruebas experimentales para obtener los primeros datos.
- **Diciembre (Primera Semana)**:
  - Análisis de los resultados obtenidos y preparación de gráficos y tablas.
  - Redacción del informe final, con conclusiones.

Referencias

- https://sing.stanford.edu/cs303-sp11/
- https://gradstudies.engineering.utoronto.ca/wp-content/uploads/sites/4/2021/06/Experimental-Design-Template-Examples.pdf
- https://caranha.github.io/ExperimentDesignCS/syllabus.html

- *FreeRTOS*: Real-Time Operating System for Embedded Devices. FreeRTOS Foundation https://www.freertos.org
- *Zephyr OS*: Zephyr Project Documentation. Linux Foundation Projects. https://docs.zephyrproject.org
- *ESP32 Technical Reference Manual*. Espressif Systems, 2023. Disponible en: https://www.espressif.com
- *Raspberry Pi Pico Datasheet* y documentación. Raspberry Pi Foundation. https://www.raspberrypi.org/documentation
- *MicroPython for ESP32 and Raspberry Pi Pico Documentation*. MicroPython Foundation. https://docs.micropython.org