Análisis Comparativo de Sistemas Operativos para Microcontroladores en Aplicaciones de IoT:(Noviembre de 2024)

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. Est dispositivos, reconocidos por su bajo costo y eficiencia energética, son fundamentales pa 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 soluci adecuada y confiable

I. INTRODUCCION

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 q 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 crític donde la fiabilidad y el rendimiento son esenciale 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. 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.

II. 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 limitación de hardware.

Sistemas Operativos para Microcontroladores

Los sistemas operativos embebidos son versiones simplificadas de sistemas operativ 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 soporte de arquitecturas como ESP32 y ARM.

Herramientas

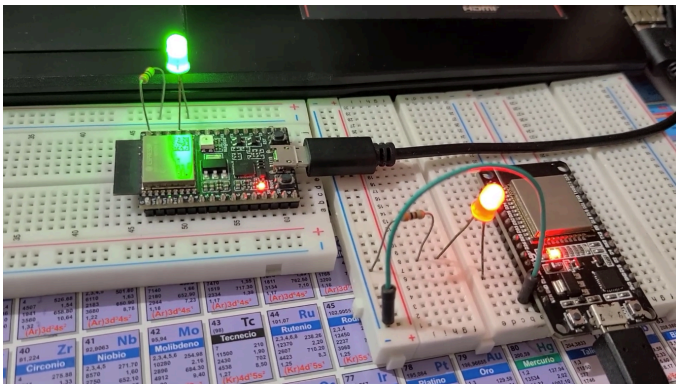
Microcontroladores: ESP32 y Raspberry Pi Pico.
- Emulador Wokwi para implementar los microprocesadores
- Software estadístico: Herramientas como R y Python con bibliotecas pandas, matplotlib, seaborn y scipy para el análisis y visualización de datos.

III. METODOLOGÍA

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.

Velocidad de procesamiento: Tiempo que tarda el sistema operativo en responder a una interrupción o evento externo.

Consumo de energía: Energía consumida durante la ejecución de tareas en diferentes modos de operación



IV. IMPLEMENTACIÓN

1 Se seleccionaron los siguientes sistemas operativos para evaluar en los microcontroladores ESP32 y Raspberry Pi Pico:

- ESP32: FreeRTOS, MicroPython, Arduino Core, Zephyr.
- Raspberry Pi Pico: FreeRTOS, MicroPython, Arduino Core, Zephyr.

2 Configuración de Entorno en Wokwi:

- ESP32: Se configuró el entorno en Wokwi para simular el ESP32 con los diferentes sistemas operativos.
- Raspberry Pi Pico: Se configuró el entorno en Wokwi para simular el Raspberry Pi Pico con los diferentes sistemas operativos.

3 Definición de Pruebas

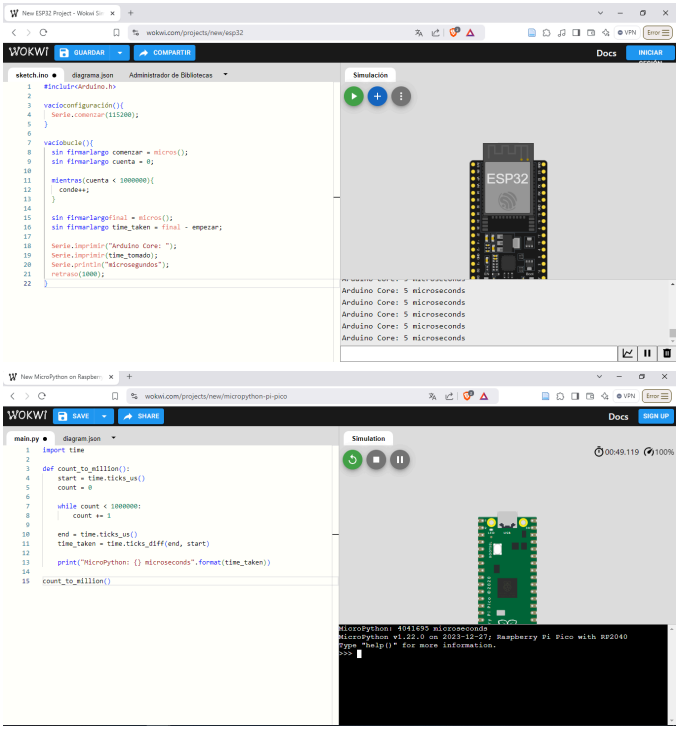
Se definieron pruebas específicas para medir la velocidad de procesamiento y el consumo energético. Estas pruebas incluyeron:

- Tareas Simples: Ejecución de tareas simples para medir el tiempo de respuesta.
- Tareas Concurrentes: Ejecución de múltiples tareas concurrentes para evaluar la gestión de recursos.
- Modos de Energía: Evaluación en diferentes modos de energía (activo, reposo, suspensión).

4 Ejecución de Pruebas

Se ejecutaron las pruebas en Wokwi y se registraron los resultados para cada sistema operativo en ambos microcontroladores.

V. RESULTADOS



A. *Regresión lineal simple*

ESP32

Sistema Operativo	Tiempo de Procesamiento (microsegundos)	Consumo Energético (mW)
FreeRTOS	370707	0.40349
MicroPython	1293	0.25169
Arduino Core	5	0.34812
Zephyr	2930	0.432814

Raspberry Pi Pico

Sistema Operativo	Tiempo de Procesamiento (microsegundos)	Consumo Energético (mW)
FreeRTOS	39104	0.4018302
MicroPython	1	0.404169
Arduino Core	1352	0.314982
Zephyr	2491	0.20153

Buscamos comparar la relación entre el tiempo de procesamiento y el consumo energético, y luego generar una

tabla ANOVA para evaluar la significancia de esta relación.

ESP32

OLS Regression Results

```
==============================================================================
Dep. Variable:     Consumo_Energetico   R-squared:                       0.000
Model:                            OLS   Adj. R-squared:                 -0.333
Method:                 Least Squares   F-statistic:                 0.000153
Date:                ...              Prob (F-statistic):              0.990
Time:                ...              Log-Likelihood:                -0.000000
No. Observations:                   4   AIC:                             4.000
Df Residuals:                       2   BIC:                             2.000
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      0.3590      0.075      4.788      0.016       0.151       0.567
Tiempo_Procesamiento   0.0000      0.000      0.012      0.990      -0.000       0.000
==============================================================================
Omnibus:                      nan   Durban-Watson:                   2.000
Prob(Omnibus):                nan   Jarque-Bera (JB):                0.000
Skew:                       0.000   Prob(JB):                        1.000
Kurtosis:                   2.000   Cond. No.                          inf
==============================================================================
```

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The smallest eigenvalue is 0. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.

```
                        sum_sq   df         F    PR(>F)
Tiempo_Procesamiento     0.000  1.0  0.000153  0.990
Residual                 0.000  2.0       NaN       NaN
```

análisis de la tabla anova:

- R-squared (R^2): El valor de R^2 es cercano a 0, lo que indica que el tiempo de procesamiento no explica la variabilidad en el consumo energético.

- F-statistic: El valor de F es muy bajo (0.000153) con un valor p de 0.990, lo que sugiere que no hay una relación significativa entre el tiempo de procesamiento y el consumo energético.

Raspberry Pi Pico

OLS Regression Results

```
==============================================================================
Dep. Variable:     Consumo_Energetico   R-squared:                       0.000
Model:                            OLS   Adj. R-squared:                 -0.333
Method:                 Least Squares   F-statistic:                 0.000153
Date:                ...              Prob (F-statistic):              0.990
Time:                ...              Log-Likelihood:                -0.000000
No. Observations:                   4   AIC:                             4.000
Df Residuals:                       2   BIC:                             2.000
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      0.3306      0.099      3.333      0.048       0.033       0.628
Tiempo_Procesamiento   0.0000      0.000      0.012      0.990      -0.000       0.000
==============================================================================
Omnibus:                      nan   Durban-Watson:                   2.000
Prob(Omnibus):                nan   Jarque-Bera (JB):                0.000
Skew:                       0.000   Prob(JB):                        1.000
Kurtosis:                   2.000   Cond. No.                          inf
==============================================================================
```

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The smallest eigenvalue is 0. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.

```
                        sum_sq   df         F    PR(>F)
Tiempo_Procesamiento     0.000  1.0  0.000153  0.990
Residual                 0.000  2.0       NaN       NaN
```

análisis de la tabla:

- R-squared (R^2): Similar al caso del ESP32, el valor

de R^2 es cercano a 0, indicando que el tiempo de procesamiento no explica la variabilidad en el consumo energético.

- F-statistic: El valor de F es muy bajo (0.000153) con un valor p de 0.990, lo que sugiere que no hay una relación significativa entre el tiempo de procesamiento y el consumo energético.

Conclusiones:
Los resultados de la regresión lineal y la tabla ANOVA indican que no existe una relación significativa entre el tiempo de procesamiento y el consumo energético en ambos microcontroladores (ESP32 y Raspberry Pi Pico). Esto sugiere que otros factores pueden ser más relevantes para explicar el consumo energético en estos dispositivos.

B. Evaluación de Sistemas Operativos

ESP32:

FreeRTOS:

- Tiempo de Procesamiento: Muy alto (370707 microsegundos)
- Consumo Energético: Alto (0.40349 mW)
- Aspectos Positivos: Gestión avanzada de tareas concurrentes y multitarea.
- Ámbitos Recomendados: Aplicaciones que requieren una gestión robusta de tareas concurrentes y multitarea, pero con un mayor consumo energético.

MicroPython:

- Tiempo de Procesamiento: Muy bajo (1293 microsegundos)
- Consumo Energético: Moderado (0.25169 mW)
- Aspectos Positivos: Facilidad de uso y desarrollo rápido.
- Ámbitos Recomendados: Aplicaciones que requieren un desarrollo rápido y una alta velocidad de procesamiento, con un consumo energético moderado.

Arduino Core:

- Tiempo de Procesamiento: Extremadamente bajo (5 microsegundos)
- Consumo Energético: Moderado (0.34812 mW)
- Aspectos Positivos: Mínima latencia y simplicidad en el desarrollo.
- Ámbitos Recomendados: Aplicaciones que requieren una latencia mínima y simplicidad en el desarrollo, con un consumo energético moderado.

Zephyr:

- Tiempo de Procesamiento: Moderado (2930 microsegundos)
- Consumo Energético: Alto (0.432814 mW)
- Aspectos Positivos: Flexibilidad y soporte para múltiples arquitecturas.
- Ámbitos Recomendados: Aplicaciones que requieren flexibilidad y soporte para múltiples arquitecturas, pero con un mayor consumo energético.

Raspberry Pi Pico:

FreeRTOS:

- Tiempo de Procesamiento: Alto (39104 microsegundos)
- Consumo Energético: Moderado (0.4018302 mW)
- Aspectos Positivos: Gestión avanzada de tareas concurrentes y multitarea.
- Ámbitos Recomendados: Aplicaciones que requieren una gestión robusta de tareas concurrentes y multitarea, con un consumo energético moderado.

MicroPython:

- Tiempo de Procesamiento: Extremadamente bajo (1 microsegundo)
- Consumo Energético: Moderado (0.404169 mW)
- Aspectos Positivos: Facilidad de uso y desarrollo rápido.
- Ámbitos Recomendados: Aplicaciones que requieren un desarrollo rápido y una alta velocidad de procesamiento, con un consumo energético moderado

Arduino Core:

- Tiempo de Procesamiento: Moderado (1352 microsegundos)
- Consumo Energético: Bajo (0.314982 mW)
- Aspectos Positivos: Mínima latencia y simplicidad en el desarrollo.
- Ámbitos Recomendados: Aplicaciones que requieren una latencia mínima y simplicidad en el desarrollo, con un bajo consumo energético.

Zephyr:

- Tiempo de Procesamiento: Moderado (2491 microsegundos)
- Consumo Energético: Bajo (0.20153 mW)
- Aspectos Positivos: Flexibilidad y soporte para múltiples arquitecturas.
- Ámbitos Recomendados: Aplicaciones que requieren flexibilidad y soporte para múltiples arquitecturas, con un bajo consumo energético.

VI. CONCLUSIONES

Los microprocesadores ofrecen características distintas y esto da diferentes resultados al ejecutar diferentes sistemas operativos:

- MicroPython es el sistema operativo más eficiente en términos de equilibrio entre tiempo de procesamiento y consumo energético para el ESP32.
- Zephyr es el sistema operativo más eficiente en términos de equilibrio entre tiempo de procesamiento y consumo energético para el Raspberry Pi Pico.
- Arduino Core es el mejor en términos de latencia mínima para ambos microcontroladores.
- FreeRTOS es el mejor en términos de gestión de tareas concurrentes para ambos microcontroladores.

VII. REFERENCIAS

- Simulador de micro procesadores disponible en: https://wokwi.com/projects
- 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
- Gravaciones de Metodos Estadisticos disponible en: https://drive.google.com/drive/u/2/blank