UNIVERSIDAD POLITÉCNICA SALESIANA

SEDE QUITO

CARRERA DE COMPUTACIÓN

COMPARACIÓN DEL RENDIMIENTO DE LOS SISTEMAS EMBEBIDOS ESP32 y M5STACK

Trabajo de titulación previo a la obtención del
Título de Ingeniero en Ciencias de la Computación

AUTOR: SEBASTIÁN ISAÍAS CÓRDOVA RECALDE
TUTOR: MANUEL RAFAEL JAYA DUCHE

Quito - Ecuador
2024

CERTIFICADO DE RESPONSABILIDAD Y AUTORÍA DEL TRABAJO DE TITULACIÓN

Yo, Sebastián Isaías Córdova Recalde con documento de identificación N.º 1750000463; manifiesto que:

Soy autor y responsable del presente trabajo; y, autorizo a que sin fines de lucro la Universidad Politécnica Salesiana pueda usar, difundir, reproducir o publicar de manera total o parcial el presente trabajo de titulación.

Quito, 12 de agosto del 2024

Atentamente,

Sebastián Isaías Córdova Recalde
1750000463

CERTIFICADO DE CESIÓN DE DERECHOS DE AUTOR DEL TRABAJO DE TITULACIÓN A LA UNIVERSIDAD POLITÉCNICA SALESIANA

Yo, Sebastián Isaías Córdova Recalde con documento de identificación No. 1750000463, expreso mi voluntad y por medio del presente documento cedo a la Universidad Politécnica Salesiana la titularidad sobre los derechos patrimoniales en virtud de que soy autor del Artículo Académico: "Comparación del rendimiento de los sistemas embebidos ESP32 y M5STACK", el cual ha sido desarrollado para optar por el título de: Ingeniero en Ciencias de la Computación, en la Universidad Politécnica Salesiana, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

En concordancia con lo manifestado, suscribo este documento en el momento que hago la entrega del trabajo final en formato digital a la Biblioteca de la Universidad Politécnica Salesiana.

Quito, 12 de agosto del 2024

Atentamente,

Sebastián Isaías Córdova Recalde
1750000463

CERTIFICADO DE DIRECCIÓN DEL TRABAJO DE TITULACIÓN

Yo, Manuel Rafael Jaya Duche con documento de identificación N° 1710631035, docente de la Universidad Politécnica Salesiana, declaro que bajo mi tutoría fue desarrollado el trabajo de titulación: COMPARACIÓN DEL RENDIMIENTO DE LOS SISTEMAS EMBEBIDOS ESP32 y M5STACK, realizado por Sebastián Isaías Córdova Recalde, con documento de identificación N.º 1750000463, obteniendo como resultado final el trabajo de titulación bajo la opción Artículo Académico que cumple con todos los requisitos determinados por la Universidad Politécnica Salesiana.

Quito, 12 de agosto del 2024

Atentamente,

Ing. Manuel Rafael Jaya Duche, MSc.
1710631035

DEDICATORIA

El presente artículo representa el testimonio continuo de gratitud hacia mis amados padres, Rocío Recalde y Mario Córdova, quienes con sus jubilaciones me brindaron el apoyo necesario para convertirme en profesional. Agradezco a Dios porque, a pesar de los desafíos y tropiezos en mi trayectoria, nunca permitió que me quedara en el suelo; por el contrario, me fortaleció con cada experiencia para seguir adelante con determinación en la vida.

Durante mi recorrido universitario, he enfrentado una mezcla de momentos buenos, difíciles e indecisiones que han culminado en la conclusión de este artículo académico. Ha sido un camino marcado por lágrimas, sonrisas, recuerdos, decepciones y distanciamientos, pero sobre todo, por el descubrimiento de aquellos en quienes puedo confiar de ahora en adelante.

Para finalizar, este escrito es un ofrecimiento especial a mi hija Ada Fiorella Córdova Coba, Adriana Stefania Coba Correa, Alisha Jácome Córdova, Henry Cordova, Mery Cordova, Eduardo Cordova, Matias Cordova y primos. A través de mis experiencias, deseo transmitirles que, a pesar de mis imperfecciones y decisiones, siempre he sido resiliente. Continuaré avanzando, siendo el ejemplo de que, cuando te enfrentas a caídas, debes sacudirte, levantarte y mirar hacia adelante, luchando incansablemente por tus sueños y objetivos.

Sebastián Isaías Córdova Recalde

COMPARACIÓN DEL RENDIMIENTO DE LOS SISTEMAS EMBEBIDOS ESP32 y M5STACK

1st Sebastián Isaías Córdova Recalde
scordovar1@est.ups.edu.ec

2nd Manuel Rafael Jaya Duche
mjaya@ups.edu.ec

***Resumen*—En este artículo se presenta un análisis comparativo entre los SoC ESP32 y M5STACK Core 2. La investigación se centró en tres aspectos clave: velocidad de procesamiento, eficiencia energética y velocidad de transferencia inalámbrica. Mediante pruebas, se evaluó el rendimiento de cada dispositivo en condiciones de carga intensiva, consumo energético y velocidad de transferencia inalámbrica.Los resultados indican que el M5STACK Core 2 ofrece una mejor velocidad de procesamiento, completando tareas en menor tiempo comparado con el ESP32. En cuanto a eficiencia energética, el M5STACK Core 2 demostró un menor consumo de energía, siendo más adecuado para aplicaciones con restricciones energéticas. En términos de comunicación inalámbrica, ambos dispositivos mostraron rendimientos similares, aunque el M5STACK Core 2 presentó una ligera ventaja en la velocidad de transferencia. Estas diferencias reflejan las particularidades de diseño y arquitectura de cada SoC. Mientras que el ESP32 se destaca por su flexibilidad y soporte amplio, el M5STACK Core 2 es preferido por su diseño compacto y facilidad de uso, lo que facilita el desarrollo rápido de prototipos. Este análisis proporciona información valiosa para elegir el hardware más adecuado en el desarrollo de aplicaciones IoT, optimizando tanto el rendimiento como la fiabilidad en diversas condiciones operativas.**

***Palabras Clave*—ESP32, M5STACK, comparación, rendimiento, sistemas embebidos, SoC.**

***Abstract*—This paper presents a comparative analysis between the ESP32 and M5STACK Core 2 SoC. The research focused on three key aspects: processing speed, power efficiency and wireless transfer speed. Through rigorous testing, the performance of each device was evaluated under intensive load conditions, power consumption and data transfer speed via Wi-Fi. The results indicate that the M5STACK Core 2 offers better processing speed, completing tasks in less time compared to the ESP32. In terms of power efficiency, the M5STACK Core 2 demonstrated lower power consumption, making it more suitable for power-constrained applications. In terms of wireless communication, both devices showed similar performances, although the M5STACK Core 2 showed a slight advantage in transfer speed. These differences reflect the design and architectural particularities of each Soc. While the ESP32 stands out for its flexibility and broad support, the M5STACK Core 2 is preferred for its compact design and ease of use, which facilitates rapid prototyping. This analysis provides valuable information for choosing the most suitable hardware in IoT application development, optimizing both performance and reliability under various operating conditions.**

***Keywords*—ESP32, M5STACK, SoC, comparison, performance, embedded systems.**

I. Introducción

En el estudio realizado para evaluar y comparar el rendimiento de los sistemas embebidos ESP32 y M5STACK, se llevaron a cabo diversas pruebas con el objetivo de analizar sus capacidades en términos de velocidad de procesamiento, consumo de energía y velocidad de transferencia inalámbrica. Estas pruebas fueron diseñadas para proporcionar una visión integral de las capacidades de cada plataforma en escenarios relevantes para aplicaciones prácticas.

Inicialmente, se evaluó la velocidad de procesamiento de ambos sistemas mediante la ejecución de algoritmos de procesamiento intensivo y operaciones básicas de manejo de datos. El consumo de energía se midió durante períodos de carga variados para determinar la eficiencia energética relativa de cada dispositivo en diferentes condiciones de uso. Además, se realizó un análisis exhaustivo de la velocidad de transferencia inalámbrica utilizando protocolos estándar de comunicación WiFi y Bluetooth.

Se espera que los resultados revelen diferencias significativas en términos de rendimiento entre el ESP32 y el M5STACK, destacando las fortalezas y limitaciones de cada uno en contextos específicos de aplicación. Se anticipa que el ESP32, conocido por su versatilidad y bajo consumo de energía, mostrará ventajas en eficiencia energética, mientras que el M5STACK, con su hardware adicional y capacidades extendidas, podría sobresalir en rendimiento bruto y capacidades de procesamiento avanzadas.

II. Metodología y Materiales

II-A. Materiales

Los materiales principales utilizados para todas las pruebas son:

- M5STACK Core 2
- ESP32
- Modulo SD para ESP32
- Tarjeta sd de 4gb
- Cable usb a tipo B para ESP32
- Cable usb a tipo C para M5STACK Core 2
- Laptop Lenovo Ideapad 14 gaming core i5 10th y tarjeta Nvidea 1050
- Matlab
- Arduino IDE

II-B. Diseño experimental

Se establecen los criterios de prueba para cada SoC. En las pruebas el criterio está basado en benchmarking, para la primera prueba se van a utilizar para medir la velocidad de

procesamiento de cada sistema. Los sistemas serán evaluados con procesos matemáticos complejo realizado en la plataforma Arduino y midiendo el tiempo que demora en completar el proceso.

A continuación, la lógica de programación para desarrollar la prueba de velocidad de procesamiento:

Algoritmo 1 Medición de velocidad de procesamiento en ESP32

```
Define NUM_ITERACIONES ← 1000
Define FILE_NAME ← "/velocidad_procesamiento.txt"
procedure SETUP
    Initialize Serial at 115200 baud rate
    if SD card initialization fails then
        Print "Error al inicializar la tarjeta SD."
        halt
    end if
    Attempt to open FILE_NAME for writing
    if file is open then
        if file is empty then
            Write "Iteración,Tiempo (ms)" to file
        end if
        Close file
    else
        Print "Error al abrir el archivo!"
        halt
    end if
    for i ← 0 to NUM_ITERACIONES do
        startTime ← current time in milliseconds
        STRESSDEVICE
        endTime ← current time in milliseconds
        tiempoTranscurrido ← endTime − startTime
        Open FILE_NAME for appending
        if file is open then
            Write i + 1, tiempoTranscurrido to file
            Close file
        else
            Print "Error al abrir el archivo!"
        end if
    end for
    Print "Proceso completado. Verifica la tarjeta SD para los resultados."
end procedure
```

Figura 1: Lógica de programación para ESP32

Algoritmo 4 Medición de velocidad de procesamiento en M5Stack Core2

```
for i ← 0 to NUM_ITERACIONES do
    startTime ← current time in milliseconds
    STRESSDEVICE
    endTime ← current time in milliseconds
    tiempoTranscurrido ← endTime − startTime
    Open FILE_NAME for appending
    if file is open then
        Write i + 1, tiempoTranscurrido to file
        Close file
    else
        Print "Error al abrir el archivo!"
    end if
end for
Print "Proceso completado. Verifica la tarjeta SD para los resultados."
procedure STRESSDEVICE
    Define matrixA, matrixB, result as 10x10 arrays
    for i ← 0 to 9 do
        for j ← 0 to 9 do
            matrixA[i][j] ← random number between 0 and 99
            matrixB[i][j] ← random number between 0 and 99
        end for
    end for
    for i ← 0 to 9 do
        for j ← 0 to 9 do
            result[i][j] ← 0
            for k ← 0 to 9 do
                result[i][j] ← result[i][j] + matrixA[i][k] × matrixB[k][j]
            end for
        end for
    end for
end procedure
```

Figura 2: Lógica de programación para M5STACK

En la segunda prueba se evalúa la capacidad energética, es decir, el consumo de energía que cada SoC gasta al procesar los cálculos matemáticos complejos. Se mide en mA y será evaluado en tres modos energéticos, es decir, en modo activo, suspensión ligera e hibernación.

En la tercera prueba sobre la velocidad de transferencia inalámbrica. Se debe tomar en cuenta que el ambiente de prueba es con un router HUAWEI Echolife EG8145V5 con un ISP Netlife de 70Mbps, el sistema esp32 tiene un chip Wi-Fi/BT Chip ESP32-D0WD-V3 y el m5stakc tiene un ESP32-D0WDQ6-V3 que tiene integrado wifi y bluetooth.

II-C. Realización de pruebas

Consta de tres fases, siendo las pruebas de velocidad de procesamiento, consumo de energía y monitoreo de comunicación.

Las pruebas de velocidad de procesamiento se realizan en el laboratorio de computación avanzada ubicada en el bloque d de la Universidad Politécnica Salesiana.

En primera instancia, se inserta el código desarrollado en Arduino IDE a los sistemas embebidos. a través del canal serial 115200. durante 3 semanas se recopilaron los datos que serán analizados en Matlab. Para la obtención de datos en el ESP32 se implementó un módulo sd, en cambio, el M5STACK se encuentra integrada el módulo sd.

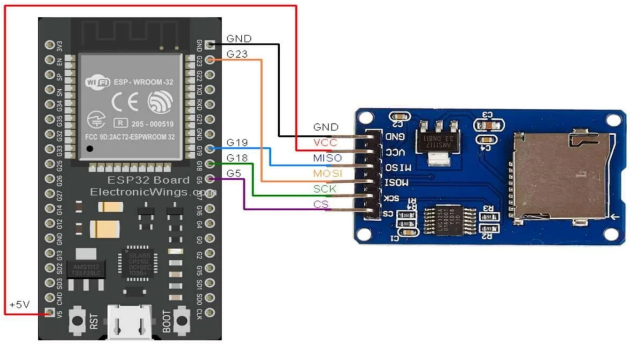


Figura 3: Diagrama de conexión de un esp32 a un modulo sd

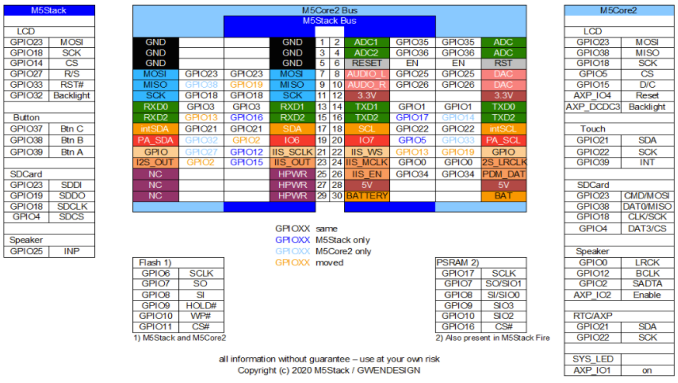


Figura 4: Esquema técnico de un M5STACK core 2

Tomando en cuenta que la comunicación de transferencia de datos del ESP32 se encuentra en el pin cs, es decir, pin 5 y que en el M5STACK es el pin 14. En Arduino se define estos puertos lógicos para que todos los resultados se guarden en la tarjeta sd.

Para realizar un benchmarking correcto se extrae el tiempo que demora en procesar o resolver el ejercicio matemático en una iteración.

```
// Ejecutar las iteraciones
for (int i = 0; i < NUM_ITERACIONES; i++) {
  unsigned long startTime = millis(); // Tiempo de
   inicio del procesamiento

  // Resolver ecuaciones complejas
  solveEquations();

  unsigned long endTime = millis(); // Tiempo de
  fin del procesamiento
  unsigned long tiempoTranscurrido = endTime -
  startTime; // Calcular tiempo transcurrido
```

Figura 5: Cálculo de velocidad de procesamiento en milisegundos del ESP32 y M5STACK

Una vez cargado el código en los SoCs y se guarda los datos automáticamente en un archivo csv, procedemos al análisis de datos en Matlab.

El consumo de energía se calcula en cada uno de los modos de manera independiente llegando a tener tres resultados sobre la capacidad de energía de cada SoC. Para el cálculo de consumo de energía en cualquier modo se ocupa la siguiente formula que da como resultado en [mAh], siendo consu = consumo de energía.

$$\text{Consu} = \text{Corriente (mA)} \times \frac{\text{Tiempo de ejecución ms}}{1 \times 10^6 \times 3600} \quad (1)$$

Por último, se realiza la prueba de velocidad de comunicación inalámbrica. A continuación, se presenta parte del código donde se obtiene la velocidad de transferencia de datos.

```
unsigned long duration = endTime - startTime;

// Calcular velocidad de transferencia (estimada
)
float transferSpeed = 0.0;
if (duration > 0) {
  // Estimaci n simple de velocidad en bits por
 segundo
  transferSpeed = (float)(1000 * 8) / duration;
// Conversi n a bits por segundo
}
```

Figura 6: Cálculo de velocidad de comunicaciòn inalámbrica

Se calcula los tiempos de inicio y fin de la transferencia inalámbrica, luego se transforma a bps. Para tener un cálculo en Mbps el resultado se divide para 1×10^6.

El tiempo de respuesta esta expresada en [ms].

II-D. Análisis de datos

En las pruebas de velocidad de procesamiento en el análisis de los datos se debe tomar en cuenta que existen dos campos la iteración (conteo para saber que a finalizado por una vez el proceso) y tiempo (periodo que demora en resolver el problema matemático, expresado en minutos). Para aquello se exportaron los csv generados por cada sistema y se graficaron en Matlab.

Para las pruebas de consumo de energía. Se debe tomar en cuenta los tres modos que tiene cada sistema, para esto, se debe realizar tres graficas y promedios diferentes de cada modo para saber qué sistema obtuvo un menor consumo de energía.

Para el análisis sobre los datos de velocidad de comunicación inalámbrica, de igual manera, se grafican y promedia los resultados en Matlab sobre la velocidad de transferencia en [Mbps] y el tiempo de respuesta en [ms].

Una vez obtenido los resultados, graficados y promediados en cada prueba se emite un criterio basado en los resultados.

III. Resultados

Para un mayor entendimiento de los resultados se desarrollaron gráficas y se calculò un promedio de todos los resultados extraídos en cada prueba, esto se realizò para una mayor precisión a la hora de exponer el producto de las pruebas desarrolladas en un periodo de 3 meses (se desarrolla las pruebas y análisis de cada prueba en un mes). Teniendo un total de mil datos por prueba. Cabe recalcar que las pruebas se realizaron en días diferentes y con un total de recolección de datos de 3 a 4 horas aproximadamente por día.

Antes del desarrollo del código se investigó algunas plataformas y aplicaciones para estresar al máximo a los sistemas. Un articulo que investiga la optimizaciòn de rendimiento DSP en sistemas embebidos con recursos limitados concluye que ”la arquitectura RISC tradicional no es ideal para càlculos complejos y se necesita tècnicas especìficas para mejorar el rendimiento”[9]. Por tal motivo, se desarrolló códigos personalizados para cada sistema y cada prueba a realizar.

A continuación, los resultados de las pruebas de velocidad de rendimiento.

El tiempo esta expresado en minutos. El ESP32 tiene una media de 1655.3 [min] y el M5STACK tiene 1521.1 [min]. Por tanto, quien tiene mayor velocidad de procesamiento es el M5STACK.

En una gráfica de barras se expresa los promedios resultantes de cada sistema, esto sirve para un mejor entendimiento de la diferencia entre los dos sistemas.

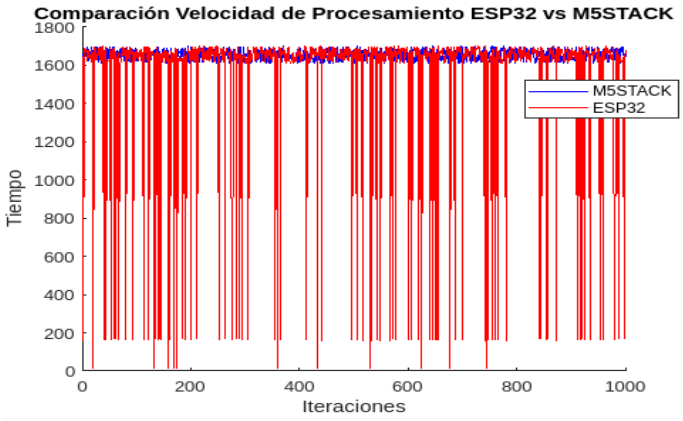


Figura 7: Comparativa de velocidad de procesamiento de ESP32 vs M5STACK

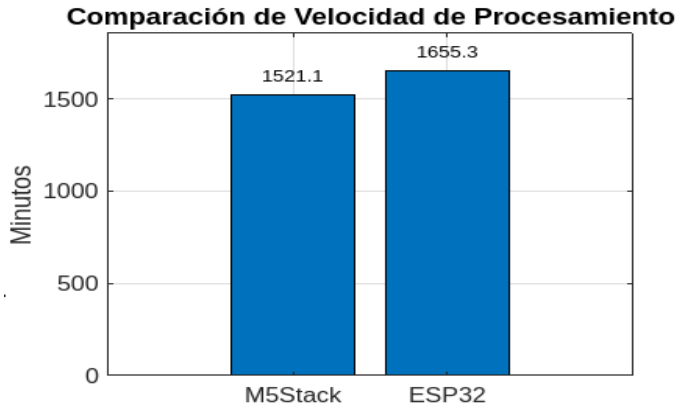


Figura 8: Grafica de barras de velocidad de procesamiento de ESP32 vs M5STACK

Para la segunda prueba de consumo de energía se estresò a los sistemas y se midió el consumo de energía en tres modos: activo, suspensión ligera e hibernación.

A continuación, la gráfica de consumo de energía en los distintos modos.

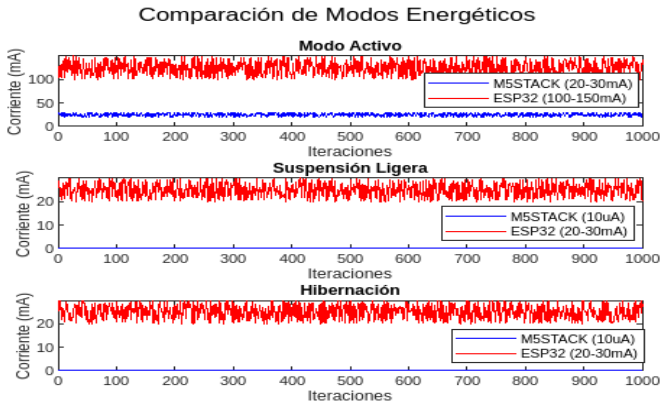


Figura 9: Consumo de energia en diferentes modos de energia

Para cada uno de los modos se promedió y como resultado se obtuvo que en el modo activo el ESP32 tiene un promedio de 124.3 [mA] y el M5STACK, 25.1 [mA] de consumo de energía; por otro lado, en el modo de suspensión ligera se identificò que el M5STACK tiene 0.00010 [mA] y el esp32, 24.9 [mA]; por último, en hibernación se tiene que el ESP32 tiene 0.25 [mA] y el M5STACK mantiene los 0.00010 [mA] de consumo de energía.

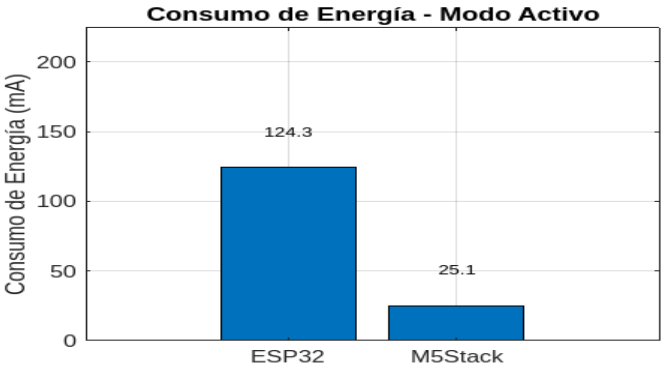


Figura 10: Consumo de energía modo de activo

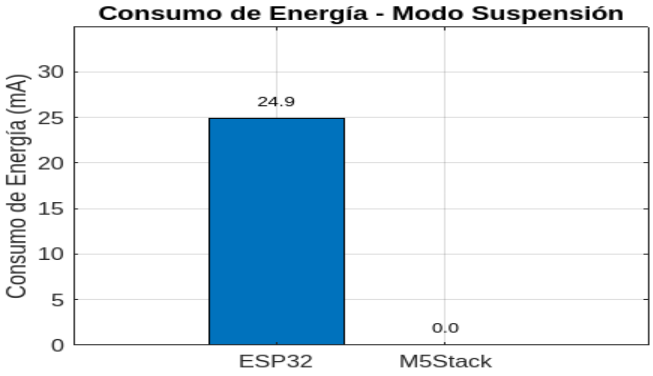


Figura 11: Consumo de energía modo de suspensión ligera

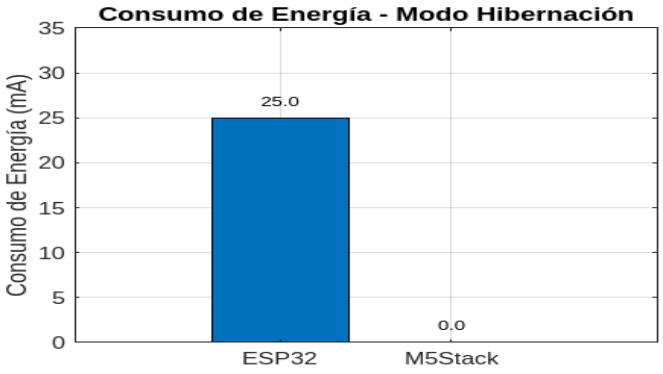


Figura 12: Consumo de energía modo hibernación

En la tercera prueba de velocidad de comunicación wifi tenemos la siguiente grafica.

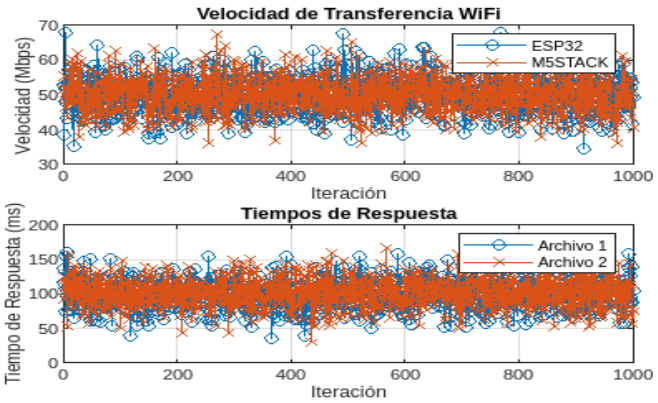


Figura 13: Velocidad de transferencia inalámbrica

En un ambiente practico de 70Mbps, se obtuvo un margen de diferencia bastante corto. El esp32 tiene una velocidad de 50.0465 [Mbps] y 99.4657 [ms] de tiempo de respuesta. El m5stack 50.1549 [Mbps] y 101.1262 [ms] de tiempo de respuesta.

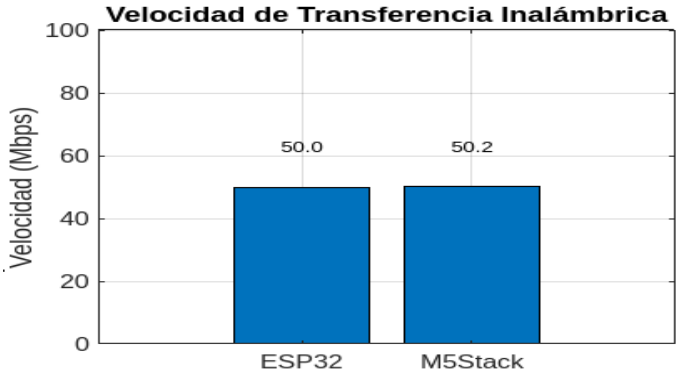


Figura 14: Velocidad de transferencia inalámbrica - Mbps

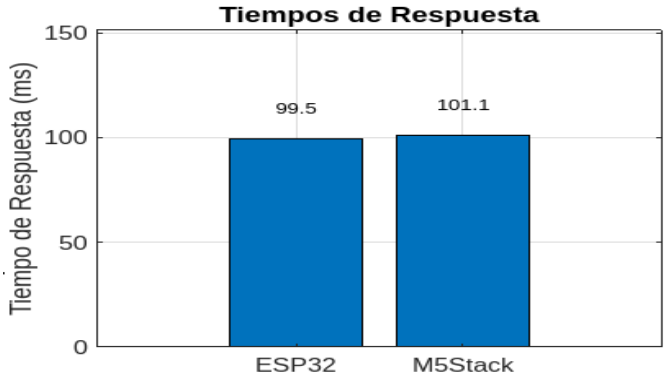


Figura 15: Velocidad de transferencia inalámbrica - tiempos de respuesta

Al observar y analizar los resultados de cada prueba, se puede afirmar que el M5Stack es el SoC con mejores calificaciones, superando al ESP32 en dos de las tres pruebas realizadas. En términos de velocidad de procesamiento, el M5STACK es un 8.11 % más rápido que el ESP32. En cuanto al consumo de energía, el M5STACK demuestra una eficiencia energética significativamente mayor, con un consumo del 79.8 % menor en modo activo, un 99.9996 % menor en modo de suspensión ligera y un 99.9996 % mayor en modo de hibernación. Para la tercera prueba, los resultados indican que la velocidad de transferencia de datos del M5STACK es un 0.2166 % mayor que la del ESP32, y su tiempo de respuesta es un 1.67 % más alto.

IV. Conclusion

De acuerdo con las pruebas el M5STACK en velocidad de procesamiento es 8.11 % mejor que el ESP32, asi mismo, en el consumo de energía en los diferentes modos de energia le supera en un 78.8 % modo activo, 99.9996 % modo suspensión ligera y 99.9996 % modo hibernación. Por ultimo, en velocidad de transferencia inalámbrica se obtuvo un 0.2166 % y tiempo de respuesta de 1.67 % a favor del M5STACK.

Referencias

[1] E. Systems, “ESP32 Technical Reference Manual,” 2024, [En línea]. Disponible: https://www.espressif.com/sites/default/files/documentation/esp32_technical_reference_manual_en.pdf.

[2] M5Stack, “M5Stack Official Documentation,” 2024, disponible: https://docs.m5stack.com/.

[3] Y. H. Lee and D. H. Shin, “An Enhanced Low-Power Wireless Sensor Node Using ESP32 Microcontroller for IoT Applications,” *Sensors*, vol. 20, no. 20, p. 5882, 2020.

[4] Y. W. Q. Zhang and H. Zhang, “Application and development of M5Stack in teaching demonstration,” *Revista Científica: Science and Technology Applications*, vol. 3, no. 4, pp. 299–304, 2019.

[5] T. T. N. H. N. Phan and K. K. R. Choo, “ESP32: The New Hope for IoT Security?” *IEEE Transactions on Consumer Electronics*, vol. 66, no. 2, pp. 118–125, 2020.

[6] Z. L. L. Qiao and Y. Liu, “An M5Stack-Based Intelligent Control System for LED Plant Factory,” *IEEE Access*, vol. 9, pp. 116 556–116 563, 2021.

[7] Y. L. Y. Cheng and Z. Zhang, “Research on the Design of ESP32-Based Smart Home System,” in *2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*, 2019, pp. 2275–2279.

[8] J. H. Y. Du and Z. Xie, “Design and implementation of a wireless remote control terminal based on M5Stack,” in *2020 2nd International Conference on Computers, Networks and Communication (ICNC)*, 2020, pp. 274–277.

[9] TI.com, “Analog, Embedded processing, Semiconductor company,” 2024, accedido el 1 de julio de 2024. [En línea]. Disponible: https://www.ti.com/lit/wp/spry089/spry089.pdf?ts=1719851951705.

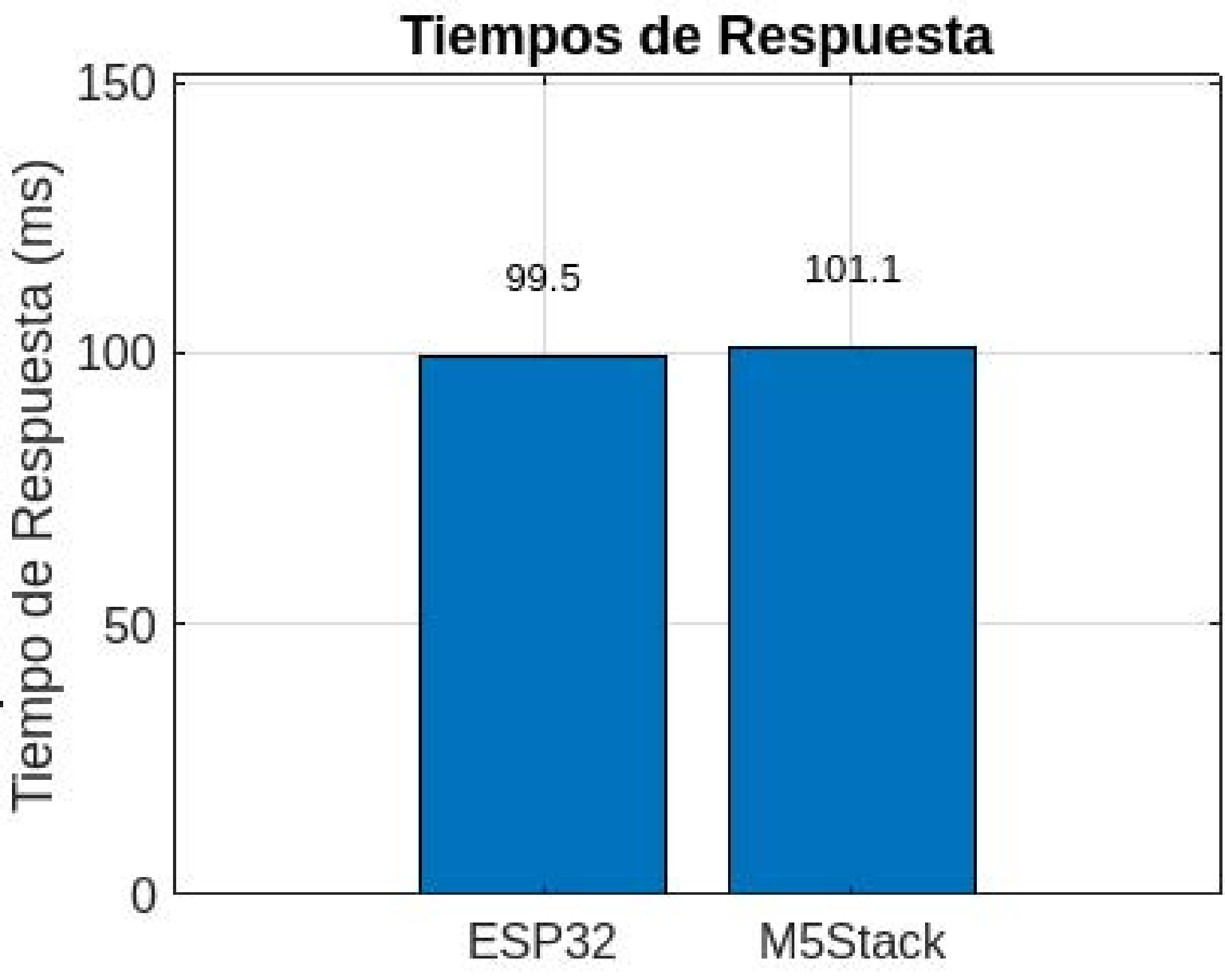
ANEXOS

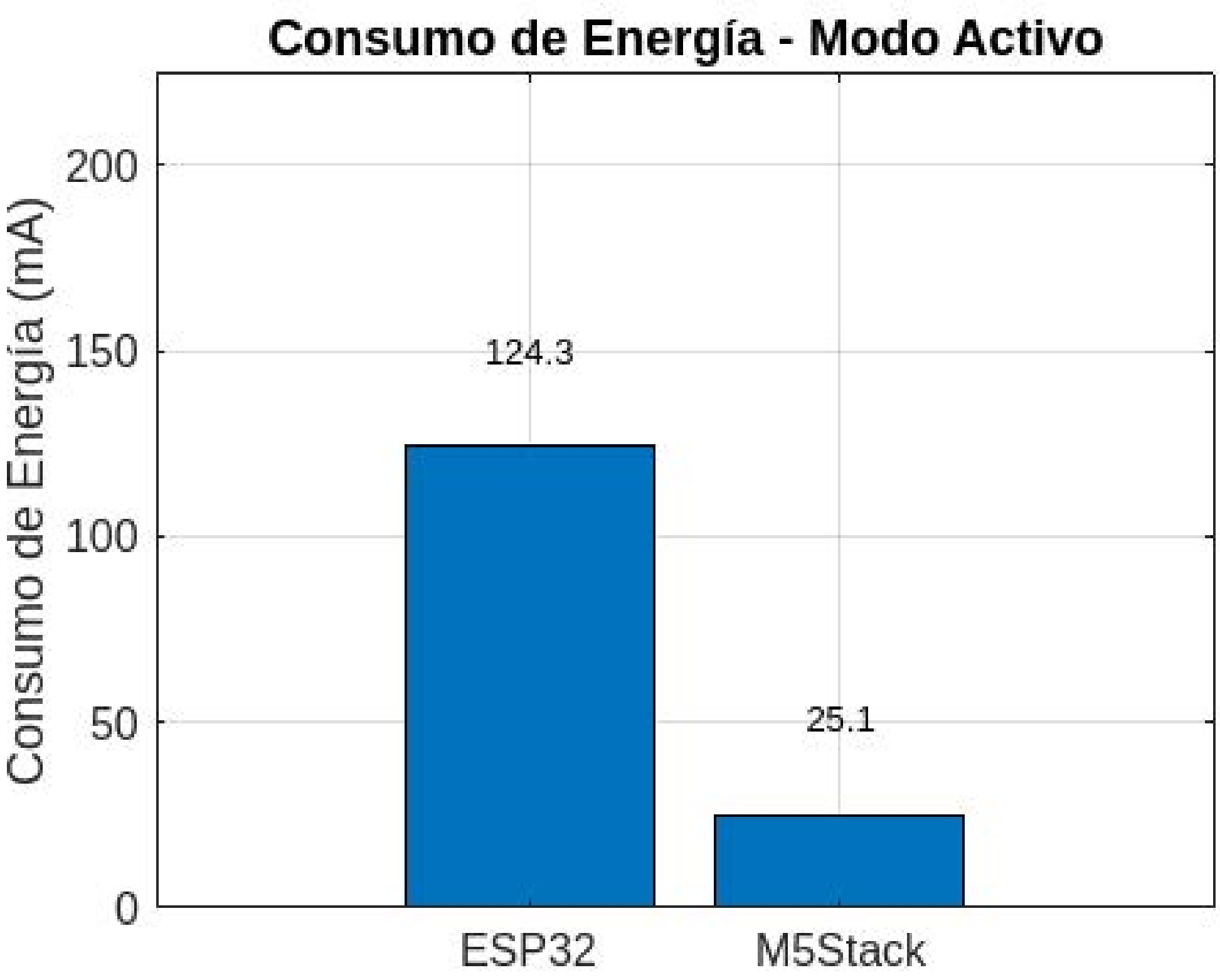
Algoritmo 1 Medición de velocidad de procesamiento en ESP32

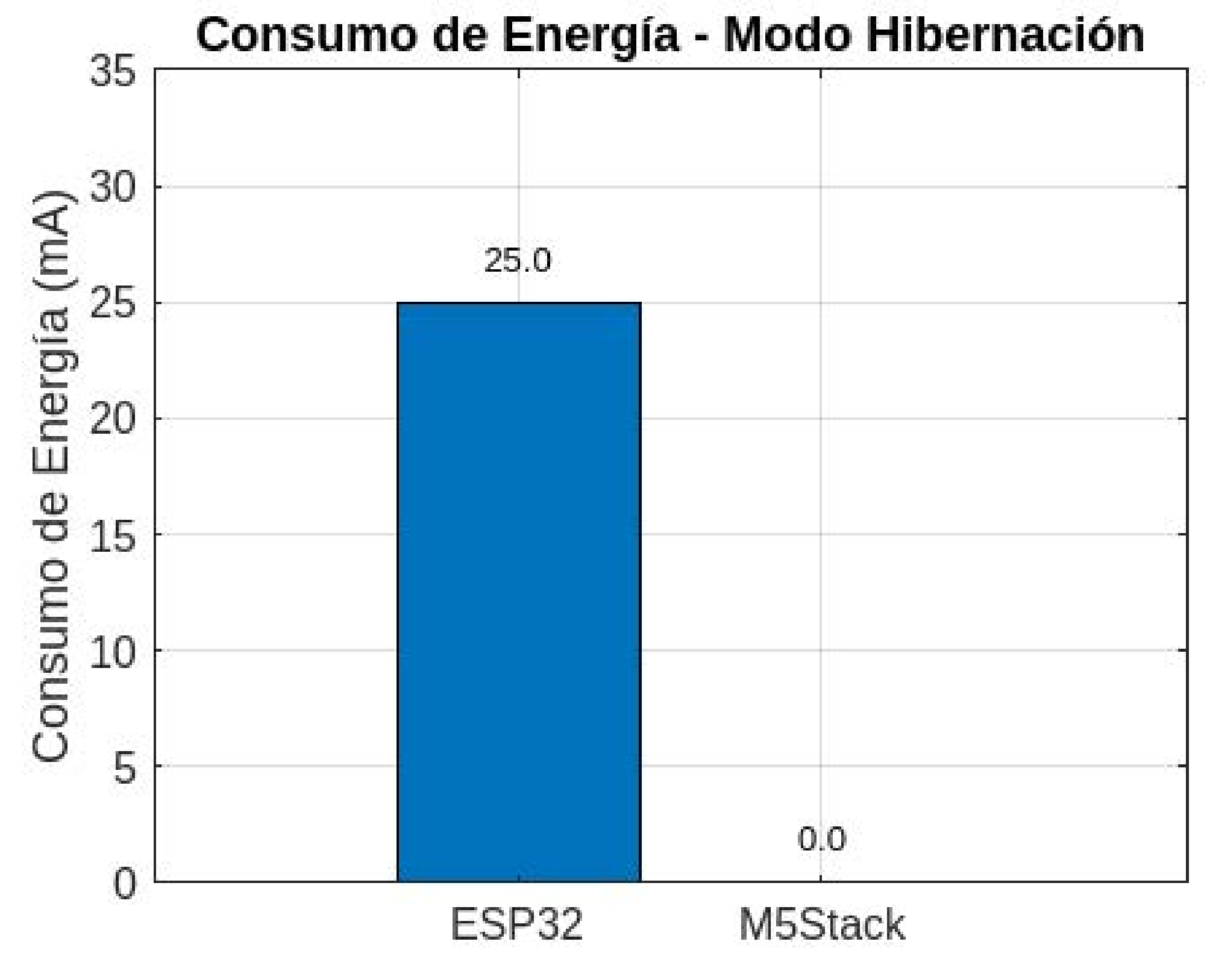
```
Define NUM_ITERACIONES ← 1000
Define FILE_NAME ← "/velocidad_procesamiento.txt"
procedure SETUP
    Initialize Serial at 115200 baud rate
    if SD card initialization fails then
        Print "Error al inicializar la tarjeta SD."
        halt
    end if
    Attempt to open FILE_NAME for writing
    if file is open then
        if file is empty then
            Write "Iteración,Tiempo (ms)" to file
        end if
        Close file
    else
        Print "Error al abrir el archivo!"
        halt
    end if
    for i ← 0 to NUM_ITERACIONES do
        startTime ← current time in milliseconds
        STRESSDEVICE
        endTime ← current time in milliseconds
        tiempoTranscurrido ← endTime − startTime
        Open FILE_NAME for appending
        if file is open then
            Write i + 1, tiempoTranscurrido to file
            Close file
        else
            Print "Error al abrir el archivo!"
        end if
    end for
    Print "Proceso completado. Verifica la tarjeta SD para los resultados."
end procedure
```

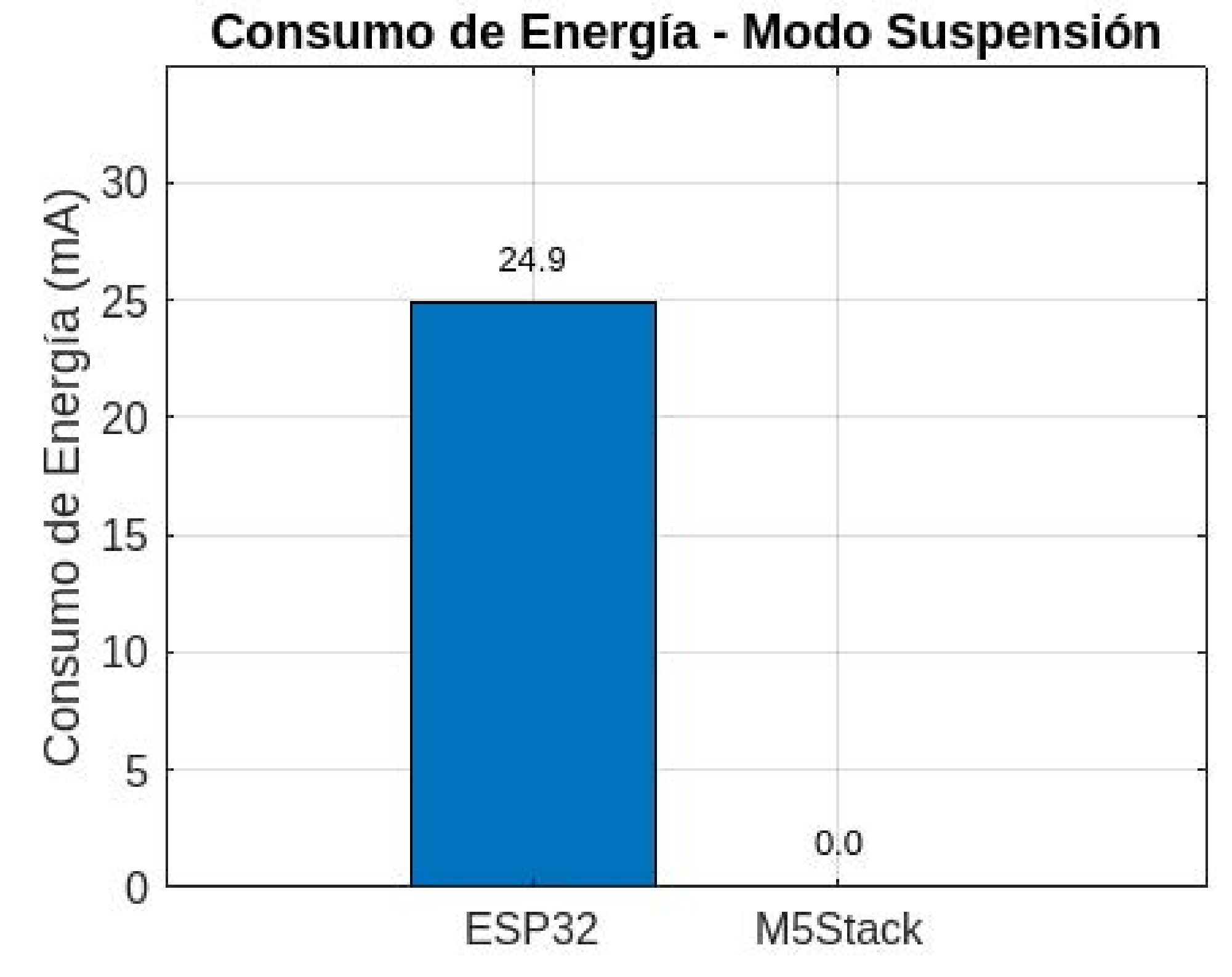
Algoritmo 4 Medición de velocidad de procesamiento en M5Stack Core2

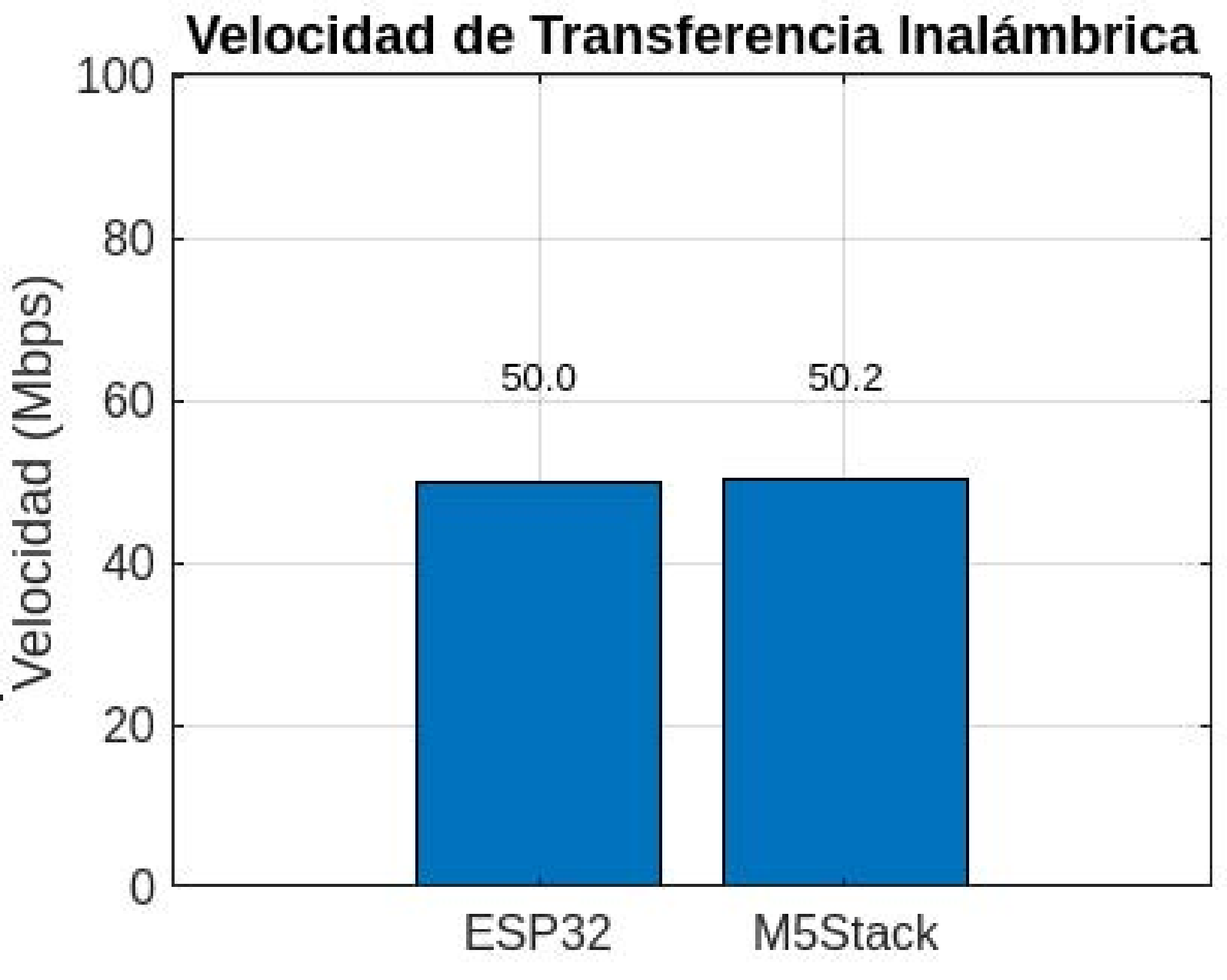
for $i \leftarrow 0$ to NUM_ITERACIONES **do**
    $startTime \leftarrow$ current time in milliseconds
    StressDevice
    $endTime \leftarrow$ current time in milliseconds
    $tiempoTranscurrido \leftarrow endTime - startTime$
    Open FILE_NAME for appending
    if file is open **then**
        Write $i + 1, tiempoTranscurrido$ to file
        Close file
    else
        Print "Error al abrir el archivo!"
    end if
end for
Print "Proceso completado. Verifica la tarjeta SD para los resultados."
procedure StressDevice
    Define matrixA, matrixB, result as 10x10 arrays
    for $i \leftarrow 0$ to 9 **do**
        for $j \leftarrow 0$ to 9 **do**
            $matrixA[i][j] \leftarrow$ random number between 0 and 99
            $matrixB[i][j] \leftarrow$ random number between 0 and 99
        end for
    end for
    for $i \leftarrow 0$ to 9 **do**
        for $j \leftarrow 0$ to 9 **do**
            $result[i][j] \leftarrow 0$
            for $k \leftarrow 0$ to 9 **do**
                $result[i][j] \leftarrow result[i][j] + matrixA[i][k] \times matrixB[k][j]$
            end for
        end for
    end for
end procedure


Tiempos de Respuesta
Tiempo de Respuesta (ms)
150
100
50
0
99.5
101.1
ESP32
M5Stack

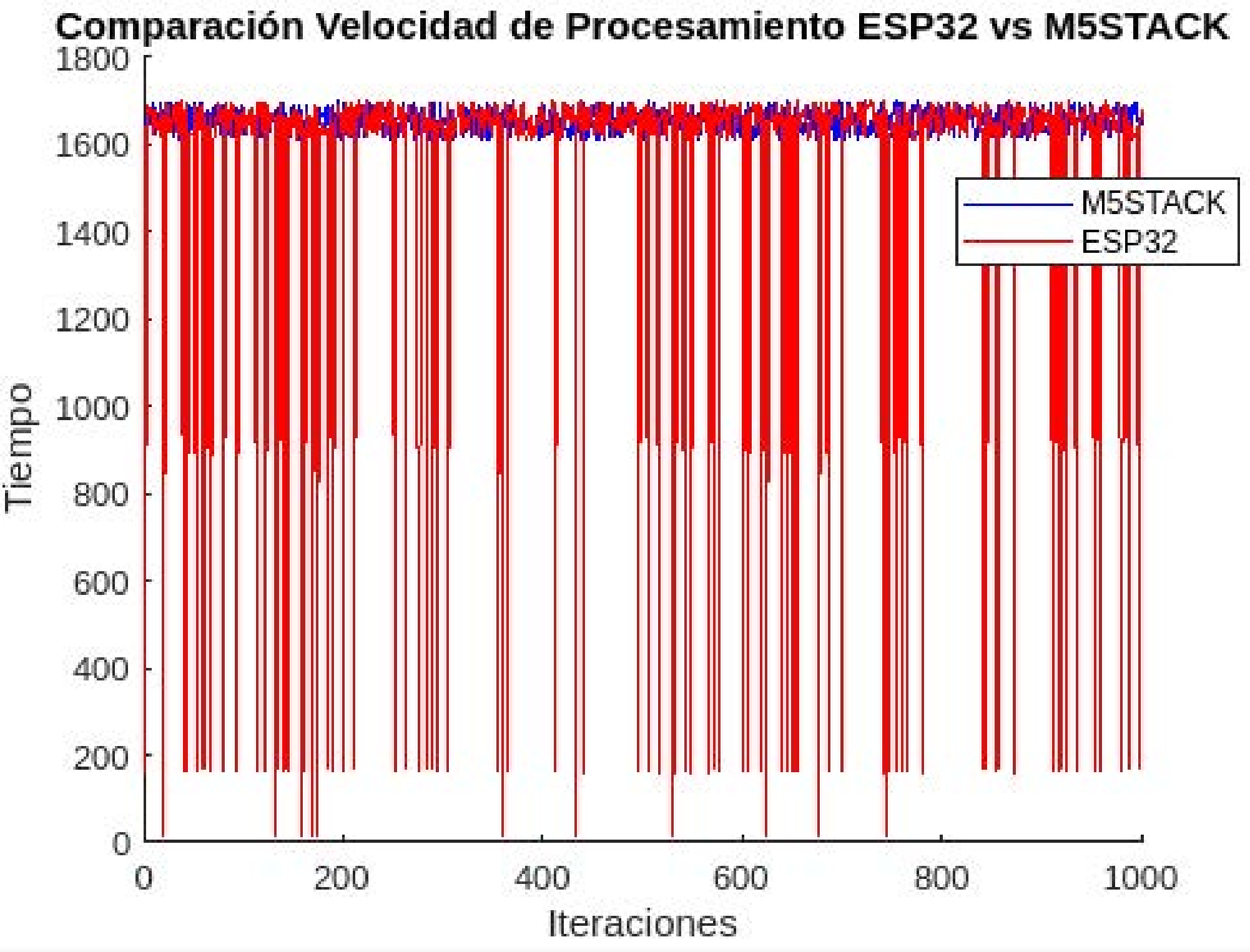

Consumo de Energía - Modo Activo
Consumo de Energía (mA)
200
150
100
50
0
124.3
25.1
ESP32
M5Stack


Consumo de Energía - Modo Hibernación
Consumo de Energía (mA)
35
30
25
20
15
10
5
0
25.0
0.0
ESP32
M5Stack

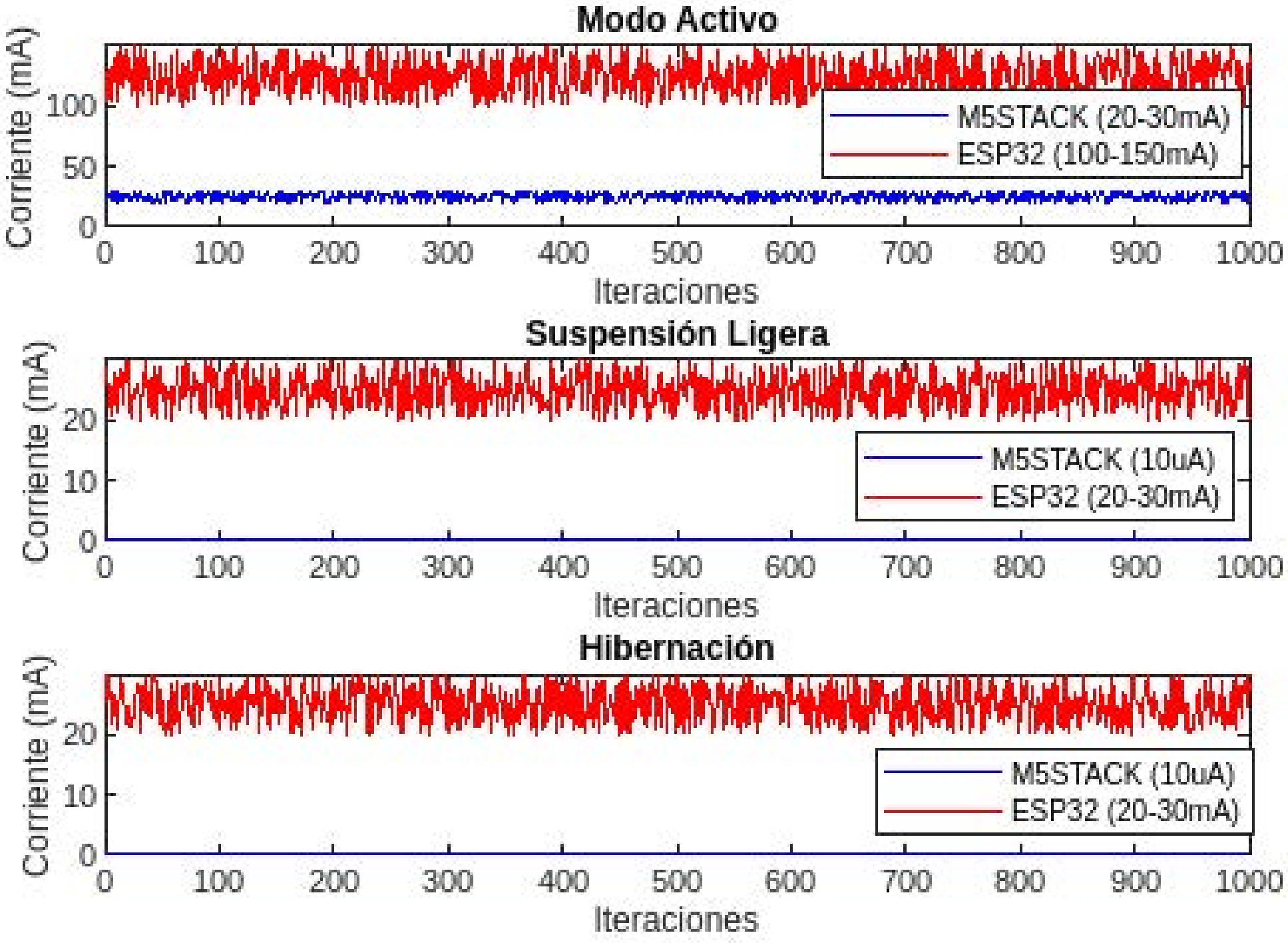

Consumo de Energía - Modo Suspensión
Consumo de Energía (mA)
30
25
20
15
10
5
0
24.9
0.0
ESP32
M5Stack

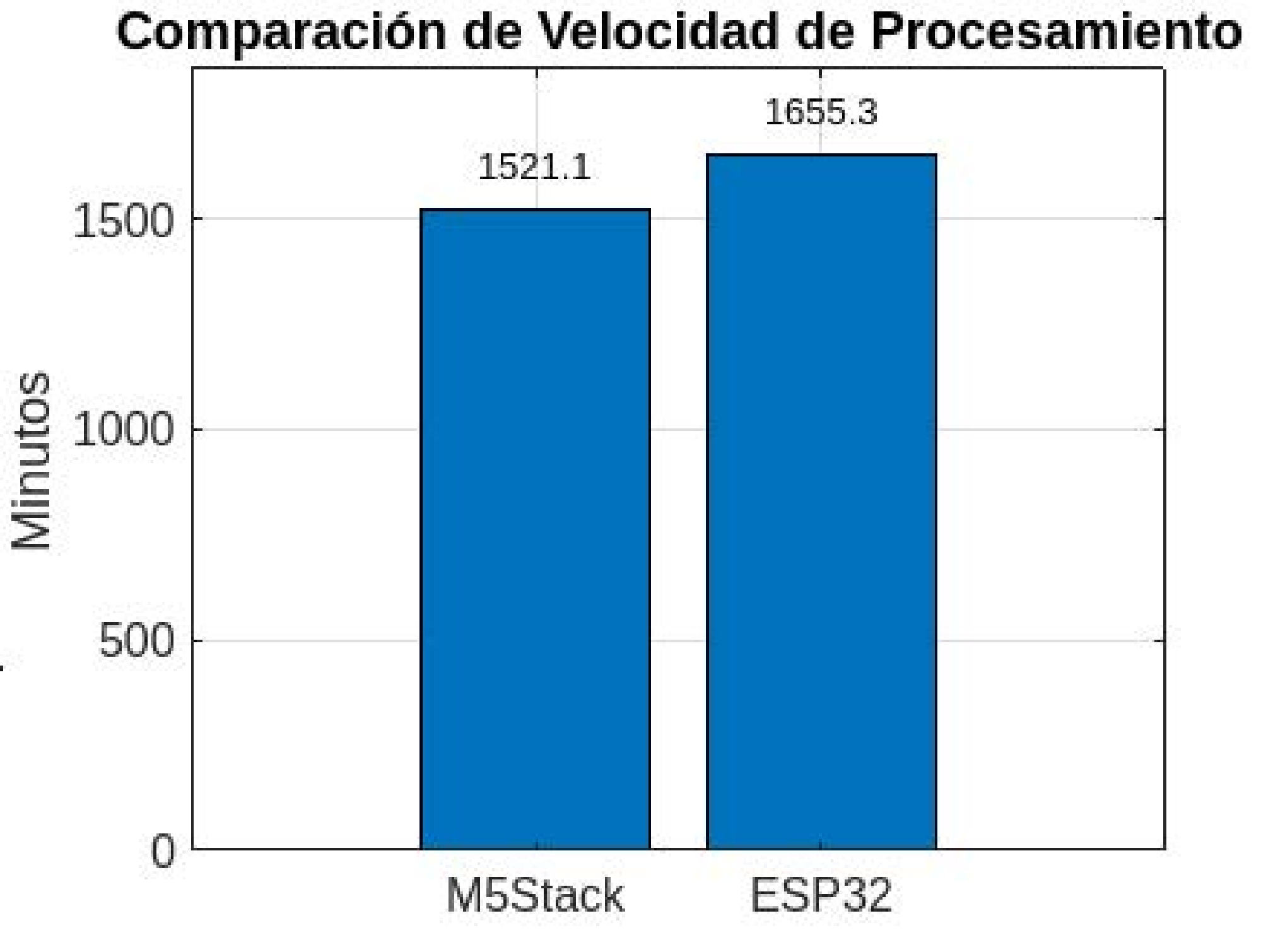

Velocidad de Transferencia Inalámbrica
Velocidad (Mbps)
100
80
60
40
20
0
50.0
50.2
ESP32
M5Stack

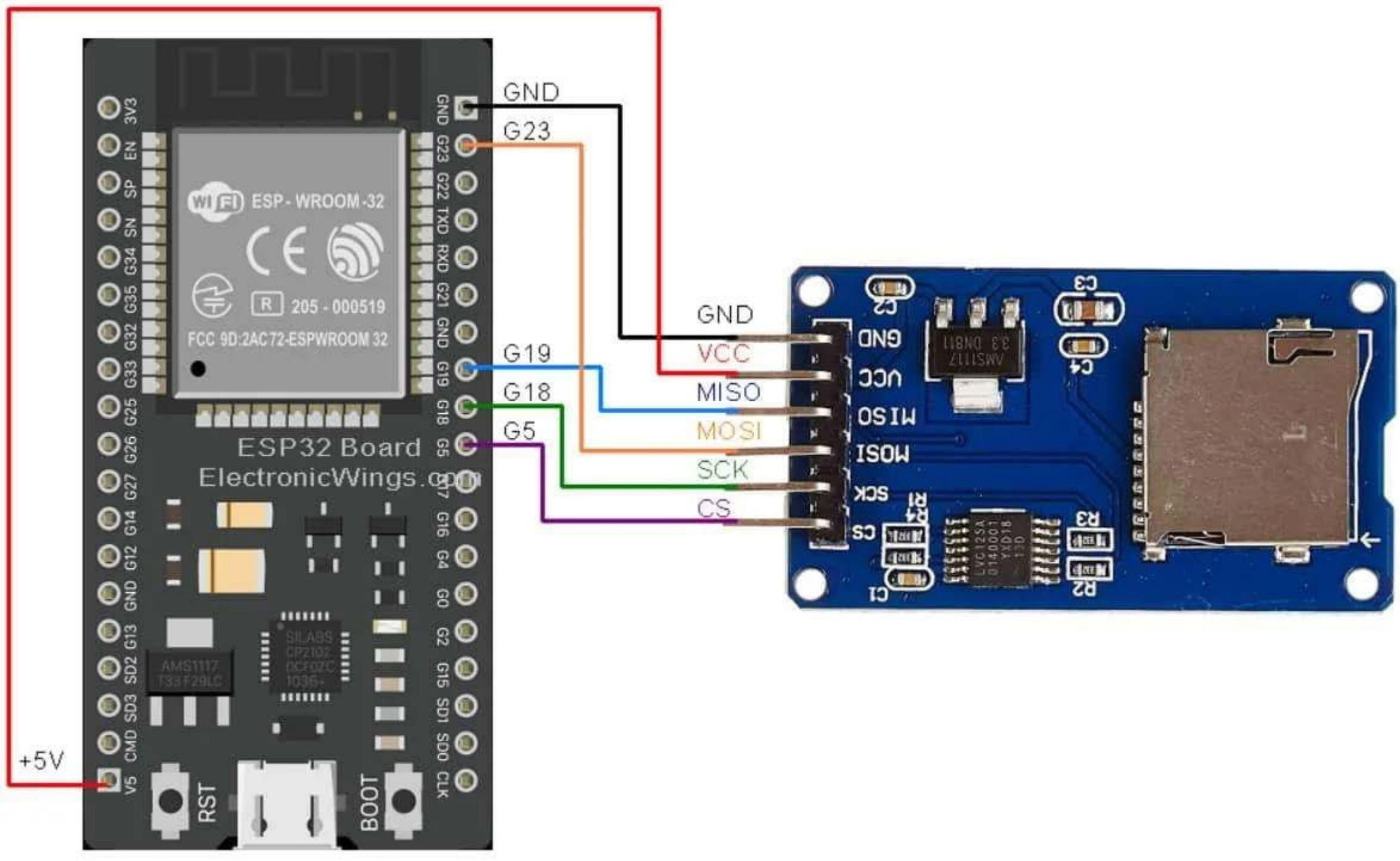
```
// Ejecutar las iteraciones
for (int i = 0; i < NUM_ITERACIONES; i++) {
  unsigned long startTime = millis(); // Tiempo de
   inicio del procesamiento

  // Resolver ecuaciones complejas
  solveEquations();

  unsigned long endTime = millis(); // Tiempo de
  fin del procesamiento
  unsigned long tiempoTranscurrido = endTime -
  startTime; // Calcular tiempo transcurrido
```

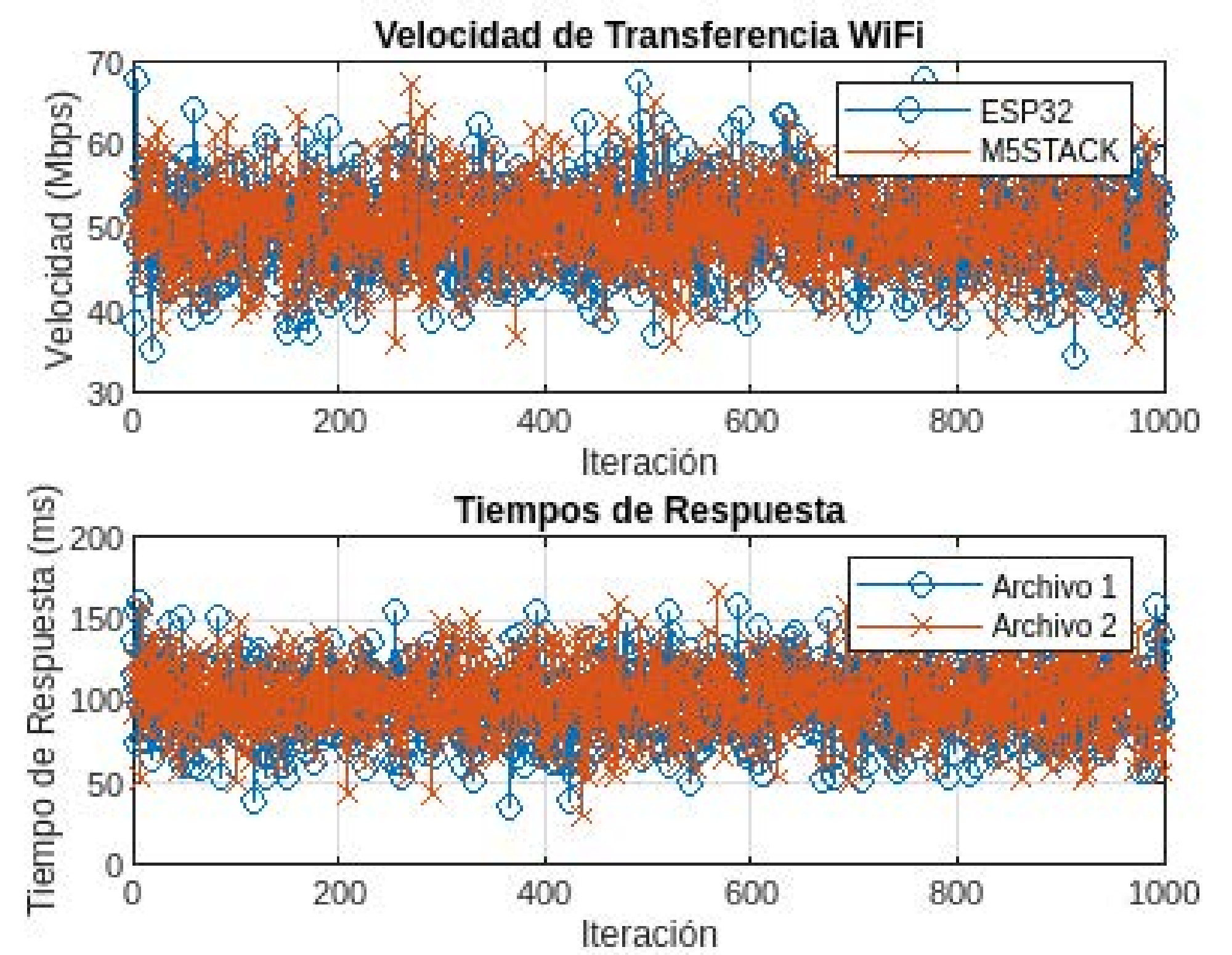

Comparación Velocidad de Procesamiento ESP32 vs M5STACK
1800
1600
1400
1200
1000
800
600
400
200
0
Tiempo
0
200
400
600
800
1000
Iteraciones
M5STACK
ESP32

Comparación de Modos Energéticos




Comparación de Velocidad de Procesamiento
1655.3
1521.1
1500
1000
500
0
Minutos
M5Stack
ESP32


GND
G23
GND
VCC
MISO
MOSI
SCK
CS
G19
G18
G5
+5V
ESP-WROOM-32
205 - 000519
FCC 9D:2AC72-ESPWROOM 32
ESP32 Board
ElectronicWings.com
RST
BOOT


Velocidad de Transferencia WiFi
Velocidad (Mbps)
70
60
50
40
30
0
200
400
600
800
1000
Iteración
ESP32
M5STACK
Tiempos de Respuesta
Tiempo de Respuesta (ms)
200
150
100
50
0
0
200
400
600
800
1000
Iteración
Archivo 1
Archivo 2

```
unsigned long duration = endTime - startTime;

// Calcular velocidad de transferencia (estimada
)
float transferSpeed = 0.0;
if (duration > 0) {
  // Estimaci n simple de velocidad en bits por
 segundo
  transferSpeed = (float)(1000 * 8) / duration;
// Conversi n a bits por segundo
}
```