Ciencia Latina Revista Científica Multidisciplinar, Ciudad de México, México.
ISSN 2707-2207 / ISSN 2707-2215 (en línea), mayo-junio 2026,
Volumen 10, Número 3.

https://doi.org/10.37811/cl_rcm.v10i3

DESARROLLO DE UN SISTEMA DE TELEMETRÍA EN TIEMPO REAL CON RT-LINUX PARA EL MONITOREO DE DESCARGA DE BATERÍAS LIPO EN UN VEHÍCULO ELÉCTRICO A ESCALA

DEVELOPMENT OF A REAL-TIME TELEMETRY SYSTEM USING RT-LINUX FOR MONITORING THE DISCHARGE OF LIPO BATTERIES IN A FULL-SCALE ELECTRIC VEHICLE

Josue Jimenez Ramirez
Instituto Politécnico Nacional ESIME, México

Pedro Guevara López
Instituto Politécnico Nacional ESIME, México

Rubén Vázquez Medina
Instituto Politécnico Nacional CICATA, México

Leobardo Hernández González
Instituto Politécnico Nacional, México

Omar Jiménez Ramírez
Instituto Politécnico Nacional, México

DOI: https://doi.org/10.37811/cl_rcm.v10i3.24666

Desarrollo de un Sistema de Telemetría en Tiempo Real con RT-Linux para el Monitoreo de Descarga de Baterías LiPo en un Vehículo Eléctrico a Escala

Josue Jimenez Ramirez[1]
jjimenezr1402@alumno.ipn.mx
https://orcid.org/0000-0002-3847-0554
Instituto Politécnico Nacional
ESIME- Culhuacan
México

Pedro Guevara López
pguevara@ipn.mx
https://orcid.org/0000-0001-5373-1403
Instituto Politécnico Nacional
ESIME- Culhuacan
México

Rubén Vázquez Medina
ruvazquez@ipn.mx
https://orcid.org/0000-0002-6210-4097
Instituto Politécnico Nacional
CICATA, Querétaro
México

Leobardo Hernández González
lhernandezg@ipn.mx
https://orcid.org/0000-0002-4555-8695
Instituto Politécnico Nacional
México

Omar Jiménez Ramírez
ojimenezr@ipn.mx
https://orcid.org/0000-0003-1566-8604
Instituto Politécnico Nacional
México

RESUMEN

En la actualidad la gestión eficiente de energía aplicada a la electromovilidad requiere el uso de métodos precisos de monitoreo con el objetivo de garantizar la seguridad y gestionar la vida útil de los sistemas de almacenamiento (baterías). Debido a los costos elevados y el riesgo que implica la realización de pruebas experimentales de vehículos eléctricos de tamaño real, el uso de vehículos eléctricos a escala se posiciona como una herramienta experimental para la validación de sistemas de telemetría en un entorno seguro. En este sentido, en este trabajo se presenta el desarrollo de un sistema de telemetría en tiempo real (RT) orientado al monitoreo de baterías de LiPo, utilizando una plataforma y un auto eléctrico a escala como sistema de pruebas controladas. El sistema se basa en una arquitectura que usa Python y un protocolo de comunicación MAVLink ejecutados en un sistema operativo Linux con kernel de tiempo real (PREEMPT_RT). Se implementó un método de conteo de Coulomb para la estimación del Estado de Carga (SoC) y un controlador de tipo PID (Proporcional Integral Derivativo), para la ejecución de perfiles controlados de carga en un motor de tipo brushless. Los resultados validan la medición de métricas críticas en la batería, proporcionando una base experimental escalable para futuros trabajos de optimización en los sistemas energéticos de vehículos eléctricos a gran escala.

Palabras clave: LiPo, estado de carga, MAVLink, sistemas embebidos, telemetría en tiempo real

[1] Autor principal.
Correspondencia: pguevara@ipn.mx

Development of a Real-Time Telemetry System Using RT-Linux for Monitoring the Discharge of LiPo Batteries in a Full-Scale Electric Vehicle

ABSTRACT

Currently, efficient energy management in electromobility requires the use of precise monitoring methods to ensure safety and manage the service life of storage systems (batteries). Due to the high costs and risks involved in conducting experimental tests on full-scale electric vehicles, the use of scaled-down electric vehicles serves as an experimental tool for validating telemetry systems in a safe environment. In this regard, this paper presents the development of a Real-time (RT) telemetry system designed for monitoring LiPo batteries, using a platform and a scaled-down electric car as a controlled testing system. The system is based on an architecture that uses Python and the MAVLink communication protocol running on a Linux operating system with a real-time kernel (PREEMPT_RT). A Coulomb counting method was implemented to estimate the State of Charge (SoC), and a PID controller was used to execute controlled charging profiles on a brushless motor. The results validate the measurement of critical battery metrics, providing a scalable experimental basis for future optimization work on large-scale electric vehicle power systems.

***Keywords*:** embedded systems, lipo batteries, mavlink, real-time telemetry, state of charge

Artículo recibido 25 abril 2026
Aceptado para publicación: 25 mayo 2026

INTRODUCCIÓN

La gestión de energía en la electromovilidad depende de la caracterización dinámica de las baterías, posicionandola como el componente crítico para la autonomía de vehículos eléctricos (Mundige et al., 2023; Pérez, 2024). La problemática de la investigación radica en que la telemetría de descarga de baterías en condiciones de operación dinámica o reales suele verse afectada por diferentes situaciones, tales como la adquisición de datos causado por las latencias lo que genera una falla o inconsistencia en el diagnóstico del estado de carga de las baterías. La importancia de este estudio busca encontrar una estabilidad de telecontrol para prevenir degradaciones o daños prematuros de las celdas de la batería. (Pérez, 2024). La presente investigación sustenta la teoría sobre sistemas en tiempo real, ya que la reconstrucción de los tiempos de respuesta y transporte, cálculos matemáticos y el filtrado digital es vital para compensar los retardos en las plataformas RT-Linux, haciendo que los datos de voltaje y corriente que sean capturados puedan demostrar la dinámica del sistema (Martínez et al., 2019; Martínez & López, 2019). En este contexto, el presente trabajo se utiliza un vehículo eléctrico a escala (Tamiya F104 Pro II) como banco de pruebas experimentales, aplicando una arquitectura basada en el protocolo de comunicación MAVLink, el cual sirve como puente entre la controladora encargada del telecontrol (Pixhawk), un sistema operativo basado en Linux instalado en una Raspberry Pi 5 optimizado con un parche PREEMPT_RT para logar el determinismo necesario (Guamushig & Manobanda, 2023) con el propósito de monitorear el proceso de descarga (SoC) de una batería LiPo de 3 celdas. El objetivo principal es la capacidad de obtención de datos críticos en la descarga de una batería, la corriente consumida, el voltaje, el cálculo de SoC, y así asentar bases para futuros trabajos que sean de utilidad para encontrar nuevas técnicas de optimización de energía en este tipo de baterías y se pueda aplicar en vehículos eléctricos a gran escala.

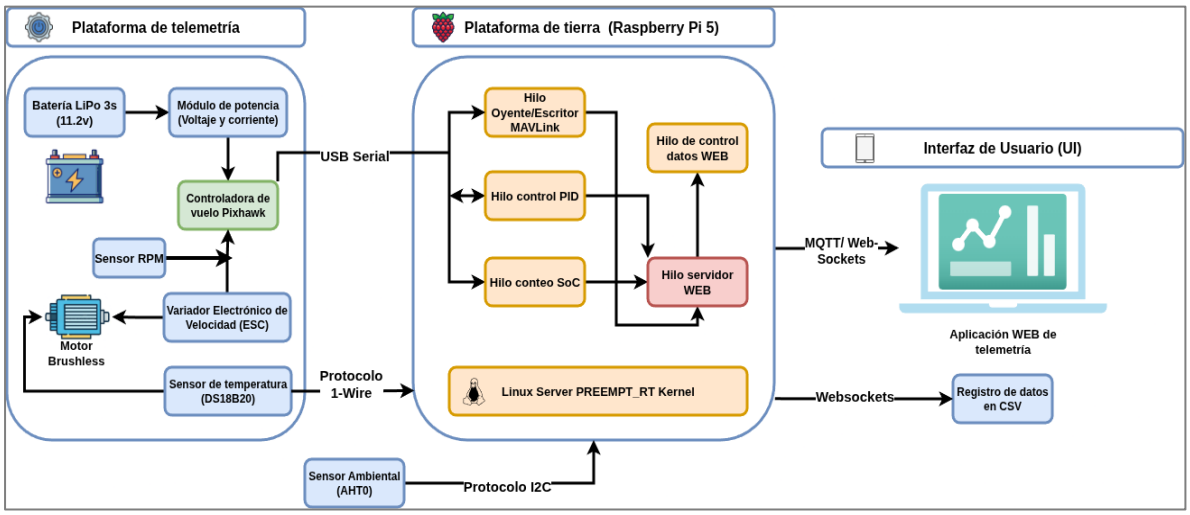
En este contexto, la principal contribución de este trabajo consiste en el desarrollo e integración de una arquitectura de telemetría en tiempo real orientada al monitoreo dinámico de baterías LiPo en un vehículo eléctrico a escala, utilizando una plataforma basada en Raspberry Pi 5 y una controladora Pixhawk bajo el protocolo MAVLink. La propuesta incluye la implementación Rt-Linux mediante el parche PREEMPT_RT, permitiendo la evaluación experimental del comportamiento del sistema a través del análisis de tiempos de ejecución, cumplimiento de plazos máximos de respuesta y variaciones

no determinísticas (jitter). También se valida experimentalmente la capacidad del sistema para ejecutar tareas concurrentes de adquisición, procesamiento, control y visualización de datos en tiempo real enfocadas a sistemas de electromovilidad.

METODOLOGÍA

El desarrollo del presente proyecto se orienta al diseño, integración y validación experimental en laboratorio de una plataforma de telemetría en tiempo real (RT), orientada al análisis y monitoreo de la descarga dinámica en baterías LiPo. La arquitectura del sistema se muestra en la Figura 1, la cual muestra las fronteras operativas entre el hardware de descarga (Tamiya F104 II Pro), los nodos de adquisiciones de datos de los sensores (temperatura, corriente, voltaje y velocidad angular), la capa de procesamiento determinista en la Estación de Tierra (Raspberry Pi 5) y la interfaz de usuario remota.

Figura 1. Diagrama de metodología de proyecto de telemetría



Configuración de Kernel PREEMPT_RT en la estación de tierra

Como estación de tierra (Raspberry Pi 5), se implementó un entorno de operación determinista para disminuir la variación no determinística del tiempo de respuesta y latencias que se generan en los sistemas operativos de propósito general. Se optó por la instalación de Ubuntu Server como sistema operativo, ya que cuenta con integración nativa con Raspberry Pi 5. Posteriormente, se instaló el parche PREEMPT_RT través del metapaquete de compilación nativa que ofrece Ubuntu, la cual se ejecuta con el comando en terminal: pro enable realtime-kernel --variant=raspi.

La validación de la correcta instalación del parche PREEMPT_RT se realizó de manera formal mediante la instrucción **`uname -r`**, comprobando la correcta inicialización del planificador de hilos críticos en tiempo real. La captura de terminal que demuestra el éxito de instalación se muestra en la Figura 2.

Figura 2. Instalación correcta del kernel PREEMPT_RT.

```
josue@Josue:~/Telemetria$ uname -r
6.8.0-2038-raspi-realtime
josue@Josue:~/Telemetria$
```

Ya en nivel de software de alto nivel (script de Python), se programó los privilegios a través de la biblioteca **`os`**. Al iniciar el script captura su identificador único **`pid = os.getpid()`** y se configura la política de planificación bajo el algoritmo de prioridad Primero en Entrar, Primero en Salir (FIFO), dando el nivel máximo del planificador del kernel (**`RT_PRIORITY = 99`**), quitando prioridad al resto de servicios en segundo plano del sistema operativo.

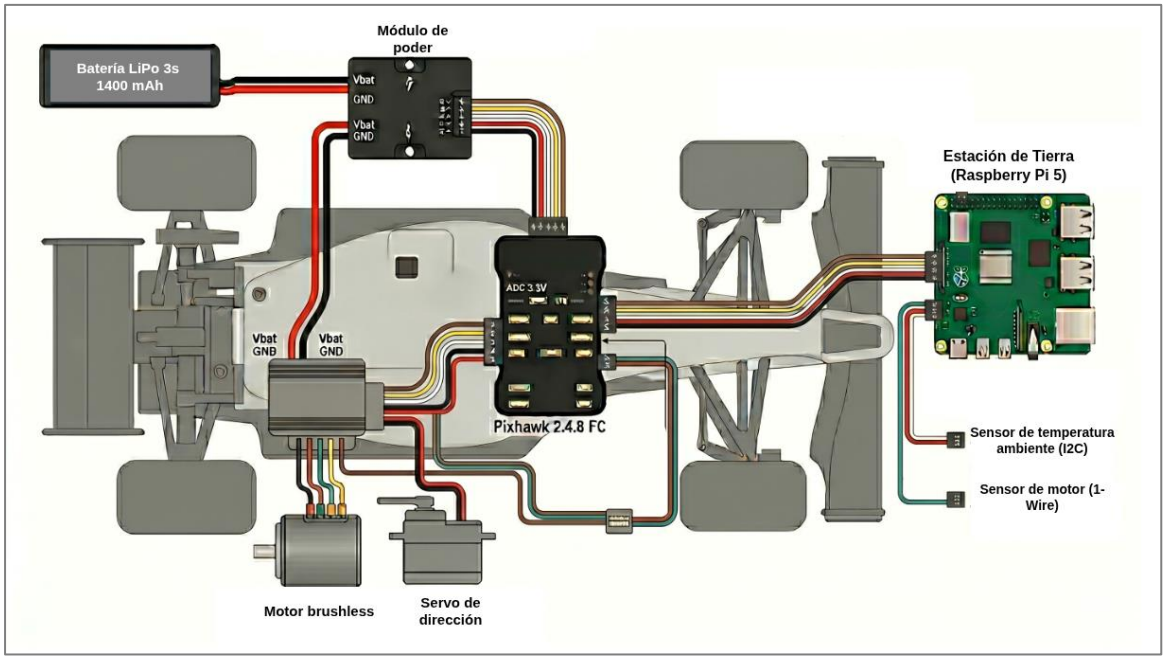
Integración de chasis, competentes electrónicos y sensores

La siguiente fase metodológica indica la instrumentación e integración de la plataforma de telemetría, usando un chasis de un vehículo eléctrico a escala (Tamiya F104 Pro II) el cual cuenta con un motor brushless hobbywing ezrun 3652 y una ESC (Electronic Speed Crontroller) hobbywing ezrun max10 g2. Todos esto está montado en una estructura fabricada con paredes de acrílico y soportes estructurales impresos en 3D, usando como material el ABS por su dureza. La batería es de LiPo de 3 celdas en serie (3S1P) con un voltaje nominal de 11.1 V (12.6 V en carga máxima) y una capacidad total nominal de 1300 mAh.

Para la captura de las variables de interés (corriente, voltaje y velocidad angular) se utilizó y configuró la controladora de vuelo Pixhawk (Meier et al., 2011) basada en Ardupilot, conectada a un módulo de poder para la medición de tensión y corriente en el bus principal de corriente continua. Ademas, se incluye un sensor de velocidad de efecto Hall para medir las RPM del motor y obtener su velocidad.

Para la medición de variables térmicas y ambientales que puedan influir en la tasa de descarga de la batería, se integró un sensor de temperatura DS18B20 a la carcasa metálica del motor brushless usando el protocolo 1-Wire, y un sensor de temperatura y humedad ambiente AH10, conectado por el bus I2C de la estación de tierra, la cual registra temperatura y humedad del laboratorio en el cual están haciendo pruebas. El esquema físico de las conexiones del hardware se muestra en la Figura 3.

Figura 3. Conexiones electrónicas físicas del chasis Tamiya F104 Pro II



Hilos y decodificación de mensajes MAVLink

Para evitar el bloqueo de procesos en el sistema de telemetría, se implementó una arquitectura multihilo aplicando la biblioteca threading en python. La comunicación física se realizó mediante comunicación serial USB a través del puerto de dispositivo /dev/ttyACM0 a una velocidad fija de 115,200 bps. Sobre este canal de comunicación se usa el protocolo MAVLink. El flujo del sistema comienza con un subproceso independiente el cual se denomina “Hilo oyente / escritor “. Este opera en un modo de no bloqueo el cual llama asíncronamente a una función que se encarga de extraer del búfer de entrada de datos con nombre especifico (SYS_STATUS), valores de voltaje de batería, corriente y RPM. A su vez se implementó un objeto de exclusión mutua, el cual permite un acceso seguro y concurrente a los hilos encargados del control del auto y servidor web.

Evaluación de rendimiento computacional en tiempo real

Una parte fundamental de la metodología es la gestión determinista del bucle principal, el cual está programado para una ejecución de un periodo de muestreo objetivo fijo de $T_s = 50.0\ ms(20\ Hz)$. Al comenzar de cada ciclo, el hilo con alta prioridad captura una marca de tiempo en la SBC Raspberry Pi 5, invocando una función para este fin. Después de esto, el hilo toma el control exclusivo de la CPU para que este procese de forma secuencial la decodificación de las variables físicas, estimación de estado de carga y la llamada a un lazo cerrado de control de velocidad basado en un control tipo Proporcional

Integral Derivativo (PID) que envía las señales PWM hacia el ESC (Electronic Speed Crontroller). Al finalizar las tareas correspondientes, el algoritmo registra una segunda marca de tiempo para deducir el tiempo de ejecución (C), el cual es el coste real computacional consumido por el procesador antes del plazo máximo de respuesta (50 ms). En base a la literatura enfocada en sistemas de gestión de baterías o BMS (Plett, 2015) indica que para mantener un error de integración bajo para el método de estimación SoC por conteo de Coulomb durante caídas fuertes de tensión, es necesario operar en una frecuencia entre 10 Hz y 50 Hz, por lo tanto, se justifica el uso de 20 Hz se puede dar seguimiento a la dinámica de descarga de la batería sin pérdida de información o errores de integración altos. Si el coste computacional supera ese plazo $C > 50.0ms$, el sistema registra un sobrepaso temporal e incumpliendo restricciones de tiempo real (Buttazzo, 2024). Se calcula la variación no determinista del tiempo de respuesta real frente al plazo máximo de respuesta, almacenando la magnitud de esta fluctuación.

Algoritmo para estimación de SoC

La estimación del estado de carga (SoC) se ejecuta concurrentemente, el cual aplica el método de integración por conteo de Coulomb**.** Tomando en cuenta el intervalo de tiempo $\Delta t = 0.05s$, la capacidad acumulada de consumo se calcula a partir de la ecuación 1:

$$\Delta mAh = \frac{I(A)*\Delta t(s)}{3.6} \quad (1)$$

La capacidad de energía remanente se va descontando respecto a la capacidad nominal previamente calibrada (1,300 mAh), lo cual permite analizar la descarga de la batería LiPo bajo diferentes perfiles de conducción experimentales y controlados. Para proteger la batería y evitar posibles daños, el sistema integra un protocolo de seguridad (failsafe) (Lu. et al., 2013). El failsafe evalúa continuamente tres criterios de fallas:

Corte por voltaje crítico

Si el voltaje en la batería cae por debajo de 9.6 V (umbral mínimo máximo para evitar daño en batería de 3 celdas), se ejecuta un contador el cual, si persiste el voltaje por debajo del umbral crítico por más de 3 segundos, ejecuta un paro de emergencia de todo el hardware.

Límite inferior de SoC

Si la capacidad que se calculó llega a disminuir por debajo de un 15% de carga en la batería, cualquier prueba que se encuentre en ejecución se detendrá por completo.

Falla en sensores: Si existen incongruencias en las RPM (RPM altas) y un valor bajo de consumo de corriente y esto continua por más de 5 segundos consecutivos, se ejecuta un paro de emergencia en el motor burshless para evitar daños al sensor de efecto Hall.

Interfaz gráfica y servidor WEB

El diseño implementado en la interfaz gráfica utiliza hilos independientes, esto es para evitar que el renderizado de las gráficas no perjudique el determinismo computacional del lazo cerrado. Ante esto el backend del sistema de telemetría está ejecutando asíncronamente un hilo secundario, utilizado para el servidor Flask, el cual se encarga de exponer los estados internos de la memoria, usando un formato JSON.

El servicio frontend programado en HTML, establece un canal asíncrono el cual esta periódicamente consultando al backend con una tasa de actualización síncrona de 50 ms. Los datos se mapean en indicadores visuales y gráficas. La distribución y diseño del frontend se puede observar en la Figura 4. la cual muestra los indicadores visuales de los datos mapeados y las gráficas de interés.

El diseño de la interfaz web se basó en los Principios Heurísticos de Usabilidad de Jakob Nielsen (Nielsen, 1994), los cuales son considerados el estándar global. Los principios utilizados son los siguientes:

- Visibilidad del estado del sistema.
- Diseño estético y minimalista.
- Control y libertad de usuarios.
- Consistencia y estándares.
- Coincidencia entre el sistema y el mundo real.

Figura 4. Interfaz gráfica de usuario.

Finalmente, los datos obtenidos son guardados en un archivo plano CSV local con un bufer de memoria directo, guardando la marca de tiempo, valores eléctricos y las métricas computacionales de tiempo real para posteriormente el análisis de descarga y tiempos de ejecución del sistema y así ver el comportamiento dinámico de la plataforma de telemetría y validar si se considera un sistema en tiempo real.

RESULTADOS Y DISCUSIÓN

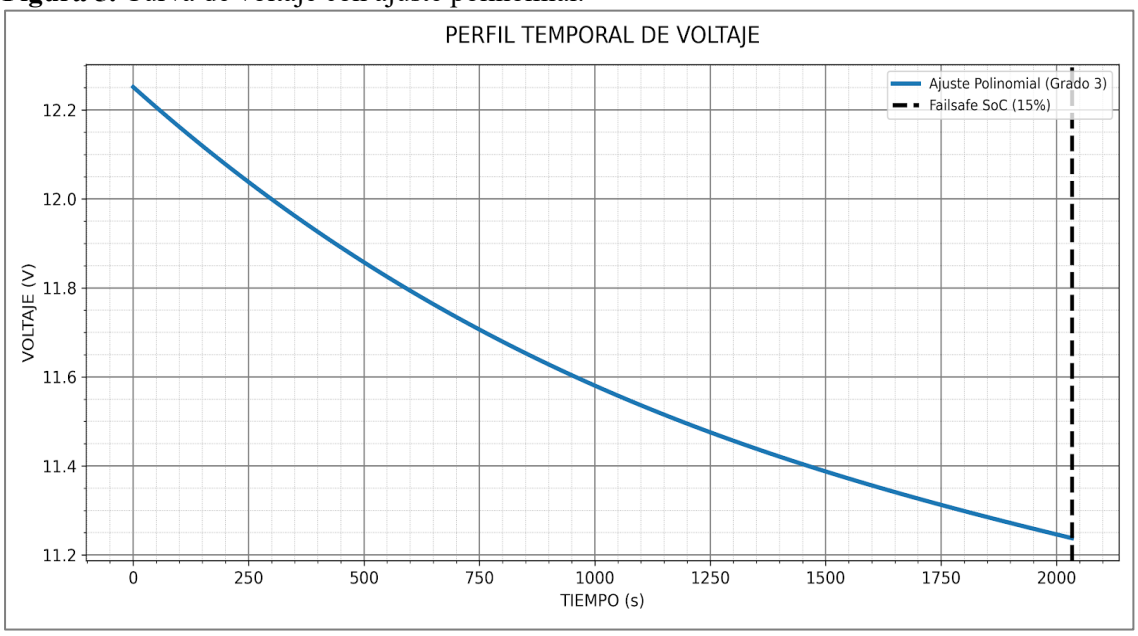
La validación del sistema de telemetría se llevó a cabo mediante una prueba de laboratorio. Se puso en accionamiento un ciclo completo de descarga bajo un perfil de conducción económico. Se pudieron registrar un total de 40,579 ciclos de procesamiento determinista, con una duración de operación total de 2,033.36 segundos (33.89 minutos de adquisición de datos sin interrupción), dando una base de datos sólida para analizar el comportamiento dinámico de la descarga y consumo computacional del prototipo.

Dinámica del perfil de descarga de batería LiPo

El análisis del comportamiento de descarga de la batería de LiPo de 3 celdas inicio con un estado de carga (SoC) de 99.27% y un valor de voltaje en circuito abierto de 12.40 V. Durante la prueba controlada de laboratorio, la demanda de corriente impuesta por el motor brushless registro un flujo de corriente continuo promedio de 1.95 A, registrando picos máximos de corriente de hasta 4.19 A con rampas de aceleración cíclicas máximas de 3,000 RPM.

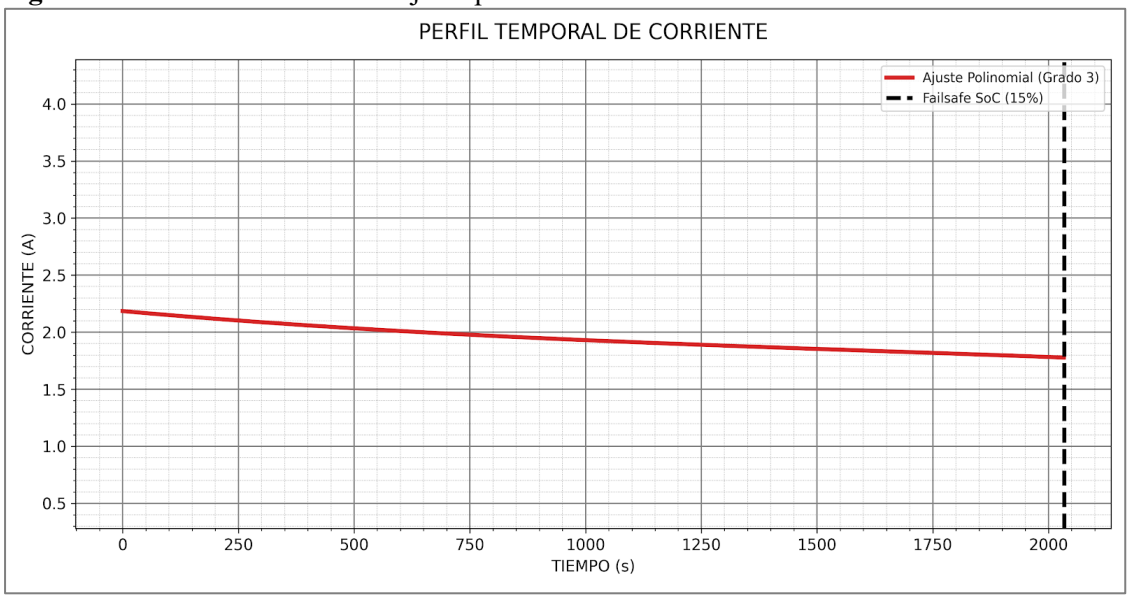
Como resultado a esta demanda de trabajo, la curva de voltaje tuvo una caída progresiva de voltaje, combinada con caídas de tensión causadas por la resistencia interna (sag), alcanzando un mínimo de 11.09 V bajo carga. Los resultados afirman el éxito del algoritmo de paro de emergencia (Failsafe), ya que la lógica del algoritmo detuvo la misión de manera controlada y con éxito al detectar un SoC del 14.93%, exactamente a los 2,033.36 segundos, cumpliendo con la restricción programada de no pasar por debajo del 15% de carga, esto para prevenir una sobre-descarga que pueda dañar físicamente la batería LiPo. La Figura 5. Muestra el comportamiento dinámico del voltaje ante la carga de trabajo generada por el motor, aplicando como post procesado de los datos un ajuste polinomial.

Figura 5. Curva de voltaje con ajuste polinomial.



Así mismo, la Figura 6, muestra el comportamiento dinámico de la carga de corriente demandada por el motor aplicando el ajuste polinomial.

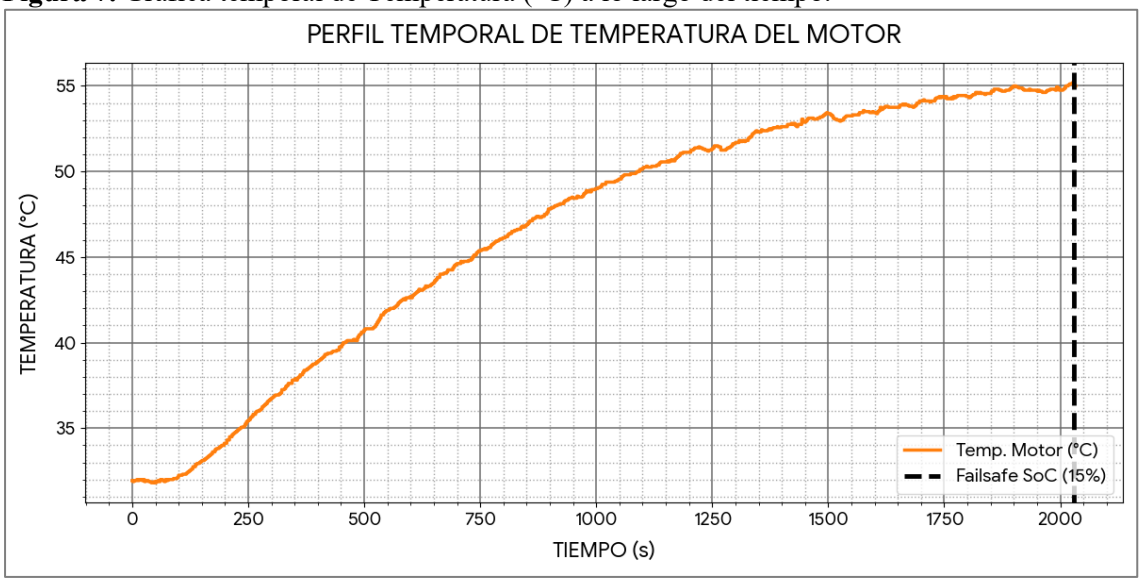
Figura 6. Curva de corriente con ajuste polinomial.



Resultados térmicos del sistema motriz

La monitorización de las lecturas térmicas se valida gracias a los resultados obtenidos por el sensor DS18B20, el cual registró una temperatura inicial en el motor de 31.94 °C. debido al calentamiento generado por la conducción de corriente en el motor y la carga de trabajo demandada, la prueba experimento un crecimiento a lo largo de los 33 minutos de duración de la prueba, la cual se estabilizó en un valor final de 55.00 °C y registro un pico máximo de temperatura de 55.19 °C. Con esto se valida la necesidad de nodos sensoriales de temperatura para la evaluación de la eficiencia térmica del motor. Esto se puede confirmar en la Figura 7.

Figura 7. Gráfica temporal de Temperatura (°C) a lo largo del tiempo.

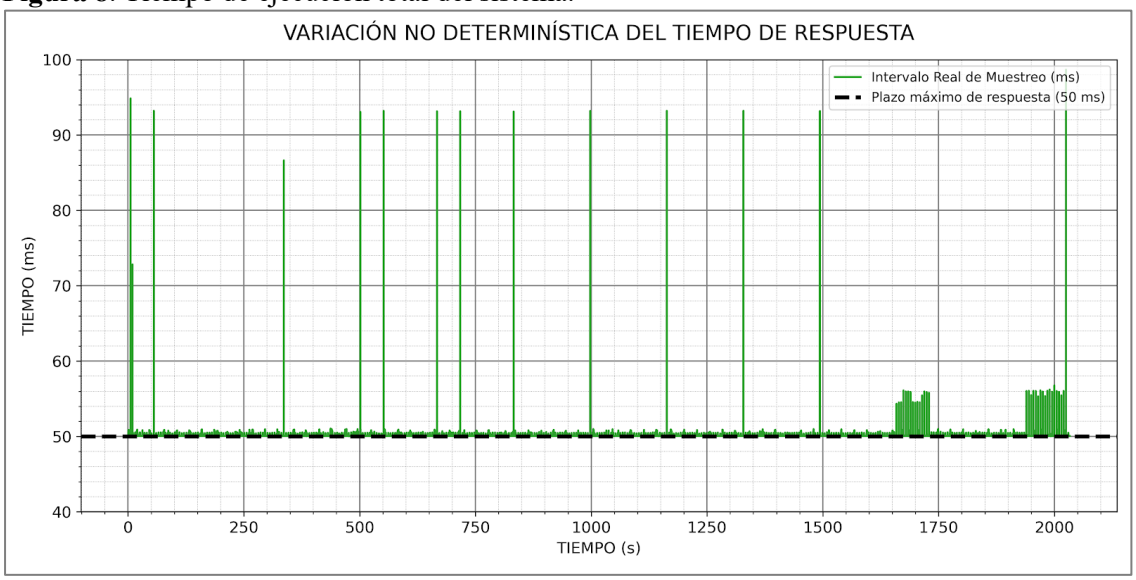


Evaluación computacional de sistema en tiempo real

Una parte fundamental es la validación de la capacidad del software para cumplir de manera determinista con las restricciones temporales bajo el parche del kernel PREEMPT_RT. El tiempo de muestreo objetivo o plazo máximo de respuesta se estableció en 50.0 ms. Durante esta prueba los datos obtenidos demostraron un rendimiento estable y bueno, el tiempo que le tomo a todo el sistema de decodificar los mensajes MAVLink, calcular variables, despliegue de interfaz web y escritura en el disco, promedio apenas 1.49 ms.

Añadiendo, bajo condicione de estrés o picos de escritura en el bufer de almacenamiento, el tiempo de ejecución máximo que se registro fue de 5.11 ms, ante estos resultados se puede decir que el sistema logro un 100% de cumplimiento del plazo máximo de respuesta a lo largo de los 40,579 ciclos, demostrando que en el peor de los casos el procesador utilizo únicamente el 10.22% del tiempo disponible antes de llegar al plazo máximo de respuesta de 50 ms. La Figura 8 muestra como la curva de tiempo de ejecución se mantiene en promedio alejada del plazo máximo, por lo cual genera una bandera de validación exitosa de un 100&.

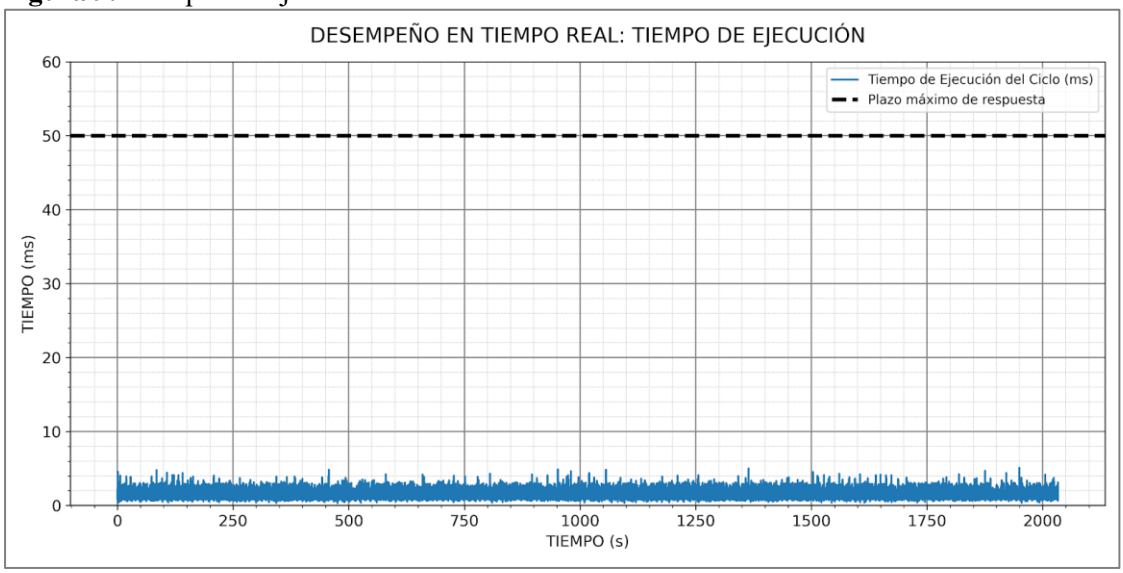
Figura 8. Tiempo de ejecución total del sistema.



Sin embargo, el análisis del periodo transcurrido entre cada actividad de cada ciclo revela el impacto de la arquitectura del software sobre el determinismo global. Se demostró que hay una variación no determinística del tiempo de respuesta, con picos de hasta 98.67 ms. Estas latencias se pueden observar en la Figura 9 y no se atribuyen a la planificación de tareas del núcleo PREEMPT_RT.

Como primer posible razón a estos picos de latencia, se les atribuye a las operaciones bloqueantes que se generan por persistencia síncrona de datos de la memoria MicroSD, lo cual provoca en el hilo principal, estados de espera y pausas temporales. Otra de las posibles razones se debe al lenguaje de programación Python, el cual no está diseñado para la implementación de sistemas en tiempo real, como lo podría ser C o C++. Ante esto el prototipo de telemetría desarrollado se puede catalogar como Tiempo Real Suave (Soft Real -Time).

Figura 9. Tiempos de ejecución.



DISCUSIÓN

El desarrollo del sistema de telemetría demostró ser efectivo para el análisis de la descarga de baterías de LiPo en una plataforma vehicular a escala. El análisis entre la demanda de corriente (picos de 4.19 A) y la evaluación térmica del motor (55 °C) valida a este entorno como un banco de pruebas seguro y escalable para una futura implementación. A su vez el método de integración numérica por conteo de Coulomb demostró un buen comportamiento para gestionar el Estado de Carga (SoC) de forma concurrente. Se pudo garantizar que el sistema de seguridad programado (Failsafe), pudo detener la descarga de forma autónoma en el umbral crítico del 14.93% demostrando que las arquitecturas embebidas de bajo costo pueden ser de utilidad para el desarrollo de sistemas de telemetría en baterías a escala o en un tamaño real.

En cuanto al desempeño computacional, se identificaron las limitaciones de la arquitectura de software; Aunque la configuración del kernel PREEMPT_RT mantuvo el tiempo de ejecución en promedio por debajo de los 5.11 ms, el periodo de muestreo presentó latencias provocadas por operaciones bloqueantes en la memoria MicroSD y el uso de un lenguaje de programación no diseñado para sistemas en tiempo real Hard. Ante todo, esto el sistema opera bienbajo un régimen de Tiempo Real no Crítico.

CONCLUSIONES

El desarrollo del sistema de telemetría en tiempo real permitió validar experimentalmente una arquitectura embebida orientada al monitoreo dinámico de baterías LiPo en un vehículo eléctrico a escala, integrando una Raspberry Pi 5 con kernel PREEMPT_RT, comunicación MAVLink y una plataforma de adquisición basada en Pixhawk. Los resultados obtenidos demostraron que el sistema fue capaz de mantener una operación estable y continua durante más de 40 mil ciclos de ejecución, garantizando la captura y procesamiento de variables críticas como voltaje, corriente, temperatura, velocidad del motor y estado de carga (SoC). Por otra parte, la implementación del método de conteo de Coulomb permitió estimar de manera adecuada el comportamiento de descarga de la batería bajo perfiles controlados de conducción, mientras que la integración del sistema de seguridad (Failsafe) protegió la integridad de la batería al detener automáticamente las pruebas cuando el nivel de carga alcanzó el umbral crítico establecido. Esto demuestra que las plataformas embebidas de bajo costo pueden utilizarse como herramientas confiables para el análisis energético y la validación experimental en aplicaciones relacionadas con electromovilidad.

Desde el punto de vista computacional, la integración del kernel PREEMPT_RT permitió reducir los tiempos de ejecución y mantener el cumplimiento de las restricciones temporales establecidas para la adquisición y procesamiento de datos. Aunque se identificaron variaciones temporales ocasionadas principalmente por operaciones de entrada/salida y limitaciones propias del lenguaje Python, el sistema mantuvo un comportamiento estable dentro de un esquema de Tiempo Real Suave (Soft Real-Time), lo cual resulta suficiente para aplicaciones de monitoreo, telemetría y supervisión energética.

Uno de los principales aportes del proyecto consiste en la integración completa de hardware, software y análisis temporal en una plataforma experimental funcional y escalable, capaz de servir como banco

de pruebas para futuras investigaciones relacionadas con sistemas de gestión de baterías, vehículos eléctricos y arquitecturas embebidas en tiempo real. Asimismo, el trabajo proporciona una base tecnológica accesible para el desarrollo de sistemas de supervisión energética en aplicaciones académicas, industriales y de investigación.

REFERENCIAS BIBLIOGRAFICAS

Adam, G. K., Petrellis, N., & Doulos, L. T. (2021). Performance assessment of Linux kernels with PREEMPT_RT on ARM-based embedded devices. *Electronics*, 10*(11), 1331. https://doi.org/10.3390/electronics10111331

Bermúdez Pérez, M. A. (2024). *Sistema de telemetría para optimización de rendimiento de vehículo eléctrico* [Informe de pasantía de investigación, Universidad Autónoma de Occidente]. Repositorio Institucional UAO. https://hdl.handle.net/10614/15952

Buttazzo, G. C. (2024). *Hard real-time computing systems: Predictable scheduling algorithms and applications* (4th ed.). Springer. https://doi.org/10.1007/978-3-031-45410-3

Canchignia Guamushig, G. A., & Manobanda Manobanda, D. A. (2023). *Implementación de un sistema de telemetría para el monitoreo y gestión de consumo de energía eléctrica para el prototipo Shell Eco-Marathon de la UPS sede Quito, Campus Sur* [Tesis de licenciatura, Universidad Politécnica Salesiana].

González-Baldovinos, D. L., Guevara-López, P., Cano-Rosas, J. L., Valdez-Martínez, J. S., & López-Chau, A. (2022). Response times reconstructor based on mathematical expectation quotient for a high priority task over RT-Linux. *Mathematics*, 10*(1), 134. https://doi.org/10.3390/math10010134

Lu, L., Han, X., Li, J., Hua, J., & Ouyang, M. (2013). A review on the key issues for lithium-ion battery management in electric vehicles. *Journal of Power Sources, 226*, 272–288. https://doi.org/10.1016/j.jpowsour.2012.10.060

Meier, L., Tanskanen, P., Fraundorfer, F., & Pollefeys, M. (2011). PIXHAWK: A system for autonomous flight using onboard computer vision. En *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2011)* (pp. 2992–2997). IEEE.

https://doi.org/10.1109/ICRA.2011.5980229

Mundige, R. M., Joseph, S. A., Badachi, C., Sharma, A., & Jha, A. (2023). Development of telemetry system for electric vehicle. En *2023 7th International Conference on Computer Applications in Electrical Engineering – *Recent Advances (CERA)** (pp. 1–6). IEEE. https://doi.org/10.1109/CERA59325.2023.10455470

Nielsen, J. (1994). Enhancing the explanatory power of usability heuristics. En **Proceedings of the SIGCHI Conference on Human Factors in Computing Systems** (pp. 152–158). Association for Computing Machinery. https://doi.org/10.1145/191666.191729

Plett, G. L. (2015). *Battery management systems, Volume I: Battery modeling*. Artech House.

Reghenzani, F., Massari, G., & Fornaciari, W. (2020). The real-time Linux kernel: A survey on PREEMPT_RT. **ACM Computing Surveys*, 52*(1), Article 18, 1–36. https://doi.org/10.1145/3297714

Valdez, J., Delgado, G., Guevara, P., & Cano, J. (2019). Transmission times reconstruction in a telecontrolled real-time system. **IEEE Latin America Transactions*, 17*(3), 349–357. https://doi.org/10.1109/TLA.2019.8863304