Infografía de Consumo y Rendimiento

Proyecto Hardware 2025 Juego Beat Hero

Autores:

Pablo Villa Camañes
Alejandro Lacosta Ramos

Contexto general:

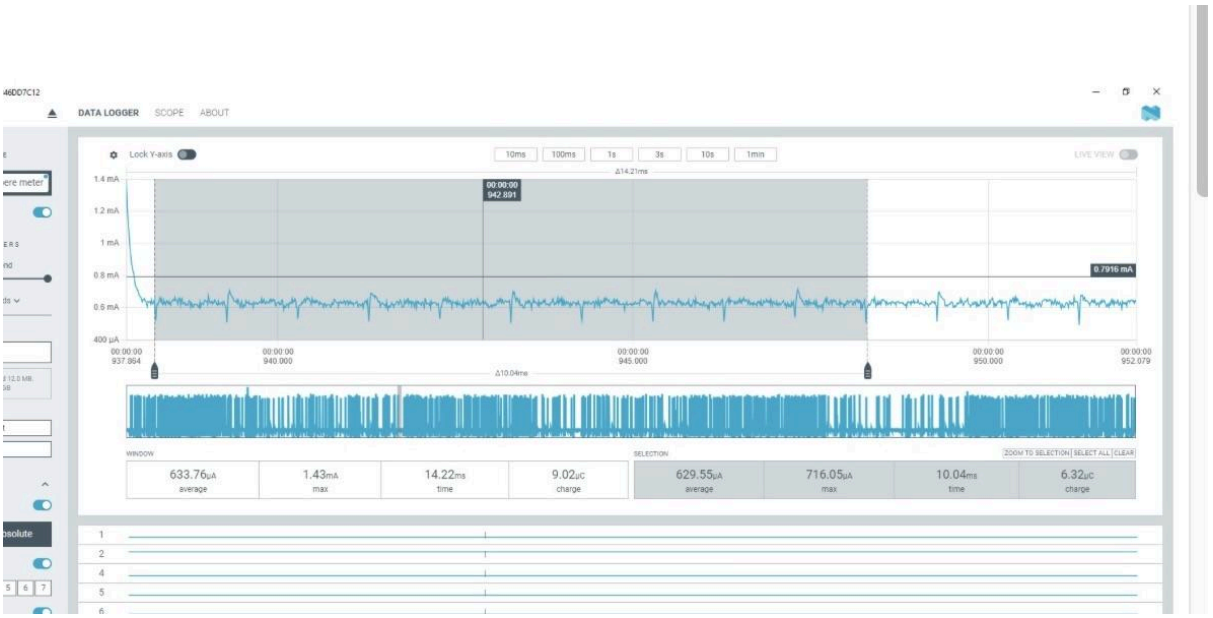
A lo largo del proyecto de Hardware hemos ido evolucionando desde un simple “blink” de LED hasta un minijuego completo tipo “*Beat Hero”*, pero siempre con una idea fija en mente: no sólo que el sistema funcione, sino que lo haga de forma eficiente y medible en términos de consumo energético y rendimiento. En las primeras prácticas se partía de un código muy ingenuo, con bucles de espera activa y la CPU ocupada permanentemente, que nos sirvió como línea base de consumo para ver qué ocurre cuando no se aprovechan los periféricos ni los modos de bajo consumo del micro. A partir de ahí introdujimos timers hardware, interrupciones y primitivas como “drv_consumo_esperar()” y” drv_consumo_dormir()”, pasando de corrientes medias de varios miliamperios a escenarios donde el núcleo duerme casi todo el tiempo y sólo se despierta por un evento concreto (temporizador o botón), reduciendo el consumo en casi un orden de magnitud. En la práctica de bajo consumo (P4) consolidamos estos mecanismos, diferenciando claramente entre un “sleep ligero” para espera de eventos frecuentes y un modo de suspensión profunda (System OFF) pensado para largos periodos de inactividad, y midiendo con el Power Profiler el impacto real de cada decisión de diseño. Finalmente, en la práctica 5 integramos todo en una aplicación real: el juego *Beat Hero*, construido como una máquina de estados que coordina LEDs, botones con antirrebotes, gestor de eventos, servicio de alarmas y watchdog, y que además entra automáticamente en suspensión profunda cuando termina la partida o no hay interacción del usuario.

Análisis de consumo energético:

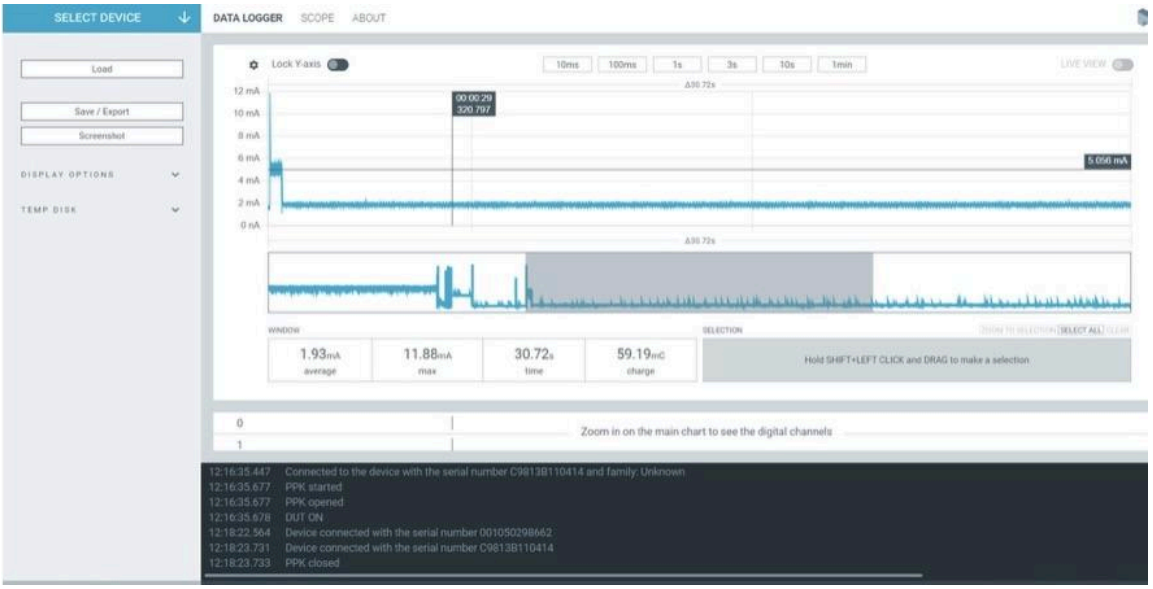
A nivel global, las medidas del Power Profiler muestran muy bien la historia del proyecto: partimos de un firmware que mantiene la CPU y los periféricos activos de forma casi constante y vamos introduciendo, práctica a práctica, mecanismos que permiten que el micro pase la mayor parte del tiempo dormido. En todas las capturas se distinguen dos métricas clave: la corriente media (lo que realmente determina la autonomía de la batería) y la corriente pico (los instantes de máxima actividad, típicamente al encender LEDs, arrancar el debug o hacer operaciones puntuales).

En la Práctica 3, el patrón típico es el de un sistema casi siempre despierto. En el primer escenario (blink con timers / idle “razonable”) la corriente media ronda el orden del 0,6–0,7

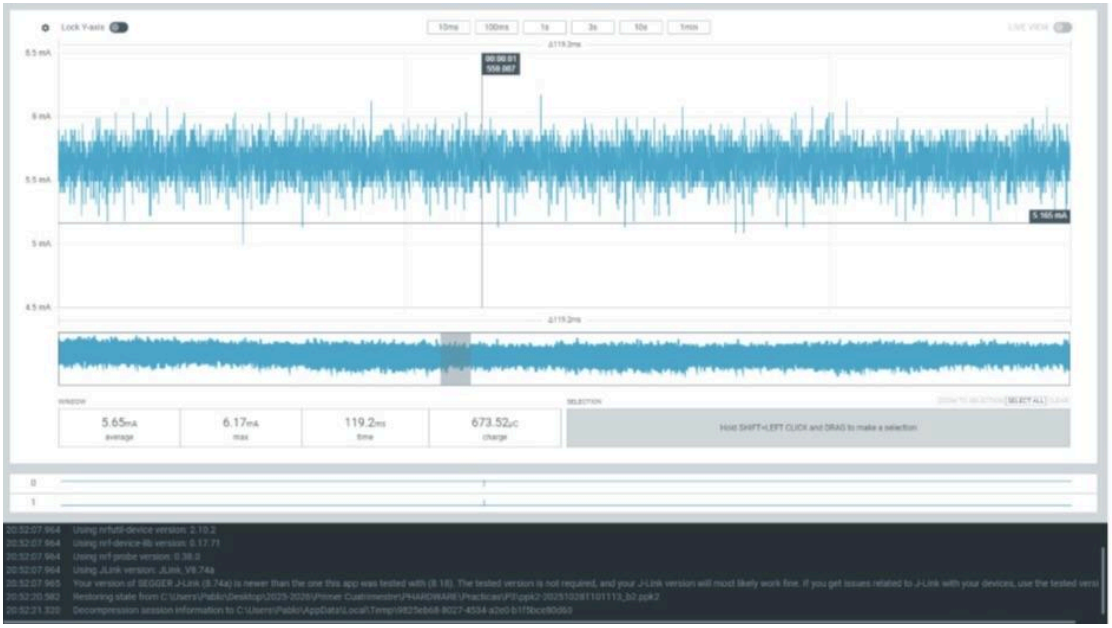
mA, con picos alrededor de 1–1,5 mA, lo que refleja una CPU que sigue trabajando con temporizadores y GPIO activos, pero sin grandes ráfagas de actividad.



En el segundo escenario (blink ejecutándose de forma más agresiva) la media sube cerca de los 2 mA y aparecen picos de 6–12 mA, claramente asociados al parpadeo de LEDs y al código que está continuamente actuando sobre ellos.

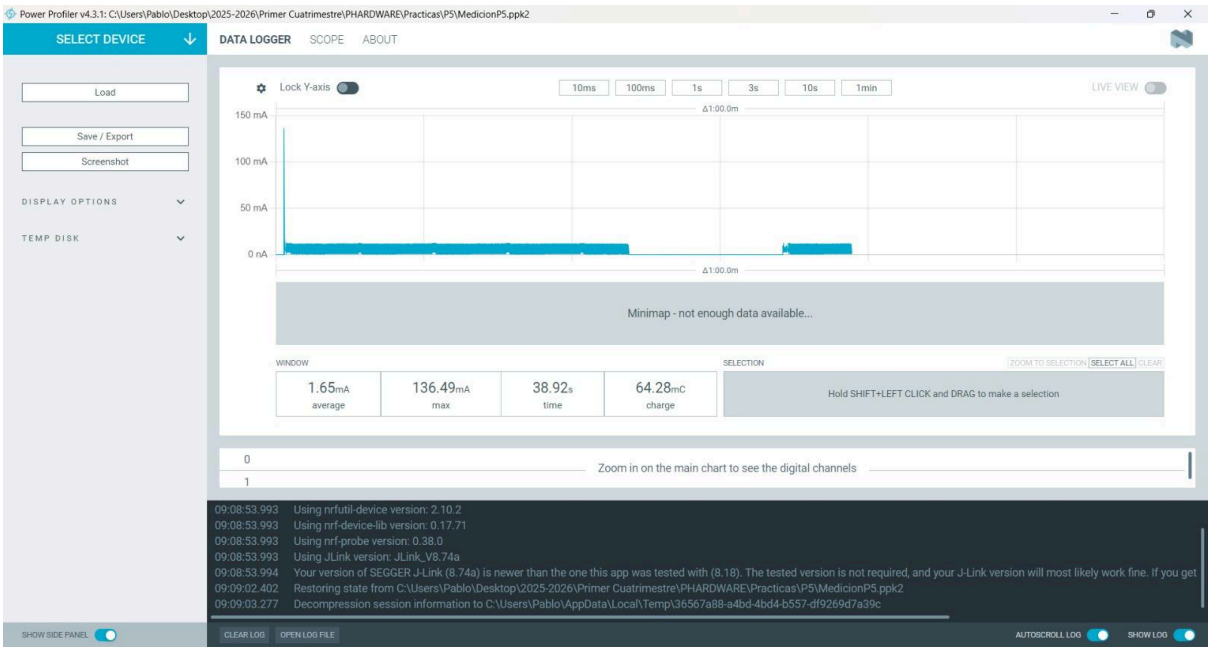


La tercera captura de P3, con consumos medios en torno a 5–6 mA, en realidad está dominada por el J-Link y el proceso de depuración: ruido, picos al flashear y al arrancar el micro, que sirven para entender que medir con el programador conectado no representa el consumo real en régimen.



En la Práctica 4 se ve el salto cualitativo: pasamos de “hacer cosas todo el rato” a esperar eventos durmiendo. Cuando el firmware entra en modo espera con temporizador (RTC/SysTick + drv_consumo_esperar()), las trazas muestran corrientes medias en el entorno de 0,5 mA, con pequeños picos cuando salta el timer o se atiende un evento. Y cuando se prueba el modo de suspensión profunda (drv_consumo_dormir()), el perfil se aplana prácticamente hasta el fondo de la escala: sólo se ven pequeños dientes cuando el sistema se despierta para medir, estando el resto del tiempo en unos pocos µA. Esta práctica deja clara la diferencia entre un sleep ligero para esperas cortas y el System OFF pensado para inactividad larga. Este comportamiento, queda perfectamente representado en el algoritmo “blink_v3_bis”, donde el bucle principal no está haciendo una espera activa “while(1){}”, sino que realiza “sleeps” ligeros mientras espera a los timers si no tiene tareas que realizar. Cada vez que se realiza una interrupción, el micro se despierta, ejecuta la ISR, así como el estado del blink, y vuelve a dormir. De no existir actividad, realiza un “POWER OFF” o “sleep” profundo, lo que permite ahorrar mucha energía, pero solamente puede salirse mediante interrupciones externas.

En la Práctica 5, con el juego Beat Hero completo, las capturas son más ricas porque mezclan varias fases de la aplicación. En la traza global se ve primero un pico muy alto (del orden de decenas o incluso >100 mA) asociado al encendido, inicialización de la placa y arranque del entorno de depuración; a continuación, la corriente se estabiliza en torno a 1,5–2 mA de media, que es el coste de tener la FSM del juego, los timers, el gestor de eventos y el watchdog funcionando mientras se muestran secuencias de LEDs y se leen botones.



Cuando se hace zoom sobre la parte central (fase de juego puro: estados SHOW_SEQUENCE y WAIT_FOR_INPUT), la selección muestra medias en torno a 0,9–1,0 mA, con picos regulares de 10–12 mA: son exactamente las ráfagas de actividad cuando se encienden LEDs, se procesa el antirrebotes y se actualiza la lógica del compás. Finalmente, en el tramo donde se ejecuta la secuencia de fin de partida y se llama a “drv_consumo_dormir()”, la traza pasa de un patrón de picos moderados (animación final) a un nivel casi plano en la base, indicando que el micro ha entrado en suspensión profunda y sólo se volverá a despertar por una interrupción de los botones 3/4. El watchdog, que está activo durante todo el juego, no introduce escalones visibles en las capturas: su impacto en el consumo medio es despreciable frente al coste de LEDs y lógica del juego, pero aporta robustez frente a bloqueos.

En resumen, las curvas del Power Profiler demuestran de forma cuantitativa la evolución del diseño: del “miliamperio constante” de las primeras prácticas, a un Beat Hero que consume alrededor de 1 mA mientras el usuario juega y que cae a unos pocos µA cuando termina la partida, gracias a los modos de bajo consumo y a la arquitectura basada en eventos.

Comparativa de eficiencia energética:

El salto en eficiencia del algoritmo “Blink_v3_bis”, se comprende mejor al compararlo con las implementaciones previas:

- **Blink_v2** (Espera Activa): Mantiene la CPU activa al 100% en un bucle (busy-waiting). Esto resulta en un consumo medio elevado y constante de 5.64 mA. Es el método menos eficiente energéticamente.

- **Blink_v3** (Bajo Consumo - WFI): Utiliza interrupciones y el modo "Sleep" (Wait For Interruption “WFI”). La CPU permanece dormida la mayor parte del tiempo, despertándose sólo brevemente para conmutar el LED. Esto, permite reducir drásticamente el consumo medio a 617.46 µA.

- **Blink_v3_bis** (Sueño Profundo - System OFF): Lleva la optimización un paso más allá. Mientras que el “Blink_v3” ahorra energía durante la operación, “Blink_v3_bis” optimiza la energía en caso de inactividad total por parte del usuario (esperando al usuario), lo que logra un consumo de 0 µA.

Desde un punto de vista cuantitativo, la evolución entre las distintas versiones del algoritmo de parpadeo permite caracterizar muy bien el impacto de las decisiones arquitectónicas sobre el consumo energético. En **Blink_v2** (espera activa), la CPU permanece ocupada permanentemente en bucles de espera (busy–waiting), lo que se traduce en un consumo medio del orden de 5,64 mA. Este valor constituye una referencia clara de “peor caso” en términos de eficiencia: el microcontrolador está continuamente despierto, incluso cuando la aplicación no está realizando trabajo útil. Al pasar a **Blink_v3** (bajo consumo con WFI), el modelo de ejecución cambia a un esquema dirigido por interrupciones, en el que el procesador entra en modo *sleep* y sólo despierta brevemente para atender el temporizador y conmutar el LED. El consumo medio desciende así a aproximadamente 0,62 mA, lo que supone una reducción cercana al **89 %** respecto a la versión con espera activa; dicho de otra forma, se obtiene el mismo comportamiento funcional con un coste energético casi nueve veces menor. Finalmente, en **Blink_v3_bis** (System OFF) se explota el modo de apagado profundo del microcontrolador: en ausencia de eventos, el sistema entra en un estado de consumo de microamperios (prácticamente 0 µA a la escala de medida), lo que implica una reducción superior al 99 % frente a la implementación inicial durante los periodos de inactividad.

Si se compara este marco de referencia con el comportamiento del juego **Beat Hero** en la práctica final, se observa que las mismas ideas de diseño se reutilizan para contener el consumo en un escenario funcionalmente mucho más complejo. Durante la ejecución normal del juego (estados SHOW_SEQUENCE y WAIT_FOR_INPUT, con LEDs, lectura de

botones, gestor de eventos y watchdog activos), las mediciones se sitúan en torno a 0,9–1,6 mA de corriente media, con picos asociados a la conmutación de LEDs y a la actividad de CPU. Aunque el consumo es lógicamente superior al de un simple “blink”, porque el sistema realiza más cómputo y gestiona más periféricos, sigue estando en el mismo orden de magnitud que Blink_v3, lo que indica que la arquitectura basada en interrupciones y estados permite mantener el coste energético bajo control. Por otro lado, al finalizar la partida, el firmware ejecuta una breve animación (del orden de 0,5 mA de media) y a continuación invoca “drv_consumo_dormir()”, entrando en un modo de suspensión profunda análogo al de Blink_v3_bis, donde el consumo vuelve a la zona de los µA y el sistema sólo puede reactivarse mediante los botones 3/4. En conjunto, la comparación cuantitativa muestra que el salto desde algoritmos de prueba hasta la aplicación final no rompe la eficiencia conseguida: gracias a la combinación de *sleep* ligero durante la actividad periódica y *System OFF* en inactividad prolongada, el juego hereda los beneficios energéticos demostrados previamente en los experimentos de las prácticas 3 y 4.

Resultados comparativos:

Tabla comparativa de rendimientos y consumo

Algoritmo	Método de espera / Estado de la CPU	Consumo medio aproximado	Carga típica por periodo	Mejora frente al blink_v2	Comentario principal
Blink_v2 (Espera activa)	Bucle “busy wait” con CPU siempre activa	5.64mA	672.33 µA	Referencia de peor caso	La CPU ejecuta instrucciones continuamente mientras espera. No explota ningún modo de bajo consumo. Sirve como línea base de lo que no ha de hacerse en un sistema eficiente.
Blink_v3 (bajo consumo, WFI)	Modo “Sleep” (Wait for Interrupt). Mayoritariamente dormida (mientras no haya nada que ejecutar)	617.46 µA	30.17 µA	Aproximadamente un 89% de menor consumo medio	El temporizador genera interrupciones periódicas, pero, entre interrupciones, la CPU duerme (sleep). Reduce en casi un orden de magnitud el consumo respecto a la espera activa, manteniendo el comportamiento funcional.
Blink_v3_bis (System Off)	Modo "Sleep profundo"; CPU apagada en espera larga	Prácticamente 0 µA en espera; 0.85 mA en ventana de actividad	8,52nC (en actividad de 10 s)	Aproximadamente un 85% de menor consumo medio	Optimiza el escenario de inactividad total: cuando el usuario no interactúa con el sistema, este entra “System OFF” o apagado profundo, donde consume mínima energía. Solo puede ser despertado mediante una interrupción externa.
Práctica 3 Reposo con temporizadores	IDLE con temporizador activo (sleep en la CPU)	0.63 mA en fase de ejecución	-	Aproximadamente un 88.9% de menor consumo medio	Al introducir el modo de espera “sleep” y apagar periféricos no necesarios, el consumo medio baja al rango de cientos de microamperios, alineándose con los resultados del “blink_v3”
Práctica 5 (Beat Hero - juego en ejecución)	FSM del juego funcionando, watchdog, temporizadores, y demás periféricos (botones activos). CPU con IDLE.	Entre 0.9 y 2.1 mA según la fase (picos de 10-12mA)	Actividad intermitente	Aproximadamente un 84% de menor consumo medio	Durante el juego, coexisten los LEDs, timers, lectura de botones, gestor de eventos, programación de alarmas, watchdog y generación de patrones de manera simultánea. Gracias a la arquitectura modular, las esperas de bajo consumo y los apagados profundos por inactividad, el consumo medio es capaz de mantenerse en torno al miliamperio a pesar de esta gran carga de trabajo.
Práctica 5 (Beat Hero - reposo/fin de partida)	Animación final y “System OFF”	0.5mA durante la animación, y unos pocos µA en suspensión	Carga despreciable mientras no exista interacción	Energéticamente comparable con el “Blink_v3_bis” . Consumo despreciable	Tras la secuencia de fin, el firmware apaga los LED, así como el resto de periféricos, y entra en una suspensión profunda sin prácticamente consumo. El sistema solo despierta mediante las interrupciones externas de los botones 3/4.

Conclusión General

A la vista de los resultados comparativos puede extraerse una conclusión clara: el consumo energético de un sistema embebido no depende sólo del hardware, sino sobre todo de cómo se programa la espera y de cómo se organiza la arquitectura del firmware.

En primer lugar, Blink_v2 (espera activa) representa el caso base y también el peor escenario. Su virtud principal es la simplicidad: es trivial de entender, muy útil para depuración inicial y para verificar rápidamente el funcionamiento de reloj, GPIO y herramienta de medida. Sin embargo, su defecto es estructural: la CPU permanece ejecutando instrucciones de forma continua mientras espera, lo que se traduce en un consumo medio del orden de 5,64 mA y en una carga por periodo muy elevada. Desde el punto de vista energético, este enfoque sólo es aceptable como referencia didáctica o para pruebas muy puntuales.

La transición a Blink_v3 (bajo consumo con WFI) introduce por primera vez un uso sistemático de los modos *sleep*. Aquí la CPU deja de “quemar ciclos” y pasa la mayor parte del tiempo dormida, despertándose únicamente cuando llega una interrupción del temporizador. El resultado cuantitativo es una reducción aproximada del 89 % del consumo medio respecto a Blink_v2, sin modificar el comportamiento visible (el LED sigue parpadeando igual). La principal virtud de esta aproximación es que ofrece una mejora enorme con un coste de complejidad relativamente bajo: basta con reestructurar la espera en torno a interrupciones. Como contrapartida, sigue siendo un esquema “siempre en marcha”: el sistema nunca llega a apagarse por completo; es adecuado para tareas periódicas continuas, pero no maximiza la autonomía en escenarios de inactividad prolongada.

Este último aspecto se aborda con Blink_v3_bis (System OFF), que explota el modo de apagado profundo de la nRF. Mientras no hay interacción, el microcontrolador entra en *System OFF* con consumo prácticamente nulo, y sólo un evento externo (botón) lo despierta. En los intervalos de actividad medidos se observan corrientes en torno a 0,85 mA, lo que supone alrededor de un 85 % de reducción frente a Blink_v2 durante la ventana activa, y consumo prácticamente cero el resto del tiempo. La utilidad de este enfoque es evidente en sistemas que pasan la mayor parte del tiempo esperando al usuario (sensores remotos, mandos a distancia, etc.). El precio que se paga es una mayor complejidad de arranque y de gestión de contexto: hay que reconfigurar módulos tras el despertar y asumir cierta latencia hasta que el sistema queda operativo.

Las mediciones de la Práctica 3 en reposo con temporizadores confirman que estas ideas son trasladables a otros entornos y no se limitan al ejemplo académico del “blink”. Al introducir modos de espera tipo *sleep* e inhabilitar periféricos no esenciales, el consumo medio se sitúa en torno a 0,63 mA, lo que supone aproximadamente un 88,9 % menos de consumo que el caso base de 5,64 mA en la misma arquitectura. Esta versión funciona como puente entre los ejercicios de temporización y el proyecto final: demuestra que, aplicando las mismas técnicas (espera por eventos, apagado selectivo de hardware), es posible reducir drásticamente la energía incluso en aplicaciones de propósito general.

El caso más interesante es el del juego Beat Hero de la Práctica 5, porque ya no se trata de un ejemplo sintético, sino de una aplicación completa con múltiples módulos: FSM de juego, drivers de LEDs y botones, gestor de eventos, temporizadores, watchdog, generación pseudoaleatoria de patrones, filtrado de rebotes, etc. A pesar de esta complejidad, las medidas muestran que, durante la ejecución del juego, el consumo medio se mantiene entre 0,9 y 2,1 mA, es decir, entre un 63 % y un 84 % menos de consumo que la referencia Blink_v2, con picos de 10–12 mA asociados a la conmutación de LEDs y a ráfagas de actividad. La virtud principal aquí es arquitectónica: gracias a la modularidad y al uso sistemático de esperas en bajo consumo, el juego consigue un perfil energético razonable sin sacrificar respuesta ni jugabilidad. El defecto, inevitable, es que la complejidad del código y la interacción constante con el usuario impiden alcanzar los niveles de ahorro extremos de Blink_v3_bis; sin embargo, para una aplicación interactiva, el compromiso alcanzado es muy favorable.

Por último, la fase de reposo y fin de partida de Beat Hero integra las lecciones de las prácticas anteriores en un contexto real: tras la animación final, el firmware apaga los LEDs, detiene la actividad innecesaria y entra en un modo de suspensión profunda equivalente al *System OFF* de Blink_v3_bis. En esta fase el consumo se reduce a unos pocos microamperios, y el sistema sólo se reanuda cuando el usuario pulsa los botones 3 ó 4. De este modo, entre partidas, el juego presenta un comportamiento energético equiparable al del mejor algoritmo de bajo consumo, pero con la ventaja de ofrecer una experiencia de usuario completa cuando está activo