Lectura correcta y eficiente de ENCODERS de cuadratura en ESP32: análisis comparativo entre el periférico pcnt e interrupciones externas

[1]Erik Reyes Reyes *[1]Areli Rojo Hernández* *[1]Alejandro Lara Caballero*
[1]Alicia Montserrat Alvarado González *[1]Angel Adrian Orta Quintana* *[2]Rogelio Ernesto Garcia Chavez*

[1]Universidad Autónoma Metropolitana, Unidad Cuajimalpa
Departamento de Matemáticas Aplicadas y Sistemas
[2]Colegio de Postgraduados, Campus Montecillo
PSEI-Cómputo Aplicado

ereyes@cua.uam.mx arojoh@correo.cua.uam alarac@cua.uam.mx
aalvarado@cua.uam.mx orta@cua.uam.mx garcia.rogelio@colpos.mx

Referencia de este artículo [1].

Resumen

Este artículo explora la metodología adecuada para leer encoders de cuadratura en el SoC ESP32, contrastando el uso de Interrupciones Externas (ISR) frente al periférico dedicado PCNT. Se de- muestra que para realizar una lectura correcta, es imperativo utilizar el periférico PCNT. Los resul- tados experimentales evidencian que las ISR introducen latencia no determinista y pérdida de pasos a frecuencias medias (>20 kHz), mientras que el PCNT garantiza la integridad de la señal y libera al CPU para tareas críticas.

1. Introducción

La lectura correcta de un encoder de cuadratura no se reduce a la simple acción de contar pulsos; implica asegurar que cada transición física del dispositivo se registre de manera confiable en el software, evitando retardos que puedan comprometer la ejecución de otras tareas críticas y garantizando un uso eficiente de los recursos del sistema. En aplicacio- nes de control y comunicación, donde la simultaneidad de procesos es indispensable, esta precisión resulta aún más relevante.

En el SoC ESP32, se presenta una dicotomía técnica que condiciona la estrategia de imple- mentación:

- Por un lado, el uso del método genérico de interrupciones (GPIO ISR), cuya simpli- cidad lo hace atractivo para desarrolladores principiantes, pero que acarrea un costo significativo en rendimiento. Cada interrupción genera una carga adicional sobre la CPU, lo que puede derivar en latencias acumuladas y pérdida de eventos cuando el sistema se encuentra bajo alta demanda.

- Por otro lado, el empleo del periférico dedicado Pulse Counter (PCNT), diseñado específicamente para la captura y conteo de pulsos. Aunque su configuración inicial es más compleja, este periférico ofrece ventajas sustanciales: descarga al procesador de la tarea de conteo, asegura la integridad de los datos y permite que el sistema ejecute en paralelo otras funciones críticas como algoritmos de control, comunicación inalámbrica o procesamiento de señales.

El hecho de que el PCNT esté integrado directamente en el SoC del ESP32 representa un beneficio adicional: elimina la necesidad de recurrir a implementaciones externas de hard- ware, como los contadores dedicados reportados en [1]. Esto no solo simplifica el diseño electrónico, sino que también reduce costos, consumo energético y posibles puntos de falla.

En este sentido, el presente artículo busca demostrar, con argumentos técnicos y ejemplos prácticos, por qué el PCNT constituye la vía más robusta y eficiente para lograr una lectura correcta de encoders en aplicaciones que requieren conteos precisos de pulsos, especial- mente en escenarios donde el microcontrolador debe atender simultáneamente múltiples tareas o deba estar en comunicación o interacción con otros periféricos.

2. Encoder de Cuadratura

El encoder incremental de cuadratura es un dispositivo electromecánico diseñado para convertir el desplazamiento angular de un eje en señales digitales discretas. Su principio de funcionamiento se basa en un disco con ranuras y dos sensores ópticos o magnéticos dispuestos de tal manera que generan dos señales de salida, denominadas Canal A y Canal
B. La característica de este tipo de encoder es que las señales de ambos canales presentan un desfase eléctrico de 90° ($\pi/2$ radianes) entre sí. Este desfase permite distinguir no solo la magnitud del movimiento (velocidad angular), sino también el sentido de giro. La Figura 1 muestra el tren de pulsos ideal generado en un ciclo completo.

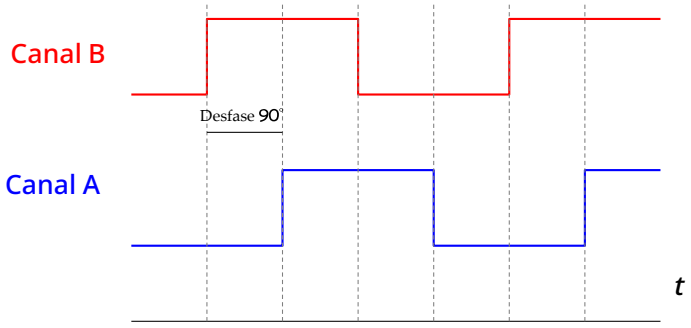

Figura 1: Señales de un encoder de cuadratura.

La interacción de las señales A y B genera una secuencia de cuatro estados lógicos distin- tos (00, 10, 11, 01). La dirección de transición entre estos estados determina el sentido de rotación. En la Figura 2 se detallan estos estados lógicos y su evolución temporal durante un ciclo completo.

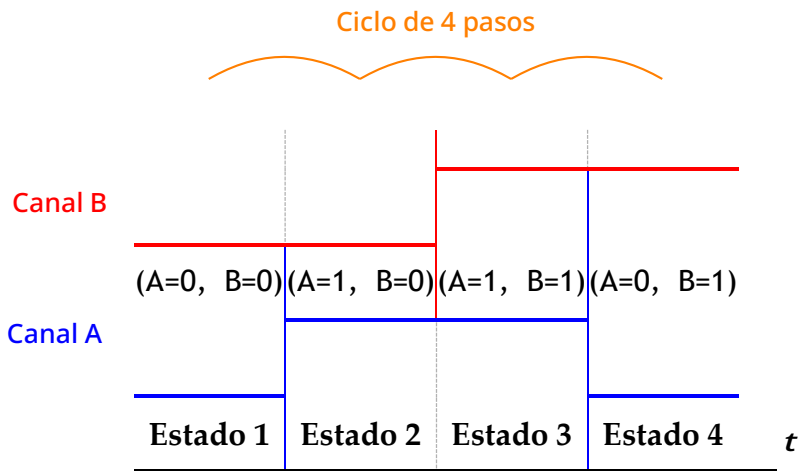

Figura 2: Diagrama de tiempos mostrando los 4 estados lógicos únicos (00, 10, 11, 01) en un ciclo de cuadratura.

Tal como se observa en la Figura 2, el desfase entre canales impone un orden estricto de lectura. Dependiendo de si el Canal A adelanta al Canal B o viceversa, se generan los si- guientes patrones de estados que definen la dirección de rotación de la siguiente manera:

- Sentido horario: 00 → 10 → 11 → 01 → 00
- Sentido antihorario: 00 → 01 → 11 → 10 → 00

2.1 Consideraciones de lectura

La resolución efectiva de un encoder de cuadratura está intrínsecamente ligada a la estra- tegia de detección de flancos empleada, dado que es posible monitorear únicamente los flancos de subida, los de bajada, o ambos. Para los fines de esta investigación, se imple- mentó un modo de decodificación que evalúa ambos flancos (subida y bajada) en ambos canales (A y B). Este enfoque permite identificar las cuatro transiciones lógicas posibles dentro de un mismo periodo eléctrico, logrando así cuadruplicar la resolución física no- minal del encoder.

En consecuencia, la frecuencia de eventos de conteo (f_{ev}) que el microcontrolador debe procesar aumenta proporcionalmente y se define mediante la ecuación:

$$f_{ev} = \frac{\text{RPM} \times \text{PPR} \times 4}{60} \tag{1}$$

donde RPM representa la velocidad angular del eje y PPR los pulsos físicos por revolución del disco. Es crucial destacar que, si f_{ev} excede la capacidad de respuesta del CPU (limitada por la latencia de las interrupciones), se producirán lecturas erróneas o pérdida de pasos.

3. Enfoque basado en Interrupciones (GPIO ISR)

Una estrategia convencional para abordar el conteo de pulsos del encoder en el ESP32 es la utilización de Interrupciones Generales (GPIO ISR). Si bien es un método directo, conlleva una penalización temporal intrínseca y acumulativa conocida como *overhead*.

En el contexto de la arquitectura Xtensa LX6 del ESP32, cada vez que se detecta un cambio de estado lógico (flanco) en los canales del encoder, el controlador de interrupciones fuerza al CPU a realizar una transición de contexto estricta. Como se detalla a continuación, este proceso consume ciclos de reloj críticos antes y después de ejecutar la lógica de conteo real:

1. Ruptura del Pipeline: El procesador detiene la instrucción actual, vacía la tubería de ejecución y salta al vector de interrupción.

2. Preservación de Contexto: Se deben salvaguardar los registros de estado (PS), el contador de programa (PC) y los registros generales activos, ya sea en la pila (Stack) o mediante la conmutación de la ventana de registros (Windowed Register File).

3. Ejecución de la Rutina (ISR): Se procesa el código de usuario: lectura de los registros GPIO, evaluación de la lógica de cuadratura y actualización de las variables volátiles de conteo.

4. Restauración de Contexto: Se recuperan los registros previos y se ejecuta la instruc- ción de retorno (RFI), devolviendo el control al flujo principal.

La Figura 3 ilustra la densidad de eventos generada por esta configuración. Al habilitar la detección tanto en los flancos de subida como en los de bajada, la frecuencia de solicitu- des de interrupción se cuadruplica respecto a la frecuencia fundamental de la señal. En consecuencia, durante operaciones a alta velocidad, la acumulación del tiempo dedicado al *overhead* (pasos 1, 2 y 4: gestión de pipeline y contexto) puede saturar los ciclos disponi- bles del CPU, impidiendo que el microcontrolador atienda otras tareas críticas en tiempo real.

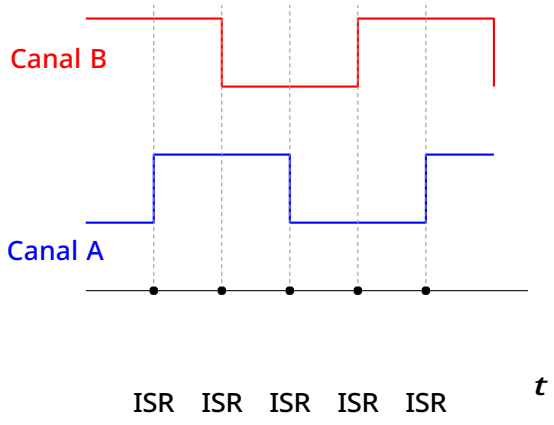

Figura 3: Cronograma de los eventos de interrupción

4. Enfoque de conteo por hardware dedicado (PCNT)

Como alternativa robusta a la gestión por software (ISR), el ESP32 integra el periférico de Conteo de Pulsos (PCNT). Este módulo opera como una unidad de hardware dedicada, diseñada para decodificar señales de pulsos de manera asíncrona y paralela a la Unidad Central de Procesamiento (CPU).

La arquitectura del PCNT (detallada en la Fig. 4) introduce un cambio de paradigma en el flujo de datos, caracterizado por tres etapas críticas:

1. Filtrado de Glitches por Hardware: Antes de procesar la señal lógica, las entradas atraviesan un filtro digital configurable. Este bloque compara la duración del pulso entrante con un umbral de ciclos de reloj del bus APB. Cualquier perturbación o ruido de alta frecuencia (glitch) inferior a este umbral es descartado automáticamente. Esto elimina la necesidad de algoritmos de *debouncing* por software, liberando recursos computacionales.

2. Decodificación y Acumulación en Tiempo Real: Una Máquina de Estados Finitos (FSM) interna interpreta los cambios de fase de cuadratura (modos x1, x2 o x4) y actualiza un acumulador de 16 bits con signo. Este proceso es atómico y ocurre inde- pendientemente de la carga de trabajo del procesador.

3. Acceso Asíncrono (Polling): A diferencia de las ISR que interrumpen el flujo de ejecución, el valor del contador reside en un registro de memoria mapeada. La CPU consulta este registro únicamente cuando el lazo de control lo requiere (e.g., cada T_s = 10 ms), desacoplando la adquisición de datos de la velocidad del actuador.

Gracias a este desacoplamiento, la complejidad computacional del conteo se reduce a $O(1)$, garantizando una carga de CPU constante e inmune a la saturación, incluso a altas veloci- dades angulares.

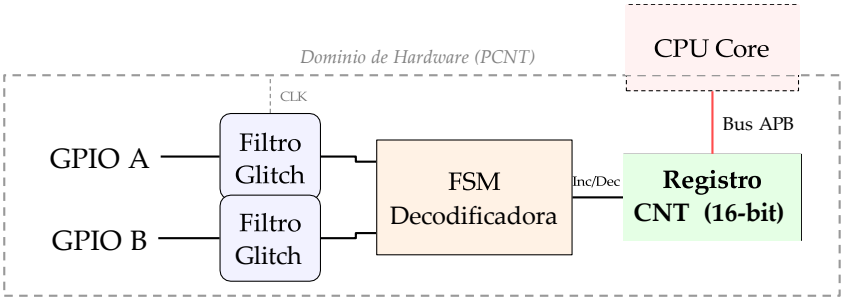

Figura 4: Arquitectura funcional del periférico PCNT. El flujo de señal (izquierda a dere- cha) es procesado íntegramente por hardware, permitiendo que la CPU lea el acumulador bajo demanda sin interrupciones.

5. Análisis Comparativo y Validación Experimental

Para validar y cuantificar las diferencias entre el enfoque basado en interrupciones (ISR) y el conteo por hardware (PCNT), se configuró un banco de pruebas utilizando un motor de corriente continua con caja reductora modelo GM-25-370 (12V, 140 RPM nominales).

Este actuador incorpora un encoder magnético de efecto Hall de dos canales acoplado al eje trasero del motor (antes de la caja de engranajes). Las especificaciones relevantes para el experimento se detallan a continuación:

- Velocidad nominal: 140 RPM.
- Relación de reducción: 1:45 (aproximada).
- Resolución del encoder: 11 pulsos por revolución (PPR) del eje motor.
- Resolución efectiva: $11 \times 45 \approx 495$ PPR en el eje de salida.

Dado que el encoder monitorea el eje rápido del motor, la frecuencia de eventos que el microcontrolador debe procesar es significativamente alta. A la velocidad nominal de 140 RPM en la salida, el motor interno gira a aprox. 6300 RPM. Aplicando la Ecuación 1, la tasa de interrupciones generada es:

$$f_{ev} = \frac{6300 \text{ RPM} \times 11 \text{ PPR} \times 4}{60} \approx 4,620 \text{ Hz} \quad (2)$$

Esto implica que, bajo el método ISR, el ESP32 debe detener su flujo principal más de 4,600 veces por segundo.

6. Resultados Experimentales y Validación

Para cuantificar el impacto del método de lectura en el rendimiento global del sistema, se diseñó un protocolo experimental dividido en dos fases complementarias. La primera fase evaluó el comportamiento con el hardware mecánico real (Motor GM-25-370), mien- tras que la segunda fase consistió en una prueba de estrés sintética mediante inyección de señales para determinar los límites de ruptura del microcontrolador. La variable crítica evaluada en ambas fases fue la carga de la CPU (*CPU Load*), monitorizada mediante las herramientas de trazado de FreeRTOS.

6.1 Fase 1: Rendimiento en Condiciones Nominales

Se sometió al sistema a pruebas de velocidad escalonada para comparar el método tradicio- nal basado en Interrupciones (ISR en ambos flancos) frente al uso del periférico PCNT. Los

resultados, detallados en la Tabla 1, evidencian el consumo de recursos ante frecuencias de eventos reales generadas por el encoder.

Tabla 1: Comparativa de consumo de recursos y precisión de conteo a distintas velocidades angulares.

Condición Operativa	Velocidad (RPM)	Frecuencia (f_{ev} ≈)	Uso de CPU (%)		Integridad de Datos
			ISR	PCNT	
Reposo	0	0 Hz	0.01 %	0.01 %	100 %
Baja Carga	70	2.31 kHz	4.25 %	0.02 %	100 %
Nominal	140	4.62 kHz	8.80 %	0.02 %	> 99.9 %
Sobrecarga*	>200	> 6.60 kHz	>12.50 %	0.02 %	Pérdida detectada

*Extrapolación basada en picos transitorios de corriente.

6.2 Fase 2: Validación de Límites por Inyección de Señales

Dado que la mecánica del motor limita la frecuencia máxima a ≈ 5 kHz, se implementó una prueba sintética para validar la robustez del sistema ante escenarios de alta velocidad (útil para escalabilidad a motores de mayor RPM). Utilizando un generador de funciones, se inyectaron señales cuadradas en cuadratura directamente a los pines del ESP32, realizando un barrido desde 10 kHz hasta 500 kHz.

- Punto de Fallo ISR: A partir de los 85 kHz, el método por interrupciones saturó el núcleo de la CPU (100 % de carga), provocando el disparo del *Watchdog Timer* y el reinicio del microcontrolador.
- Estabilidad PCNT: El módulo PCNT mantuvo una precisión de conteo absoluta y una carga de CPU constante del 0.02 % incluso a frecuencias de 400 kHz. La limitación en este punto fue el filtro de *glitch* hardware, no el procesamiento.

6.3 Discusión del Análisis Computacional

El análisis de los datos revela una divergencia fundamental en la escalabilidad de ambos métodos:

1. Correlación Lineal (ISR): El consumo de CPU utilizando interrupciones exhibe un comportamiento lineal ($R^2 \approx 0{,}99$). Al duplicar la frecuencia de entrada, la carga del procesador escala proporcionalmente debido al *overhead* del cambio de contexto (*context switching*) en cada interrupción.
2. Comportamiento Determinista (PCNT): Por el contrario, el periférico PCNT des- acopla la adquisición de datos del ciclo de instrucción de la CPU. El uso de CPU se mantiene residual (0,02 %) independientemente de la velocidad, correspondiente únicamente a una operación de lectura de memoria ($O(1)$) en el bucle de control.

7. Conclusión

La validación experimental confirma que, si bien el método basado en interrupciones es funcional para prototipado rápido a bajas revoluciones, resulta insostenible para aplicacio- nes de robótica móvil profesional. El crecimiento lineal del consumo de CPU en el método ISR compromete la capacidad del microcontrolador para ejecutar tareas concurrentes crí- ticas.

La implementación del periférico PCNT demostró ser la solución óptima, ofreciendo tres ventajas determinantes:

- Eficiencia: Libera aproximadamente un 9 % de potencia de cálculo en condiciones nominales, recursos esenciales para algoritmos de control complejos (PID, Odome- tría) o pilas de comunicación (Wi-Fi/micro-ROS).
- Robustez: Garantiza la integridad de los datos incluso ante picos de velocidad que saturarían un sistema basado en interrupciones.
- Escalabilidad: Permite la migración futura a motores de mayor resolución o RPM sin requerir cambios en la arquitectura del software ni en el microcontrolador.

8. Referencias bibliográficas

Márquez, C., Martínez, D. & Sandoval. (2025, mayo - junio). Lectura de encoders con el circuito LS7366. *Boletín UPIITA*, 20 (109). https://www.boletin.upiita.ipn.mx/index.php/ciencia/1095-cyt-numero-109/ 2420-lectura-de-encoders-con-el-circuito-ls7366

Referencia del artículo

Reyes, E., Rojo, A., Lara, A., Alvarado, A., Orta, A. & Garcia, R. *(marzo – abril, 2026).* Lectura correcta y eficiente de ENCODERS de cuadratura en ESP32: análisis comparativo entre el periférico PCNT e interrupciones externas. *Boletín UPIITA. año 20, (113) 2026.*

https://www.boletin.upiita.ipn.mx/index.php/ciencia/1105-cyt-numero-113/2482-lectura-correcta-y-eficiente-de-encoders-de-cuadratura-en-esp32-analisis-comparativo-entre-el-periferico-pcnt-e-interrupciones-externas