HAMMING

CARABALLO CARDENAS ANDRES FELIPE, UNIVERSIDAD MILITAR NUEVA GRANADA
(est.andres.caraballo@unimilitar.edu.co)

Resumen La práctica de laboratorio se centra en la aplicación de la codificación Hamming (7,4) a los datos adquiridos por el acelerómetro MPU6050, empleando como plataforma de procesamiento una Raspberry Pi Pico 2W. Cada lectura de 16 bits se fragmenta en bloques de 4 bits, los cuales se transforman en palabras de 7 bits mediante el esquema de codificación, para luego unirse en una trama final de 28 bits. Estas tramas se transmiten a través de la interfaz UART y se analizan con un osciloscopio con el fin de verificar su correcta conformación. Finalmente, se implementa la etapa de decodificación para evaluar la detección y corrección de errores de un solo bit, demostrando la importancia de la codificación de Hamming en la preservación de la integridad de la información dentro de sistemas embebidos.

I. INTRODUCCIÓN

En esta práctica se trabajó con una Raspberry Pi Pico 2W y el sensor MPU6050, encargado de entregar lecturas de aceleración. A partir de los datos obtenidos, se aplicó la codificación Hamming (7,4), dividiendo cada muestra en pequeños bloques de 4 bits que luego se transformaron en palabras de 7 bits. Estas palabras se agruparon en tramas de 28 bits y se enviaron a través de UART, para después observarlas en el osciloscopio. Con este proceso fue posible no solo visualizar la transmisión de la información, sino también comprobar cómo la decodificación permite detectar y corregir errores de un solo bit, resaltando la importancia de este tipo de técnicas para mantener la integridad de los datos en sistemas embebidos.

II. PROCEDIMIENTO

Para el desarrollo del montaje, se conectó la Raspberry Pi Pico 2W al acelerómetro MPU6050 utilizando el bus de comunicación I²C, mientras que la salida de transmisión (Tx) del puerto UART se enlazó al osciloscopio digital para poder observar las señales generadas. Una vez establecida la comunicación, se configuró el sistema para leer datos de 16 bits provenientes del acelerómetro, los cuales se fragmentaron en cuatro bloques de 4 bits cada uno.

Cada bloque fue sometido al proceso de codificación Hamming (7,4) con paridad par, lo que permitió obtener cuatro palabras de 7 bits. Estas se concatenaron en una trama de 28 bits, que posteriormente fue enviada a través de la interfaz UART de la Pico. En el osciloscopio, se analizó la estructura de la señal transmitida, verificando su correcta conformación.

Con el fin de evaluar la robustez del sistema, se implementó la etapa de decodificación para reconstruir los datos originales y, de manera intencional, se introdujeron errores de un solo bit en algunas tramas. Esto permitió comprobar en la práctica la capacidad del algoritmo Hamming para detectar y corregir fallos, asegurando la integridad de la información. Finalmente, se compararon los datos recuperados con las lecturas iniciales, validando el funcionamiento tanto de la codificación como de la transmisión en condiciones reales de prueba.

III. ANÁLISIS DE RESULTADOS

Para iniciar la práctica, se programó la Raspberry Pi Pico 2W con un código que habilitó la adquisición y visualización de los datos. En primera instancia, los valores fueron mostrados en la pantalla OLED, lo que permitió una verificación rápida del funcionamiento del sistema. Posteriormente, la información se desplegó en la consola, facilitando un análisis más detallado y la validación de los resultados obtenidos.

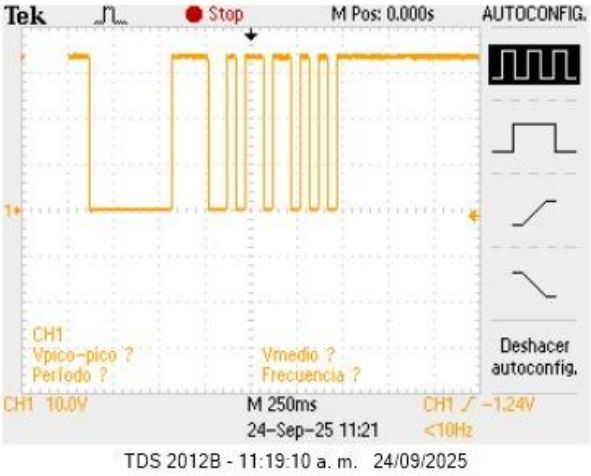

Figura 1. Señal resultante de los datos del acelerómetro

La gráfica capturada en el osciloscopio corresponde a la señal transmitida por la Raspberry Pi Pico 2W a través del puerto UART, una vez generadas las tramas de 28 bits producto de la codificación Hamming (7,4). En la señal se distinguen claramente los niveles lógicos alto y bajo que representan los "1" y "0" transmitidos. El primer pulso extendido corresponde al bit de inicio (start bit), seguido de una secuencia de transiciones rápidas que forman la palabra codificada.

Se observa que la trama presenta la estructura esperada: después del inicio, se transmiten de manera ordenada los bits que conforman cada palabra de 7 bits concatenada. Esto confirma que los nibbles de 4 bits del acelerómetro fueron correctamente transformados en palabras de 7 bits mediante la codificación Hamming.

IV. CONCLUSIONES

- La práctica mostró de manera clara cómo la codificación Hamming (7,4) ayuda a que los datos viajen de forma más segura, corrigiendo errores simples y garantizando que la información del acelerómetro llegue sin alteraciones.

- La Raspberry Pi Pico 2W demostró ser un microcontrolador práctico y flexible, capaz de manejar tanto la lectura del sensor como la codificación, transmisión y recuperación de datos en tiempo real.

- Por medio del osciloscopio fue posible ver físicamente la transmisión de las tramas por UART, comprobando que la estructura de los bits se generó de manera correcta y que la velocidad de envío se mantuvo estable.

V. REFERENCIAS

[1]. Rurik, W., & Mazumdar, A. (2016). Hamming codes as error-reducing codes. En 2016 IEEE Information Theory Workshop (ITW) (pp. 404–408). IEEE. https://doi.org/10.1109/ITW.2016.7606865

[2]. MicrocontrollersLab. (s. f.). *Interface MPU6050 with Raspberry Pi Pico using MicroPython*. Recuperado de https://microcontrollerslab.com/mpu6050-raspberry-pi-pico-micropython-tutorial/

[3]. Tim Hanewich. (2023, 3 de noviembre). How to use an MPU-6050 with a Raspberry Pi Pico using MicroPython. Medium. Recuperado de https://timhanewich.medium.com/how-to-use-an-mpu-6050-with-a-raspberry-pi-pico-using-micropython-cd768ea9268d

[4]. Interfacing MPU6050 with Raspberry Pi Pico & MicroPython." (s.f.). How2Electronics. Recuperado de https://how2electronics.com/interfacing-mpu6050-with-raspberry-pi-pico-micropython/