CODIFICACIÓN HAMMING (7,4) DE LECTURAS DEL MPU6050 CON RASPBERRY PI PICO 2W

Tomas Alejandro Ramirez Velasquez
Universidad Militar Nueva Granada
Est.tomas.ramirez@unimilitar.edu.co
https://github.com/tomasramirez20/Practica-7-Com.Digitales

A. Resumen

En este laboratorio se implementó un sistema de codificación y transmisión de datos utilizando el código Hamming (7,4) con paridad par, aplicado a lecturas del acelerómetro MPU6050 mediante una Raspberry Pi Pico 2W. El proceso consistió en leer datos de 16 bits del sensor, dividirlos en cuatro nibbles de 4 bits, codificar cada uno con Hamming, y transmitir la trama resultante de 28 bits a través del puerto UART. La trama fue visualizada tanto en la consola de Thonny como en un osciloscopio digital. Los resultados demostraron la efectividad del código para la detección y corrección de errores en la transmisión de datos.

2. Introducción

La codificación Hamming es un método ampliamente utilizado en sistemas de comunicaciones digitales para la detección y corrección de errores en la transmisión de datos [1]. En este laboratorio, se emplea la variante Hamming (7,4), que permite codificar palabras de 4 bits en palabras de 7 bits, añadiendo tres bits de paridad. El objetivo principal fue aplicar este esquema a las lecturas del acelerómetro MPU6050, transmitiendo las tramas codificadas a través de UART para su posterior visualización y análisis. Este proceso permite evaluar la robustez del código frente a posibles errores de transmisión y su implementación en sistemas embebidos.

2. Marco Teórico y procedimiento

El proceso comenzó con la lectura de datos del acelerómetro MPU6050 a través del protocolo I2C, obteniendo valores de 16 bits por eje. Cada valor se dividió en cuatro nibbles de 4 bits, los cuales fueron codificados individualmente utilizando la función `hamming74_encode`. Esta función calcula tres bits de paridad (P1, P2, P4) según las siguientes relaciones:

$$P1 = d3 \oplus d2 \oplus d0$$

$$P2 = d3 \oplus d1 \oplus d0$$

$$P4 = d2 \oplus d1 \oplus d0$$

La palabra codificada resultante tiene la estructura: [P1, P2, d3, P4, d2, d1, d0]. Las cuatro palabras codificadas se concatenaron para formar una trama de 28 bits, la cual se transmitió por el puerto UART de la Raspberry Pi Pico 2W. La decodificación se realizó mediante la función `hamming74_decode`, que calcula el síndrome para detectar y corregir errores en un solo bit.

Segun lo anterior se puede confirmar en el codigo implementado `Hamming[16,28]` para la codificacion de solo el vector x del acelerometro, el cual se puede encontrar dentro del repositorio de github [2]

3. Resultados y Análisis

A continuación, se presentan los resultados obtenidos durante la ejecución del laboratorio. Se incluyen tanto la visualización de la trama codificada en la consola de Thonny como la lectura mostrada en la pantalla OLED del MPU6050.

Imagen 1: Lectura del MPU6050 en pantalla OLED

En esta imagen se observa la lectura en bruto del acelerómetro, mostrando un valor de 16 bits correspondiente al eje X: `1111111111001010`.

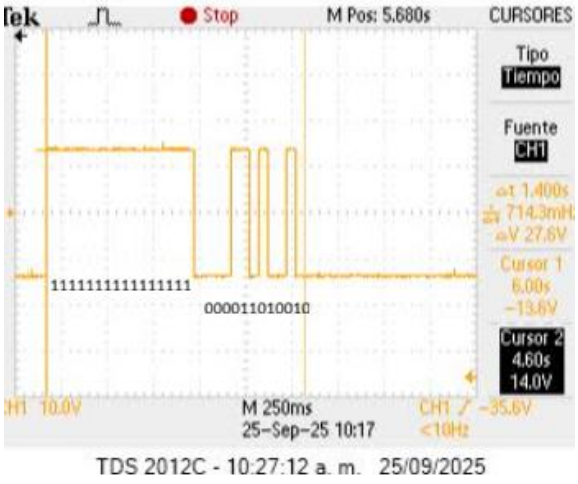
Imagen 2 y 3 : Trama codificada en consola de Thonny y osciloscopio

```
==================================
==================================
AX (16 bits): 1111111111001010
D (bits15..12) -> 1111
Codificado: 1111111
C (bits11..8) -> 1111
Codificado: 1111111
B (bits7..4) -> 1100
Codificado: 1100001
A (bits3..0) -> 1010
Codificado: 1010010
Trama final (28 bits): 1111111111111111000
011010010
```

En esta captura se muestra la trama codificada de 28 bits obtenida tras aplicar Hamming (7,4) a cada nibble del dato original. La trama final fue:

`1111111111111110000011010010`

Tambien, se hace la captura en el osciloscopio, donde el resultado fue el siguiente:



Se mide con los cursores la duracion de un bit y de toda la trama entera, esto para confirmar que sean 28 bits:

$$\#\, bits = ⍁\{t\_t\}\{t\_b\}$$

Donde t_t es el tiempo de la trama : 1,4s
Y t_b es el tiempo de cada bit: 50ms

$$\#\, bits = 28$$

Se observa que cada nibble fue codificado correctamente, y la estructura de la trama coincide con lo esperado según el proceso descrito.

La implementación del código Hamming (7,4) permitió codificar exitosamente los datos del acelerómetro, generando una trama de 28 bits que incluye redundancia para la corrección de errores. La función de decodificación fue verificada mediante pruebas con errores simulados, demostrando su capacidad para detectar y corregir un bit erróneo. La comparación con los cálculos manuales realizados previamente en el taller de Hamming confirmó la validez de los resultados obtenidos. La visualización de la trama en el osciloscopio permitió validar la integridad de la transmisión UART.

4. Conclusiones

Se logró implementar exitosamente un sistema de codificación Hamming (7,4) en una Raspberry Pi Pico 2W para transmitir datos del acelerómetro MPU6050 a través de UART. El proceso permitió comprobar la eficacia del código para la detección y corrección de errores en un entorno real. La visualización de la trama codificada en el osciloscopio y en la consola de Thonny confirmó la correcta generación y transmisión de los datos. Este laboratorio refuerza los conceptos teóricos de codificación de canal y su aplicación en sistemas embebidos.

5. Referencias

[1] R. W. Hamming, “Error detecting and error correcting codes,” *The Bell System Technical Journal*, vol. 29, no. 2, pp. 147–160, Apr. 1950.

[2] https://github.com/tomasramirez20/Practica-7-Com.Digitales