Laboratorio 2: Detección de Errores en Comunicación Serial usando ESP8266MOD y Raspberry Pi Pico

1st Santiago Hernández Ávila
Facultad de Ingeniería Electrónica
Universidad Santo Tomás
Bogotá, Colombia
santiago.hernandez@usantotomas.edu.co

2nd Diego Alejandro Rodríguez Gámez
Facultad de Ingeniería Electrónica
Universidad Santo Tomás
Bogotá, Colombia
diegoarodriguezg@usantotomas.edu.co

3rd Camila Pérez Mercado
Facultad de Ingeniería Electrónica
Universidad Santo Tomás
Bogotá, Colombia
camila.perez@usantotomas.edu.co

***Abstract*—Este informe documenta el desarrollo de la segunda práctica de laboratorio en el curso de comunicaciones, enfocada en la implementación de técnicas de detección de errores en transmisión serial. Para esta práctica se utilizó una ESP8266MOD como emisor y una Raspberry Pi Pico como receptor, empleando UART como protocolo de comunicación. Se implementaron las técnicas de *Checksum* y un protocolo simple de retransmisión basado en *ACK/NACK*. El tercer punto de la guía, correspondiente a la comparación entre LRC y VRC, no fue implementado debido a limitaciones técnicas. Se presentan los objetivos, metodología, resultados experimentales y un análisis crítico de la efectividad de cada método. Finalmente, se discute la importancia de documentar y versionar el código en repositorios colaborativos.**

***Index Terms*—ESP8266MOD, Raspberry Pi Pico, UART, Detección de Errores, Checksum, ARQ, ACK/NACK**

I. Introducción

En la transmisión de datos digitales, los errores son inevitables debido a interferencias, ruido eléctrico y problemas de sincronización. Para garantizar la confiabilidad, se emplean mecanismos de detección y corrección de errores.

El presente trabajo se centra en el análisis práctico de dos de estos mecanismos:

- Checksum: técnica simple basada en la suma de los datos transmitidos.
- ARQ (Automatic Repeat reQuest) con ACK/NACK: protocolo de retransmisión confiable.

La práctica se realizó sustituyendo la placa Arduino Uno de la guía por una ESP8266MOD, lo que permitió explorar la flexibilidad de esta plataforma en conjunto con la Raspberry Pi Pico.

II. Metodología General

A. Plataformas utilizadas

- **ESP8266MOD**: configurada como emisor de datos mediante UART.
- **Raspberry Pi Pico**: configurada como receptor, programada en MicroPython.
- **Conexión**: Comunicación directa UART (TX $\rightarrow$ RX, RX $\rightarrow$ TX, GND compartida).

B. Ambiente de desarrollo

- **ESP8266MOD**: Programada en Arduino IDE utilizando comunicación serial por software.
- **Raspberry Pi Pico**: Programada en Thonny IDE con MicroPython.
- **Repositorio**: El código completo se encuentra documentado en GitHub[1].

III. Punto 1: Configuración y Checksum Simple

A. Objetivo

Validar la transmisión UART e implementar un método de detección de errores mediante suma módulo 256 (Checksum).

B. Metodología

El emisor transmitió secuencias de datos junto con un valor de verificación calculado como la suma módulo 256. El receptor comparó la suma calculada localmente con el valor recibido. Se realizaron pruebas con datos correctos y con alteraciones intencionales para verificar la capacidad de detección.

C. Evidencia experimental

En la Fig. 1 se muestra la evidencia del funcionamiento, donde se aprecia el código en la ESP8266MOD y la salida de la Raspberry Pi Pico validando la transmisión y detectando errores cuando se introdujeron alteraciones.

D. Resultados

El sistema detectó de manera correcta las alteraciones introducidas. Cuando los datos llegaban completos, el receptor mostraba confirmación de integridad; al introducir errores, se evidenció la detección inmediata. Esto confirma la utilidad del método para errores simples, aunque no garantiza la detección de múltiples errores en bloque.

[1]El enlace al repositorio puede ser agregado posteriormente.

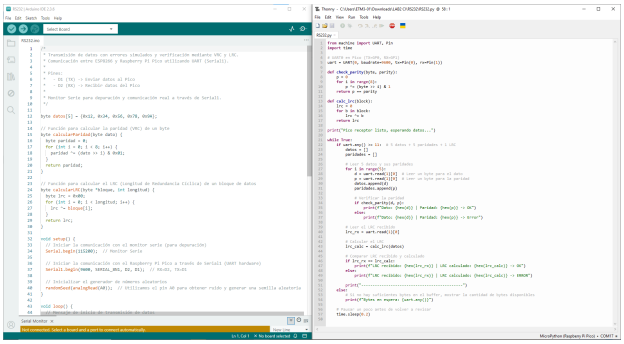

Fig. 1. Evidencia del Punto 1: transmisión con checksum entre ESP8266MOD y Raspberry Pi Pico.

IV. Punto 2: Protocolo ARQ Simple con ACK/NACK

A. Objetivo

Implementar un protocolo de retransmisión confiable basado en confirmaciones positivas (ACK) o negativas (NACK) desde el receptor hacia el emisor.

B. Metodología

El receptor validó los datos recibidos y respondió con un mensaje de confirmación o rechazo. En caso de rechazo, el emisor retransmitió el paquete hasta lograr confirmación. Para evaluar el desempeño, se simuló la introducción de errores con probabilidades del 10%, 30%, 60% y 90%.

C. Evidencia experimental

En la Fig. 2 se presenta la captura de la ejecución del protocolo ARQ, donde se observan tanto el código como la consola de la ESP8266MOD y de la Raspberry Pi Pico. Se evidencia la retransmisión de paquetes y las respuestas `ACK/NACK` según la probabilidad de error configurada.

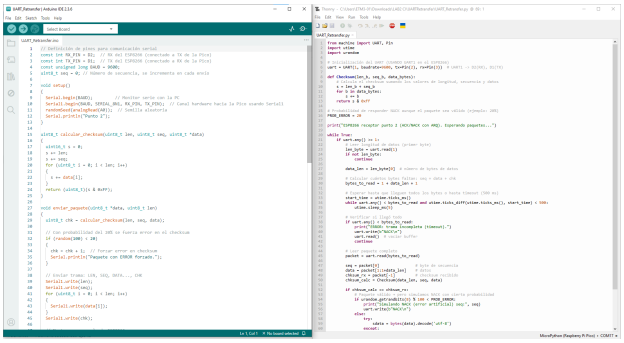

Fig. 2. Evidencia del Punto 2: protocolo ARQ (ACK/NACK) entre ESP8266MOD y Raspberry Pi Pico.

D. Resultados

Los resultados obtenidos se consolidan en la Tabla I, donde se observa cómo la probabilidad de error afecta directamente la eficiencia del protocolo.

TABLE I
Retransmisiones observadas con ARQ (ACK/NACK) para distintas probabilidades de error p

Prob. de error (p)	Retransmisión típica	Máxima observada	Comentarios
10%	1	2	Casi siempre basta un reintento; esporádicamente dos. Eficiencia alta y latencia estable.
30%	2	4	Se presentan ráfagas de NACK; algunas tramas requieren 3–4 reintentos. Eficiencia media.
60%	3–4	7	Múltiples retransmisiones frecuentes; la latencia crece y la eficiencia baja.
90%	7–9	14	Retransmisiones muy altas y *timeouts* ocasionales; canal casi inutilizable.

V. Punto 3: Implementación de LRC y VRC

Este punto no fue implementado en el presente laboratorio. La complejidad adicional en la configuración del hardware y el tiempo disponible impidieron su desarrollo. Se reconoce, sin embargo, la importancia de estas técnicas para el análisis de bloques de datos y se deja como trabajo pendiente para futuras prácticas.

VI. Discusión

Los experimentos permiten resaltar que:

- El Checksum es útil y sencillo, pero no garantiza detección completa de errores múltiples.
- El protocolo ARQ asegura la entrega de la información, aunque introduce un costo en tiempo y eficiencia.
- Sustituir el Arduino por la ESP8266MOD representó un reto adicional, pero permitió ampliar la experiencia hacia el uso de plataformas IoT.
- Documentar los resultados y centralizar el código en un repositorio es fundamental para la replicabilidad y mejora continua.

VII. Conclusiones

Se concluye que:

- El Checksum permitió validar la transmisión y detectar errores simples de forma efectiva.
- El protocolo ARQ con ACK/NACK garantiza confiabilidad, siendo adecuado para enlaces donde los errores no son excesivos.
- La práctica demuestra la importancia de conocer distintos mecanismos de detección y corrección de errores en sistemas embebidos.
- La documentación en repositorios como GitHub representa una buena práctica de ingeniería para asegurar trazabilidad y colaboración.

References

[1] W. Stallings, *Data and Computer Communications*, 10th ed. Pearson, 2014.
[2] A. Tanenbaum, *Computer Networks*, 5th ed. Pearson, 2011.
[3] Documentación oficial de Raspberry Pi Pico y MicroPython: https://www.raspberrypi.com/documentation/micropython/
[4] Espressif Systems, “ESP8266EX Datasheet”, https://www.espressif.com/en/products/socs/esp8266