Implementación y Comparación de Protocolos de Detección de Errores en Comunicaciones Serie Industriales

Luis Felipe Garzón Camacho
Facultad de Ingeniería Electrónica
Universidad Santo Tomás
Bogotá, Colombia
luisgarzonc@usantotomas.edu.co

David Stiven Quinchanegua Cely
Facultad de Ingeniería Electrónica
Universidad Santo Tomás
Bogotá, Colombia
davidquinchanegua@usantotomas.edu.co

***Abstract*—Este trabajo presenta la implementación y análisis comparativo de diferentes técnicas de detección de errores en comunicaciones serie industriales, utilizando ESP32 y Raspberry Pi Pico. Se implementaron tres protocolos: checksum simple, protocolo ARQ stop-and-wait, y comparación entre VRC (Vertical Redundancy Check) y LRC (Longitudinal Redundancy Check). Los resultados experimentales demuestran que el protocolo ARQ alcanza 100% de efectividad hasta 30% de probabilidad de error, mientras que LRC presenta 49% menos overhead que VRC manteniendo eficacia comparable. La comunicación TTL directa entre dispositivos de 3.3V elimina la necesidad de conversores RS232, simplificando el diseño sin comprometer la funcionalidad.**

***Index Terms*—Detección de errores, ARQ, VRC, LRC, comunicaciones industriales, UART**

I. Introducción

Las comunicaciones serie industriales requieren métodos confiables de detección y corrección de errores para garantizar la integridad de datos en ambientes con interferencia electromagnética. Este estudio implementa y compara tres enfoques fundamentales: checksum simple para detección básica, protocolo ARQ para recuperación automática, y técnicas VRC/LRC para diferentes granularidades de verificación.

La selección de ESP32 y Raspberry Pi Pico permite implementar comunicación UART a 9600 baudios con niveles TTL de 3.3V nativos, eliminando la necesidad de conversores MAX3232 tradicionalmente requeridos para interfaces RS232.

II. Metodología

A. Configuración Hardware

La comunicación se estableció mediante conexión TTL directa:

- ESP32 GPIO17 (TX) → Pico GP1 (RX)
- ESP32 GPIO16 (RX) → Pico GP0 (TX)
- GND común entre dispositivos

La compatibilidad de voltajes 3.3V entre ambos dispositivos justifica la omisión del convertidor MAX3232, simplificando el diseño mientras mantiene estándares industriales de comunicación.

B. Protocolos Implementados

1) Checksum Simple: Implementación de checksum módulo 256 con formato de trama STX-LEN-PAYLOAD-CS-ETX. Se evaluó efectividad mediante inyección controlada de errores cada tercer paquete.

2) Protocolo ARQ: Sistema stop-and-wait con ACK/NACK y hasta 3 reintentos por paquete. Se midió efectividad con probabilidades de error de 0%, 20%, 30% y 50%.

3) VRC vs LRC: Comparación entre Vertical Redundancy Check (paridad por byte) y Longitudinal Redundancy Check (XOR de bloque). Sincronización mediante marcadores de control (0xAA, 0xBB, 0xCC, 0xDD).

III. Resultados

A. Protocolo ARQ

La Tabla I presenta los resultados experimentales del protocolo ARQ con diferentes probabilidades de error simulado.

TABLE I
Efectividad del Protocolo ARQ

Error %	Reintentos Prom.	Tasa Entrega %	Overhead %
0	0.00	100.0	0
20	0.24	100.0	24
30	0.29	100.0	29
50	0.90	76.2	90

B. Comparación VRC vs LRC

La Tabla II muestra el análisis comparativo de overhead y eficiencia para un bloque de 5 bytes.

TABLE II
Comparación VRC vs LRC

Método	Overhead %	Eficiencia %	Bytes Total
VRC	140	41.7	12
LRC	60	62.5	8

El overhead se calcula como: $\frac{\text{bytes extra}}{\text{bytes datos}} \times 100\%$
Para VRC: $\frac{7}{5} \times 100\% = 140\%$ (5 VRC + 2 marcadores)
Para LRC: $\frac{3}{5} \times 100\% = 60\%$ (1 LRC + 2 marcadores)

IV. Discusión

A. Función del Convertidor MAX3232

El MAX3232 convierte señales TTL (0-3.3V) a estándares RS232 (±12V) para comunicación de larga distancia. Sin embargo, la comunicación directa TTL entre ESP32 y Pico (ambos 3.3V) es compatible nativamente, eliminando la necesidad del convertidor y reduciendo complejidad del circuito.

B. Inyección de Errores Artificiales

Se implementaron tres métodos de inyección:

- Alteración de payload antes de calcular checksum
- Simulación probabilística en el receptor
- Desconexión física de cables

C. Efectividad de Detección

VRC detecta errores de bit único con alta granularidad, mientras LRC es más eficiente para errores distribuidos en bloques. El protocolo ARQ garantiza recuperación hasta 30% de error con 100% de entrega.

D. Análisis de Overhead

Para bloques de tamaño n:

- VRC: Overhead = $\frac{n+2}{n} \times 100\%$
- LRC: Overhead = $\frac{3}{n} \times 100\%$

LRC mejora escalabilidad con bloques grandes, mientras VRC mantiene overhead constante independiente del tamaño.

E. Aplicaciones Industriales

VRC: Sistemas críticos de seguridad donde cada byte requiere validación individual (controladores de vuelo, sistemas médicos).

LRC: Comunicaciones de alta velocidad con grandes volúmenes de datos (SCADA, redes industriales Ethernet).

V. Conclusiones

Este estudio demuestra que la selección del método de detección de errores debe balancear overhead vs. granularidad según la aplicación específica. El protocolo ARQ ofrece excelente confiabilidad hasta 30% de error, LRC proporciona mayor eficiencia que VRC (49% menos overhead), y la comunicación TTL directa simplifica diseños sin comprometer funcionalidad.

Para aplicaciones industriales futuras, se recomienda la combinación ARQ + LRC para optimizar confiabilidad y eficiencia en comunicaciones de bloques de datos grandes.

References

[1] D. A. Barragán, "Comunicaciones Industriales - Laboratorio 2," Universidad Santo Tomás, Bogotá, 2025.

[2] Analog Devices (Maxim), "MAX3232 Data Sheet," [Online]. Available: https://www.analog.com/media/en/technical-documentation/data-sheets/max3222-max3241.pdf

[3] Raspberry Pi Foundation, "RP2040 Datasheet," [Online]. Available: https://datasheets.raspberrypi.org/rp2040/rp2040-datasheet.pdf

[4] Espressif Systems, "ESP32 Series Datasheet," [Online]. Available: https://www.espressif.com/sites/default/files/documentation/esp32$_datasheet_en.pdf$