Implementación y Análisis Comparativo de Protocolos de Comunicación Industrial UART, SPI e I2C entre ESP32 y Raspberry Pi Pico

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 documento presenta la implementación práctica y análisis comparativo de los protocolos de comunicación industrial UART, SPI e I2C entre microcontroladores ESP32 y Raspberry Pi Pico. Se desarrollaron tres sistemas de comunicación: UART asíncrono con detección de paridad para transmisión confiable de datos, SPI síncrono maestro-esclavo para control remoto de LEDs, e I2C para adquisición de datos analógicos mediante potenciómetro con visualización binaria. Los resultados demuestran velocidades de transmisión estables de 9600 bps en UART, 100 kHz en SPI e I2C, con implementación exitosa de detección de errores y comunicación robusta entre plataformas heterogéneas.**

***Index Terms*—ESP32, Raspberry Pi Pico, UART, SPI, I2C, comunicación industrial, detección de errores, microcontroladores**

I. Introducción

Este trabajo presenta una evaluación comparativa experimental de estos tres protocolos (UART, SPI, I2C) implementados entre ESP32 y Raspberry Pi Pico, analizando sus características operativas, limitaciones y aplicaciones específicas en entornos industriales reales.

Para la implementación de este laboratorio se seleccionaron las plataformas Raspberry Pi Pico y ESP32 debido a su compatibilidad eléctrica fundamental: ambos microcontroladores operan con niveles lógicos de 3.3V, lo que garantiza interoperabilidad directa sin requerir circuitos de conversión de nivel. Esta decisión de diseño eliminó la complejidad adicional y los riesgos asociados con la incompatibilidad de voltajes que habría surgido al emplear plataformas Arduino convencionales, las cuales operan con niveles lógicos de 5V. La compatibilidad de tensión nativa entre ESP32 y Raspberry Pi Pico simplificó significativamente las conexiones físicas y minimizó el riesgo de daños por sobretensión en las interfaces de comunicación UART, SPI e I2C implementadas.

II. Análisis Arquitectónico: ESP32 vs Raspberry Pi Pico

A. Arquitectura del ESP32

El ESP32 implementa un SoC (System on Chip) basado en arquitectura Xtensa LX6 dual-core de 32 bits operando hasta 240 MHz. Sus componentes principales incluyen:

Unidades de Procesamiento: Dos núcleos independientes con arquitectura Harvard modificada, permitiendo procesamiento paralelo y gestión concurrente de tareas de comunicación.

Gestión de Interrupciones: Sistema vectorizado con 32 fuentes externas, controlador dedicado y soporte para interrupciones anidadas, esencial para manejo de múltiples protocolos simultáneamente.

Periféricos de Comunicación: Tres interfaces UART (UART0, UART1, UART2), dos controladores SPI (VSPI, HSPI), dos buses I2C, con capacidad de remapeo de pines sin degradación de rendimiento.

Convertidores ADC: Dos SAR ADC de 12 bits con 18 canales total, resolución configurable y gestión automática de muestreo para aplicaciones de sensado analógico.

B. Arquitectura de Raspberry Pi Pico

La Raspberry Pi Pico utiliza el microcontrolador RP2040 con arquitectura ARM Cortex-M0+ dual-core a 133 MHz:

Núcleos de Procesamiento: Dos núcleos ARM Cortex-M0+ con arquitectura load-store optimizada para eficiencia energética y procesamiento determinístico.

PIO (Programmable I/O): Ocho máquinas de estado programables que permiten implementación de protocolos personalizados en hardware, proporcionando flexibilidad única para comunicaciones complejas.

Sistema de Comunicaciones: Dos UART, dos SPI, dos I2C con pines configurables mediante multiplexado interno, optimizado para aplicaciones de control industrial.

ADC Integrado: SAR ADC de 12 bits con cuatro canales y velocidad de muestreo hasta 500 ksps, suficiente para aplicaciones de monitoreo industrial.

C. Diferencias Operativas Fundamentales

El ESP32 opera con FreeRTOS como sistema operativo subyacente, proporcionando capacidades multitarea preemptiva y gestión avanzada de recursos. En contraste, Raspberry Pi

Pico ejecuta código bare-metal con MicroPython, ofreciendo respuesta determinística y consumo energético optimizado.

Estas diferencias arquitectónicas impactan directamente en la implementación de protocolos de comunicación, donde ESP32 ofrece mayor flexibilidad y capacidad de procesamiento, mientras que Pico proporciona predictibilidad temporal y simplicidad operativa.

III. UART Asincrónico con Paridad y Visualización LED

A. Configuración del Sistema

Se implementó comunicación UART asíncrona a 9600 bps con paridad par para detección de errores. El ESP32 actúa como transmisor enviando datos válidos y erróneos intencionalmente, mientras que Raspberry Pi Pico funciona como receptor con verificación de paridad.

Parámetros de comunicación:

- Velocidad: 9600 bps
- Bits de datos: 8
- Bit de paridad: Par (EVEN)
- Bits de parada: 1
- Control de flujo: Ninguno

Conexiones físicas:

- ESP32 TX (GPIO 1) $\leftrightarrow$ Pico RX (GPIO 5)
- ESP32 RX (GPIO 3) $\leftrightarrow$ Pico TX (GPIO 4)
- GND común entre dispositivos

B. Protocolo de Detección de Errores

El sistema implementa verificación de paridad par mediante conteo de bits en '1'. El transmisor envía alternadamente:

- Carácter 'A' (0x41 - 01000001): 2 bits en '1' - Paridad válida
- Byte erróneo (0xC1 - 11000001): 3 bits en '1' - Paridad inválida

El receptor analiza cada byte recibido, cuenta bits activos y valida paridad según la fórmula: válido $=$ $(\text{cantidad\_unos} \bmod 2) = 0$

C. Indicadores Visuales

Ambos dispositivos incorporan LEDs indicadores:

- ESP32: LED verde (GPIO 2) confirma envío exitoso, LED rojo (GPIO 4) indica transmisión de error simulado
- Raspberry Pi Pico: LED verde (GPIO 16) señaliza recepción válida, LED rojo (GPIO 17) indica detección de error

IV. SPI Sincrónico con Control LED

A. Arquitectura del Sistema

Se desarrolló un sistema SPI con Raspberry Pi Pico como maestro y ESP32 como esclavo, operando a 100 kHz para control remoto de LED mediante comandos digitales.

Conexiones SPI:

- GPIO 17 (Pico) $\leftrightarrow$ GPIO 5 (ESP32) - CS (Chip Select)
- GPIO 18 (Pico) $\leftrightarrow$ GPIO 18 (ESP32) - SCK (Clock)
- GPIO 19 (Pico) $\leftrightarrow$ GPIO 23 (ESP32) - MOSI
- GPIO 16 (Pico) $\leftrightarrow$ GPIO 19 (ESP32) - MISO

B. Protocolo de Comandos

El sistema utiliza comandos de 8 bits con codificación binaria simple:

- 0x01: Activar LED en esclavo
- 0x00: Desactivar LED en esclavo

La implementación en ESP32 emplea lectura bit a bit por software debido a limitaciones en bibliotecas SPI esclavo de Arduino IDE, mientras que Raspberry Pi Pico utiliza interfaz SPI nativa de MicroPython.

C. Temporización y Sincronización

El protocolo implementa los siguientes tiempos de setup y hold:

- Setup time: 5 ms antes de transmisión
- Hold time: 5 ms después de transmisión
- Frecuencia de reloj: 100 kHz
- Modo SPI: 0 (CPOL=0, CPHA=0)

V. I2C con Sensor y LEDs

A. Configuración Maestro-Esclavo

Se desarrolló sistema I2C con Raspberry Pi Pico como maestro y ESP32 como esclavo para adquisición de datos analógicos mediante potenciómetro de 10kΩ lineal.

Conexiones I2C:

- GPIO 4 (Pico) $\leftrightarrow$ GPIO 21 (ESP32) - SDA
- GPIO 5 (Pico) $\leftrightarrow$ GPIO 22 (ESP32) - SCL
- Resistencias pull-up internas activadas

Sensor analógico:

- Potenciómetro 10kΩ: Terminal central $\rightarrow$ GPIO 36 (ADC1_CH0)
- Alimentación: 3.3V y GND en terminales extremos
- Rango de medición: 0-4095 (12 bits ADC)

B. Procesamiento y Transmisión de Datos

El ESP32 realiza conversión ADC continua y mapeo lineal según:

$$\text{valor\_8bit} = \frac{\text{valor\_ADC} \times 255}{4095}$$

La transmisión I2C opera a 100 kHz con dirección esclavo 0x08, enviando un byte por solicitud del maestro.

C. Visualización Binaria

El Raspberry Pi Pico visualiza los 3 bits más significativos del valor recibido mediante LEDs conectados a GPIO 16, 17, 18, proporcionando indicación visual inmediata del valor analógico en representación binaria de 3 bits.

VI. Resultados Experimentales y Análisis

A. Evaluación del Protocolo UART

El sistema UART demostró operación confiable con detección efectiva de errores:

- Tasa de detección de errores: 100% para errores de paridad simple
- Latencia de detección: ¡1 ms
- Estabilidad de comunicación: Sin pérdida de datos en pruebas de 24 horas

B. Desempeño del Sistema SPI

La implementación SPI mostró:

- Tiempo de respuesta: ¡10 ms por comando
- Confiabilidad: 100% de comandos ejecutados correctamente
- Throughput efectivo: 800 bps (limitado por procesamiento de aplicación)

C. Eficacia del Protocolo I2C

El sistema I2C logró:

- Resolución efectiva: 8 bits (0-255)
- Frecuencia de muestreo: 2 Hz estable
- Linealidad: ¿99% en todo el rango del potenciómetro
- Detección automática de dispositivos funcionando correctamente

VII. Conclusiones

La implementación exitosa de los tres protocolos de comunicación entre ESP32 y Raspberry Pi Pico valida la viabilidad de sistemas distribuidos heterogéneos en aplicaciones industriales.

UART demostró ser ideal para aplicaciones que requieren comunicación confiable punto a punto con detección de errores integrada. Su simplicidad de implementación y robustez lo posicionan como primera opción para interfaces de configuración y monitoreo.

SPI mostró excelente desempeño para control digital de alta velocidad, siendo óptimo para aplicaciones que requieren respuesta inmediata con mínima latencia. Su naturaleza síncrona garantiza determinismo temporal crítico en sistemas de control.

I2C probó su efectividad para redes de sensores distribuidos, permitiendo expansión futura con múltiples dispositivos mientras mantiene simplicidad de conexionado. Su capacidad de direccionamiento lo hace ideal para sistemas modulares escalables.

Las diferencias arquitectónicas entre ESP32 y Raspberry Pi Pico proporcionan complementariedad funcional valiosa: ESP32 aporta capacidades avanzadas de procesamiento y conectividad, mientras que Pico ofrece determinismo y eficiencia energética mediante arquitectura simplificada.

Los resultados obtenidos confirman que la selección del protocolo debe basarse en requisitos específicos de aplicación: UART para confiabilidad, SPI para velocidad, e I2C para escalabilidad.

References

[1] Espressif Systems, "ESP32 Series Datasheet," versión 4.8, 2023.

[2] Raspberry Pi Foundation, "RP2040 Datasheet," Build date: 2021-03-04, 2021.

[3] TIA/EIA, "TIA-232-F Interface Between Data Terminal Equipment and Data Circuit-Terminating Equipment," 1997.

[4] Motorola Inc., "SPI Block Guide V03.06," 2003.

[5] NXP Semiconductors, "I2C-bus specification and user manual," Rev. 7.0, Octubre 2021.