Práctica 15: Uso de protocolo Bluetooth LE en la Raspberry Pi Pico W

Facultad de Estudios Profesionales Zona Media - UASLP

Microcontroladores

Nombre del alumno: Sergio Adolfo Juarez Mendoza
Clave:369376
Profesor: Ing. Jesús Padrón

20 de mayo del 2025

1. Introducción

En esta práctica se implementó un sistema de comunicación inalámbrica utilizando el protocolo Bluetooth Low Energy (BLE) entre dos dispositivos Raspberry Pi Pico W. El objetivo principal fue configurar un dispositivo como servidor BLE que lee la temperatura interna del microcontrolador y la transmite a un segundo dispositivo configurado como cliente BLE, el cual recibe y muestra los datos en el monitor serial.

A través de esta práctica, se comprendieron los conceptos fundamentales de Bluetooth LE, sus diferencias con Bluetooth clásico, y se aprendió a utilizar la librería BTStack para implementar comunicación inalámbrica en sistemas embebidos. La implementación incluyó la configuración de servicios GATT, características y notificaciones BLE, así como el manejo de eventos asíncronos en el stack Bluetooth.

2. Desarrollo de la práctica

2.1. Configuración del entorno

Se utilizó el siguiente archivo CMakeLists.txt para compilar los proyectos tanto del servidor como del cliente:

```
cmake_minimum_required(VERSION 3.12)
include(pico_sdk_import.cmake)

project(Practice_15 C CXX ASM)
set(PICO_BOARD pico_w)
set(CMAKE_C_STANDARD 11)
set(CMAKE_CXX_STANDARD 17)

pico_sdk_init()

# Servidor BLE
add_executable(picow_ble_temp_sensor
    server.c server.common.c
)
target_link_libraries(picow_ble_temp_sensor
    pico_stdlib
    pico_btstack_ble
    pico_btstack_cyw43
    pico_cyw43_arch_none
    hardware_adc
)
pico_btstack_make_gatt_header(picow_ble_temp_sensor
    PRIVATE "${CMAKE_CURRENT_LIST_DIR}/temp_sensor.gatt")

# Cliente BLE
add_executable(picow_ble_temp_reader
    client.c
)
target_link_libraries(picow_ble_temp_reader
    pico_stdlib
    pico_btstack_ble
    pico_btstack_cyw43
    pico_cyw43_arch_none
)
```

2.2. Implementación del servidor BLE

El servidor BLE se configuró para proporcionar un servicio de sensor ambiental con una característica de temperatura. El archivo GATT define la estructura del servicio:

```
PRIMARY_SERVICE, GAP_SERVICE
CHARACTERISTIC, GAP_DEVICE_NAME, READ, "picow_temp"

PRIMARY_SERVICE, GATT_SERVICE
CHARACTERISTIC, GATT_DATABASE_HASH, READ,

PRIMARY_SERVICE, ORG.BLUETOOTH_SERVICE_ENVIRONMENTAL_SENSING
CHARACTERISTIC, ORG.BLUETOOTH_CHARACTERISTIC_TEMPERATURE,
    READ | NOTIFY | INDICATE | DYNAMIC,
```

El servidor implementa las siguientes funciones clave:

- Lectura del sensor de temperatura interno
- Configuración de anuncios BLE
- Manejo de eventos de conexión/desconexión
- Notificación de cambios en la temperatura

2.3. Implementación del cliente BLE

El cliente BLE se encarga de:

- Escanear dispositivos BLE cercanos
- Filtrar dispositivos que ofrecen el servicio de sensor ambiental
- Conectarse al servidor y descubrir características
- Suscribirse a notificaciones de temperatura
- Mostrar los valores recibidos en el monitor serial

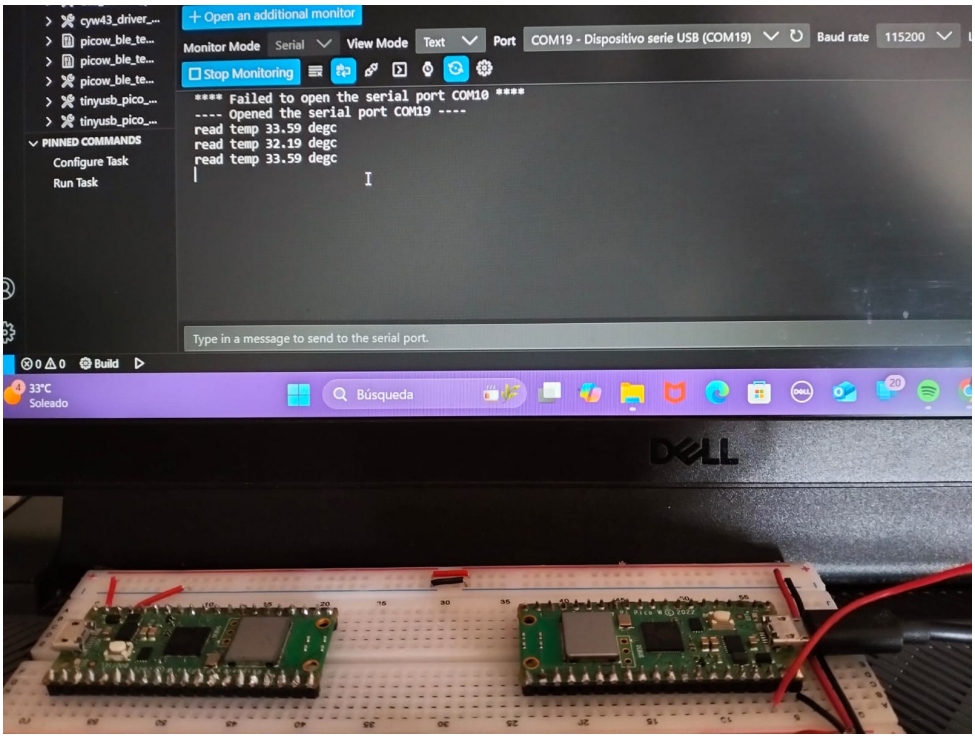

Figura 1: Diagrama de conexión entre servidor y cliente BLE

2.4. Complicaciones y soluciones

- **Problema**: Inicialmente el cliente no podía encontrar el servidor.
  Solución: Verificar que ambos dispositivos estén configurados con los mismos parámetros de anuncio y que el servicio UUID esté correctamente definido en ambos lados.
- **Problema**: Las notificaciones de temperatura no llegaban consistentemente.
  Solución: Asegurar que el cliente haya habilitado correctamente las notificaciones mediante el descriptor de configuración de cliente.
- **Problema**: El servidor dejaba de funcionar después de algunas horas.
  Solución: Implementar manejo de errores más robusto y reinicio automático en caso de fallas.

3. Preguntas de repaso

1. **¿Cuál es la principal diferencia entre Bluetooth clásico y Bluetooth LE?**

   La principal diferencia es el consumo de energía. Bluetooth LE está diseñado específicamente para bajo consumo, sacrificando ancho de banda (1-2 Mbps vs 3 Mbps del clásico). Bluetooth LE usa una arquitectura de conexión por eventos con tiempos de conexión muy cortos, mientras que el clásico mantiene conexiones continuas.

2. **¿Qué ventajas ofrece Bluetooth LE en comparación con otras tecnologías inalámbricas?**

   Ventajas clave:

   - Bajo consumo (años de operación con batería de botón)

- Baja latencia (3-6 ms)
- Compatibilidad casi universal con smartphones
- Topologías flexibles (punto a punto, broadcast, malla)
- Menor complejidad de implementación que WiFi

3. **¿Cuál es el propósito del archivo btstack_config.h en esta práctica?**

   Este archivo configura aspectos fundamentales del stack Bluetooth:

   - Habilitar funcionalidades (LE Peripheral/Central) Definir tamaños de buffers y memoria Configurar parámetros de conexión Especificar capacidades de seguridad

4. **¿Qué ventajas crees que tiene usar comunicación BLE en un sistema de monitoreo frente a WiFi o comunicación por cable?**

   Ventajas:

   - Menor consumo energético (ideal para sensores con batería)
   - Menor complejidad de configuración que WiFi
   - No requiere infraestructura de red Mayor flexibilidad de ubicación que cableado Costo reducido de implementación

5. **¿Por qué crees que es importante que el código del cliente BLE esté diseñado para reconectarse automáticamente al servidor en caso de desconexión?**

   Es crucial porque:

   - Las conexiones BLE pueden perderse por interferencia o movimiento
   - Garantiza continuidad en el monitoreo sin intervención humana
   - Mejora la experiencia del usuario al no requerir reconexión manual
   - Especialmente importante en aplicaciones IoT donde la supervisión debe ser constante

6. **En un ambiente con muchos dispositivos BLE, ¿qué estrategias se podrían usar para evitar interferencias o confusiones al conectar dispositivos?**

   Estrategias efectivas:

   - Usar UUIDs de servicio específicos para la aplicación
   - Implementar filtrado por dirección MAC o nombre de dispositivo
   - Usar RSSI para seleccionar el dispositivo más cercano Emplear bonding/pairing para conexiones persistentes Canales de anuncio personalizados (en Bluetooth 5+)

7. **¿Cómo modificarías esta práctica para mostrar los datos recolectados en una aplicación móvil usando Bluetooth LE?**

   Modificaciones necesarias:

   - Desarrollar app móvil con frameworks como Flutter o React Native

- Usar APIs como CoreBluetooth (iOS) o BluetoothAdapter (Android)
- Mantener la misma estructura GATT para compatibilidad Añadir interfaz gráfica para visualización histórica Implementar notificaciones push para valores críticos

4. Conclusiones

Esta práctica permitió comprender los fundamentos de Bluetooth LE y su implementación en sistemas embebidos. Se logró establecer comunicación inalámbrica confiable entre dos Raspberry Pi Pico W, demostrando la utilidad de BLE para aplicaciones IoT con requerimientos de bajo consumo.

Los principales aprendizajes fueron:

- Configuración de servicios y características GATT
- Manejo de eventos asíncronos en BTStack
- Lectura del sensor de temperatura interno de la Pico W
- Implementación de patrones cliente-servidor en BLE

Como posibles mejoras se podrían considerar:

- Añadir seguridad mediante pairing con PIN
- Implementar historial de temperaturas en el cliente Reducir aún más el consumo energético optimizando intervalos de conexión Extender para manejar múltiples sensores simultáneamente