Práctica 15: Comunicación BLE entre dos Raspberry Pi Pico W

José Josef Medellin Reséndez

19 de mayo de 2025

Introducción

En esta práctica trabajamos con dos Raspberry Pi Pico W para realizar una comunicación Bluetooth Low Energy (BLE). El objetivo fue que una de las placas actuara como servidor, midiendo la temperatura interna de su microcontrolador, y que la otra funcionara como cliente, recibiendo esos datos y mostrándolos por consola.

Para lograr esto, usamos la pila BTstack junto con archivos `.c`, `.gatt` y `CMakeLists.txt`, que nos permitieron compilar y ejecutar correctamente los dos programas. Esta práctica nos ayudó a entender cómo enviar y recibir información entre dos dispositivos usando BLE.

Desarrollo

Primero, se configuró el proyecto para que el archivo `CMakeLists.txt` generara dos ejecutables: uno para el **servidor** y otro para el **cliente**. También se incluyó la pila de BTstack para poder usar la funcionalidad BLE.

El **servidor BLE** mide la temperatura usando el sensor interno de la Pico, y la envía por medio de notificaciones BLE usando un servicio definido en el archivo `temp_sensor.gatt`.

El **cliente BLE** busca al servidor, se conecta, y empieza a recibir las temperaturas, las cuales imprime en el monitor serial.

Ambas placas se programaron en C y se compilaron usando CMake. Para probarlo, se cargaron los programas en las dos placas y se comprobó que el cliente recibía la temperatura correctamente. Fue importante tener ordenados todos los archivos fuente y configurar bien el entorno.

Conclusión

Esta práctica fue útil para aprender cómo se puede usar Bluetooth Low Energy entre dos Raspberry Pi Pico W. Pude ver cómo se configura un servicio BLE, cómo enviar datos y cómo recibirlos del otro lado.

También entendí mejor cómo se usa el ADC interno para medir temperatura, y cómo organizar un proyecto con varios archivos en C usando BTstack. Aunque al principio fue

algo confuso, con paciencia logré que todo funcionara bien. Fue una práctica muy completa y aplicable a proyectos reales de comunicación inalámbrica.

Anexo: Contenido de `CMakeLists.txt`

```
cmake_minimum_required(VERSION 3.12)

# Pull in SDK (must be before project)
include(pico_sdk_import.cmake)

if (PICO_SDK_VERSION_STRING VERSION_LESS "1.5.0")
    message(FATAL_ERROR "Raspberry Pi Pico SDK version
    1.5.0 (or later) required. Your version is ${
        PICO_SDK_VERSION_STRING}")
endif()

project(Practica 15 C CXX ASM)
set(PICO_BOARD pico_w)
set(CMAKE_C_STANDARD 11)
set(CMAKE_CXX_STANDARD 17)

# Initialize the SDK
pico_sdk_init()

# Standalone example that reads from the on board
# temperature sensor and sends notifications via BLE
# Flashes slowly each second to show it's running
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
)

target_include_directories(picow_ble_temp_sensor PRIVATE
    ${CMAKE_CURRENT_LIST_DIR} # For btstack config
)
```

```
pico_btstack_make_gatt_header(picow_ble_temp_sensor
    PRIVATE "${CMAKE_CURRENT_LIST_DIR}/temp_sensor.gatt"
)

pico_add_extra_outputs(picow_ble_temp_sensor)

# Flashes twice quickly each second when connected to
# another device and reading its temperature
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
    hardware_adc
)

pico_enable_stdio_usb(picow_ble_temp_reader 1)
pico_enable_stdio_uart(picow_ble_temp_reader 0)

target_include_directories(picow_ble_temp_reader PRIVATE
    ${CMAKE_CURRENT_LIST_DIR} # For btstack config
)

target_compile_definitions(picow_ble_temp_reader PRIVATE
    RUNNING_AS_CLIENT=1
)

pico_add_extra_outputs(picow_ble_temp_reader)
```