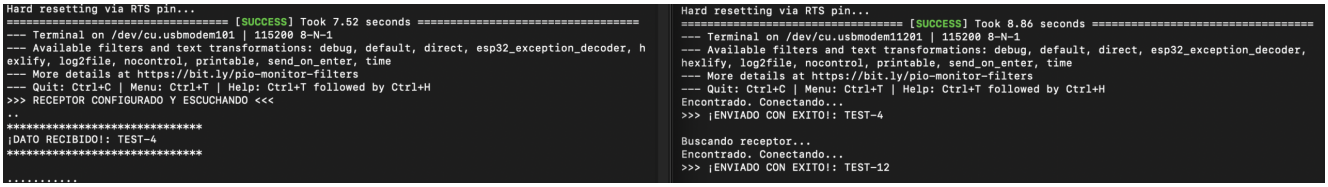
Comunicación BLE entre ESP32-S3 (SuperMini y Zero)



Esta guía explica cómo conectar dos placas ESP32-S3 mediante **Bluetooth Low Energy (BLE)** usando la librería **NimBLE**, optimizada para dispositivos con 4MB de memoria Flash en macOS.

1. Hardware Necesario

- **1x ESP32-S3 SuperMini** (Emisor).
- **1x ESP32-S3 Zero** (Receptor).
- **2x Cables USB-C de alta calidad** (Asegúrate de que sean de **datos**, no solo de carga).
- **Ordenador:** macOS (Probado en MacBook Air M1/M2/M3).

---

2. Preparación del Entorno (macOS)

Instalamos **PlatformIO Core** para gestionar los proyectos desde la terminal.

1. **Instalar PlatformIO:** `curl -fsSL https://raw.githubusercontent.com/platformio/platformio-core-installer/master/get-platformio.py -o get-platformio.py python3 get-platformio.py`
2. **Configurar el Path:** `echo 'export`

```
PATH=$PATH:$HOME/.platformio/penv/bin'  >>  ~/.zshrc
source ~/.zshrc
```

---

3. Proyecto A: El Receptor (ESP32-S3 Zero)

Este dispositivo actúa como **Servidor BLE**, exponiendo un «buzón» (Característica) donde otros pueden escribir.

Paso 1: Inicializar

Bash

```
mkdir -p ~/S3_Receptor && cd ~/S3_Receptor
pio project init --board esp32-s3-devkitc-1
```

Paso 2: Configurar platformio.ini

Es crucial configurar los **4MB de Flash** y activar el USB CDC.

Ini, TOML

```
[env:esp32-s3-zero]
platform = espressif32
board = esp32-s3-devkitc-1
framework = arduino
monitor_speed = 115200
board_build.arduino.memory_type = qio_qspi
board_build.flash_mode = dio
board_upload.flash_size = 4MB
board_build.partitions = min_spiffs.csv
build_flags =
    -D ARDUINO_USB_MODE=1
    -D ARDUINO_USB_CDC_ON_BOOT=1
lib_deps = h2zero/NimBLE-Arduino @ ^1.4.1
```

Paso 3: Código src/main.cpp

C++

```
#include <Arduino.h>
#include <NimBLEDevice.h>

#define SERVICE_UUID        "1234"
#define CHARACTERISTIC_UUID "5678"

class MyCallbacks : public NimBLECharacteristicCallbacks {
    void onWrite(NimBLECharacteristic* pCharacteristic) {
        std::string value = pCharacteristic->getValue();
            Serial.printf("\n¡DATO RECIBIDO!: %s\n",
value.c_str());
    }
};

void setup() {
  Serial.begin(115200);
  NimBLEDevice::init("S3-Zero-Receptor");
  NimBLEServer* pServer = NimBLEDevice::createServer();
               NimBLEService*       pService       =
pServer->createService(SERVICE_UUID);
             NimBLECharacteristic*      pChar      =
pService->createCharacteristic(
                                CHARACTERISTIC_UUID,
                                NIMBLE_PROPERTY::WRITE
                               );
  pChar->setCallbacks(new MyCallbacks());
  pService->start();
           NimBLEAdvertising*     pAdvertising     =
NimBLEDevice::getAdvertising();
  pAdvertising->addServiceUUID(SERVICE_UUID);
  pAdvertising->start();
  Serial.println("Receptor listo...");
}

void loop() { delay(2000); }
```

4. Proyecto B: El Emisor (ESP32-S3 SuperMini)

Este dispositivo actúa como **Cliente BLE**, escaneando el aire y enviando datos al receptor.

Paso 1: Inicializar

Bash

```
mkdir -p ~/S3_Emisor && cd ~/S3_Emisor
pio project init --board esp32-s3-devkitc-1
```

Paso 2: Configurar platformio.ini (Igual al receptor)

Paso 3: Código src/main.cpp

C++

```
#include <Arduino.h>
#include <NimBLEDevice.h>

#define SERVICE_UUID        "1234"
#define CHARACTERISTIC_UUID "5678"

void setup() {
  Serial.begin(115200);
  NimBLEDevice::init("S3-SuperMini-Emisor");
}

void loop() {
  NimBLEScan* pScan = NimBLEDevice::getScan();
  NimBLEScanResults results = pScan->start(4, false);

  for (int i = 0; i < results.getCount(); i++) {
```

```
    NimBLEAdvertisedDevice device = results.getDevice(i);
    if (device.getName() == "S3-Zero-Receptor") {
      NimBLEClient* pClient = NimBLEDevice::createClient();
      if (pClient->connect(&device)) {
                        NimBLERemoteService*   pSvc   =
pClient->getService(SERVICE_UUID);
        if (pSvc) {
                   NimBLERemoteCharacteristic*  pChr  =
pSvc->getCharacteristic(CHARACTERISTIC_UUID);
          if (pChr) {
            String valor = "TEST-" + String(millis()/1000);
            pChr->writeValue(valor.c_str());
            Serial.println(">>> ENVIADO: " + valor);
          }
        }
        pClient->disconnect();
      }
      NimBLEDevice::deleteClient(pClient);
    }
  }
  delay(3000);
}
```

---

5. Publicación y Ejecución

Para que todo funcione, sigue este orden de comandos:

1. **Identificar Puertos:** ls /dev/cu.usbmodem*
2. **Subir Receptor:**Bashpio run --target upload --upload-port /dev/cu.usbmodem101 pio device monitor -p /dev/cu.usbmodem101
3. **Subir Emisor:**Bashpio run --target upload --upload-port /dev/cu.usbmodem11201 pio device monitor -p /dev/cu.usbmodem11201

⚠ Lecciones Aprendidas (Troubleshooting)

- **Flash 4MB vs 8MB:** La mayoría de placas genéricas S3 son de 4MB. Si usas la config de 8MB, la placa entrará en bootloop.
- **Librería NimBLE:** Es mucho más eficiente en memoria que la librería BLE estándar de Arduino, ideal para el S3.
- **Botón Reset:** En macOS, si el monitor no arranca, desconecta y conecta el USB o pulsa el botón físico de RESET en la placa.