 Icono

El contenido generado por IA puede ser incorrecto.

|  |
| --- |
| **Diseños C++**  **SPRINT 0** |

**PROYECTO APLICACIONES DE BIOMETRÍA Y MEDIO AMBIENTE**

**Nombre del Alumno:** Alan Guevara Martínez

**Enlace Git: [https://github.com/ALANGMupv/ProyectoBiometria2025.git](https://github.com/ALANGMupv/ProyectoBiometria2025.git" \t "_blank)**

**Enlace trello:** [**https://trello.com/invite/b/68c9273bf08b76ab09c96593/ATTI19e0d5a059994ed24df6c6455b577bae3E1C4F40/gtiproyectobiometria**](https://trello.com/invite/b/68c9273bf08b76ab09c96593/ATTI19e0d5a059994ed24df6c6455b577bae3E1C4F40/gtiproyectobiometria)

**Curso:** 2025/2026

# 

**ÍNDICE / TABLA DE CONTENIDOS**

[Introducción 2](#_Toc209638393)

[HolaMundoIBeacon (archivo principal) 2](#_Toc209638394)

[Clase EmisoraBLE 2](#_Toc209638395)

[Clase LED 3](#_Toc209638396)

[Clase Medidor 4](#_Toc209638397)

[Clase Publicador 4](#_Toc209638398)

[Clase Puerto Serie 5](#_Toc209638399)

[Clase ServicioEnEmisora 5](#_Toc209638400)

**TABLA DE ILUSTRACIONES / DISEÑOS**

[Ilustración 1: Diseño EmisoraBLE 3](#_Toc209638445)

[Ilustración 2: Diseño LED 4](#_Toc209638446)

[Ilustración 3: Diseño Medidor 5](#_Toc209638447)

[Ilustración 4: Diseño Publicador 5](#_Toc209638448)

[Ilustración 5: Diseño PuertoSerie 6](#_Toc209638449)

[Ilustración 6: Diseño ServicioEnEmisora 7](#_Toc209638450)

##### Introducción

En este documento se han realizado los diseños de las clases en C++ que estructuran el proyecto. Cada clase cumple una responsabilidad concreta, facilitando la modularidad y la escalabilidad del código.

El archivo principal (*HolaMundoIBeacon*) actúa como coordinador: contiene las funciones setup() y loop(), que son el punto de entrada en Arduino, y desde ahí se realizan las llamadas al resto de clases.

Los diagramas completos de diseño pueden consultarse en el siguiente enlace de Figma:

<https://www.figma.com/board/M5GoSHtcgihRU9eOloRw9p/Ingenier%C3%ADa-Inversa-C----Dise%C3%B1o-?node-id=0-1&t=fQN68Fkj8KLjSxMq-1>

##### HolaMundoIBeacon (archivo principal)

Este archivo define la lógica central del programa:

* **setup()**: se ejecuta una vez al encender la placa. Inicializa el puerto serie, llama a inicializarPlaquita(), enciende la emisora BLE, inicia el medidor y muestra un mensaje de fin de configuración.
* **loop()**: se repite indefinidamente mientras la placa esté encendida. En cada ciclo incrementa un contador, hace parpadear el LED, mide el valor de CO₂ y temperatura, publica ambos datos por Bluetooth y finalmente emite un beacon libre de prueba.

El archivo principal no implementa lógica compleja, sino que orquesta la ejecución llamando a los métodos definidos en las clases auxiliares.

**Clases principales:**

##### Clase EmisoraBLE

Encargada de manejar el chip Bluetooth Low Energy (nRF52) y controlar los anuncios.

* **EmisoraBLE(nombreEmisora, fabricanteID, txPower)**: constructor.
* **encenderEmisora()**: inicializa la emisora BLE (función destacada).
* **detenerAnuncio()**: detiene cualquier anuncio en curso (función destacada).
* **emitirAnuncioIBeacon(uuid, major, minor, rssi)**: emite un anuncio iBeacon con formato estándar (UUID, major, minor, TxPower) (función destacada).
* **emitirAnuncioIBeaconLibre(carga, tamaño**): permite enviar libremente 21 bytes de datos en el anuncio iBeacon (función destacada).
* **anyadirServicio(servicio)**: añade un servicio BLE a la emisora.
* **anyadirServicioConSusCaracteristicas(...)**: añade servicio y características.
* **getConexion(connHandle)**: devuelve un objeto de conexión BLE activo

Imagen que contiene Diagrama

El contenido generado por IA puede ser incorrecto.

Ilustración 1: Diseño EmisoraBLE

##### Clase LED

Responsable de controlar el LED de la placa.

* **LED (int número)**: constructor que configura el pin del LED.
* **encender ()**: activa el LED.
* **apagar ()**: desactiva el LED.
* **alternar ()**: cambia el estado actual (si estaba encendido, lo apaga y viceversa).
* **brillar (long tiempo)**: enciende el LED durante un tiempo determinado y lo apaga (función destacada).

Diagrama

El contenido generado por IA puede ser incorrecto.

Ilustración 2: Diseño LED

##### Clase Medidor

Encargada de simular la medición de valores ambientales (CO₂ y temperatura).

* **Medidor ()**: constructor.
* **iniciarMedidor()**: prepara el medidor (de momento no hace nada en el sprint 0).
* **medirCO2()**: devuelve un valor entero que representa el CO₂ medido (función destacada).
* **medirTemperatura()**: devuelve un valor entero que representa la temperatura medida (función destacada).

Imagen que contiene Gráfico

El contenido generado por IA puede ser incorrecto.

Ilustración 3: Diseño Medidor

##### Clase Publicador

Se encarga de empaquetar los valores medidos y transmitirlos mediante la emisora BLE.

* **Publicador ()**: constructor.
* **encenderEmisora()**: inicializa la emisora.
* **publicarCO2(valorCO2, contador, tiempoEspera)**: empaqueta y envía un anuncio con el valor de CO₂ medido (función destacada).
* **publicarTemperatura(valorTemperatura, contador, tiempoEspera)**: empaqueta y envía un anuncio con el valor de temperatura medido (función destacada).

Diagrama

El contenido generado por IA puede ser incorrecto.

Ilustración 4: Diseño Publicador

##### Clase Puerto Serie

Permite comunicarse con el ordenador a través del puerto serie, útil para depuración.

* **PuertoSerie(long baudios)**: constructor que inicializa la comunicación a la velocidad indicada.
* **esperarDisponible()**: bloquea la ejecución hasta que el puerto serie esté listo (función destacada).
* **escribir (T mensaje)**: imprime un mensaje en el puerto serie (función destacada).

Diagrama

El contenido generado por IA puede ser incorrecto.

Ilustración 5: Diseño PuertoSerie

##### Clase ServicioEnEmisora

Permite implementar un servicio BLE completo, con UUID propio y características asociadas, posee una clase interna Caracteristica.

* **ServicioEnEmisora(nombreServicio)**: constructor que crea un servicio con UUID.
* **escribeUUID ()**: muestra el UUID del servicio.
* **anyadirCaracteristica(caracteristica)**: añade una característica al servicio.
* **activarServicio ()**: activa el servicio y todas sus características asociadas (función destacada).
* **Operador BLEService&**: permite usar la clase como un servicio BLE nativo.

***Clase interna Caracteristica (dentro de ServicioEnEmisora)***

Representa un valor concreto dentro de un servicio BLE.

* **Caracteristica(nombreCaracteristica)**: constructor que crea la característica.
* **asignarPropiedadesPermisosYTamanyoDatos (props, permisos, tam)**: configura la característica.
* **escribirDatos(str)**: escribe datos en la característica (función destacada).
* **notificarDatos(str)**: notifica datos al cliente conectado (función destacada).
* **instalarCallbackCaracteristicaEscrita(cb)**: asocia un callback para detectar escrituras en la característica (función destacada).
* **activar ()**: activa la característica (función destacada).

Diagrama

El contenido generado por IA puede ser incorrecto.

Ilustración 6: Diseño ServicioEnEmisora