Documento de diseño

PROYECTO DE APLICACIONES DE BIOMETRÍA Y MEDIO AMBIENTE

Sprint #0

Autor: Santiago Fuenmayor Ruiz

26 de septiembre de 2025

Tabla de contenido

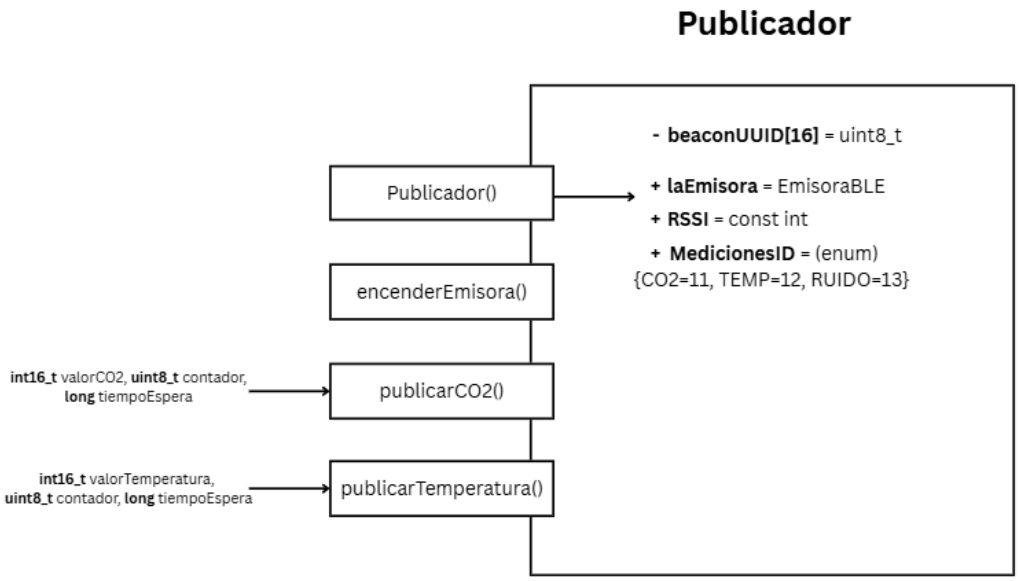
ARDUINO

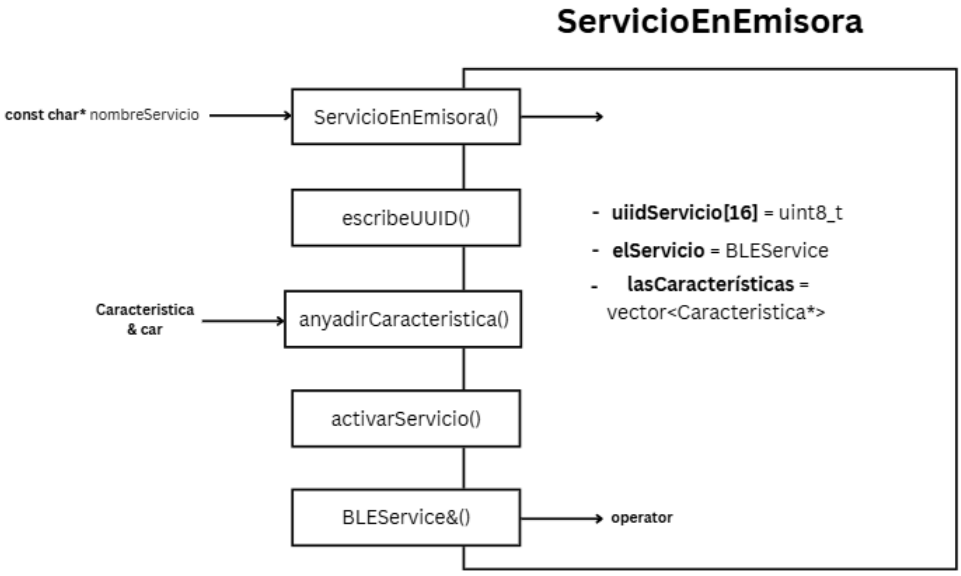
En la parte de Arduino, el código implementa el nodo sensor encargado de simular la lectura de parámetros ambientales y publicarlos mediante Bluetooth Low Energy (BLE) en formato iBeacon. Para ello se estructura en distintas clases con funciones bien definidas:

Medidor se encarga de obtener valores de CO_2 y temperatura (actualmente simulados), **Publicador** gestiona la creación de los paquetes iBeacon y su emisión, y **EmisoraBLE** abstrae la configuración del módulo BLE para encenderlo, detenerlo o iniciar la transmisión. Además, se incluyen clases de apoyo como **PuertoSerie**, utilizada para depuración a través del monitor serie, **LED**, que actúa como indicador visual del envío de datos, y **ServicioEnEmisora**, pensada para manejar servicios y características BLE en futuras ampliaciones.

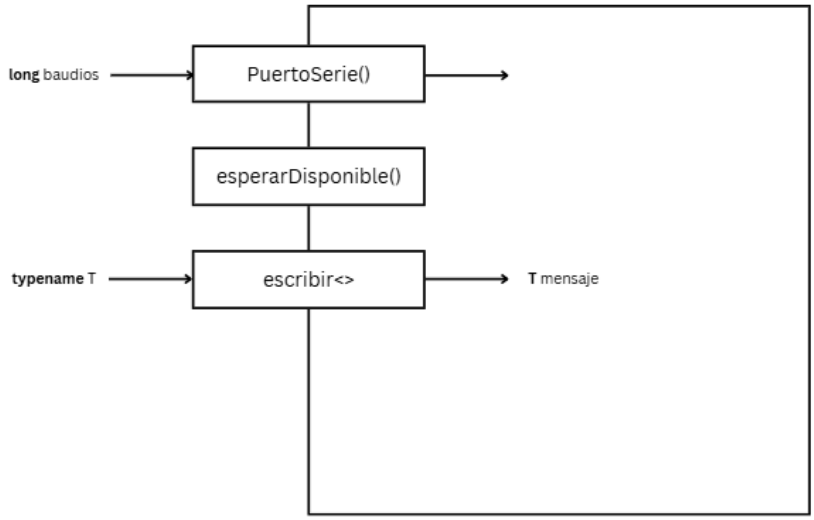
El flujo básico del sistema consiste en inicializar la emisora y el puerto serie, simular las medidas, empaquetarlas en un anuncio iBeacon y transmitirlas durante un tiempo definido, con la posibilidad de indicar el proceso mediante un parpadeo del LED. De este modo, el Arduino actúa como un beacon emisor de datos ambientales, que posteriormente son detectados por la aplicación móvil y enviados al servidor para su almacenamiento y consulta.

INGENIERÍA INVERSA

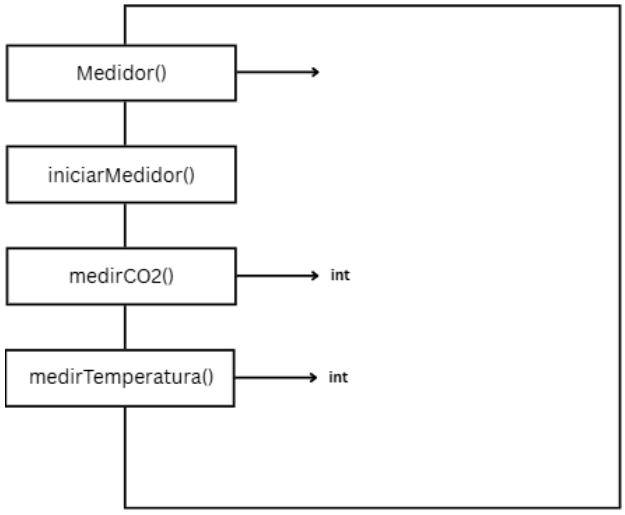




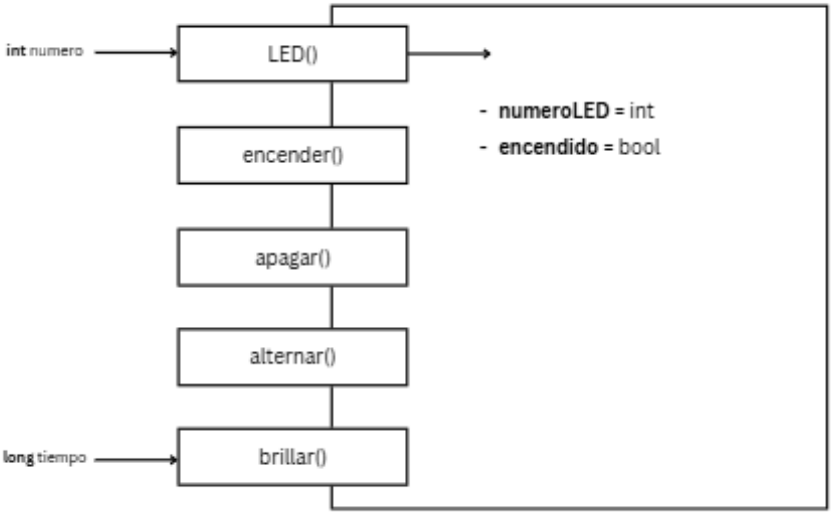
PuertoSerie



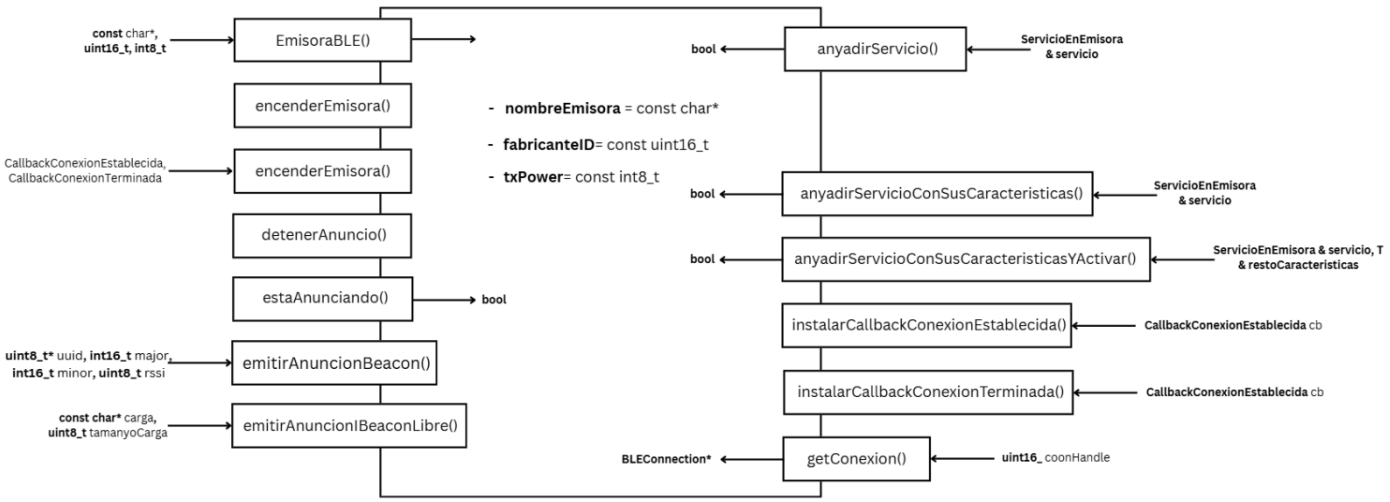
Medidor



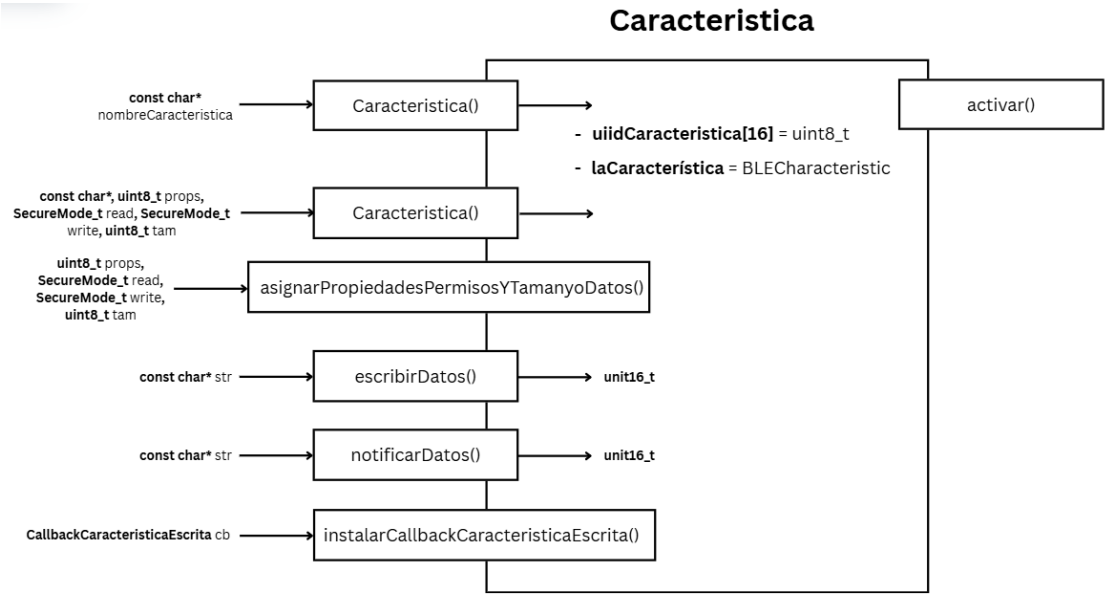
LED



EmisoraBLE



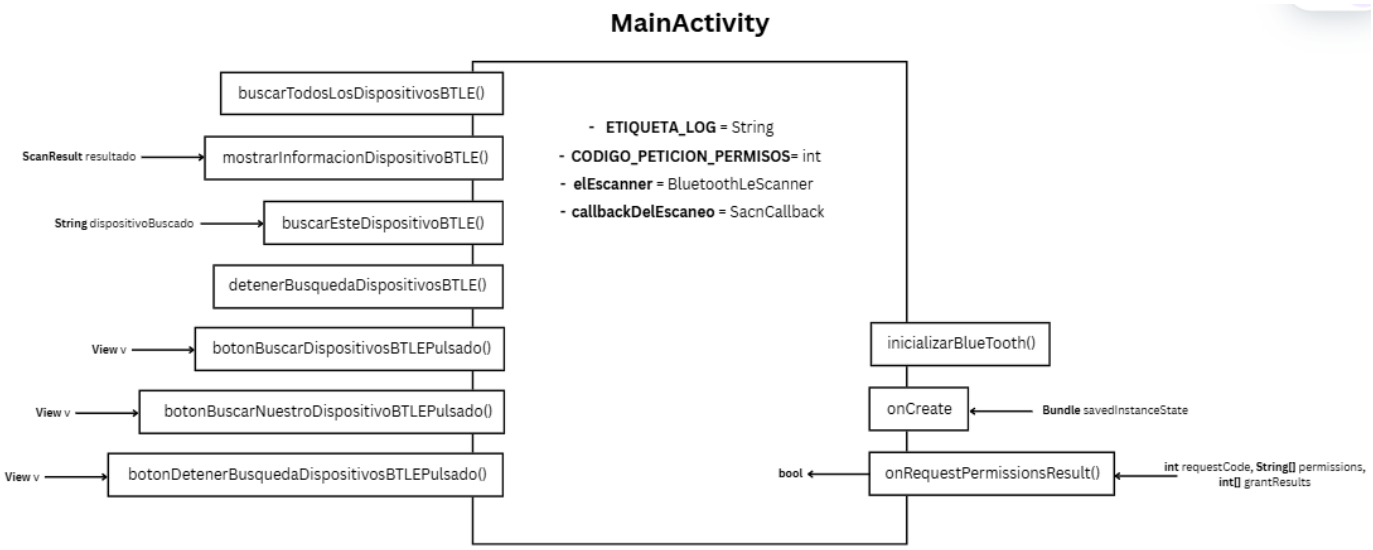
Caracteristica

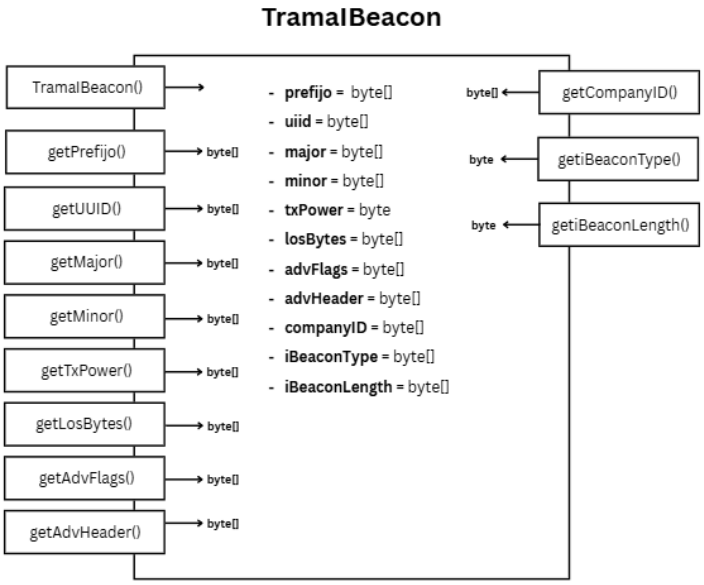


TELÉFONO

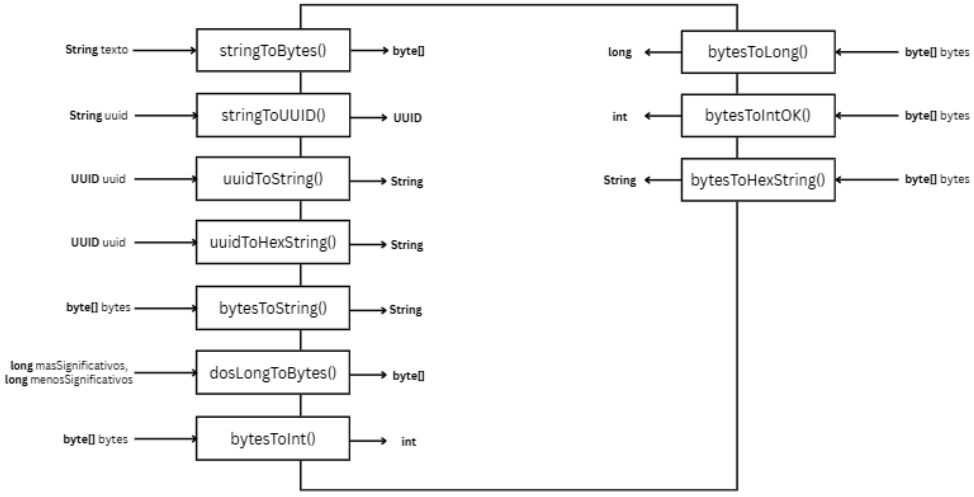
En la parte de **Android**, la aplicación funciona como receptor de los anuncios iBeacon enviados por el Arduino, interpretando las tramas y mostrando los valores medidos. Se apoya en las clases MainActivity para la gestión de la interfaz y detección, TramaIBeacon para decodificar los paquetes recibidos y Utilidades para funciones auxiliares, mientras que el AndroidManifest.xml define los permisos de Bluetooth y localización. En este Sprint 0 su funcionamiento es básico o simulado, mostrando cómo se prepararían los datos para su posterior envío al servidor.

INGENIERIA INVERSA





Utilidades


String texto
stringToBytes()
byte[]
String uuid
stringToUUID()
UUID
UUID uuid
uuidToString()
String
UUID uuid
uuidToHexString()
String
byte[] bytes
bytesToString()
String
long masSignificativos,
long menosSignificativos
dosLongToBytes()
byte[]
byte[] bytes
bytesToInt()
int
long
bytesToLong()
byte[] bytes
int
bytesToIntOK()
byte[] bytes
String
bytesToHexString()
byte[] bytes