

Enviar hola por Bluetooth

Enlaces interesantes sobre el tema

- <https://www.jesusninoc.com/08/27/obtener-un-listado-en-python-de-los-dispositivos-bluetooth-cercanos>
- <https://www.jesusninoc.com/08/27/escanear-dispositivos-bluetooth-y-caracteristicas-escribibles-desde-python>
- <https://www.jesusninoc.com/08/27/escanear-dispositivos-bluetooth-caracteristicas-escribibles-y-el-atributo-handle-de-cada-caracteristicas-desde-python/>
- <https://www.jesusninoc.com/08/27/crear-una-aplicacion-para-iphone-que-permita-recibir-mensajes-por-bluetooth/>

Código para enviar un mensaje a los dispositivos Bluetooth con características escribibles desde Python

[crayon-6a9d55b0a17fc440862963/]

Explicación línea por línea del código

```
import asyncio
from bleak import BleakScanner, BleakClient
```

- **import asyncio:** Importa el módulo asyncio, que proporciona una infraestructura para escribir código asíncrono en Python. Esto incluye corutinas y bucles de eventos para ejecutar operaciones de manera no

bloqueante.

- **from bleak import BleakScanner, BleakClient:** Importa BleakScanner y BleakClient desde la biblioteca bleak. BleakScanner se usa para escanear dispositivos Bluetooth Low Energy (BLE), y BleakClient se usa para conectarse a esos dispositivos.

```
# Reemplaza esto con el mensaje que deseas enviar  
MESSAGE = "Hola"
```

- **MESSAGE = "Hola":** Define una variable MESSAGE que contiene el texto «Hola». Este mensaje se enviará a las características escribibles de los dispositivos BLE.

```
async def scan_for_devices():  
    print("Escaneando dispositivos BLE...")  
    devices = await BleakScanner.discover()  
    return devices
```

- **async def scan_for_devices():** Define una corutina asincrónica llamada scan_for_devices. Las corutinas permiten realizar operaciones de forma asincrónica utilizando await.
- **print("Escaneando dispositivos BLE..."):** Imprime un mensaje en la consola indicando que el escaneo de dispositivos BLE ha comenzado.
- **devices = await BleakScanner.discover():** Llama a BleakScanner.discover(), una función asincrónica que escanea los dispositivos BLE cercanos y devuelve una lista de ellos. await espera a que esta operación se complete antes de continuar.
- **return devices:** Devuelve la lista de dispositivos BLE encontrados.

```
async def connect_and_send_message(device, message):
```

```

try:
    async with BleakClient(device) as client:
        print(f"Conectado a {device.name} ({device.address})")
        try:
            services = await client.get_services()
            for service in services:
                print(f"Servicio: {service.uuid}")
                for characteristic in service.characteristics:
                    print(f" Característica: {characteristic.uuid} - Propiedades:
                    {characteristic.properties}")
                    if "write" in characteristic.properties:
                        try:
                            # Utilizar el handle en lugar del UUID
                            handle = characteristic.handle
                            await client.write_gatt_char(handle, message.encode('utf-8'))
                            print(f"Mensaje '{message}' enviado a la característica
                            {characteristic.uuid}.")
                        except Exception as e:
                            print(f"Error al enviar mensaje a la característica
                            {characteristic.uuid}: {e}")
                        except Exception as e:
                            print(f"Error al obtener servicios: {e}")
                        except Exception as e:
                            print(f"Error al conectar con el dispositivo {device.name}:
                            {e}")

```

- **async def connect_and_send_message(device, message):**
Define una corutina asincrónica llamada `connect_and_send_message` que se conecta a un dispositivo BLE y envía un mensaje a sus características escribibles.
- **try::** Inicia un bloque try para manejar posibles errores durante la conexión y el envío de mensajes.
- **async with BleakClient(device) as client::** Usa `BleakClient` para establecer una conexión con el dispositivo BLE especificado. `async with` asegura que la

conexión se maneje correctamente y se cierre al finalizar el bloque.

- **print(f"Conectado a {device.name} ({device.address})"):** Imprime un mensaje confirmando la conexión exitosa al dispositivo, mostrando su nombre y dirección.
- **try::** Inicia un bloque try anidado para manejar errores al obtener los servicios del dispositivo.
- **services = await client.get_services():** Obtiene los servicios del dispositivo de manera asincrónica. await espera a que esta operación termine.
- **for service in services::** Itera sobre cada servicio del dispositivo.
- **except Exception as e::** Maneja cualquier excepción que ocurra al conectar con el dispositivo.

```
async def main():
```

```
    devices = await scan_for_devices()
```

```
    if not devices:
```

```
        print("No se encontraron dispositivos BLE.")
```

```
        return
```

```
    for device in devices:
```

```
        if device.name: # Verificar que el dispositivo tenga un nombre
```

```
            print(f"Dispositivo encontrado: {device.name} - {device.address}")
```

```
            await connect_and_send_message(device, MESSAGE)
```

```
        else:
```

```
            print(f"Dispositivo encontrado sin nombre: {device.address}")
```

- **async def main():** Define la corutina principal main que coordina el escaneo y el envío de mensajes.
- **devices = await scan_for_devices():** Llama a la corutina scan_for_devices para obtener la lista de dispositivos BLE cercanos.
- **if not devices::** Verifica si la lista de dispositivos está vacía.

- **for device in devices::** Itera sobre cada dispositivo en la lista.

```
# Ejecutar el escaneo y el envío de mensajes  
asyncio.run(main())
```

- **asyncio.run(main()):** Ejecuta la corutina principal main utilizando el bucle de eventos de asyncio. Este método crea un nuevo bucle de eventos, ejecuta la corutina main y cierra el bucle al finalizar, gestionando todo el flujo de ejecución asíncronica.

Resultado del envío

```
Servicio: d0611e78-bbb4-4591-a5f8-487910ae4366  
Característica: 8667556c-9a37-4c91-84ed-54ee27d90049 - Propiedades: ['write', 'notify', 'extended-properties']  
Mensaje 'Hola' enviado a la característica 8667556c-9a37-4c91-84ed-54ee27d90049.
```

19:32



Mensaje recibido: Hola