Escanear dispositivos Bluetooth y características escribibles desde Python

Resultado de la ejecución (servicio y caracterísica)

Código

[crayon-6a9d495bb4013209721486/]

Explicación detallada de cada parte del código

Importaciones

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

- **import asyncio**: Importa el módulo asyncio, que es una biblioteca estándar en Python para escribir código concurrente utilizando corutinas y eventos. Permite la ejecución de operaciones asincrónicas de manera eficiente.
- **from bleak import BleakScanner, BleakClient**: Importa las clases BleakScanner y BleakClient del paquete bleak. Estas clases permiten escanear dispositivos Bluetooth Low Energy (BLE) y conectarse a ellos, respectivamente.

Función para escanear dispositivos

```
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 son funciones que pueden ser pausadas y reanudadas, permitiendo operaciones asincrónicas.
- **print("Escaneando dispositivos BLE...")**: Imprime un mensaje en la consola para indicar que se está iniciando el escaneo de dispositivos BLE.
- **devices = await BleakScanner.discover()**: Llama a BleakScanner.discover(), que es una función asincrónica que realiza el escaneo y devuelve una lista de dispositivos BLE encontrados. await se usa para esperar a que esta operación termine antes de continuar.
- **return devices**: Devuelve la lista de dispositivos BLE encontrados.

Función para listar características escribibles

```
async def list_writable_characteristics(device):
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:
if "write" in characteristic.properties:
print(f" Característica escribible: {characteristic.uuid} -
```

```
Propiedades: {characteristic.properties}")
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 list_writable_characteristics(device)**: Define una corutina asincrónica para conectar a un dispositivo BLE y listar sus características escribibles.
- **try**: Inicia un bloque try para manejar posibles excepciones que puedan surgir durante la conexión y obtención de servicios.
- **async with BleakClient(device) as client**: Usa BleakClient para conectar al dispositivo BLE especificado. async with asegura que la conexión se maneje correctamente y se cierre al final del bloque.
- **print(f"Conectado a {device.name} ({device.address})")**: Imprime un mensaje indicando que se ha conectado exitosamente al dispositivo, mostrando su nombre y dirección.
- **services = await client.get_services()**: Obtiene los servicios del dispositivo BLE de manera asincrónica. await espera a que la operación termine.
- **for service in services:**: Itera sobre cada servicio disponible en el dispositivo.
- **except Exception as e**: Maneja cualquier excepción que pueda ocurrir durante la conexión o la obtención de servicios.
- **except Exception as e**: Maneja cualquier excepción que pueda ocurrir durante la conexión al dispositivo.

Función principal

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

- **async def main()**: Define la corutina principal main, que coordina el escaneo de dispositivos y la lista de características escribibles.
- **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 no se encontraron dispositivos.
- **for device in devices:**: Itera sobre cada dispositivo encontrado.

Ejecución del programa

```
asyncio.run(main())
```

- **asyncio.run(main())**: Ejecuta la corutina principal main utilizando el bucle de eventos de asyncio. Este método configura el entorno necesario para que las corutinas se ejecuten y se encargue de la ejecución y finalización del bucle de eventos.

Ejemplo de servicio y característica configurado en una aplicación para iPhone (Swift)

[crayon-6a9d495bb4019796342478/]