SIC Lab
Exploring Bluetooth LE

version 1.0

Authors: André Zúquete, Vitor Cunha

Version log:

- 1.0: Initial version

1 Introduction

In a previous practical lecture we have introduced and explored BlueZ, the default Bluetooth stack built into the Linux Kernel, and the PyBluez module to program classic Bluetooth applications. In this guide we will develop services using the new Bluetooth Low-Energy (BLE) protocol, which introduces some useful skills for the project.

2 GATT Manager

In BLE there a component called GATT Manager that allows local, external applications to register services and profiles.

Registering a service profile allows remote client applications to recognize the service and make several types of interactions with it. This interactions can be read a value, write a value or get a value update notification. A notification can send just an update warning or the updated value.

A GATT service is represented by a D-Bus object hierarchy where the root node corresponds to a Service and the child nodes represent Characteristics and Descriptors that belong to that Service. Each node must implement one of org.bluez.GattService (for a Service), org.bluez.GattCharacteristic (for a Characteristic) or org.bluez.GattDescriptor (for a Descriptor) interfaces. Each node must also implement the standard D-Bus Properties interface to expose their properties. These objects collectively represent a GATT service definition.

To make service registration simple, bluetoothd requires that all objects that belong to a GATT service be grouped under a D-Bus Object Manager that solely manages the objects of that service. Hence, the standard DBus.ObjectManager interface must be available on the root service path.

2.1 Application registration

An application object is registered with the 'org.bluez' service using its 'org.bluez.GattManager1' interface. The application object is represented by an arbitrary path followed by '/hciX', where this last component represents the local BLE device device (hci0, hci1, etc.). The 'org.bluez.GattManager1' interface allows to register and unregister an application.

An application must implement the 'org.freedesktop.DBus.ObjectManager' interface. The method 'GetManagedObjects' of this interface is called upon the application registration to provide all the Services, Characteristics and Descriptors provided by the application. Each of these have a path (for their identification) and a set of properties.

2.2 Service

A Service aggregates a set of related functionalities; each of these are represented by a Characteristic. Typically, a Characteristic is a value that can be read or written. Also, the update of this value can be notified to subscribing clients to minimize useless reading operations by them.

Each Service has a UUID that is used to recognise its functionalities. There are standard UUID for well-known services, but new UUIDs can be used for publishing new services.

Services and Characteristics' UUIds are 128-bit values written as 00000000-0000-1000-0000-000000000000 where each digit is a hexadecimal number. Standard UUID have a set of fixed low-order bits: 0000XXXX-0000-1000-8000-00805f9b34fb.

2.3 Characteristics

A Characteristic is a value used in a service along with properties and configuration information about how the value is accessed and information about how the value is displayed or represented. In GATT, a characteristic is defined by its characteristic definition. A characteristic definition contains a characteristic declaration, characteristic properties, and a value and may contain descriptors that describe the value or permit configuration of the server with respect to the characteristic.

Each Characteristic possesses a UUID.

The properties of a Characteristic, aka flags, can be a union of the following alternatives:

- Broadcast: it is transmitted in the Server Characteristic Configuration Descriptor;
- Read: it can be read by clients;
- Write without response: it can be written by clients without response;
- Write: it can be written by clients;
- Notify: clients are notified of its update without acknowledgment;
- Indicate: clients are notified of its update with acknowledgment;
- Authenticated Signed Writes: it can be written by clients with signed writes;

2.4 Descriptors

Characteristic Descriptors are used to contain related information about a Characteristic value.

The Characteristic User Description is an optional characteristic descriptor that defines a UTF-8 string of variable size that is a user textual description of the Characteristic Value.

The Characteristic Presentation Format is an optional characteristic descriptor that defines the format of the Characteristic value.

3 Chat server

You have the code of a chat server in the resources distributed with this guide. Read its contents, given the explanations above provided.

4 Chat client

Implement a client for this chat. The client has to search other devices, list the services on each device, and start using the one implementing the chat server (you have to look for the service's UUID). Then, the client may subscribe notification of modifications in the chat's message queue in order to receive messages. Finally, the chat client can read user input and send it to the chat's message queue.

5 Acknowledgements

Authored by André Zúquete and Vítor Cunha

6 Bibliography

- BlueZ
- PyBluez
- pydbus