How to Connect an Android App to a Python Bluetooth Server

**This guide assumes you know the basics of Python programming and Android development.*

Python Bluetooth Server

Firstly you need to install the relevant libraries in order to get setup:

```
sudo aptget install bluez
sudo pip3 install pybluez
```

Once that is done, plug in your USB Bluetooth module.

Run:

```
hciconfig
```

You should see output like this:

```
hci0:   Type: Primary  Bus: USB
        BD Address: 5C:F3:70:94:64:EF  ACL MTU: 1021:8  SCO MTU: 64:1
        UP RUNNING PSCAN
        RX bytes:1534 acl:0 sco:0 events:79 errors:0
        TX bytes:2810 acl:0 sco:0 commands:79 errors:0
```

Next set your adapter to scan mode so it can send and receive messages:

```
sudo hciconfig hci0 piscan
```

Python code example by Albert Huang:

```
#!/usr/bin/env python3
"""PyBluez simple example rfcomm-server.py
Simple demonstration of a server application that uses RFCOMM sockets.
Author: Albert Huang <albert@csail.mit.edu>
$Id: rfcomm-server.py 518 2007-08-10 07:20:07Z albert $
"""

import bluetooth

server_sock = bluetooth.BluetoothSocket(bluetooth.RFCOMM)
server_sock.bind(("", bluetooth.PORT_ANY))
server_sock.listen(1)

port = server_sock.getsockname()[1]

uuid = "94f39d29-7d6d-437d-973b-fba39e49d4ee"

bluetooth.advertise_service(server_sock, "SampleServer", service_id=uuid,
                            service_classes=[uuid, bluetooth.SERIAL_PORT_CLASS],
                            profiles=[bluetooth.SERIAL_PORT_PROFILE],
                            # protocols=[bluetooth.OBEX_UUID]
```

```
                            )

print("Waiting for connection on RFCOMM channel", port)

client_sock, client_info = server_sock.accept()
print("Accepted connection from", client_info)

try:
    while True:
        data = client_sock.recv(1024)
        if not data:
            break
        print("Received", data)
except OSError:
    pass

print("Disconnected.")

client_sock.close()
server_sock.close()
print("All done.")
```

Source: https://github.com/pybluez/pybluez/blob/master/examples/simple/rfcomm-server.py

This is a great baseline for your Bluetooth server.

The uuid is a unique identifier that the client must also use to communicate to the server. Think of it like a passcode.

Android Bluetooth Client

Next we can write a simple Android app for sending messages to the server. You should first build an empty app to get setup.

Declare objects for your Bluetooth adapter, device, and socket:

```
public BluetoothAdapter btAdapter;
public BluetoothDevice btDevice;
public BluetoothSocket btSocket;
```

Declare a String to store your adapter's BD Address that was found in the last section of the guide.

```
public static final String SERVICE_ADDRESS = "5C:F3:70:94:64:EF";
```

Declare a string to store the uuid you will be using for communication. This is the same uuid you set in your Python Bluetooth sever:

```
public static final String SERVICE_ID = "94f39d29-7d6d-437d-973b-fba39e49d4aa";
```

Instantiate your adapter and device object to the adapter's MAC address:

```
btAdapter = BluetoothAdapter.getDefaultAdapter();
btDevice = btAdapter.getRemoteDevice(SERVICE_ADDRESS);
```

Then add this code snippet which handles Android permission for Bluetooth, and starts a thread for connecting if permission is granted:

```
        if (btAdapter == null) {
            Toast.makeText(getApplicationContext(), "Bluetooth not
available", Toast.LENGTH_LONG).show();
        } else {
            if (!btAdapter.isEnabled()) {
                Intent enableIntent = new
Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
                if (ActivityCompat.checkSelfPermission(this,
Manifest.permission.BLUETOOTH_CONNECT) != PackageManager.PERMISSION_GRANTED)
{
                    // TODO: Consider calling
                    //    ActivityCompat#requestPermissions
                    ActivityCompat.requestPermissions(MainActivity.this, new
String[]{Manifest.permission.BLUETOOTH_CONNECT}, 1);
                }
                startActivityForResult(enableIntent, 3);
            } else {
                System.out.println("starting thread");
                ConnectThread connectThread = new ConnectThread(btDevice);
                connectThread.start();
            }
        }
```

Once connected you can create a UI for communication. I suggest starting with a simple EditText and Button for sending the message. Within the button's OnClickListener, you can send a message over bluetooth like this:

```
OutputStream out = btSocket.getOutputStream();
byte[] byteArray;
String msg = "insert message to send here"
byteArray = msg.getBytes();
out.write(byteArray);
```

Troubleshooting

- Your Bluetooth communication was working, but it has stopped. Try the following:
    - Unplug & replug bluetooth adapter
    - Re-run `sudo hciconfig hci0 piscan`
    - Restart your bluetooth server
    - Close and re-open your Android app
- Nothing is showing up under hciconfig
    - Ensure your adapter is plugged in secure
    - Make sure you see it being enumerated by using the command `dmesg`

- Bluetooth server does not receive messages from Android app
    - Make sure both are running the same uuid
    - Make sure Android app has the right MAC address of the bluetooth adapter
    - Make sure Bluetooth is enabled
    - Try troubleshooting steps from first bullet