BLUETOOTH FW FOR SMART SPEAKER DEVICE

I. Requirements

The device SDK runs on Raspberry Pi, Beaglebone. It requires C++11 or later.

II. Dependencies

- This is list of requirement and dependencies for the Device SDK for C++.
  Building with Bluetooth is optional and is currently limited to Raspberry Pi and Beaglebone. `A2DP-SINK`,`A2DP-SOURCE`, `AVRCPTarget` and `AVRCPController` profiles are supported.
  If you choose to build with Bluetooth, these libraries and modules, and their dependencies, must be installed:

Library	**Minimum version**
SBC Library	1.3
Bluez5	5.37 or earlier
Libpulse-dev. Only require if enabling Cmake variable : BLUETOOTH_BLUEZ_PULSEAUDIOINITIALIZER	8.0
Module	**Minimum version**
Pulseaudio	12.2 or earlier
Pulseaudio Bluetooth	
Cmake	3.7

III. Setup Environment on Debian Image

Download the latest firmware for your Beaglebone as website: https://beagleboard.org/latest-images

In case of document, the version is Debian 9.5 2018-10-07 4GB SD IoT

1. **Pulseaudio**

- Installing:
  sudo apt-get update
  sudo apt-get install pulseaudio pulseaudio-module-bluetooth
- Automatically change sound Input Output device:
  Add the following line into our **"/etc/pulse/default.pa"**
  load-module module-switch-on-connect

2. **BlueZ 5**

- Install Dependencies:
  sudo apt-get update

sudo apt-get install -y libusb-dev libdbus-1-dev libglib2.0-dev libudev-dev libical-dev libreadline-dev

- Download, Compile and Install:

  wget http://www.kernel.org/pub/linux/bluetooth/bluez-5.48.tar.xz

  tar xvf bluez-5.48.tar.xz

  cd bluez-5.48

  ./configure --prefix=/usr --mandir=/usr/share/man --sysconfdir=/etc --localstatedir=/var --enable-experimental

  make

  sudo make install

3. **Cmake**

   sudo apt-get install cmake

4. **SBC**

   sudo apt-get install libsbc-dev

IV. Build and run the example

- Install libsoc just for testing:
  - Git clone libsoc from git repository:

    git clone https://github.com/jackmitch/libsoc.git
  - Enter the libsoc.git directory:

    cd libsoc
  - Run autoreconf to generate the libsoc configure scripts

    autoreconf -i
  - Configure the libsoc library with the required features

    ./configure --enable-debug --enable-board=beaglebone_black --with-board-configs
  - Compile and Install

    make && sudo make install
- Git clone source code from git repository:

  git clone https://github.com/olli-ai/omni-device-sdk.git
- Go to direction of example

  cd omni-device-sdk/BluetoothDevice/BlueZ/test/Discoverable
- Create folder for building

mkdir build && cd build

- Use Cmake to generate Makefiles

  cmake ..

- Compile the code using make

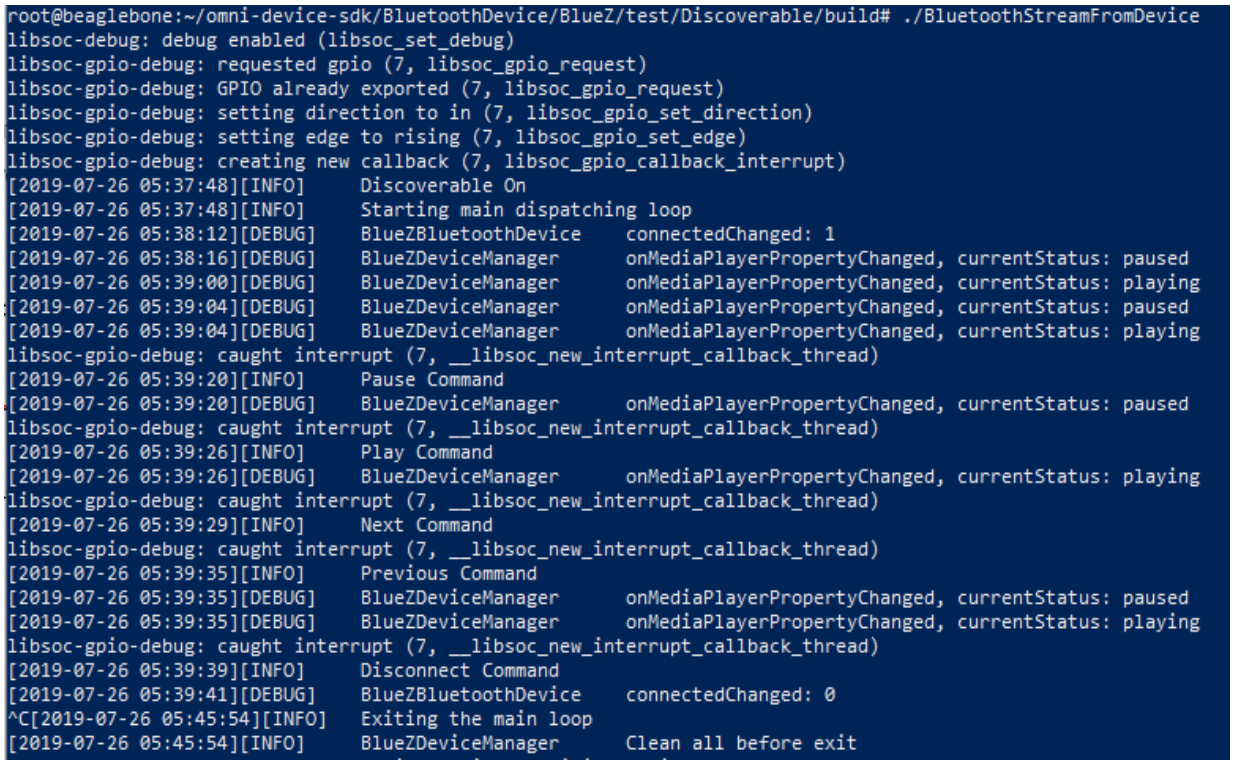
  make

❖ Run the example as 2 cases

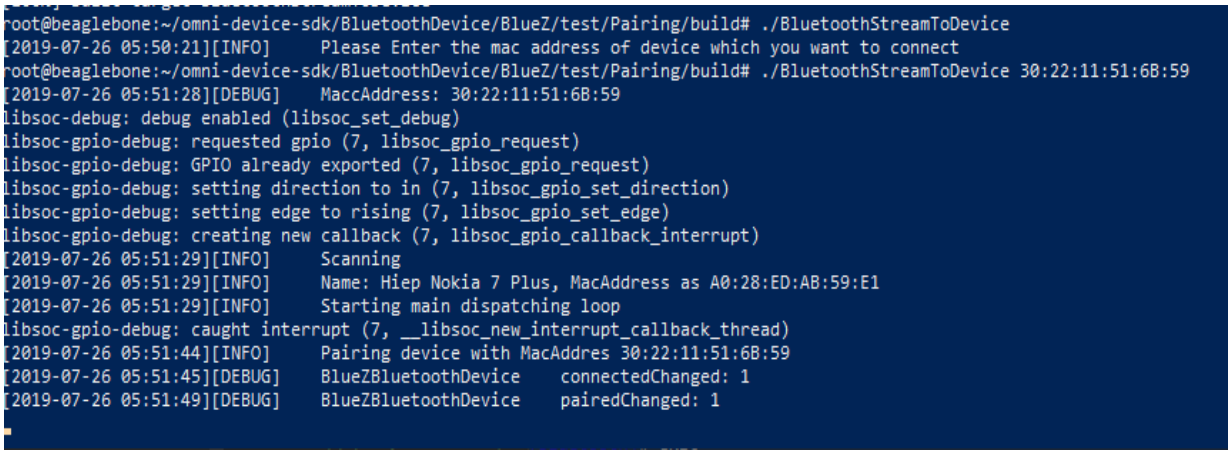
Start the pulseaudio with command:

pulseaudio --start

In discoverable Mode:



And Pairing Mode:

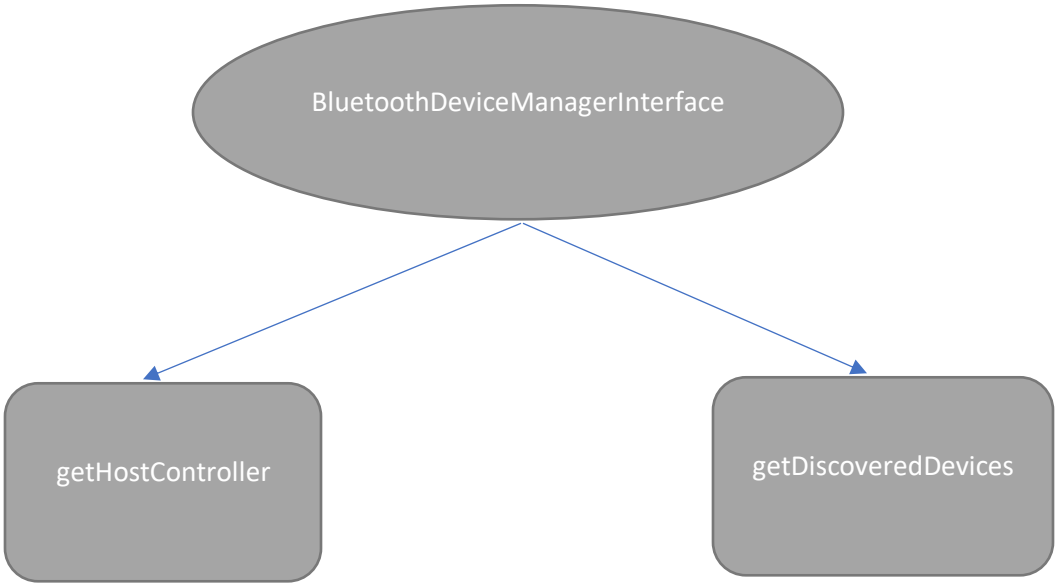


V. Issue

If Pulseaudio can not auto switch on new Audio Output Device. Please follow instructions below:

- Now let's check that A2DP streaming is working. We start by checking that PulseAudio is listing the Bluetooth sound card:
  pacmd list-cards
- The Bluetooth card will be index #1, you can also see the supported profiles (a2dp, hsp, off…). Set A2DP as active profile:
  pacmd set-card-profile bluez_card.xx_xx_xx_xx_xx_xx a2dp_sink
- Set the Bluetooth device as output audio:
  pacmd set-default-sink bluez_sink.xx_xx_xx_xx_xx_xx.a2dp_sink

VI. API



1. HostController:

An interface to represent the HostControllerInterface on the local system.

This is responsible for Scanning and Discovery.

- **getMac**(): @return the MAC address of the adapter.
- **getFriendlyname**(): @return the friendly name of the adapter.
- **isDiscoverable**(): @return the device is current discoverable by other devices. If right, return True. Otherwise, False.
- **enterDiscoverableMode**(): Set the adapter to become discoverable.
  @return True if the operation was successful. Otherwise, False.

- **exitDiscoverableMode**(): Set the adapter become non-discoverable.
  @return True if the operation was successful. Otherwise, False.
- **isScanning**(): Getter for the scanning state of the device. This must wait until any priority startScan and StopScan methods have finished.
  @return the device is currently scanning for other device. If right, return True. Otherwise, False.
- **startScan**(): Set the adapter to start scanning
  @return True if the operation was successful. Otherwise, False.
- **stopScan**(): Set the adapter to stop scanning.
  @return True if the operation was successful. Otherwise, False.

2. **BluetoothDeviceInterface:**
   - **getMac**(): return the MAC address of the Bluetooth Device.
   - **getFriendlyName**(): return the friendly name of the Bluetooth Device.
   - **getDeviceState**(): return the DeviceState of the current device. Like: FOUND, UNPAIRED, PAIRED, IDLE, DISCOVERED, CONNECTED.
   - **isPaired**(): Getter for the paired state of the device. This should return the state after any pending state changes have been resolved.
     @return if the device is paired, return True. Otherwise, False.
   - **pair**(): Initiate a pair with this device.
     @return if the pairing was successful, return True. Otherwise, False.
   - **unpair**(): Initiate an unpair with this device.
     @return if the unpairing was successful, return True. Otherwise, False.
   - **isConnected**(): Getter for the paired state of the device. This should return the state after any pending state changes have been resolved.
     @return if the device is paired, return True. Otherwise, False.
   - **connect**(): Initiate a connect with this device.
     @return if the connecting was successful, return True. Otherwise, False.
   - **disconnect**(): Initiate an disconect with this device.
     @return if the disconnecting was successful, return True. Otherwise, False
   - **getAVRCPTarget**(): Audio/Video Remote Control Profile
     - **play**(): sends a play command to device supporting the AVRCPTarget.
       @return a Boolean indicating the success of the function.

- **pause**(): sends a pause command to device supporting the AVRCPTarget.
  @return a Boolean indicating the success of the function.
- **next**(): sends a next command to device supporting the AVRCPTarget.
  @return a Boolean indicating the success of the function.
- **previous**(): sends a previous command to device supporting the AVRCPTarget.
  @return a Boolean indicating the success of the function.