Bluetooth audio streaming

Post a reply

by **b.ramus** » Thu Feb 06, 2014 8:46 pm
Hi everyone,

I'm new on this forum and I would like to share a little project I made to stream audio content from a bluetooth source (a phone for example) to the raspberry jack output.
I spent a lot of time to get information from different sources and use a part of each to make my raspberry doing what I wanted. I didn't find any complete tutorial to do it and I think it can be interesting for people who want to have the same use of their raspberry PIs.

I recently saw some wireless bluetooth speakers that are able to play music from a remote device using a standard named A2DP (Advanced Audio Distribution Profile). A2DP is a bluetooth profile that many devices support in a native way.
This system is really interesting because you can easily listen music everywhere, just keeping your portable speaker with you. Nevertheless, when your are at home and have a complete audio system (audio amplifier with speakers), it's quite disapointting to listen music on a little portable speaker as you could have a much better sound with your audio system. You have another solution that is to connect your phone with an adapted jack wire but in this case, your phone is plugged in your audio system and you can't freely use it at the same time : it's not very practical...

The advantage of this bluetooth standard comparing to other audio streaming solutions using Upnp protocol for example is that you simply redirect your phone audio output and use what you want to play music. In other terms, your are not dependent of a particular player or application to listen music on your phone : the only condition is that the thing you use makes sound. I was particularly interested in this solution because I use a lot online music providers such as Spotify or Deezer, so I couldn't easily use Upnp solutions because my music is not saved on my phone as classical Mp3 files. With this solution you can play sound from any source on your phone : a particular application, your internet browser, your local Mp3 player, a call you receive...

So let me introduce my solution.

First of all the required things :

- a raspberry Pi (I have a raspberry type B but I think it can also work with a type A)
- an SD card with the lastest raspbian distribution (I use 2013-09-25-wheezy-raspbian)
- a compatible USB bluetooth adapter (I bought one very simple from the french provider "Boulanger" which perfectly works whithout any additionnal driver. The internal bluetooth chip is made by Cambridge Silicon Radio. A list of compatible hardware is available at the following address : http://elinux.org/RPi_VerifiedPeripherals)

- a bluetooth audio device which supports the A2DP audio profile. I personnaly use my phone to stream some music : a samsung galaxy s4 which runs android 4.3. I think that most of android phones support it. I haven't tried with apple devices or Windows phones but it probably works. Even some older phones should work.
- an headset or a jack wire (connection to an audio amplifier input) to get sound from the raspberry jack ouput
- some basic knowledge of linux (use of terminal commands, edit files)

And... some good patience !

Remark : if you already have others USB devices connected to your raspberry, you can use an external powered usb hub. It can solve some problems if your devices require important power to work : the power that the raspberry can provide your usb devices with is limited. If you only use a bluetooth adapter, it is not necessary.

I assume that you have a ready-to-use raspberry pi and you know how to access it (with a screen, keyboard and mouse or directly from another computer thanks to an ssh connection). If not, there are a lot of good tutorials on the internet and I'm sure you will manage to do it (Google is your friend).
If you already have a raspbian distribution on your raspberry and you customized it with a lot of personnal settings, I suggest you to re-install a clean version : it will prevents you from facing some addtionnal configuration problems (you can save your current image before trying it).

So the first thing you have to do is to plug your bluetooth USB adapter and to switch on your raspberry. Your adapter should begin to blink if it is equiped with an activity LED.

Then, you have to log on your raspberry as the default "pi" user and to open a terminal (if you use a ssh connection you are already on a terminal window) : all will be done from it !

1. Install required packages

Update your repositories list to make sure you will find all the required packages :

Code: Select all

```
sudo apt-get update
```

We will use in this solution a particular linux component called pulse-audio which is responsible for the sound management and another one called bluez for the bluetooth management.
Download and install the bluez component, the pulse-audio bluetooth module and some other required dependencies :

Code: Select all

```
sudo apt-get install bluez pulseaudio-module-bluetooth python-gobject python-gobject-2
```

2. Change audio default settings

Add the default user "pi" to the lp group (you will be able to see bluetooth sources and to change some bluetooth settings) :

Code: Select all

```
sudo usermod -a -G lp pi
```

Change the default bluetooth audio settings :

Code: Select all

```
sudo nano /etc/bluetooth/audio.conf
```

Add/Complete the following line in the [General] section :

Code: Select all

```
Enable=Source,Sink,Media,Socket
```

Change the resampling algorithm from the pulse-audio configuration file :

Code: Select all

```
sudo nano /etc/pulse/daemon.conf
```

Add the following line after the commented example ";resample-method = speex-float-3" (don't use the ";" symbol which indactes a commented setting) :

Code: Select all

```
resample-method = trivial
```

Reboot your raspberry to apply settings :

Code: Select all

```
sudo reboot
```

3. Make sure your bluetooth adapter is working

Display the current bluetooth configuration :

Code: Select all

```
sudo hciconfig -a
```

You should see your bluetooth usb adapter reference on the hci0 interface as following :

Code: Select all

```
   hci0:   Type: BR/EDR  Bus: USB
   BD Address: XX:XX:XX:XX:XX:XX  ACL MTU: 192:8  SCO MTU: 64:8
   UP RUNNING PSCAN
   RX bytes:2035 acl:0 sco:0 events:119 errors:0
   TX bytes:467 acl:0 sco:0 commands:50 errors:0
   Features: 0xff 0xff 0x8f 0xf8 0x18 0x18 0x00 0x80
   Packet type: DM1 DM3 DM5 DH1 DH3 DH5 HV1 HV2 HV3
   Link policy: RSWITCH HOLD SNIFF PARK
   Link mode: SLAVE ACCEPT
   Name: 'raspberryMedia'
   Class: 0x4e0100
   Service Classes: Networking, Rendering, Capturing, Telephony
   Device Class: Computer, Uncategorized
   HCI Version: 1.2 (0x2)  Revision: 0x7c5
   LMP Version: 1.2 (0x2)  Subversion: 0x7c5
   Manufacturer: Cambridge Silicon Radio (10)
```

You can see the bluetooth address of your USB adapter on the line BD Address: XX:XX:XX:XX:XX:XX

You can change the name of your USB bluetooth adapter as it will be visible by remote devices from the following file, where XX:XX:XX:XX:XX:XX represents the bluetooth address of your USB adapter :

Code: Select all

```
sudo nano /var/lib/bluetooth/XX:XX:XX:XX:XX:XX/config
```

Remark : if you change your adapter name, it will be active on next reboot

Another configuration file can be interesting :

Code: Select all

```
sudo nano /etc/bluetooth/main.conf
```

You can use some other useful bluetooth commands to make sure it is working with your phone (or other device) :

Enable the bluetooth connection on your phone (or other device), and activate the public visibilty

Launch a bluetooth scan on your raspberry :

Code: Select all

```
sudo hcitool scan
```

You should see your phone with its bluetooth address under the form XX:XX:XX:XX:XX:XX

Try to send a bluetooth ping to your phone, where XX:XX:XX:XX:XX:XX represents your phone bluetooth address :

Code: Select all

```
sudo l2ping XX:XX:XX:XX:XX:XX
```

Try to open a connection (XX:XX:XX:XX:XX:XX represents your phone bluetooth address) :

Code: Select all

```
sudo hcitool cc XX:XX:XX:XX:XX:XX
```

4. Pair your phone and your raspberry

Activate the ISCAN service on your raspberry to make it visible from other devices :

Code: Select all

```
sudo hciconfig hci0 piscan
```

You should now see the following line on your current bluetooth configuration (sudo hciconfig -a) :

Code: Select all

```
UP RUNNING PSCAN ISCAN
```

Launch the following service to begin the pairing step (just type it as a command and press enter) :

Code: Select all

```
bluez-simple-agent
```

Launch a bluetooth scan on your phone and select your raspberry (it should appear as an available bluetooth

audio device in the list). Then, you should be asked for a PIN code : choose the code you want on your phone.

A "Enter Pin :" message should appear on your raspberry asking you for the PIN code : enter the same code you just defined on your phone and press enter. If you try to connect your phone without stopping the bluez-simple-agent, you should be asked to authorize the connection (message "Authorize connection (yes/no):").

Your phone is now paired. You can stop the bluez service by pressing Ctrl+C.

Add your phone to the bluetooth trusted devices (you will not be obligated to enter the PIN code each time you want to connect your phone to your raspberry), where XX:XX:XX:XX:XX:XX represents your phone bluetooth address :

Code: Select all

```
bluez-test-device trusted XX:XX:XX:XX:XX:XX yes
```

5. Connect the bluetooth source to the default sink

Once your phone is connected, you will see it as the bluetooth source n°1 :

Code: Select all

```
pactl list sources short
```

The command result should be (XX:XX:XX:XX:XX:XX represents your phone bluetooth address) :

Code: Select all

```
0	alsa_output.platform-bcm2835_AUD0.0.analog-stereo.monitor	module-alsa-card.c	s16le 2ch 44100Hz	SUSPENDED
1	bluez_source.XX_XX_XX_XX_XX_XX	module-bluetooth-device.c	s16le 2ch 44100Hz	SUSPENDED
```

Check the available sinks :

Code: Select all

```
pactl list sinks short
```

The command result should be :

Code: Select all

```
0	alsa_output.platform-bcm2835_AUD0.0.analog-stereo	module-alsa-card.c
```

```
  s16le 2ch 44100Hz    SUSPENDED
```

Connect the source to the sink thanks to the loopback module (XX:XX:XX:XX:XX:XX represents your phone bluetooth address) :

Code: Select all

```
pactl load-module module-loopback source=bluez_source.XX_XX_XX_XX_XX_XX
sink=alsa_output.platform-bcm2835_AUD0.0.analog-stereo
```

6. Change the sound settings

Change the default HDMI sound ouput for the jack output :

Code: Select all

```
amixer cset numid=3 1
```

Turn up the sink volume :

Code: Select all

```
   amixer set Master 100%
  pacmd set-sink-volume 0 65537
```

If you play a song on your phone, it should now be redirected on your raspberry jack output. You can check it by plugging an headset to your raspberry. You can also directly connect your raspberry to your audio system though an adapted jack wire (raspberry jack output connected to an input of your audio amplifier).

This system works fine but requires a connection on your raspberry each time you want to listen music : you have to manually launch the command described on step 5 to connect the bluetooth source to the default sink though the loopback module. It means you must have a screen/keyboard/mouse connected or a network connection (ssh access) : it's not much practical, and maybe less than directly connecting your phone to your audio system though a jack wire...

I found that the lastest version of pulse audio (v4.0) includes a new module called "module-bluetooth-policy" that automatically activates the loopback module (connects the source to the sink) on each bluetooth device connection. Nevertheless, the pulse-audio version installed by default on raspbian is v2.0 (the command to get the version is : pulseaudio --version) and doesn't include such a module.

I installed the lastest version on my raspberry (it can be downloaded from here : http://www.freedesktop.org/wiki/Software/PulseAudio/) and tried to get it working with the bluetooth modules but I didn't manage to have something good (I didn't see any bluetooth source when I connected my phone and met other troubles). This is a more complicated job because you have to clone the source code from git repository, to configure makefiles, to resolve dependencies (many other packages to install), to build the

application (compilation) and to deploy it on the right folders (installation).

So I decided to find another solution : a regular job that looks for a bluetooth device connection and automatically launchs the loopback module.

7. Configure pulse-audio in system mode (necessary to launch pulse without being logged on the raspberry)

Activate the pulse-audio system mode and allow module loading :

Code: Select all

```
sudo nano /etc/default/pulseaudio
```

Modify the following lines :

Code: Select all

```
   PULSEAUDIO_SYSTEM_START=1
  DISALLOW_MODULE_LOADING=0
```

Add the default user "pi" to the pulse-access group :

Code: Select all

```
sudo adduser pi pulse-access
```

Modify the pulse-audio client configuration file to deactivate autospawn :

Code: Select all

```
sudo nano /etc/pulse/client.conf
```

Modify the following line :

Code: Select all

```
autospawn = no
```

Modify the pulse-audio daemon configuration file to change the default starting options :

Code: Select all

```
sudo nano /etc/pulse/daemon.conf
```

Modify the following lines :

Code: Select all

```
    allow-module-loading = yes
   load-default-script-file = yes
   default-script-file = /etc/pulse/default.pa
```

Add some bus policy for user "pulse" :

Code: Select all

```
sudo nano /etc/dbus-1/system.d/pulseaudio-system.conf
```

Add the following lines between the <busconfig> tags :

Code: Select all

```
<policy user="pulse">
      <allow own="org.pulseaudio.Server"/>
      <allow send_destination="org.bluez"/>
      <allow send_interface="org.bluez.Manager"/>
</policy>
```

8. Create a job that looks for bluetooth connections

Create a new script file that will be regularly launched to check if a new bluetooth device have been connected and automatically launch the loopback module :

Code: Select all

```
mkdir /home/pi/pulseAudioBluetooth
```

Code: Select all

```
nano /home/pi/pulseAudioBluetooth/checkForBluetoothDevice.sh
```

(path and name of the new script file)

Copy/paste the following code into the file :

Code: Select all

```
#!/bin/bash

#This script looks for a bluetooth audio source and launch the pulse audio loopback module if necessary

#log traces :
#date=$(date "+%Hh%Mm%Ss")
#echo "Script Exec "$date

bluetoothSource=$(pactl list sources short | grep bluez_source)

read loopbackStatus < /home/pi/pulseAudioBluetooth/bluetoothLoopbackStatus.txt

if [[ $bluetoothSource != "" ]] && [[ $loopbackStatus == "0" ]]
then
   source=${bluetoothSource:2:30}
   pactl load-module module-loopback source=$source sink=alsa_output.platform-bcm2835_AUD0.0.analog-stereo
   echo "1" > /home/pi/pulseAudioBluetooth/bluetoothLoopbackStatus.txt
   #echo "command launched"

else
   if [[ $bluetoothSource == "" ]] && [[ $loopbackStatus == "1" ]]
   then
      echo "0" > /home/pi/pulseAudioBluetooth/bluetoothLoopbackStatus.txt
      #echo "reset loopbackStatus"
   fi
fi
```

Save modifications and close the script

Make it executable :

Code: Select all

```
sudo chmod u+x /home/pi/pulseAudioBluetooth/checkForBluetoothDevice.sh
```

Create a new text file to save the loopback status (file used by the script) :

Code: Select all

```
echo 0 > /home/pi/pulseAudioBluetooth/bluetoothLoopbackStatus.txt
```

(0 means that loopback module is not running, 1 means that it's running)

9. Planify a regular launch of the script

To do this, linux has an interessant service called cron that enable to planify scripts launching.

Open the cron configuration :

Code: Select all

```
crontab -e
```

Add the following lines at the end of the configuration file to launch every minute the script and to redirect standard output and errors in a text file named cron.log (you can change the files paths if necessary) :

Code: Select all

```
    # automatically load loopback for new bluetooth devices (check every
minute)
    * * * * * /home/pi/pulseAudioBluetooth/checkForBluetoothDevice.sh  >>
/home/pi/pulseAudioBluetooth/cron.log 2>&1
```

Save the file and exit your text editor

Reboot your raspberry :

Code: Select all

```
sudo reboot
```

At this point, your raspberry is able to automatically detect a bluetooth source and to launch the audio streaming (the source has to be paired with your raspberry - refer to step 4 if you want to add a new source). The sound will not be immediately played as soon as you connect your bluetooth source : in the worst case, you will have to wait for 1 minute (maximum time between 2 detections).

If no sound is played after more than 1 minute, you probably have a problem : check the cron.log file (/home/pi/pulseAudioBluetooth/cron.log by default) to see if something is written in it (redirection of errors).

You can now stream music from your bluetooth device without connecting your raspberry to the network.
If you want to deactivate the automatic detection, juste delete the lines you added in the cron configuration file and reboot.

Enjoy your favorite music !

I hope I'm clear in my explanations and it will be useful to some people. Don't hesitate to give your feedbacks about this tutorial !

I would like to thanks people who spent some time to share their experience about audio streaming and more particularly the authors of these articles :

- http://kmonkey711.blogspot.fr/2012/12/a2dp-audio-on-raspberry-pi.html
- https://wiki.archlinux.org/index.php/PulseAudio

Some others articles that can be interesting :

- http://doc.ubuntu-fr.org/pulseaudio
- http://colin.guthr.ie/2010/09/compiling-and-running-pulseaudio-from-git/

Posts: 3

Joined: Thu Feb 06, 2014 6:41 pm

by **texy** » Thu Feb 06, 2014 9:18 pm
Hi and welcome to the forum
Thanks for sharing and posting on the forum. Plenty of these tutorials on the net, but it's nice to see it in completion on the pi forums.
Now what I'd like to see (as previously mentioned on similar posts here) is a way of the pi capturing the id3 tag info that must be sent from the serving device.....
Now IF you could do that...

Texy
"2.8inch TFT LCD + Touch screen" add-on boards for sale here :
http://www.raspberrypi.org/phpBB3/viewtopic.php?f=93&t=65566
50p goes to the Foundation

Forum Moderator

Posts: 2805

Joined: Sat Mar 03, 2012 10:59 am

Location: Berkshire, England

by **Douglas6** » Sun Feb 09, 2014 12:34 am
B.ramus, you might want to look into a more automatic (and quicker) method of doing the pactl loopback.
There are a couple of ways to do this:
1. Use a udev rule as described in this instructable
2. Use dbus with a python script such as one that can be found here