



A music system for the home

using Raspberry Pi

Introduction

This document describes all the steps required to configure a music system for the home using Raspberry Pi. It was put together by Steve Hawtin to accompany some interactive sessions run by the Basingstoke MakerSpace (www.basingstokemakerspace.org.uk/).

Overview

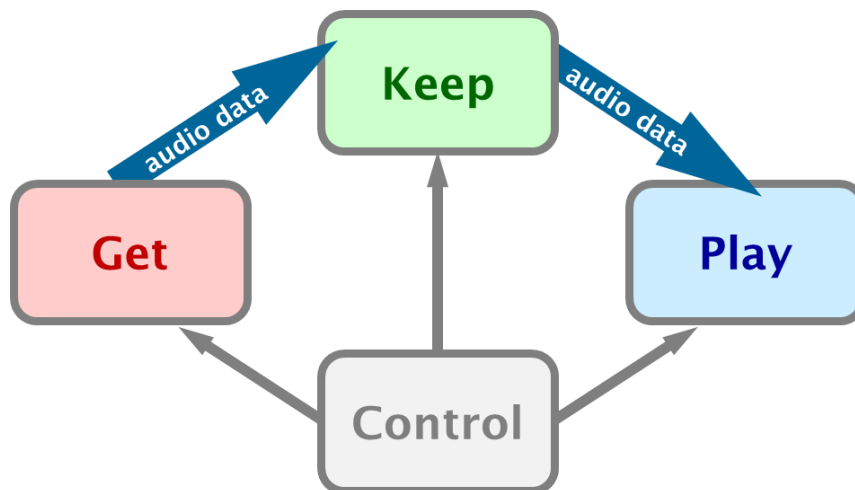


Figure 1 The essential activities when delivering recorded music

Any mechanism for delivering recorded music requires activities that can be assigned to four basic tasks: ways to **get** the audio data; a place to **keep** recorded tracks; a system to **play** the audio as sound; and finally a **control** mechanism to coordinate these activities.

Music at Home with Raspberry Pi

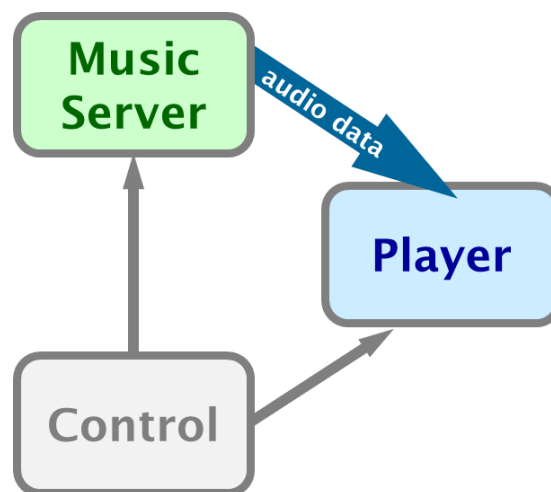


Figure 2 In a house having a music server is one good approach

This short paper describes one way to implement a “Music Server” and a collection of “Players”. That is, a system to hold a collection of music data, and various devices to convert that data into sounds. My own house has used the Logitech Media Server (LMS) to play this role for more than a decade. Originally this was because I had a number of SqueezeBox player appliances and the LMS was the best way to coordinate them. Logitech stopped selling consumer music players some time ago, however the LMS has remained a good option because Logitech made the server software open source and it is now supported by a number of other players.

This paper will show how to employ Raspberry Pi computers to implement the server (and some players). Again there are a number of ways to implement this type of setup, for example the site <https://www.max2play.com/> has some downloads that simplify the installation and management of the software. However this article takes the reader through the key steps so that anyone can do it.

Network setup

LMS runs best if you have a single “server” computer running it and if all the players are on the same network. This is the way a typical house will have its network set up. To be more technical all the devices need to be on the same subnet (often with a netmask of 255.255.255.0). If you have firewalls installed then you will need your local network expert to ensure the messages and data streams are properly forwarded, otherwise the server may not be able to find and control the players.

If all that sounds confusing then you probably have a suitable network setup already.

The configuration steps will be easier if you are able to use wired Ethernet to connect. If you don’t have wires to every room there are a wide range of devices that let you connect to the network via the power supply. If those options don’t work in your situation you will have to setup the Raspberry Pi with WiFi, the steps required to do that are described elsewhere just ask Google to find them.

Music at Home with Raspberry Pi

Installing a music server

The first step is to run the Logitech Media Server on a computer attached to your network. Mostly this is quite straightforward and there are a number of articles on-line that go through the steps. I found the article at:

<http://www.gerrelt.nl/RaspberryPi/wordpress/tutorial-stand-alone-squeezebox-server-and-player-for-bbq/>.

particularly useful.

Configure base system

Go to the site <https://www.raspberrypi.org/downloads/raspbian/> and download the latest raspbian-stretch-lite.img file. Write that image to a suitable SD card. Put the SD card into a Raspberry Pi and boot up the system with a keyboard plugged into the USB, a screen plugged into the HDMI port and your network plugged into the Ethernet port. Log in to the **pi** user (password is **raspberrypi**). Run the system configuration tool:

```
pi@raspberrypi:~$ sudo raspi-config
```

Using that tool

- Change the **pi** account password
- Change the machine's hostname (under **Network Options**)
- Enable **ssh** access (under **Interfacing Options**)

When you quit from that tool don't reboot. Before disconnecting the monitor we need to make a note of the IP address

```
pi@parrot:~$ ifconfig
...
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.186.215 netmask 255.255.255.0 broadcast 192.168.
        inet6 fe80::4bc3:d55a:a85e:99fe prefixlen 64 scopeid 0x20
    ether b8:27:eb:90:25:ec txqueuelen 1000 (Ethernet)
    RX packets 6107731 bytes 194592042 (185.5 MiB)
    RX errors 0 dropped 26 overruns 0 frame 0
    TX packets 3061629 bytes 208222552 (198.5 MiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
...
```



Music at Home with Raspberry Pi

This command outputs a whole load of information, we need the `inet` address of the `eth0` interface. In the example above this is the set of four numbers shown in cyan (`192.168.186.215`). Once we know what the address of this computer will be we can shut it down¹.

```
pi@parrot:~ sudo shutdown -h now
```

The Raspberry Pi can now be moved away from the monitor. Once it is reconnected to the Ethernet it can be rebooted without a screen attached. At this point we can log on to it from any convenient computer (with a monitor) using a secure shell. On my laptop for example I run `cygwin` with `ssh` installed, but you don't need to go to that level of complexity. The `putty` software, for example, can also let you access remote computers across the network.

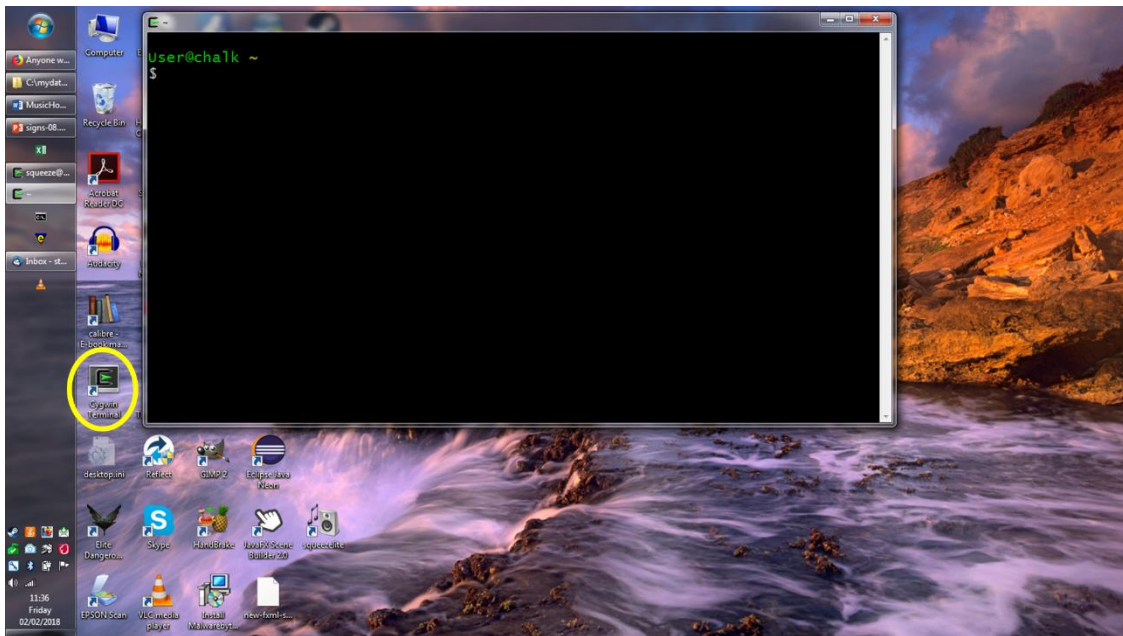


Figure 3 A Cygwin terminal running on a Windows computer

This means that I can remote log in to any computer with a simple command:

```
User@wolf ~  
$ ssh -l pi 192.168.186.215  
...  
pi@parrot:~
```

¹ In fact this relies on the local DHCP assigning leases for a reasonable time, which is the default behaviour. Again if that sounds confusing your setup probably does the right thing, if it doesn't then you'll have to read up on DHCP and talk to your network expert.

Music at Home with Raspberry Pi

Obtain LMS

```
pi@parrot:~ $ sudo apt-get update
...
Reading package lists... Done
pi@parrot:~ $ sudo apt-get upgrade
...
Do you want to continue? [Y/n] y
...
```

Ensure the package system is up to date and all necessary upgrades have been done with the commands above. Before the LMS software can be installed a number of other packages are required.

```
pi@parrot:~ $ sudo apt-get install -y libsox-fmt-all libflac-dev\
libfaad2 libmad0
...
Do you want to continue? [Y/n] y
...
```

The LMS software package is not a standard one, so it has to be obtained in a slightly different way:

```
pi@parrot:~ $ wget -O logitechmediaserver_arm.deb $(wget -q -O - \
"http://www.mysqueezebox.com/update/\
?version=7.9.1&revision=1&geturl=1&os=debarm")
...
pi@parrot:~ $ sudo dpkg -i logitechmediaserver_arm.deb
...
```

And that's it. The LMS server is now installed, next time the computer is rebooted the server should start.

One little patch

Many years ago I found that the LMS software was failing to correctly read cover art from my mp3 files. I worked out what the issue was and submitted a fix but the software maintainers didn't want to apply it. One of the great things about open source software is the fact that anyone can fix it so it works for them. Here are the steps required to apply my patch.

In the file `/usr/share/perl5/Slim/Utils/GDResizer.pm` there is a bug. The code that reads:

```
        if ( $pic->[4] ) { # offset is available
            return ( $pic->[4], $pic->[3] );
        }
```

Causes the server to fail on my system, this should be commented out. But before that take a local copy so we can undo our changes if necessary:

```
pi@parrot:~ $ cd /usr/share/perl5/Slim/Utils
```



Music at Home with Raspberry Pi

```
pi@parrot:~ $ sudo cp GDResizer.pm GDResizer.pm.18jan30
pi@parrot:~ $ sudo vi GDResizer.pm
```

You might prefer to use `nano` as your editor rather than `vi`, just change the second argument in that final command. The three lines shown above should be commented out, once you've done that they should look like:

```
# Commeneted out because it fails for some mp3s
#             if ( $pic->[4] ) { # offset is available
#                 return ( $pic->[4], $pic->[3] );
#             }
```

Overwrite the old file and we're done.

Create a special user

Modern versions of UNIX tend to restrict the use of the `root` user, in Raspbian, for example the user `pi` is used (as we have seen). However even the user `pi` has a bit too much power to be safe. So best practice is to create users that each specialise in certain tasks. We need a user that handles the music, so let's add one:

```
pi@parrot:~ $ sudo adduser music
...
```

Make sure the password is both secure and yet noted somewhere safe.

Add an external disk

In principle we could now run this music server and it would work perfectly well. However there are three reasons why keeping your music on the SD card is a bad idea:

- **Size:** Music collections tend to grow fairly fast, however big your SD card is you are likely to quickly outgrow it
- **Wearout:** SD cards tend to fail if there is repeated updates being applied. For example the files in the `/var/log` directory keep a note of every activity performed on the system, if we leave these on the SD card then it is liable to develop a fault in a few months
- **Reliability:** Finally there is the fact that if we separate the SD card containing the operating system separate from the music files and logs then the overall system will be more reliable. A failure (or stupid user error) will be less likely to make the whole system collapse

For all these reasons it is best to have a separate disk to hold the music data. The easiest way to do this is to have a USB disk. However the Raspberry Pi is a bit sensitive to the amount of electric current it consumes, so the best option is to have a USB disk that doesn't take its power from the Pi. There are three obvious ways to achieve this:

Music at Home with Raspberry Pi

- **Powered disk:** a USB disk with its own mains plug
- **Powered USB Hub:** a USB hub with its own mains plug
- **Split USB cable:** there are some USB disks with two plugs on, one for the data and the other for the power

It doesn't matter which of these three you go for, as long as the hard disk has its own power independent of the Raspberry Pi.

Format the disk

Usually any USB disk you get will be formatted so it works well with Windows. In our case we want to make it work well with Linux instead, so we need to reformat it. Plug your USB disk into the port, you can find out where it is by:

```
pi@parrot:~ $ ls -l /dev/disk/by-uuid/
```

It will point to something like `sda1`. If you are unsure of which one is yours try unplugging the disk, rerunning the command and see which one is no longer there. The various disks will be `sda`, `sdb` etc, numbered entries, such as `sda1` refer to the partitions within those disks.

Be careful to make absolutely sure that you have the correct disk, the following steps will destroy any data on the disk. Check twice that there is nothing on the disk you want to keep *before* reformatting it. The command to format the disk is `fdisk`, run this on the whole disk:

```
pi@parrot:~ $ sudo fdisk /dev/sda
Command (m for help): p
```

Typing `m` will show the available commands. The `p` command lists how the current partitions are laid out. First of all delete all the existing partitions

```
Command (m for help): d
Partition number (1,2, default 2): 2

Partition 2 has been deleted.

Command (m for help): d
Selected partition 1
Partition 1 has been deleted.
```

Then create a new partition (with the `n` command). In this case we want a primary partition with the default number, start and end sector (that will take up the whole disk):

```
Command (m for help): n
Partition type
  p   primary (0 primary, 0 extended, 4 free)
  e   extended (container for logical partitions)
Select (default p): p
Partition number (1-4, default 1):
First sector (2048-488397167, default 2048):
```



Music at Home with Raspberry Pi

```
Last sector, +sectors or +size{K,M,G,T,P} \
      (2048-488397167, default 488397167):

Created a new partition 1 of type 'Linux' and of size 232.9 GiB.
Partition #1 contains a ntfs signature.

Do you want to remove the signature? [Y]es/[N]o: y

The signature will be removed by a write command.
```

Now we set the type of the partition to 83 ('Linux'):

```
Command (m for help): t
Selected partition 1
Partition type (type L to list all types): 83
Changed type of partition 'Linux' to 'Linux'.
```

And then write these changes to the disk:

```
Command (m for help): w
```

Once the partition is defined the file system can be made:

```
pi@parrot:~ $ sudo mkfs.ext4 /dev/sda1
```

Setting the mount point

Now that the disk is formatted it needs to be assigned a location. This is done by editing the file `/etc/fstab`.

```
pi@parrot:~ $ sudo vi /etc/fstab
```

In our case we'll create a path of `/music` so the `fstab` file needs to have this line added:

```
/dev/sda1    /music      ext4        defaults    0          3
```

Then we can make the disk show up at `/music` by “mounting” it:

```
pi@parrot:~ $ mkdir /music
pi@parrot:~ $ sudo mount /music
```

Because this is listed in `/etc/fstab` we only have to run this `mount` command this one time, next time we reboot it will be attached automatically.



Music at Home with Raspberry Pi

Moving the log directory

It has already been mentioned that moving the log directories to the hard disk will help slowdown the wearout. Creating a replacement log directory on the disk (called `var_log`) and linking to it will sort that out:

```
pi@parrot:~ $ cd /music
pi@parrot:~ $ sudo mkdir var_log
pi@parrot:~ $ sudo chown root var_log
pi@parrot:~ $ cd /var
pi@parrot:~ $ sudo mv log log_18jan30
pi@parrot:~ $ sudo ln -s /music/var_log/ log
```

Configure the directories

Finally we want a set of directories that belong to the music user to allow us to store the necessary data:

```
pi@parrot:~ $ cd /music
pi@parrot:~ $ sudo mkdir import
pi@parrot:~ $ sudo mkdir active
pi@parrot:~ $ sudo mkdir playlists
pi@parrot:~ $ sudo chown music import active playlists
pi@parrot:~ $ sudo chmod 755 import active playlists
```

Once the server has all the disks configured LMS can be started. The easiest way to do that is just to reboot the computer.

```
pi@parrot:~ $ sudo shutdown -r now
```



Music at Home with Raspberry Pi

Controlling the server

LMS software can be controlled from a phone (via apps like Orange Squeeze or Squeezer) or from a desktop via the web interface. In this case the web interface will be used. By default this is to be found on port 9000 of the server. The first time that the server starts it will take quite a while sorting itself out before it is ready. The `top` command (on the Raspberry Pi) will allow you to see this activity.

Once the server is ready use a browser (on any computer that is on the same subnet) and navigate to the 9000 port. In our case above this would be `http://192.168.186.215:9000/`.

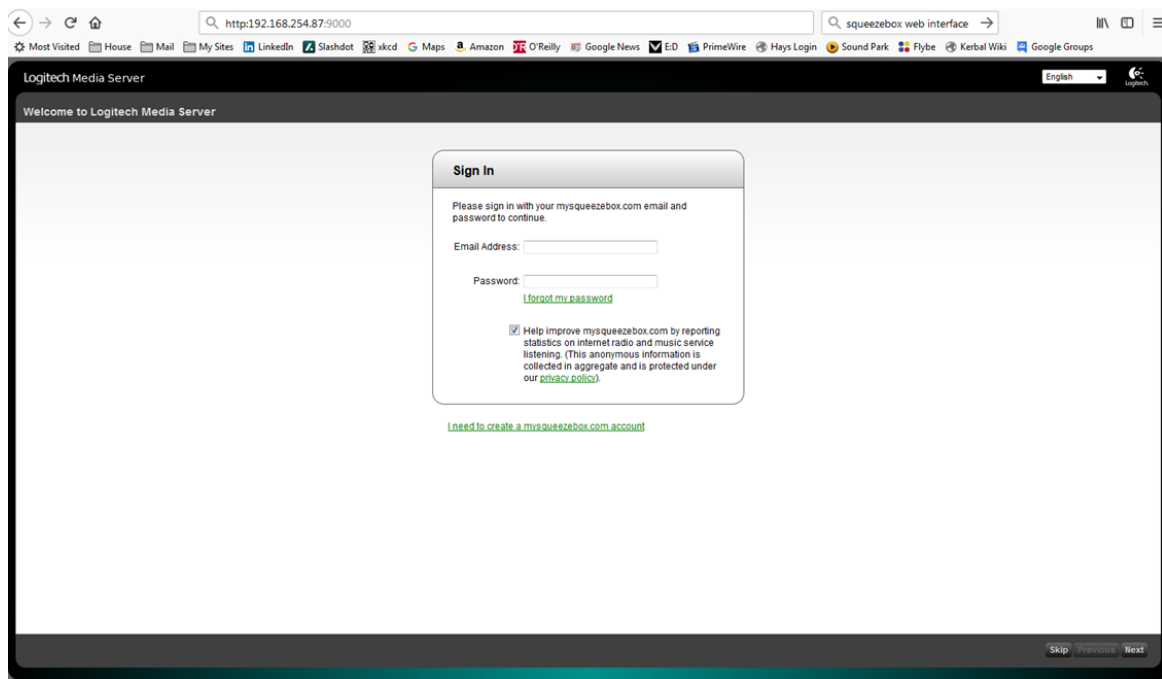


Figure 4 The LMS login screen

When this is working you will see a login screen like that shown above. You can create a new user on the Logitech site or just hit skip (bottom right). The system will then ask you to select a directory for the music (`/music/active`) and the playlists (`/music/playlists`).

Once those two have been selected the interface will show all the available music and the players that can play it. Of course in our case there is neither and music present, nor any players.

Copy music files to server

There are many ways to get music data into the appropriate directory. Installing **samba** on your server, or using **sftp** are two popular techniques.

If your desktop computer is running Linux (or you have **cygwin** installed on a Windows machine) here is an alternate approach. Suppose we have a directory `c:\mymusic\` containing subdirectories each of which has an album's worth of tagged mp3s.



Music at Home with Raspberry Pi

```
User@wolf ~  
$ cd c:  
  
User@wolf /cygdrive/c  
$ cd mymusic  
  
User@wolf /cygdrive/c/mymusic  
$ tar cf - * | ssh -l music 192.168.186.215 "(cd /music/import;\n                                          tar xvf -)"
```

The command above uses **tar** to package up all the local directories (*) sending them (|) to the target machine (192.168.186.215) logged in as **music** in the directory **/music/import** to use **tar** to unpack them.

```
$ ssh -l music 192.168.186.215  
...  
pi@parrot:~ $ cd /music/import  
pi@parrot:~ $ chmod 755 *  
pi@parrot:~ $ chmod 644 */*  
pi@parrot:~ $ mv * ../active
```

Once the copying is completed then log on to the target machine as **music**, ensure the files are readable and move them into the **/music/active** directory.



Music at Home with Raspberry Pi

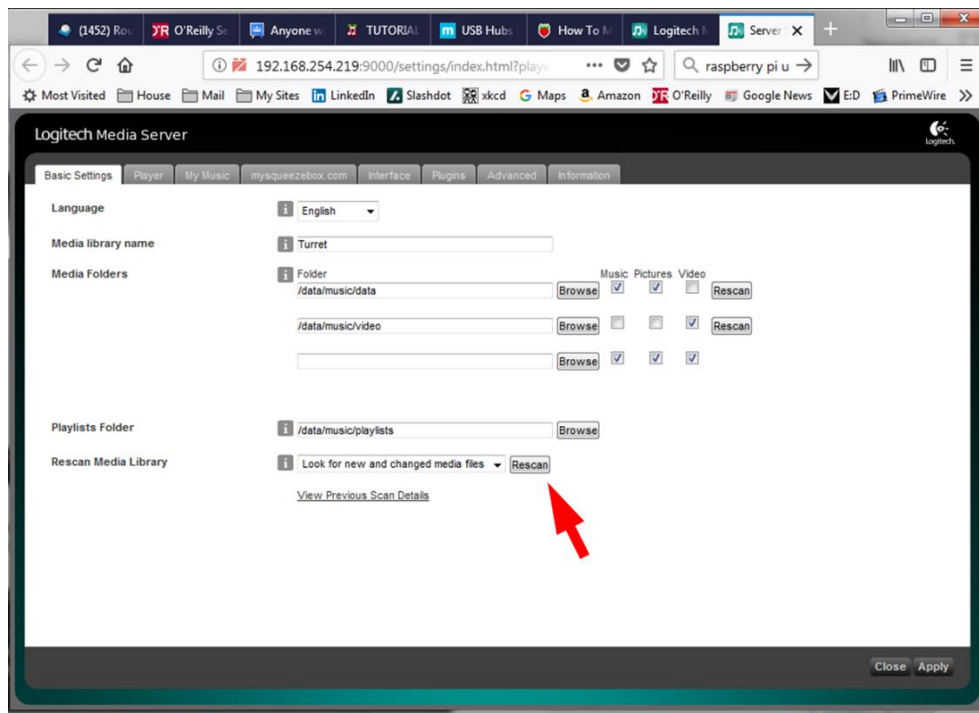


Figure 5 Tell LMS to rescan the data directory

Finally back on the desktop under the settings page there is a button that will make the sever rescan the active directory looking for changed files.

The simplest player

So having loaded some music on to the server the only remaining issue is that there are no players on the network. The simplest way to remedy this is to download the **squeezelite** program for your desktop computer (from <https://code.google.com/p/squeezelite/downloads/list>) and run it.

Playing a first song

On the web interface select your player (pulldown menu top right) and your songs (select the play icon) and the music should start to play. If you have multiple players you can either keep them separate or you can tell them to synchronise with each other.



Music at Home with Raspberry Pi

Raspberry Pi 3 player on a Bluetooth speaker

A Raspberry Pi can make a good music player. The audio circuits on the bare Raspberry Pi are really not good enough quality for music, so one option is to add a DAC board and use that to drive the speakers. On this occasion we will take a different approach, we will use an external Bluetooth speaker.

The Raspberry Pi 3 has on-board Bluetooth so that is what we'll be using. In August 2017 a new version of the Linux operating system, "stretch", was released. This has the following note:

In Jessie, we used PulseAudio to provide support for audio over Bluetooth, but integrating this with the ALSA architecture used for other audio sources was clumsy. For Stretch, we are using the bluez-alsa package to make Bluetooth audio work with ALSA itself. PulseAudio is therefore no longer installed by default...

Now this caused all sorts of frustrations because most of the web's advice about getting Bluetooth and audio to play nicely together was therefore out of date. Luckily I eventually found a great article that covered the basics at https://www.sigmdel.ca/michel/ha/rpi/bluetooth_01_en.html.

It is quite possible to install the player software on the same Raspberry Pi that is running LMS. For the purposes of this document though I have assumed that the single LMS machine will be at some central location, meanwhile a number of players will each be physically placed near where people want to listen to music. For this reason these steps show how to configure each of the players.

What you need

The components required are:

- A new Raspberry Pi 3
- A Bluetooth speaker
- An SD card with "**raspbian-stretch-lite.img**" installed
- A **wav** file containing a song or other recognisable sound for testing

I have been told that running WiFi and Bluetooth at the same time will cause some audio stuttering, in my setup it is easy to plug an Ethernet cable for the Pi so I have never actually tested this. But the following does assume you are using the wired network connection.

Preparation

Just like with the server install Raspbian Stretch Lite and use the **raspi-config** command to:

- Change the **pi** account password
- Change the machine's hostname (under **Network Options**)
- Enable **ssh** access (under **Interfacing Options**)

Then make a note of what the IP address actually is (use the command **ifconfig**). Having done all that we don't need the monitor any more we can detach it, reboot and **ssh** remotely from another machine.



Music at Home with Raspberry Pi

```
User@wolf ~
$ ssh -l pi 192.168.186.126
...
pi@jay:~ $ sudo apt-get update
...
Reading package lists... Done
pi@jay:~ $ sudo apt-get upgrade
...
Do you want to continue? [Y/n] y
...
```

The Bluetooth setup in stretch has a couple of issues, one has to do with the SIM Access Profile (sap). This is of no interest in this case, so the file `/lib/systemd/system/bluetooth.service` needs a small edit, the “`ExecStart`” line needs the plugin to be disabled with “`--noplugin=sap`”.

```
ExecStart=/usr/lib/bluetooth/bluetoothd --noplugin=sap
```

Once that option is added the line in my version is as shown above.

Extra packages

Install the extra packages that will be needed later:

```
pi@jay:~ $ sudo apt-get install bluealsa squeezelite
...
Do you want to continue? [Y/n] y
...
```

Running `squeezelite` at boot time would never work. The command couldn’t connect to the Bluetooth speaker *after* it had been started, the connection to the speaker needed to be established *before* the command is executed. A swift bit of gross cutting will stop the system from running the server too early².

```
pi@jay:~ $ cd /etc
pi@jay:~ $ sudo rm init.d/squeezelite rc*.d/[KS]01squeezelite
```

² This is the old “sysvinit” way to disable programs run at start-up. It would have been better to use `systemctl`, but that’s not what I did and I wanted to document the exact commands used.

Music at Home with Raspberry Pi

Create a new account

In order to keep the running of the player software isolated from system configuration a new account will be created. This will be called `squeeze`.

```
pi@jay:~ $ sudo adduser squeeze
Adding user `squeeze' ...
Adding new group `squeeze' (1001) ...
Adding new user `squeeze' (1001) with group `squeeze' ...
Creating home directory `/home/squeeze' ...
Copying files from `/etc/skel' ...
Enter new UNIX password: ***
Retype new UNIX password: ***
passwd: password updated successfully
Changing the user information for squeeze
Enter the new value, or press ENTER for the default
    Full Name []: Squeezelite User
    Room Number []:
    Work Phone []:
    Home Phone []:
    Other []:
Is the information correct? [Y/n] y
```

The two `***` sequences are the password for this new account, obviously those are not echoed when you run these commands. This account needs to be able to run the Bluetooth and audio software, so it has to be added to those groups.

```
pi@jay:~ $ sudo adduser squeeze bluetooth
Adding user `squeeze' to group `bluetooth' ...
Adding user squeeze to group bluetooth
Done.
pi@jay:~ $ sudo adduser squeeze audio
Adding user `squeeze' to group `audio' ...
Adding user squeeze to group audio
Done.
```

Pair the speaker

The `bluetoothctl` command provides a simple interface to the on-board controller. Since the user `squeeze` is part of the `bluetooth` group we can do the rest while logged in as that user. Running the command puts us in control, so we can power on the hardware and start scanning for devices.

```
squeeze@jay:~ $ bluetoothctl
[NEW] Controller B8:27:EB:3A:8F:46 jay [default]
[bluetooth]# power on
Changing power on succeeded
[bluetooth]# scan on
```



Music at Home with Raspberry Pi

```
Discovery started
```

Now that the on-board hardware is actively looking for devices the target Bluetooth speaker can be set to pair. When it is in pair mode this should be detected and noted by the running command.

```
[CHG] Controller B8:27:EB:3A:8F:46 Discovering: yes
[NEW] Device E0:2A:82:31:37:43 E0-2A-82-31-3E-6B
[CHG] Device E0:2A:82:31:37:43 RSSI: -91
[CHG] Device E0:2A:82:31:37:43 TxPower: 0
[CHG] Device E0:2A:82:31:37:43 Name: WOLF
[CHG] Device E0:2A:82:31:37:43 Alias: WOLF
[NEW] Device B8:D5:13:0B:61:35 SRS-XB20
```

The software has found our speaker, the fact that we have the correct device name (which we've highlighted in yellow in the text above) tells us which device it is. The other device is a laptop that we'll ignore. The string of hex digits is how we identify devices. So we want to pair, trust and connect

```
[bluetooth]# pair B8:D5:13:0B:61:35
[bluetooth]# trust B8:D5:13:0B:61:35
[bluetooth]# connect B8:D5:13:0B:61:35
Attempting to connect to B8:D5:13:0B:61:35
[CHG] Device B8:D5:13:0B:61:35 Connected: yes
[CHG] Device B8:D5:13:0B:61:35 Modalias: bluetooth:v0039p1582d2203
[CHG] Device B8:D5:13:0B:61:35 UUIDs: 00000000-deca-fade-deca-deaf
[CHG] Device B8:D5:13:0B:61:35 UUIDs: 00001108-0000-1000-8000-0080
[CHG] Device B8:D5:13:0B:61:35 UUIDs: 0000110b-0000-1000-8000-0080
[CHG] Device B8:D5:13:0B:61:35 UUIDs: 0000110c-0000-1000-8000-0080
[CHG] Device B8:D5:13:0B:61:35 UUIDs: 0000110e-0000-1000-8000-0080
[CHG] Device B8:D5:13:0B:61:35 UUIDs: 0000111e-0000-1000-8000-0080
[CHG] Device B8:D5:13:0B:61:35 UUIDs: 00001200-0000-1000-8000-0080
[CHG] Device B8:D5:13:0B:61:35 ServicesResolved: yes
[CHG] Device B8:D5:13:0B:61:35 Paired: yes
Connection successful
[CHG] Device E0:2A:82:31:37:43 RSSI: -83
[SRS-XB20]# exit
```

Once we're connected we can quit from the `bluetoothctl` program with the `exit` command.

Test the connection

Now that the speaker is connected we should be able to play audio through it. We need the `example.wav` file to run the test, so let's copy it to the Pi3.

Music at Home with Raspberry Pi

```
User@wolf ~  
$ tar cf - example.wav | ssh -l squeeze 192.168.186.126 "tar xvf -"  
squeeze@192.168.186.126's password:  
example.wav
```

This uses tar to copy the data from one machine to another just like we did for getting the music to the server. Once it is on the Pi3 lets ensure the permissions are OK

```
squeeze@jay:~ $ chmod 644 example.wav
```

The `aplay` command can send the example sound to the connected speaker.

```
squeeze@jay:~ $ aplay -D bluealsa:HCI=hci0,\  
DEV=B8:D5:13:0B:61:35,PROFILE=a2dp example.wav
```

If the sound plays then all is OK. If it doesn't I would suggest that you have a look at Michel's excellent article (linked above). Once the connection is working we can make the Bluetooth speaker the default audio device for the `squeeze` user by editing the `.asoundrc` file.

```
pcm.!default {  
    type plug  
    slave.pcm {  
        type bluealsa  
        device "B8:D5:13:0B:61:35"  
        profile "a2dp"  
    }  
}
```

Create a file called `~squeeze/.asoundrc` and containing the contents shown above (obviously the device string needs to be replaced with your speakers MAC address). If that's working correctly then the command:

```
squeeze@jay:~ $ aplay example.wav
```

Should play the example sound through your speaker.

Triggering the squeezelite process

Assuming that has all worked you now have a headless Raspberry Pi 3 that is able to use a Bluetooth speaker to play audio. The next step is the interesting bit. If we could guarantee that the speaker was always connected then just running the `squeezelite` command would complete the setup. Unfortunately real life is more complex. There are all sorts of reasons why the speaker may be disconnected, it could be out of range, have flat batteries or just be turned off. It has already been mentioned that there is no easy way to attach the `squeezelite` process to the bluetooth audio output *after* the process has started. This means we can only start-up the process when we know that speaker is already connected.



Music at Home with Raspberry Pi


	<i>squeezelite</i> running	<i>squeezelite</i> not running
<i>speaker</i> connected		Run It
<i>speaker not</i> <i>connected</i>	Kill!!!	wait for connection

Figure 6 Four possible states

This leads us to the fact that there are four possible states. The `squeezelite` process could either be running or not yet active and the speaker could be connected or absent. Each of these combinations leads to a different action, as shown in the figure above. If we regularly execute a process that determines which of these four states we are in and acts appropriately then we'll have a system that does what we want.

So here's a perl script that does exactly that:

```
#!/usr/bin/perl

use strict;
use warnings;

use IO::File;

# This is the speaker DMAC so we can see if is connected
my $speaker_dmac = "B8:D5:13:0B:61:35";
# Need to specify the user account because this is run
# in "cron" so the normal env values are absent
my $user_account = "squeeze";

# Everything must have absolute paths because we are in CRON

# If there are any issues we can turn on logging and see what
# this is doing. Uncomment the next line and comment out the one
# after
# my $logging = "/home/squeeze/respawn_log";
my $logging = "";

my $date_ran = "";
if($logging)
{
    $date_ran = `date`;
    chomp($date_ran);
}

my $speaker_connected = "";
my $sqlite_proc = "";
```



Music at Home with Raspberry Pi

```
# This uses ps to attempt to identify the process number
# of the running squeezelite process (owned by this user)
{
  my $ps_output_fh = IO::File->new("/bin/ps -U $user_account|");
  while(my $line = <$ps_output_fh>)
  {
    $sqlite_proc = $1
    if($line =~
      /\s*(\d+)\s+\S+\s+\d+:\d+:\d+\s*squeezelite/i);
    last if($sqlite_proc);
  }
  $ps_output_fh->close();
}

# Now lets see if the bluetooth speaker is connected
{
  my $bt_output_fh = IO::File->new(
    "/bin/echo -e \"info $speaker_dmac\" |".
    " /usr/bin/bluetoothctl 2>&1 |");
  while(my $line = <$bt_output_fh>)
  {
    if($line =~ /Connected\: (yes|no)/i)
    {
      my $answ = $1;
      $speaker_connected = 1 if(lc($answ) eq "yes");
      last;
    }
  }
  $bt_output_fh->close();
}

# We have four possible cases
my $state = "";
if($speaker_connected)
{
  if($sqlite_proc)
  {
    # Everything is as it should be, do nothing
    $state = "connected_running";
  }
  else
  {
    # Speaker is now connected, need to start the squeezelite
    process
    # in the background
    system("/usr/bin/squeezelite&");
    $state = "connected_absent";
  }
}
else
{
  if($sqlite_proc)
  {
    # The speaker is disconnected need to kill the running
```



Music at Home with Raspberry Pi

```
# squeezelite process
system("/bin/kill -9 $sqlite_proc");
$state = "disconnect_running";
}
else
{
    # Speaker is not connected, and the squeezelite process
    # is not running. Just leave it
    $state = "disconnect_absent";
}
}

# Finally make a note of where we ended up
if($logging)
{
    system("/bin/echo \"$date_ran $state\" >> $logging");
}
```

That file is saved as `/home/squeeze/bin/respawn` and made executable.

```
squeeze@jay:~ $ chmod 755 /home/squeeze/bin/respawn
squeeze@jay:~ $ crontab -e
```

We'll use `cron` to execute that script every minute. Run the command "`crontab -e`" and add the following line.

```
* * * * * /home/squeeze/bin/respawn
```

Now do some testing. Try rebooting the Pi with the speaker turned off. When the speaker is turned on it should do the "I'm connecting" beep and within the next minute a `squeezelite` process should start up (belonging to the user `squeeze`). You can see that happening if you run the `top` command on the Pi or when the player shows up in the pulldown list of the web control interface.

Worrying about "wearout"

With the server we reduced the chance of SD card wearout by moving the logging to the hard disk. In this case we don't have a hard disk, so the best option is to move the logging to a RAM based file system.

```
tmpfs /tmp tmpfs defaults,noatime,nosuid,size=100m 0 0
tmpfs /var/tmp tmpfs defaults,noatime,nosuid,size=30m 0 0
tmpfs /var/log tmpfs defaults,noatime,nosuid,mode=0755,size=90m 0 0
```

Add the three entries shown above into the `/etc/fstab` file and on next reboot the logging will be taken off the SD card.

