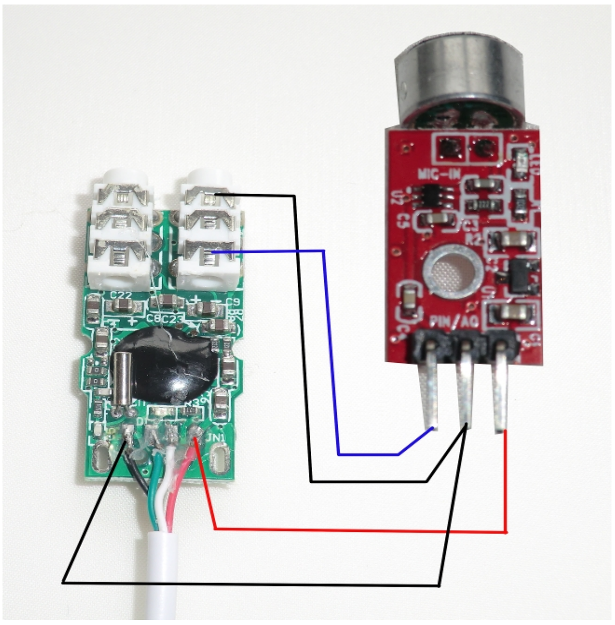
AudioSpy room microphone on a Raspberry Pi.

Plug in a USB audio adapter & MAX9812 combo.



Red = 5 volt, Black = Ground, Blue = Mic output.

The black, green, white and red wires on the USB adapter were initially covered in hot-glue, which we removed using a hairdryer and a cotton bud. If you just try and pull the lump of hot-glue off the PCB you'll pull away the solder pads on the PCB, making the USB adapter useless.

If you don't want to make your own we sell them here, soldered & glued, in a 3D printed case: http://www.ebay.co.uk/itm/High-Gain-USB-Microphone-Use-with-PC-Mac-or-Raspberry-Pi-Voice-recognition-/162571956830?ssPageName=STRK:MESE:IT for £21.95.

Increase mic volume for USB device to full using.

```
alsamixer
```

(F6 to change to USB device and arrow keys move across to Mic and up arrow increases volume to full, Esc to quit)

Make a test recording:

```
arecord -D plughw:1,0 -f S16_LE test.wav
```

CTRL-C to quit .

```
aplay test.wav
```

Install Sox audio tools with:

```
sudo apt-get install sox
```

You can make the room Mic record a mono high quality file with this one liner:

```
sox -t alsa plughw:1,0 -c 1 -r 16k test.wav
```

You can make a series of voice activated recordings with this one-liner:

```
sox -t alsa plughw:1,0 -c 1 -r 16k r.wav silence 1 0.1 0.3% 1 3.0
0.3% : newfile : restart
```

Ctrl-C to quit.

You can concatenate the voice activated recordings into a single .wav with this:

```
sox $(ls r*.wav | sort -n) out.wav
```

then remove the old files with

```
rm r*.wav
```

and playback the new file with

```
aplay out.wav
```

Transcribe room audio and email text transcript to a phone

Sign up to Google cloud platform for free at https://console.cloud.google.com/start

Google Cloud Speech API → go to Credenitials tab → Create Credentials → API Key → cut and paste your API key for use in audiospy.py or audiospypir.py later.

```
sudo apt-get install mplayer
sudo apt-get install flac
sudo apt-get install python-pycurl
```

While in folder /home/pi pull file from our server with :

```
wget www.securipi.co.uk/audiospy.py
```

Edit audiospy.py so it contains your Google Cloud Speech API key

```
nano audiospy.py
```

Save it and exit . Run it while connected to the internet

```
python audiospy.py
```

Audio samples will be uploaded to Google's Cloud Speech transcribing system and you will get text back. The returned text is stored in the file talklog.txt. You can look at the contents with

```
cat talklog.txt
```

Audio samples that contain valid words & phrases are also saved to the Pi's memory card in time & date stamped FLAC audio format. Play all the time and date stamped FLAC audio files back with

```
mplayer *.flac
```

The text you receive back from Google's Cloud Speech API will not always be 100% correct (some phrases sound a lot like others), but you can check it by listening to the correct time & date stamped FLAC audio file later. That way you can also hear who spoke the words, and hear exactly what was said.

We calculated that even if you constantly sent 15 second audio clips to the Cloud Speech API for 24 hours of a day the total cost you pay Google couldn't exceed £0.35p per day for the service. In reality you'll send a lot less audio, as the Mic is voice/noise activated. Also, the first 60 minutes of usage are free.

Our audiospy.py script also has the ability to send an email of the talklog.txt file whenever a new recording has correctly yielded some transcribed audio. First we need to setup a new email account for our Raspberry Pi to use. If you don't want to send the email, just comment out the last line in the audiospy.py file with a # symbol.

Setting up a spare Gmail account, just for your Pi to use.

The simplest way of sending emails from your Pi, is to setup a new Gmail account for the Pi to use, even if you already have an existing Gmail account you use on your phone or PC. For one thing, it gets you 15GB of new cloud storage for your alerts & secondly it removes the complication of generating application specific passwords for other apps on your existing Gmail account.

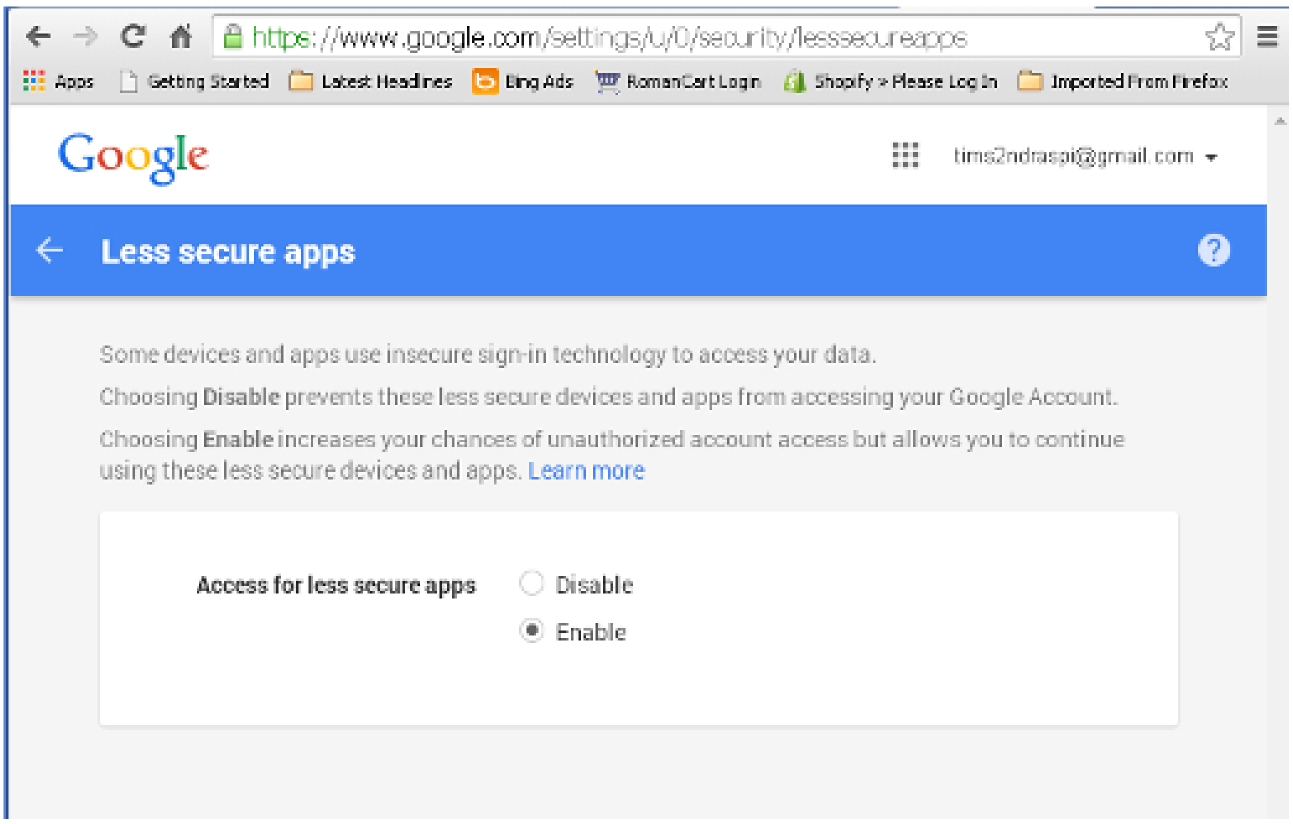
You need to create the new Gmail account in the web browser on your PC or Mac @

https://accounts.google.com/SignUp?service=mail

and note down the login & password for later. Don't use the # symbol in your password as it causes the Pi problems. Lower & upper-case letters & number combinations are always fine though.

Next, you need to set the new Gmail account to Enable "less secure apps". While logged in go to:

https://www.google.com/settings/u/0/security/lesssecureapps



Less secure apps

Some devices and apps use insecure sign-in technology to access your data.

Choosing **Disable** prevents these less secure devices and apps from accessing your Google Account.

Choosing **Enable** increases your chances of unauthorized account access but allows you to continue using these less secure devices and apps. Learn more

Access for less secure apps Disable [✓ Enable]

Now we have a working email account, just for the Pi to use when sending emails. Any emails sent from the Pi will be backed-up in the Sent folder & you only need delete old emails if you get near to the 15GB limit. Emails from the Pi can be sent to any other email address on your phone or PC.

Setup and test the email utilities

We need to install & configure several mail applications from the internet. You also need to have made a new Gmail account, just for the Pi to use and enable “less secure app” access @ https://www.google.com/settings/u/0/security/lesssecureapps

```
sudo apt-get install ssmtp
sudo apt-get install mailutils
sudo apt-get install mpack
```

Setup default settings SSMTP.

```
sudo nano /etc/ssmtp/ssmtp.conf
```

```
AuthUser=your-gmail-account@gmail.com
AuthPass=your-user-password
FromLineOverride=YES
mailhub=smtp.gmail.com:587
UseSTARTTLS=YES
```

save file & exit. Make sure you changed the AuthUser and AuthPass values to your own.

Send a test email with this command, substituting the email address below with your own.

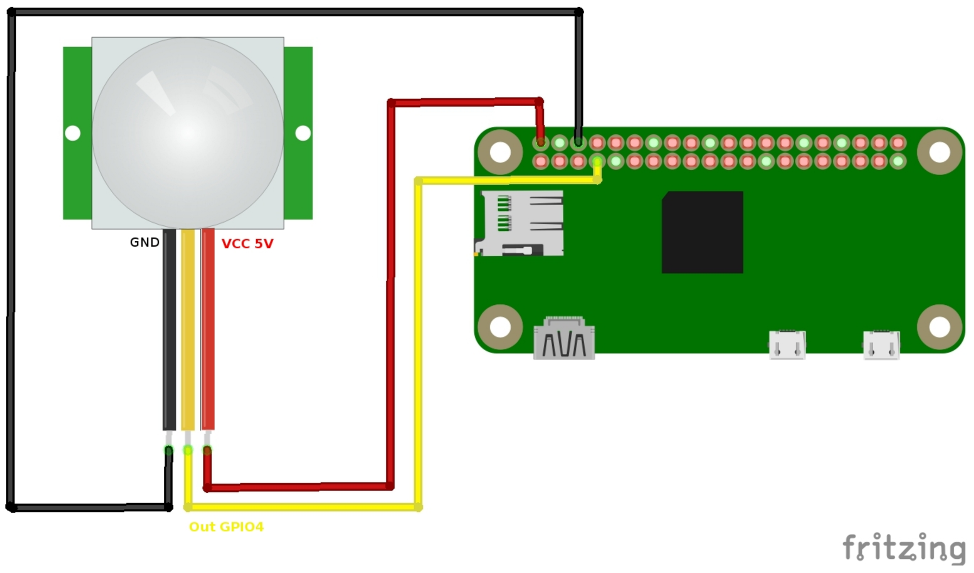
```
echo "testing 1 2 3" | mail -s "Subject" you@yourdomain.co.uk
```

Audio recording triggered by a PIR sensor

One of our other ideas is to use a Raspberry Pi with a USB room microphone fitted next to the Raspberry PiNoIR camera - which already watches outside our house between 12 midnight and 6am on a timer, and only takes photos when the PIR is triggered.

If we make the audiospy script only pay attention when the PIR is triggered too, then that's only typically 5 minutes of audio transcribing at $0.006 per 15 seconds (12 days worth making an hour of audio @ 1.44p, or less than 50p a year) – which is a good deal if you think about the amount of time you could potentially spend sifting through audio files triggered by cats and cars driving past.

The PIR is attached to GPIO4 like this:



You can pull the PIR adapted version of audiospy down from our server with

```
wget www.securipi.co.uk/audiospypir.py
```

edit the file so it contains your Google Cloud Speech API key & email address

```
nano audiospypir.py
```

save and exit, and run it with

```
python audiospypir.py
```

The PIR version saves all audio samples submitted to the API, even if they didn't appear to contain speech. You'll find time and date stamped FLAC audio files in your default folder. List them with:

```
ls
```

play them with mplayer

```
mplayer  2017-07-30---10-00-41.flac
```

Streaming Audio from one Raspberry Pi to another.

How to stream audio from one Pi to another Pi (or an Ubuntu Linux PC). We use the Netcat command to pipe the audio over a network connection from one Pi to the other.

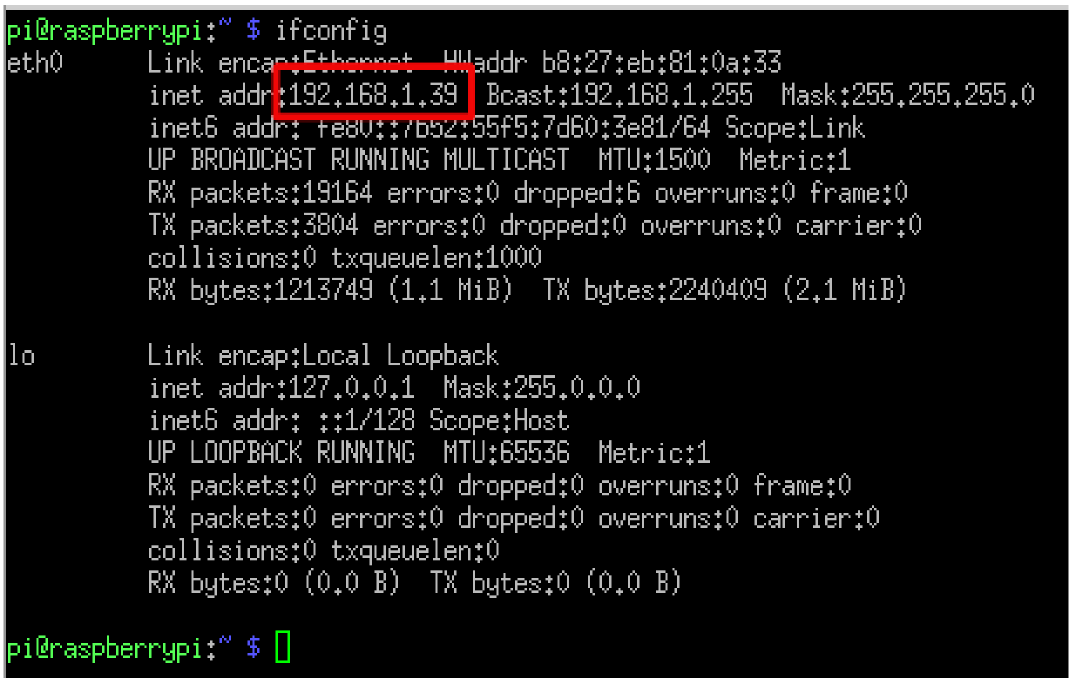
This method uses no more than 30% of processor cycles on an old model B or A+ Pi and works flawlessly. Our preferred method of setting this up is to simultaneously SSH into each Pi remotely from our desktop PC. You need to enable SSH using raspi-config command line tool.

On both the sending and receiving Raspberry Pi you need to install the Opus codec:

```
sudo apt-get install opus-tools
```

Discover and make a note of the IP address (here 192.168.1.39) of each Pi with the command :

```
ifconfig
```



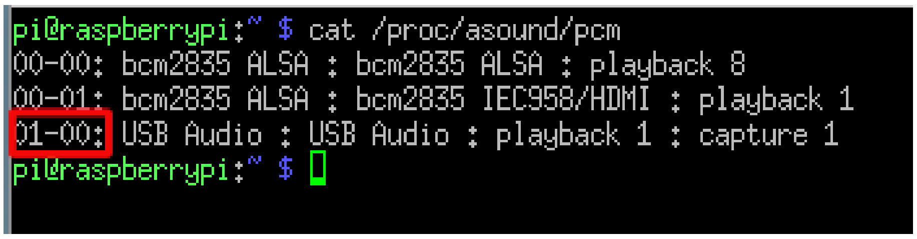
```
pi@raspberrypi:~ $ ifconfig
eth0      Link encap:Ethernet  HWaddr b8:27:eb:81:0a:33
          inet addr:192.168.1.39  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::7b52:55f5:7d60:3e81/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:19164 errors:0 dropped:6 overruns:0 frame:0
          TX packets:3804 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1213749 (1.1 MiB)  TX bytes:2240409 (2.1 MiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

pi@raspberrypi:~ $
```

Tip: setup each Pi in your router so they always get given the same static IP address.
Discover and note useful information about the USB sound cards with commands :

```
lsusb
cat /proc/asound/pcm
```



```
pi@raspberrypi:~ $ cat /proc/asound/pcm
00-00: bcm2835 ALSA : bcm2835 ALSA : playback 8
00-01: bcm2835 ALSA : bcm2835 IEC958/HDMI : playback 1
01-00: USB Audio : USB Audio : playback 1 : capture 1
pi@raspberrypi:~ $
```

In the example above our USB audio device is on port plughw:1,0

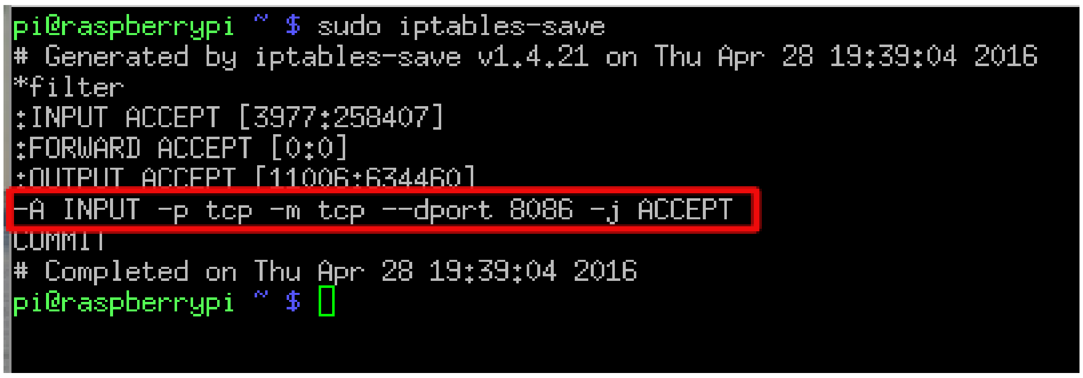
On the receiving Pi

Open up port 8086 in your firewall with:

```
sudo /sbin/iptables -I INPUT 1 -p tcp --dport 8086 -j ACCEPT
```

and check it has been accepted with

```
sudo iptables-save
```

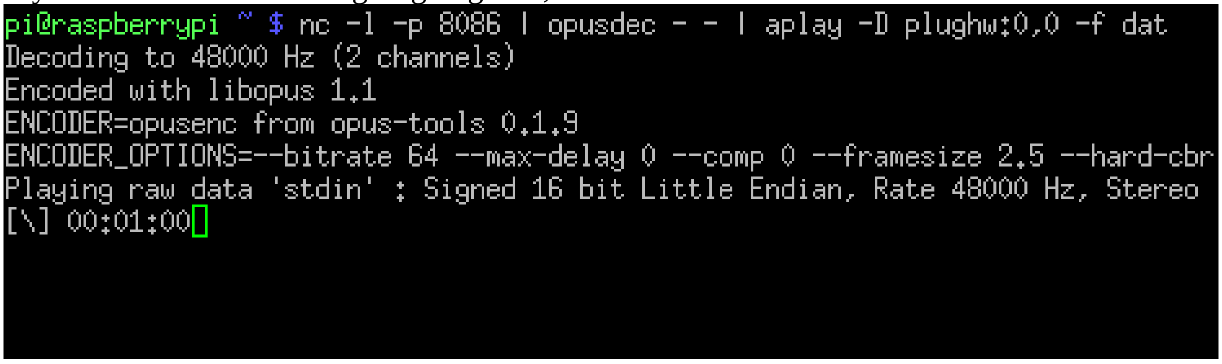


```
pi@raspberrypi ~ $ sudo iptables-save
# Generated by iptables-save v1.4.21 on Thu Apr 28 19:39:04 2016
*filter
:INPUT ACCEPT [3977:258407]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [11006:634460]
-A INPUT -p tcp -m tcp --dport 8086 -j ACCEPT
COMMIT
# Completed on Thu Apr 28 19:39:04 2016
pi@raspberrypi ~ $
```

then set the Pi listening with

```
nc -l -p 8086 | opusdec - - | aplay -D plughw:0,0 -f dat
```

if you don't set the receiving Pi going first, the sender will fail.



```
pi@raspberrypi ~ $ nc -l -p 8086 | opusdec - - | aplay -D plughw:0,0 -f dat
Decoding to 48000 Hz (2 channels)
Encoded with libopus 1.1
ENCODER=opusenc from opus-tools 0.1.9
ENCODER_OPTIONS=--bitrate 64 --max-delay 0 --comp 0 --framesize 2.5 --hard-cbr
Playing raw data 'stdin' : Signed 16 bit Little Endian, Rate 48000 Hz, Stereo
[\] 00:01:00
```

You can force the receiving Pi to use the 3.5mm headphone output, instead of HDMI audio out, with:

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

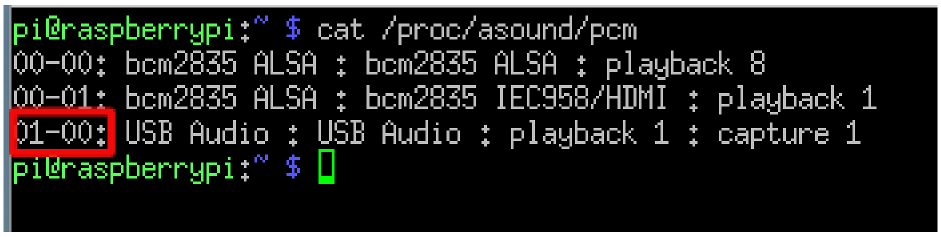
On the sending Pi:

Plug in your USB Audio Mic device and use commands:

```
lsusb
```

and

```
cat /proc/asound/pcm
```

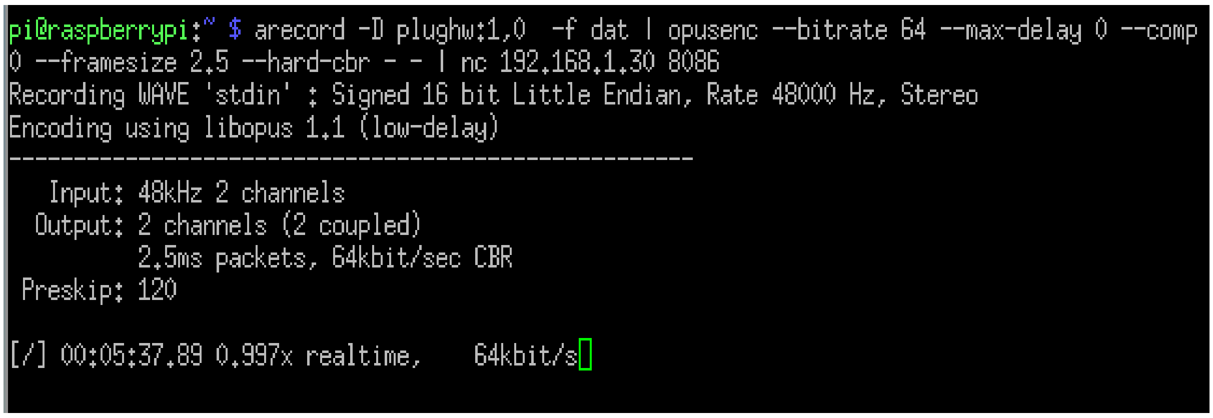


```
pi@raspberrypi:~ $ cat /proc/asound/pcm
00-00: bcm2835 ALSA : bcm2835 ALSA : playback 8
00-01: bcm2835 ALSA : bcm2835 IEC958/HDMI : playback 1
01-00: USB Audio : USB Audio : playback 1 : capture 1
pi@raspberrypi:~ $
```

The correct audio device here is plughw:1,0 - see below. The correct IP address in the example below should be the Pi you are transmitting to:

```
arecord -D plughw:1,0  -f dat | opusenc --bitrate 64 --max-delay 0
--comp 0 --framesize 2.5 --hard-cbr - - | nc 192.168.1.18 8086
```

(The command above all goes on one line)



```
pi@raspberrypi:~ $ arecord -D plughw:1,0  -f dat | opusenc --bitrate 64 --max-delay 0 --comp
0 --framesize 2.5 --hard-cbr - - | nc 192.168.1.30 8086
Recording WAVE 'stdin' : Signed 16 bit Little Endian, Rate 48000 Hz, Stereo
Encoding using libopus 1.1 (low-delay)
-----------------------------------------------------
   Input: 48kHz 2 channels
  Output: 2 channels (2 coupled)
          2.5ms packets, 64kbit/sec CBR
 Preskip: 120

[/] 00:05:37.89 0.997x realtime,    64kbit/s
```

If the command above exits immediately, then the port 8086 probably isn't open on the receiving Pi (or you haven't set the receive command running on the other Pi). Check the correct port is open on the receiving Pi with:

```
sudo iptables-save
```

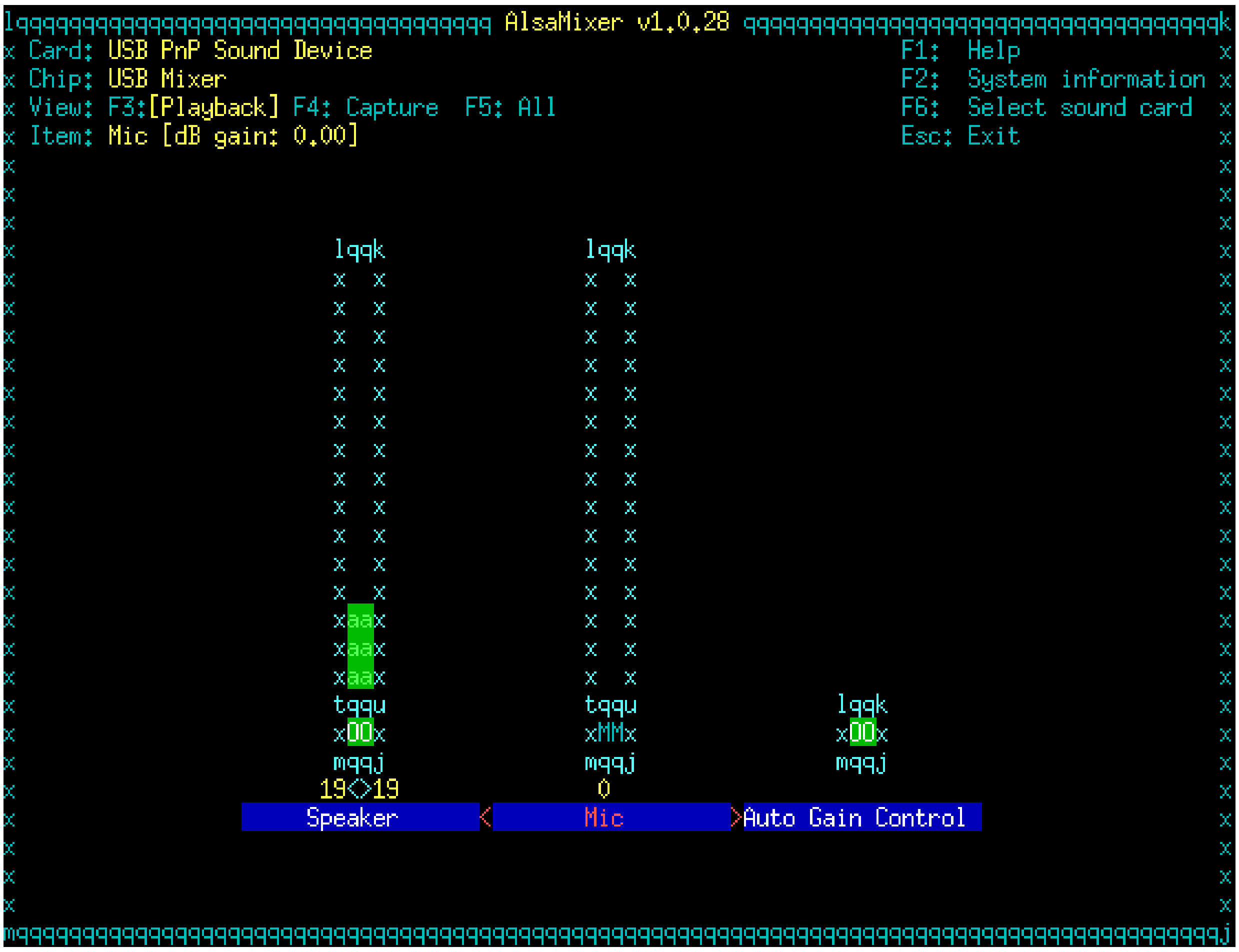
Audio Mixer Panel

On the sending Pi it might be useful to play with the USB soundcard control panel using the command:

```
alsamixer
```

Press F6 and choose the USB sound card.

You can use the arrows keys to move around and raise or lower the volume levels and use M to toggle items on or off (like Auto Gain). Esc to quit.



```
lqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqq AlsaMixer v1.0.28 qqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqk
x Card: USB PnP Sound Device                                        F1:  Help               x
x Chip: USB Mixer                                                   F2:  System information x
x View: F3:[Playback] F4: Capture  F5: All                          F6:  Select sound card  x
x Item: Mic [dB gain: 0.00]                                         Esc: Exit               x
x                                                                                           x
x                                                                                           x
x                                                                                           x
x                        lqqk               lqqk                                            x
x                        x  x               x  x                                            x
x                        x  x               x  x                                            x
x                        x  x               x  x                                            x
x                        x  x               x  x                                            x
x                        x  x               x  x                                            x
x                        x  x               x  x                                            x
x                        x  x               x  x                                            x
x                        x  x               x  x                                            x
x                        x  x               x  x                                            x
x                        x  x               x  x                                            x
x                        x  x               x  x                                            x
x                        x  x               x  x                                            x
x                        xaax               x  x                                            x
x                        xaax               x  x                                            x
x                        xaax               x  x                                            x
x                        tqqu               tqqu               lqqk                         x
x                        xOOx               xMMx               xOOx                         x
x                        mqqj               mqqj               mqqj                         x
x                       19<>19               0                                              x
x                      Speaker      <       Mic        >Auto Gain Control                   x
x                                                                                           x
x                                                                                           x
x                                                                                           x
mqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqj
```

Different audio sampling rates

At the moment we're capturing audio in stereo at 48000Hz, which is higher quality than a CD. If capturing CCTV audio and streaming it over the internet, then we can manage with a lower sample rate and single channel mono.

Either of these commands will make a local recording at a lower sample rate:

```
arecord -D plughw:1,0 -f S16_LE -c1 -r8000 -t wav test8000.wav
```

```
arecord -D plughw:1,0 -f S16_LE -c1 -r22050 -t wav test22050.wav
```

You could for example use a PIR sensor, connected to the Pi's GPIO pins, to only record audio for a set duration (-d) when motion is detected and have the filename be the date and time.

You can compare the file sizes with

```
ls -al
```

The 8000Hz sample rate generates the smallest files and is okay for speech. The 22050Hz sample rate is equivalent to FM radio. 44100Hz is CD quality and 48000Hz is DAT or DVD audio.

Try this one on the receiving Pi:

```
nc -l -p 8086 | opusdec - - | aplay -D plughw:0,0 -f S16_LE -c1
-r8000 -t wav
```

and this on the transmitting Pi:

```
arecord -D plughw:1,0 -f S16_LE -c1 -r8000 -t wav | opusenc
--bitrate 64 --max-delay 0 --comp 0 --framesize 2.5 --hard-cbr - -
| nc 192.168.1.8 8086
```

(commands all go on one line, type it in. Don't copy & paste from the PDF)

If you want to capture the audio into a file at the receiving end, you can do this instead:

```
nc -l -p 8086 | opusdec - - > test.wav
```

and play the recording back later with

```
aplay -f S16_LE -c1 -r8000 -t wav test.wav
```

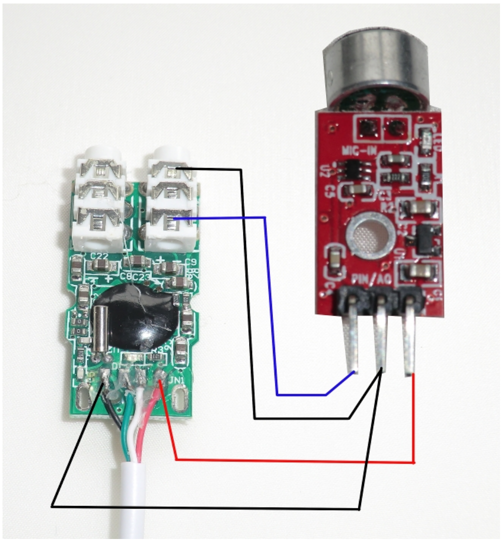
Dropping the sample rate from 48000Hz down to 8000Hz, and mono rather than stereo, means that on an original Raspberry Pi model B or A+ the %CPU cycles used with Opus (measured with command `top`) drop from around 27% to 17%. But what advantage does using Opus give us?

We used the command `iftop` to measure network traffic on each Pi while using 48000Hz and Opus compression (151Kb a second) and without compression (10x more traffic = 1.52Mb a second). At 8000Hz Mono though it turns out it's not worth using Opus (151Kb/s v 132Kb/s none).

Where to buy the hardware.

You can get a ready made USB room mic for Raspberry Pi from us here:
 http://www.ebay.co.uk/itm/High-Gain-USB-Microphone-Use-with-PC-Mac-or-Raspberry-Pi-Voice-recognition-/162571956830?ssPageName=STRK:MESE:IT for £23.95 inc VAT & P+P.

Or you can make your own using any USB Audio adapter with a Mic input and a MAX9812 mic, if you have some spare time, a 3D printer, a soldering iron and a hot glue gun.



MAX9812 room mic for £4 here:
http://www.ebay.co.uk/itm/MAX9812-Microphone-Amplifier-Module-MIC-Voice-Arduino-3-3V-5-0V-Room-Mic-Spy-Mic-/162571985482?ssPageName=STRK:MESE:IT

You can get our 3d print case design here:
www.securipi.co.uk/usb-mic-holder.stl

We sell individual PIR modules with cables for Raspberry Pi here at £3:
http://www.ebay.co.uk/itm/1x-Infrared-PIR-Motion-Sensor-Module-GPIO-cables-amp-GPIO-card-for-Raspberry-Pi-/161349464070?ssPageName=STRK:MESE:IT

We also do some other Home Security kits for Raspberry Pi:

SecuriPi PIR sensor alarm kit, photos of intruders to your phone £11.99
https://www.amazon.co.uk/dp/B00GOPJJWU

Magnetic sensor window/door alarm kit for Raspberry Pi £12.99
https://www.amazon.co.uk/dp/B00DBDT6TY

48 Infra Red LED lighting kit for Raspberry Pi PiNoIR camera £15.99
https://www.amazon.co.uk/dp/B015OA6VAI

See all our latest blog posts at www.securipi.co.uk