Installing and running Sonospy on your Raspberry Pi

V0.2, April 6 2016, Martin Pergler mp@pergler.org

# Introduction

Sonospy is an extremely flexible free third-party piece of software for serving up music from your music library to your Sonos system. In particular, it lets you get around a couple of Sonos limitations that a lot of music enthusiasts with significant libraries of ripped CDs (or other music tracks stored locally, as opposed to music services in the cloud) have encountered:

* A limitation of 65,000 tracks that can be indexed by the Sonos system itself
* A one-size-fits all set of trees to browse through your collection. For instance, you can browse by Genre, but that means the next menu will be by Artist. This makes sense for lots of types of music, but for instance not classical music, where you’re much more likely to want to browse by Composer next.

Sonospy has been developed by volunteers, principally Mark Henkelis, for about a decade. It runs on many platforms, in particular Windows, Mac, and Linux – essentially any platform that can run the Python environment. Therefore, enthusiasts whose computer is always on when they’re listening to music have for some time had this option available.

However, many of us in home environments who are not complete computer geeks don’t have a Windows, Mac, or full-fledged Linux computer running 24/7. Our music library, if sizable, is probably stored on a NAS drive accessed directly by Sonos, and computers on our home networks come and go as family members connect or disconnect laptops, and perhaps as an old desktop somewhere in the family room gets turned on and off.

Of course, “native” Sonos gets around this by building its own library database, stored on every piece of Sonos equipment in your network. But our goal is to bypass this with a better database – and that needs to run somewhere.

# Why Pi?

Of course, even if your regular computer(s) in your network are not on 24/7, you could always run Sonospy on a spare computer. Those of us who have such a spare don’t need this document.

The other obvious solution would be to run Sonospy on your actual NAS. It is, after all, a tiny computer that serves up content on the NAS’s drives, and by definition is on anytime you would have access to music stored on those drives. I tried setting this up for a while, but ultimately gave up. NAS devices typically run Linux, but with a variety of proprietary software layered on. Therefore you have to fight your way through several obstacles: i) “breaking in” to the command line of the NAS, ii) figuring out how to access the NAS drive data from the NAS computer command line – something that should be easy but often isn’t!, iii) installing Sonospy, iv) installing Python and other necessary packages, and v) installing a hook to run Sonospy when the NAS starts up, and to update its database scans. For many NAS drives, e.g. the popular ones by Synology, all of these steps are surmountable. But they change when you change NASes. Or even when your NAS manufacturer rolls out a software update, which is what happened to me. So I decided I needed a platform fully under my control.

Enter the Raspberry Pi. It costs next to nothing. Runs a standard, open operating system. Fully customizable yet doesn’t “customize” itself without telling you. Not the fastest computer around, but quite adequate for serving up a database tree. And with minuscule power consumption. It’s an ideal platform.

# About this document

These are my notes how I installed Sonospy on a Pi and set it up to serve up music on my NAS to my Sonos system. I’m not a Sonospy, Pi, or Linux geek, but I sort of know my way around computers and am not scared of the command line. If this describes you, hopefully this document will be useful.

Please improve and update it. I’m sure there are various things I’ve missed or done inelegantly.

The Sonospy distribution on github contains some documentation written by Bart(olome) Soriano called “Sonospy Installation and Quick Configuration Guide”. That doc is (currently) written from the point of view of a Windows or Mac installation. Perhaps one day someone will integrate this doc into it. In the meanwhile, you should get a hold of this doc and read it. These notes emphasize what’s different but reading that doc will help you as well.

# Overview of operation

Once we’re done, here’s how the key components work together

1. Your music will be stored on one or more shares on your NAS, just like now.
2. Sonospy, running on the Pi, will build and maintain its own database/index of this music. This is called Sonospy scanning.
3. Sonospy will also run a server process on the Pi that will let you drill down through this database with a custom searching tree. This server will run in a special way (actually one of two ways) so that the server itself is visible as a music source (or several music sources) on all your Sonos controllers. So you and your family members can access music nearly as before, just better.
4. This “special way” is either as a Windows Media Player (WMP) server (even though it’s running on a Pi, not on Windows). This is the one way Sonos generally allows you to access other music on your network not in Sonos’ own dataset. Alternatively, Sonospy can run as a so-called SMAPI service, which is the protocol Sonos has publicized for internet music services you can subscribe to. In this case, as far as Sonos is concerned, Sonospy is just like Spotify or Tunein, except it happens to be streaming from your local network (specifically your Pi) rather than somewhere in the cloud.
5. Depending on exactly how you run it, Sonospy and your Pi may need to transcode your music files on the fly, since not all music formats that Sonos can play natively can be streamed to it. Bah, humbug—and watch out for a bug mentioned below!
6. To reiterate, from Sonos’ point of view, all that has happened is that a mysterious server (or service) has appeared on the network, ready and able to serve up music with its own set of menus/trees. Nothing is changed in your actual music library, and nothing is changed in your Sonos system settings. In particular, all users can continue using their existing Sonos controller software (or hardware), and in fact if they want to bypass Sonospy’s help entirely can continue to browse your music library as they did before you did anything.
7. Finally, the Sonospy software also has the ability, if you want it to, to run as a “virtual” controller point – basically any device on your network can browse to a specific port on your Pi and there is a web-based Sonos controller running there. I personally don’t use this, but the option is there.

All of this comes with a caveat. This is free, enthusiast-maintained software. It doesn’t actually change anything on your Sonos system, so I can’t imagine it could cause any harm. But it may break, or fail to operate as expected. You can ask questions on Google Groups at sonospy-devel, and hopefully someone will answer. But they may not. And Sonos may well change some protocols later and things may stop working.

And by tweaking under the hood like this, you are giving up some of the simplicity that is Sonos’ own selling point. Plug in your Sonos and it basically just works. You have to set up your Sonospy. You need to figure out when and how often you want to update Sonospy’s own database (pt 2 above), etc. etc. Only do this if you have a bit of the tinkerer mindset, and won’t be put out when your spouse phones you when you’re out with friends that “the damned Sonos doesn’t work, what do I do?”

Disclaimer of responsibility: You are doing all such tinkering at your own risk. I don’t see right now what you could do that would cause real irrecoverable damage to you, your equipment, or your files. But if you do, it’s on your head, and not my responsibility. Or that of Mark Henkelis or any of the volunteers on the sonospy project. Even if any of us screwed up and the software is buggy or the instructions faulty.

This is as good a place as any other to remind you that you should have a good backup somewhere of your carefully created music collection, i.e. not be relying on the safety of your NAS!

# Getting started

I assume you have a Raspberry Pi and have set it up connected to your network and can log in to it via SSH. (If not, basic tutorials on the Pi available on the web can help you do that.) Note we will not need any screen or keyboard connected to the Pi. It will just interface with you via the command line and with Sonos and your NAS via your local network.

I also assume you have your music files stored on one or more shares of your NAS.

1. Create a new share on your NAS which will become part of your Pi’s filesystem (the Pi has its own filesystem which is its little SD card, but this is too small and too slow). In these notes I assume both your Pi and this new NAS share are called chickadee, and the NAS itself is named \\lucystorage. So the new share is \\lucystorage\chickadee from Windows, for instance. (Replace with whatever names you prefer). You probably have your NAS with different user accounts with different file permissions by share. Decide what user account you want to have read/write permission to this share; I created a user media with password media. You will need this below.  
    Some users have put the sonospy package in the same share as their music. I don’t recommend that, primarily for security reasons. I like the safety of the Pi (and other media devices on my network) only having read-only access to my stored music, and only specific users, on computers, can mount and access in read/write mode. It also makes various automated scanning easier, since you don’t need to exclude the sonospy directory from scanning locations.
2. Make sure this new share is accessible from your usual computer (you may need to set permissions and/or want to mount it as a drive on that computer). Later on, it will be easier for you to modify configuration files directly from your computer rather than running an editor through the shell on your Pi. On Windows, I recommend Notepad++ as a simple text editor that will happily deal with Unix-formatted text files, which is what Sonospy files are. I don’t know what is the best simple text editor to use on Mac, but you don’t need to worry about the Unix-formatted bit.
3. Access your router’s admin console through your web browser and reserve static local IP addresses to both your Pi and to your NAS, if you haven’t done so already. On my router, that means typing in routerlogin.net in the browser address bar, logging in with your admin username and password for the router (hope you remember that!), then choosing LAN Setup / Address reservation from the menu. I assigned 10.0.0.15 to LucyStorage, my NAS, and 10.0.0.17 for Chickadee, my Pi. You could probably get by without doing this, but there are places in the steps below where it is easier to hardcode the NAS IP address, and it saves timeouts and resets when either device would reboot or be taken offline and then come back up with a different IP address while your Pi or Sonos would still be looking for the old one.

From here on in, always change chickadee, LucyStorage, 10.0.0.15, and 10.0.0.17 to the appropriate names and IP addresses in your setup!

# Making Pi and the NAS talk with each other

Log in to your Pi and then enter superuser mode via sudo -i

1. If you haven’t done so, “baptise” your Pi with its new name

nano /etc/hosts Change raspberrypi to chickadee or whatever

nano /etc/hostname Ditto

/etc/init.d/hostname.sh

1. Tell chickadee about your NAS and mount the shares

apt-get install cifs-utils Install package for accessing shares via CIFS protocols

mkdir /home/pi/LucyStorage/chickadee   
Create link location where the NAS share from step 1 will mount

mkdir /home/pi/LucyStorage/music   
Ditto for the share where you have your music stored. I have mine stored in 2 shares, cmusic (for classical music) and music (other music). If you have multiple shares as well, do this for each one replacing music with the appropriate name in each instance.

From here on in, change music and cmusic always to the name of your music share(s)

nano /etc/fstab Add the following, on only one line:

//10.0.0.15/chickadee /home/pi/LucyStorage/chickadee cifs workgroup=WORKGROUP,username=media,password=media,users,auto,user\_xattr,exec 0 0

Tell the Pi OS about the share which will have sonospy (and other) files. Change the username and password to whatever you used in step 1. Note the exec parameter, which is needed to be able to execute Sonospy’s python scripts off of this share; this took some time to figure out!

//10.0.0.15/cmusic /home/pi/LucyStorage/cmusic cifs workgroup=WORKGROUP,username=media,password=media,users,auto,user\_xattr 0 0

Likewise set up access to the share with music. Repeat with other music shares each on its own line.

rpcbind (Save the file, now back at the shell). Not sure this is needed, but the script I was following for making fstab changes did it, and so did I.

mount –a Actually mount the NAS shares

Note: henceforth, if you ever reboot your Pi (deliberately, with shutdown –r now or accidentally), all you need to run from the shell to remount the drives is sudo mount –a

# Install and basic test of Sonospy

1. Download Sonospy from github

cd ~/LucyStorage/chickadee

git clone git://github.com/henkelis/sonospy.git

At this point, I renamed (from Windows) the sonospy directory that has been unpacked into sspy. You don’t need to do that, but there is a nested subdirectory in there also called sonospy and I knew I would eventually get confused if I didn’t do something.

1. Set up port numbers in pycpoint.ini using the instructions in Bart Soriano’s .doc file (in the distribution you just downloaded). I’m actually not sure if it was necessary to change from the defaults in the pycpoint.ini file but I did it anyway and it worked. The good news is that your little Pi assumes it is safely protected behind a network firewall, and therefore you don’t need to jump through any hoops to “open up” ports on it.
2. Run a small-scale test scan and try the proxy service, again following Bart Soriano’s instructions.
3. (Re)install ffmpeg if it will be needed. ffmpeg is a cross-platform package for audio (and video) file format conversion. If you will only be running sonospy in WMP server mode (see the overview of operation), you do not need to worry about this step.  
   If you will be running sonospy in SMAPI service mode, unfortunately Sonos won’t accept your Pi serving it music in some formats that it does accept natively (or using WMP). If all your music is mp3 or flac format, you also don’t need to worry about this. But if, like me, you have e.g. Apple Lossless files in ALAC format (may have e.g. ..m4a extension), sonospy will need to transcode them for Sonos. ffmpeg can do this, however it requires the ability to create mp3 files, and by default for complex intellectual property reasons the Pi operating system’s distributed compiled version does not include this built-in.   
   You have 2 choices: 1) compile your own version of ffmpeg with the LAME mp3 encoder included (the source code can be freely downloaded, it’s just free distribution of the binaries which is difficult) – instructions are available on the net in various places. 2) download one of several volunteer-compiled and under-the-radar distributed Pi ffmpeg binaries that do include it. At time of writing, one such binary is referenced and available through another package on github at <https://github.com/ccrisan/motioneye/wiki/Install-On-Raspbian>   
     
   Bug alert! *At time of writing (April 2016), using ALAC files with Sonospy is not recommended. First, for some reason, Sonos doesn’t like ffmpeg’s streamed output (tracks will not play). Second, there seems to be a bug in extracting “Composer” tags from ALAC files for use in Sonos’ indexes. Mike Henkelis is looking into these problems (not just Pi-related), but since the industry has effectively switched from ALAC to FLAC, if you have ALAC files I recommend you offline manually convert them to FLAC. This is a lossless translation. On Windows, dbPowerAmp (which you may have already used for ripping CD tracks) batch converts very swiftly – on my PC about 80x(!) faster than the Pi, with the top (lossless) compression setting saving about 3% space versus the size of the ALACs. So do it there, not on the Pi!*

# Setting it up for real

You now have all the necessary ingredients installed to get sonospy to actually serve up your music collection property.

1. First, you need to run a proper scan of your whole music library. If you need to, later you can get fancy, but for a first scan do something like this. Note this will take a long time; I have about 11000 tracks and it took 6 hours. You might want to run overnight. I have 2 separate shares with music that I will want to serve up with different indexes, hence the two separate scans. Adapt as relevant for your situation.

cd ~/LucyStorage/chickadee/sspy/sonospy  
Move to sonospy’s own directory

./scan.py -d Classical.db ~/LucyStorage/cmusic > Classical.log  
Do a scan of one music library into its database, creating a log of what was done in a file in this directory

./scan.py -d Nonclassical.db ~/LucyStorage/music > Nonclassical.log  
Do the same on another music library. Note, if you wanted to serve up both sets of music in a mixed index, you could use the same .db name and it would be added. There are fancy indexing options how you could separate out what is served up in what index at runtime rather than at scan time, but in our environment it’s easiest to keep it all separate. (If your music libraries belong to different family members, this will let them cleanly maintain music files in different directories/shares and manage their own index files)

1. Now, test it out with the default index structure. First, start the sonospy proxy server:

cd ~/LucyStorage/chickadee/sspy   
If not there already. cd .. is enough if you are in the sonospy directory from the previous step

./sonospy\_proxy -wSonospy=Classical,Classical.db –r

The above assumes you want to run in WMP server mode. If you want to run in SMAPI service mode, use –sSonospy=… instead of -wSonospy. (Note to test we are running only one database. To run several, see syntax below in the scripts step). You can change the =Classical name to whatever you would like the index to be named in your Sonos controller, but the part after the comma needs to refer to whatever you named the database in the previous step.

WMP server or SMAPI? SMAPI allows searching your tracks, and serves up album art. But SMAPI won’t accept all formats of file Sonos can actually play (see point #9 above – MP3 and FLAC are fine) and uses a bit more “fakery” to work that Sonos could disable at some point in the future.

If you are running in WMP server mode, you should see “Classical” (or whatever) automatically appear in all your Sonos controllers’ top menu (alongside Music Library, Sonos Favorites, etc.) whenever sonospy is running. If you are running in SMAPI service mode, you will need to choose “Add music services” and find it there first; it will then appear permanently in the top menu (including sitting there even when sonospy is not running, when it will appear but not work!)

./sonospy\_stop To kill the proxy once you’ve finished testing it out.

1. So far, at best your sonospy set up is very boring – since the indexes it displays are more or less exactly what Sonos would serve up on its own. Now comes the step why you’re probably doing all of this – setting up custom user indexing. This is beyond the scope of this document. There is useful info in userindex.ini and defaultindex.ini in the sonospy directly, and a variety of discussions on the sonospy-devel Google group.
2. At some point in playing around with custom indexing, you’re probably getting quite tired of typing in arcane command line parameters. It is best to set up 2 new script files (and modify one existing one) to make rescanning, starting, and stopping sonospy easy. Each of these belong in the sspy (not sspy/sonospy) directory.

New script sonospy\_scan

#!/usr/bin/env sh

# -\*- coding: utf-8 -\*-

cd /home/pi/LucyStorage/chickadee/sspy

echo `date` sonospy\_scan started >> sonospy.log

cd sonospy

./scan.py -d Classical.db ~/LucyStorage/cmusic > Classical.log

./scan.py -d Nonclassical.db ~/LucyStorage/music > Nonclassical.log

cd ..

echo `date` sonospy\_scan finished >> sonospy.log

New script sonospy\_start (note the proxy line is just one run-on line with –w not – (space) w or – (newline) w, and it shows how to start multiple indexes at once)

#!/usr/bin/env sh

# -\*- coding: utf-8 -\*-

cd /home/pi/LucyStorage/chickadee/sspy

./sonospy\_proxy -wSonospy=Classical,Classical.db,userindex\_Classical.ini -wSonospy=Nonclassical,Nonclassical.db,userindex\_Nonclassical.ini -r

# change as needed, e.g. -s rather than -w, .db and .ini files

echo `date` sonospy\_start `cat pycpoint.pid` >> sonospy.log

Change existing script sonospy\_stop (the two lines in italic are new)

#!/usr/bin/env sh

# -\*- coding: utf-8 -\*-

*cd /home/pi/LucyStorage/chickadee/sspy*

if [ -e pycpoint.pid ]

then

kill `cat pycpoint.pid`

rm pycpoint.pid

*echo `date` sonospy\_stop proxy >> sonospy.log*

fi

if [ -e web2py.pid ]

then

kill `cat web2py.pid`

rm web2py.pid

fi

Four comments. First, if you’ve mounted your chickadee share from your NAS as described in this document, you don’t need to set executable permissions for these scripts. If you’ve done something else, you might need to do a chmod u+x sonospy\_scan etc.

Second, since these and the rest of the sonospy files are stored on the NAS, you don’t have to create these scripts from within the Pi command line. You can use an editor on your computer, as long as it can handle Unix-style text files.

Third, you will see that I’ve hard coded in some directory changes – not generally good programming practice. This is in anticipation of running those scripts in the next session in an automated manner at bootup (sonospy\_start) and periodically (sonospy\_scan), when one can’t be sure the working directory is correct at startup.

Finally, the lines starting echo `date` implement a primitive overall logging mechanism, so that you can easily see when the last (re)scan was, and whether the sonospy proxy is running. This is in addition to the more detailed log files done with each scan and proxy run.

# Automated operation

1. You probably want the sonospy proxy (via sonospy\_start) to run at Pi bootup, so if the power goes down your new and improved “Sonos” doesn’t mysteriously vanish. Or if it does for some reason, and your panicking family members call you at work, you just guide them how to power cycle the Pi rather than talking them through logging in via ssh and running a script.

This ought to be easy, but I faced unexpected challenges, since sometimes the NAS shares would automount and sometimes they would not, and sometimes it would be quick and other times it would take a few seconds.

As a first step, you need to create one more script file, not in the sspy directory (which is on the NAS) but in your pi’s home directory (on its own SD card). This you have to do on your Pi, not on your computer on the NAS. One option is to type in nano ~/startup.sh and then enter the following:

#!/usr/bin/env sh

# -\*- coding: utf-8 -\*-

sleep 15 # wait for network related stuff to load, maybe shorten

sudo mount -a # ensure the NAS drives are mounted, sporadically done at start

~/LucyStorage/chickadee/sspy/sonospy\_start

Finally, you need to tell Pi to run this script at startup. There are multiple ways to do this, one is via crontab which we will also use in the next step.

crontab -e Edit crontab entries. At the end, add the line

@reboot sh /home/pi/startup.sh &

The & at the end is important. It lets execution of other startup commands continue while your script sleeps for them to catch up.

If you now reboot (with sudo reboot) then after less than a minute, you should see sonospy restart – and add a line to sspy/sonospy.log that it has done so.

1. The final step is to set up automatic database rescanning – as mentioned at the start, sonospy has its own databases, so changes to music file tags or new rips added to your library will not be automatically reflected, even though Sonos will itself reindex its own controller database. Those are independent. Crontab again comes to the rescue; add the following line via crontab –e

0 4 \* \* \* /bin/sh /home/pi/LucyStorage/chickadee/sspy/sonospy\_scan

The syntax here is particularly arcane (an example of how Linux was designed by geeks for geeks), but it means every day at 4 hours and 0 minutes, run sonospy\_scan. If you wanted to run it instead at 3:15am and only on the 6th day of the week, the syntax would be 15 3 \* \* 6 instead. Once again, any time the (re)indexing is triggered, you will see an entry in sspy/sonospy.log. On my system (about 11,000 tracks), it runs for about 10 minutes in the background if there have been no or very minimal changes.

Database integrity warning. As the developers of Sonospy have written, the database architecture sonospy uses prioritizes speed (when serving up content) at the expense of database size and scanning speed. In particular, various things are stored in separate tables. It is possible to corrupt your database quite insidiously with an unexpected Pi powerdown during the scanning process. This is statistically unlikely during a daily small update than adds 10 newly ripped tracks and that’s it. But it can easily happen if you’ve done some large-scale metadata retagging of your music tracks. After a large update, keep an eye out for matching “scan started” and “scan finished” entries in the logfile. If the scan did not finish, best to delete that database and rebuild it in case you’re unsure whether corruption could have occurred. Note an unexpected powerdown while just the proxy is running is not a problem; the proxy will restart after powerup via Step 14 above.

# A final plea

1. Hopefully by now, sonospy will be working automatically and nearly invisibly on your system. You will have better access to your music files. Hope it was not too painful. A reminder and request: Please improve this document with whatever you have learned so future users will have it easier. And join the sonospy-devel Google group to tinker with like-minded individuals to set it up to your satisfaction!