

Step by Step: Using the Rpi400 Jamulus Image

Sunday, December 6, 2020 1:12 PM

1. To make things as simple as possible, we've prepared an SD card image file that contains the official Raspberry Pi OS, pre-configured, and with Jamulus pre-installed. This card image is called **rpi400_jamulus.img.gz**, and can be downloaded using the link in Step #0 of the README.md file at the following github site:

[Link to latest rpi400_jamulus image](#)

This page also shows how to get a copy of this PDF file.

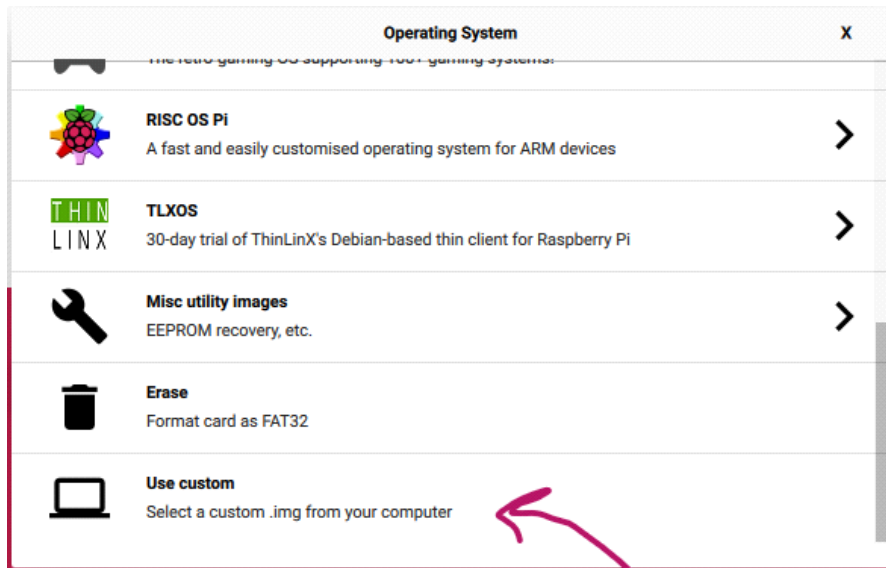
2. Next, we're going to use a tool from the Raspberry Pi foundation to load the card image into the SD card. This tool is called "Raspberry Pi Imager" and there are versions for Mac or PC.
3. If you don't already have the Raspberry Pi Imager installed, download the Raspberry Pi Imager from the link below and install it:

[Raspberry Pi Org download page](#)

1. Run the Imager program. It's main dialog should look something like:



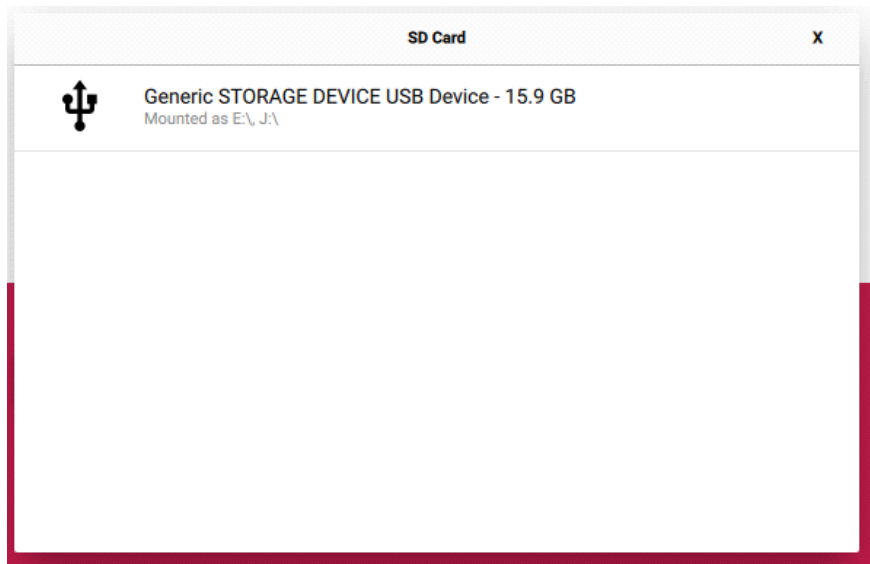
2. Next, you need to tell the program to use the SD card image from Step 1 as the source file. Click on the "Choose OS" button, and then scroll down to the very bottom of the dialog that pops up and select the "Use custom" entry.



3. Click on the "Use custom" entry, which opens a browser dialog, and browse to where you stored the **rpi400_jamulus.img.gz** file and select it. Then press "Open". The browser dialog will close, and the Raspberry Pi Imager dialog should now show that file as the "operating system":



7. Next, insert the SD card that came with your Raspberry Pi 400 (or some other SD card at least 8 GB in size) into a card reader for your Mac or PC. Verify with your operating system that you can see it. Then click on the "Choose SD Card" button on the Imager dialog, and select the appropriate drive.



WARNING: Though the Imager is pretty good at only showing drives with SD cards in them, be careful. Be sure the drive you choose is in fact the one for your SD card, because the next step is going to wipe out the contents of whatever drive you pick! Note that in Windows the drive might show more than one driver letter if you've used this card before in a Linux machine. That's because the card might have several partitions from a previous OS installation. Don't worry about it. The card will be completely overwritten.

8. After selecting the right drive, your Imager dialog should now look something like:

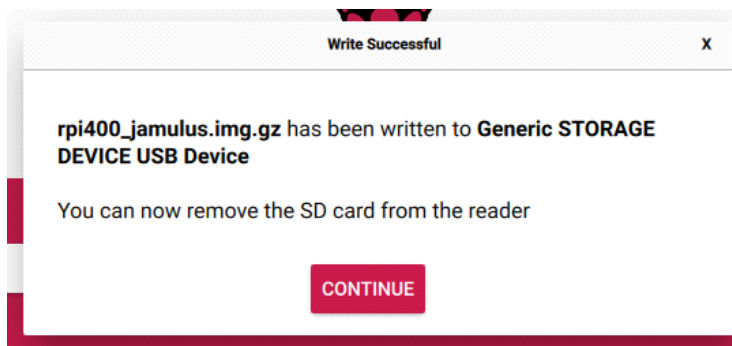


The name of the SD card should say "Generic Storage" or some other suggestive name, but be sure it's not your C: drive or any drive that has data you care about! If you goofed, just click on the button again and select another drive.

9. Next, click on the Write button. Confirm from the popup dialog that you have the right drive and wish to continue. If you say yes, the Imager will start writing to your SD card. It may take a while (5 minutes or whatever.) So be patient. During the writing process, the Imager dialog will look something like:



10. It will do a write pass and then a verify pass, and if all is well, it will eventually tell you that you can remove the card from the drive and you are good to go! You can close the imager program at this time.



WARNING: After all this, Windows may pop up a dialog asking if you want to format the drive. Just ignore that dialog and close it. The reason for this dialog is that the image you just wrote is in Linux format, which Windows doesn't recognize and thinks it's a drive that needs formatting. Again, ignore and close any such dialog!

11. After removing the SD card from the card reader, you can insert into the Raspberry Pi 400 keyboard. The SD card slot is located on the back about halfway, and is a small, thin, spring loaded slot. Push the card in (label side up) until you hear a click. Now, if you've got your monitor and mouse connected, fire up the Raspberry Pi and let it boot up. Because of the way the boot image was created, you'll see a series of flashes and flickers and blank screens. Just let nature take its course. In a minute or two, the Pi should boot up to the desktop screen:



NOTE: This OS image was configured using a 1920x1280 sized monitor with a 60 Hz refresh rate. If your monitor is different, the screen image might not be sized right, too large or too small with a black border. It's not necessarily a trivial process to fix this once the OS has been installed. We're working on an easy solution. But with any luck, you'll have a readable screen.

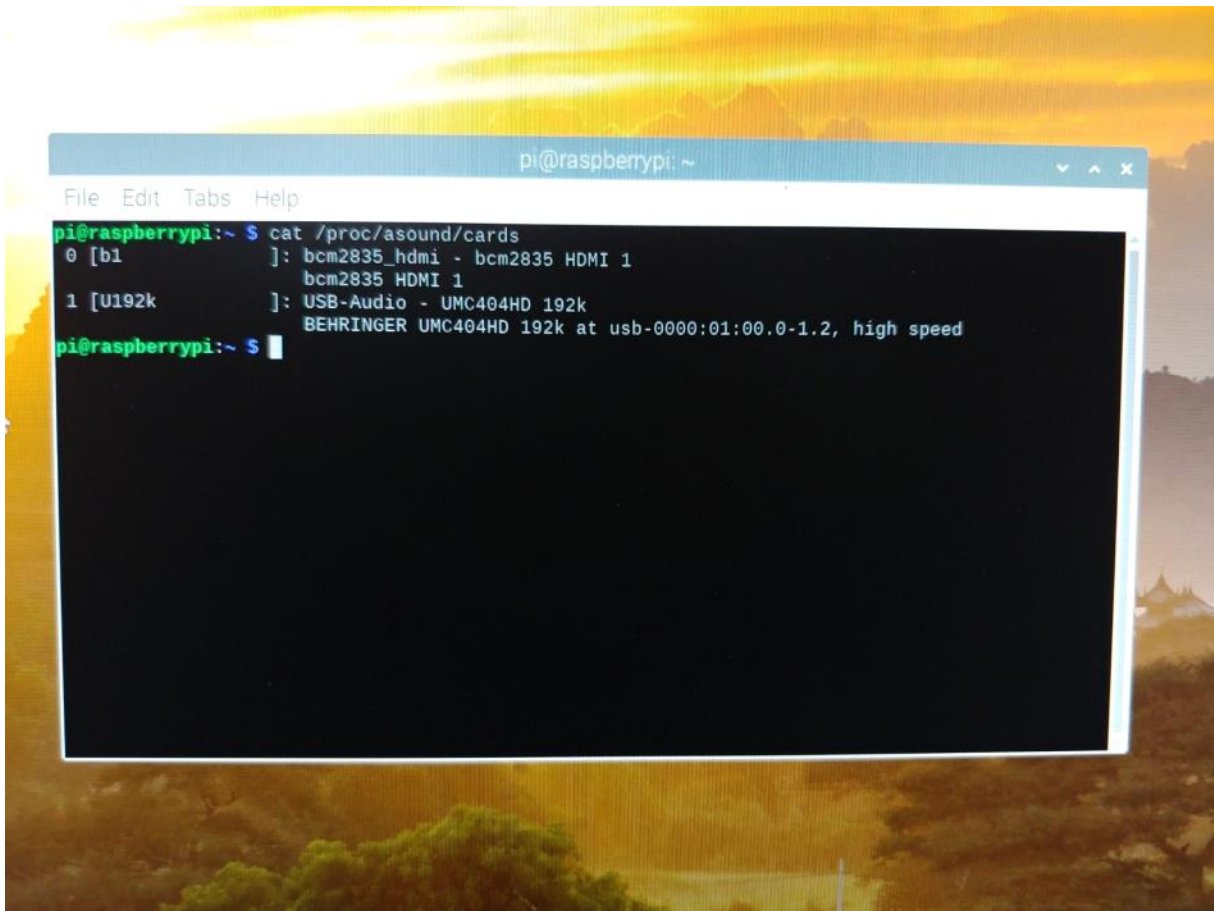
12. Okay, then! You have a Raspberry Pi with Jamulus pre-loaded. You can verify this by clicking on the Raspberry Icon and then selecting the "Sound & Video" menu entry. You'll see Jamulus listed there with its cloud icon. Now, DO NOT CLICK ON THAT LINK. Not yet. We still have a few steps to do to customize Jamulus for your audio device. So just back out of the menu.
13. Jamulus works best with a USB audio device. Many of the popular USB audio devices like the FocusRite ones, such as Scarlett Solo, Scarlett 2i2, Scarlett 4i4, etc, will work in Linux just fine. So will many of the Behringer boxes, such as the UMC202HD and UMC404HD. Basically, any USB audio device that is certified as a USB compatible device should work in Linux.

NOTE: We don't recommend using any of the cheaper Behringer boxes, such as the UM22, etc -- basically anything that doesn't have a good driver (and we don't consider the ASIO4All driver suitable for Jamulus).

14. Next, plug your USB audio device in to one of the USB slots. We highly recommend you use one of the Blue USB 3.0 slots, since there is a better chance your audio device will communicate as quickly as possible and will have less delays and buffering problems.
15. To confirm that your audio device is recognized by Linux, open up a terminal window (using the black icon at the top menu) and type the following in at the command prompt. (Below, we show a \$ for context. Don't type that \$ here, or anywhere else in this document. It's just to give you context:)

```
$ cat /proc/asound/cards
```

You should get back a list of installed cards. For this Rpi400 machine, you'll likely have only two audio devices -- the built-in one, and your USB Device. Chances are, your device will be listed last. Here's the result we got for our machine:



In our case we have a Behringer UMC404HD plugged in, and you'll see that last on the list. The other "sound card" is the built-in device for HDMI audio.

16. Pay attention to the left side of the entry for your card. That shows the shorthand "name" of the card that we'll be using in a little bit. In our case, that name is "U192k". NOTE #1: Linux is case-sensitive, so pay attention to the case. Here, the 'U' is upper case and the 'k' is lower case, for example. NOTE #2: IF you have one of the popular FocusRite Scarlett Solos, 2i2s, 4i4s, there shorthand name is most likely just simply "USB".
17. Next, we're going to set up the "Jack" audio service that Jamulus connects to internally, to use your USB audio device. We've made a script for this called "jack_setup.sh". Run the script as follows, where "xxx" should be replaced with the shorthand name of your audio device as given in Step 15.

```
$ jack_setup.sh xxx
```

For example, here's how our command line would look:

```
$ jack_setup.sh U192k
```

For many of the popular Scarlett's:

```
$ jack_setup.sh USB
```

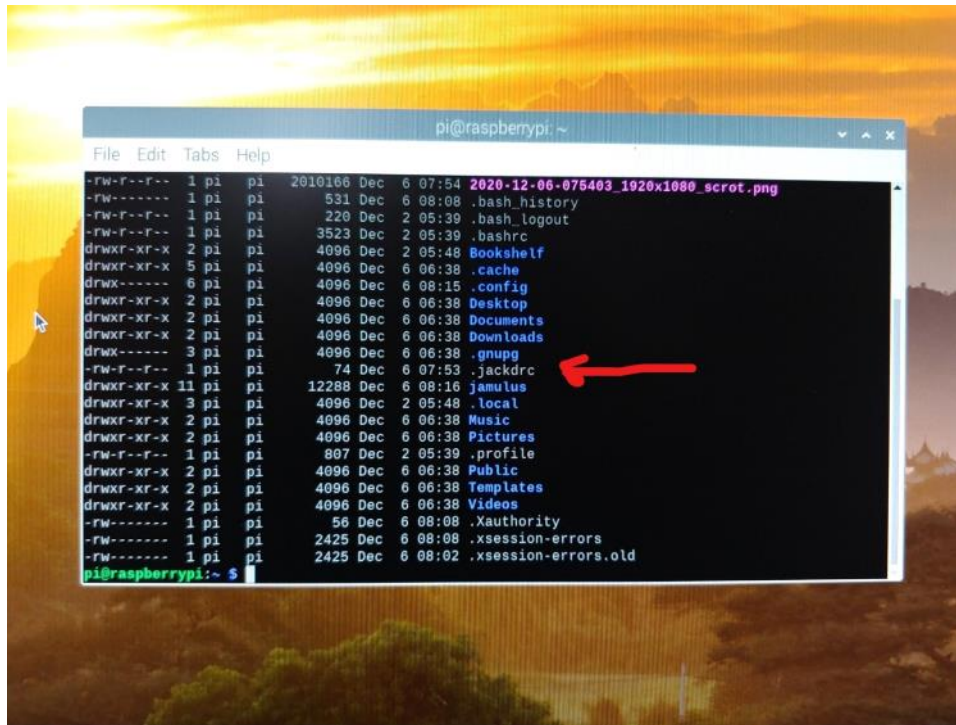
18. What the jack_setup script does is write a hidden file called ".jackdrc" (notice the beginning dot) to your home directory. That is, the file path is "/home/pi/.jackdrc". You won't normally see this file, but it's easy to verify it's there. First, ensure you are in your home directory by entering the command:

\$ cd

And then type in the command:

\$ ls -al

A list of files and sub-directories should appear, and .jackdrc should be one of them.



If you don't see this file, then something went wrong with the script. And then what? Well, "contact your system administrator". I knew you'd like us saying that!

19. For the technically inclined among you, it's useful to see what's in that .jackdrc file. You can see its contents by entering the command:

\$ cat .jackdrc

Ours shows the following:

```
"/usr/bin/jackd -T -P95 -p16 -t2000 -d alsa -dhw:U192k -p 64 -n 2 -r 48000
```

In English, this script is used to start up the Jack daemon with the specified parameters: it gives Jack a priority of 95 (so the audio communication is as fast as possible), it allows up to 16 input channels of audio, and uses the audio device called "U192k". Next, the last three entries are important, for they determine how well Jamulus is going to run. The "-p 64" entry says to use a buffer size of 64 samples, the "-n 2" entry says to use 2 periods, and the "-r 48000" says to use a 48kHz sample rate, which is what Jamulus **requires**.

Now the buffer parameters given here as defaults should work for all the decent quality USB audio devices. (If your audio device didn't cost you \$100 or more regular price, then it's probably not good enough quality). Even so, the parameters may not be appropriate for your setup. You'll know this because when running Jamulus you'll get lots of drop outs, clicks, pops, etc (these are called

xruns), so bigger buffers might be needed. For example, some people might need to set the "-n" entry to 3 periods, instead of 2, and/or set the buffer size to 128 samples, etc.

To change these defaults, you'll have to edit the .jackdrc file manually. That's out of scope of this document. Hopefully, what we give here will work fine.

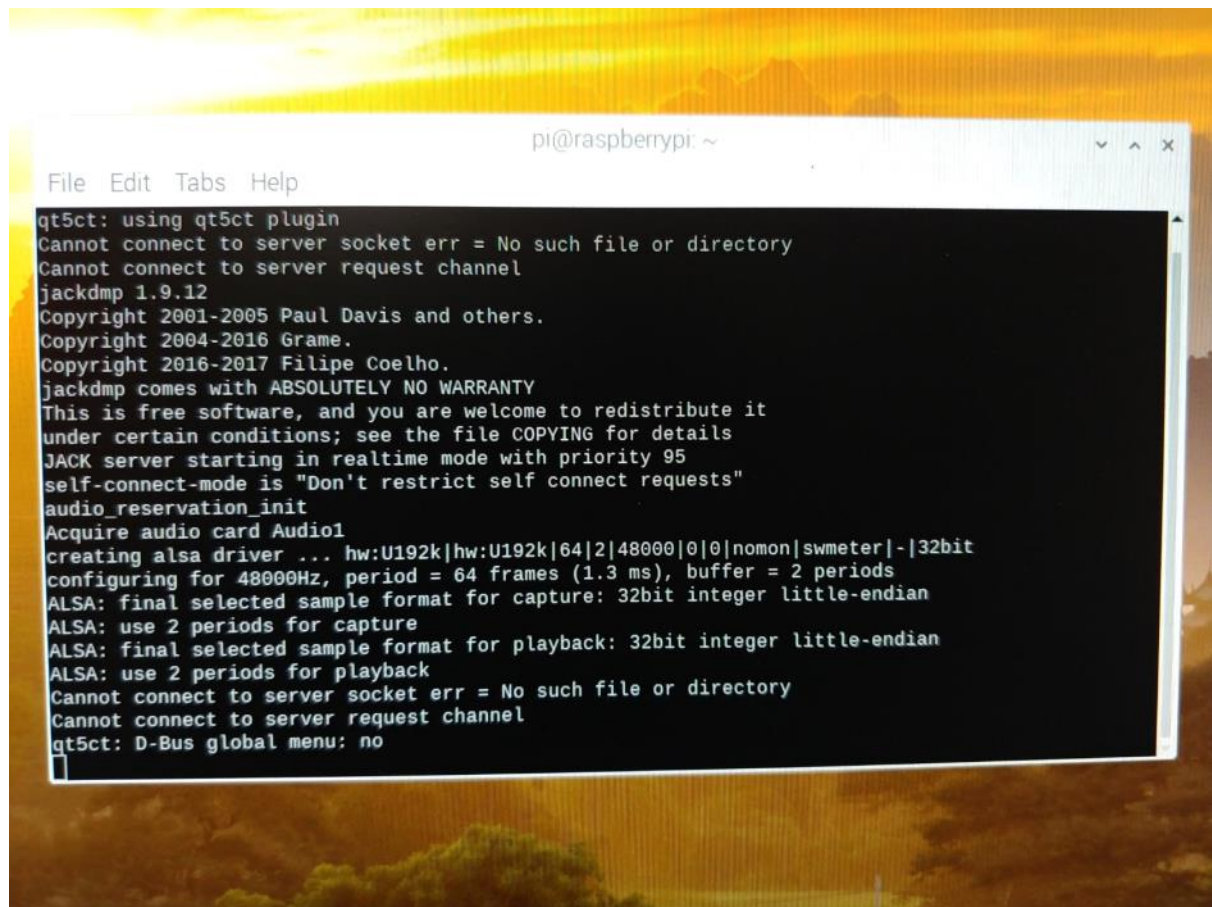
P.S. Oh and another thing: We've limited Jack to 16 channels here -- that's what the "-p 16" entry means. So no plugging in more than 16 microphones! Leave your choir back on the bus! Also, the "-T" parameter means "automatically terminate Jack after all Jack clients are finished." That way, Jack won't be running unless Jamulus is running (or some other audio application.)

1. Okay, then. You are ready to try Jamulus! Now, we noted earlier that you can conveniently run Jamulus from the GUI menu entry under "Sound & Video." However, for this first attempt, we are going to run it from the command line so we can see if it's setup properly. To do so, use the following command from the prompt:

```
$ Jamulus
```

NOTE: We remind you that Linux is case-sensitive. In our setup, the Jamulus executable is capitalized, so be sure to type that upper-case 'J', or it won't work!

20. After entering the command, Jamulus starts and spits out a bunch of messages. We're not going to worry too much about them, but will point out a few things that will tell us whether things are setup right. Here's a typical screen dump you should see the first time you run Jamulus:



```
pi@raspberrypi: ~
File Edit Tabs Help
qt5ct: using qt5ct plugin
Cannot connect to server socket err = No such file or directory
Cannot connect to server request channel
jackdmp 1.9.12
Copyright 2001-2005 Paul Davis and others.
Copyright 2004-2016 Grame.
Copyright 2016-2017 Filipe Coelho.
jackdmp comes with ABSOLUTELY NO WARRANTY
This is free software, and you are welcome to redistribute it
under certain conditions; see the file COPYING for details
JACK server starting in realtime mode with priority 95
self-connect-mode is "Don't restrict self connect requests"
audio_reservation_init
Acquire audio card Audio1
creating alsa driver ... hw:U192k|hw:U192k|64|2|48000|0|0|nomon|swmeter|-|32bit
configuring for 48000Hz, period = 64 frames (1.3 ms), buffer = 2 periods
ALSA: final selected sample format for capture: 32bit integer little-endian
ALSA: use 2 periods for capture
ALSA: final selected sample format for playback: 32bit integer little-endian
ALSA: use 2 periods for playback
Cannot connect to server socket err = No such file or directory
Cannot connect to server request channel
qt5ct: D-Bus global menu: no
```

The things to watch for is the line that says "creating alsa driver" which should show some of the parameters that are in the .jackdrc file, such as device name, (U192k), buffer size, (64) number of

periods (2), sample rate (48000). If you don't see these, something is wrong.

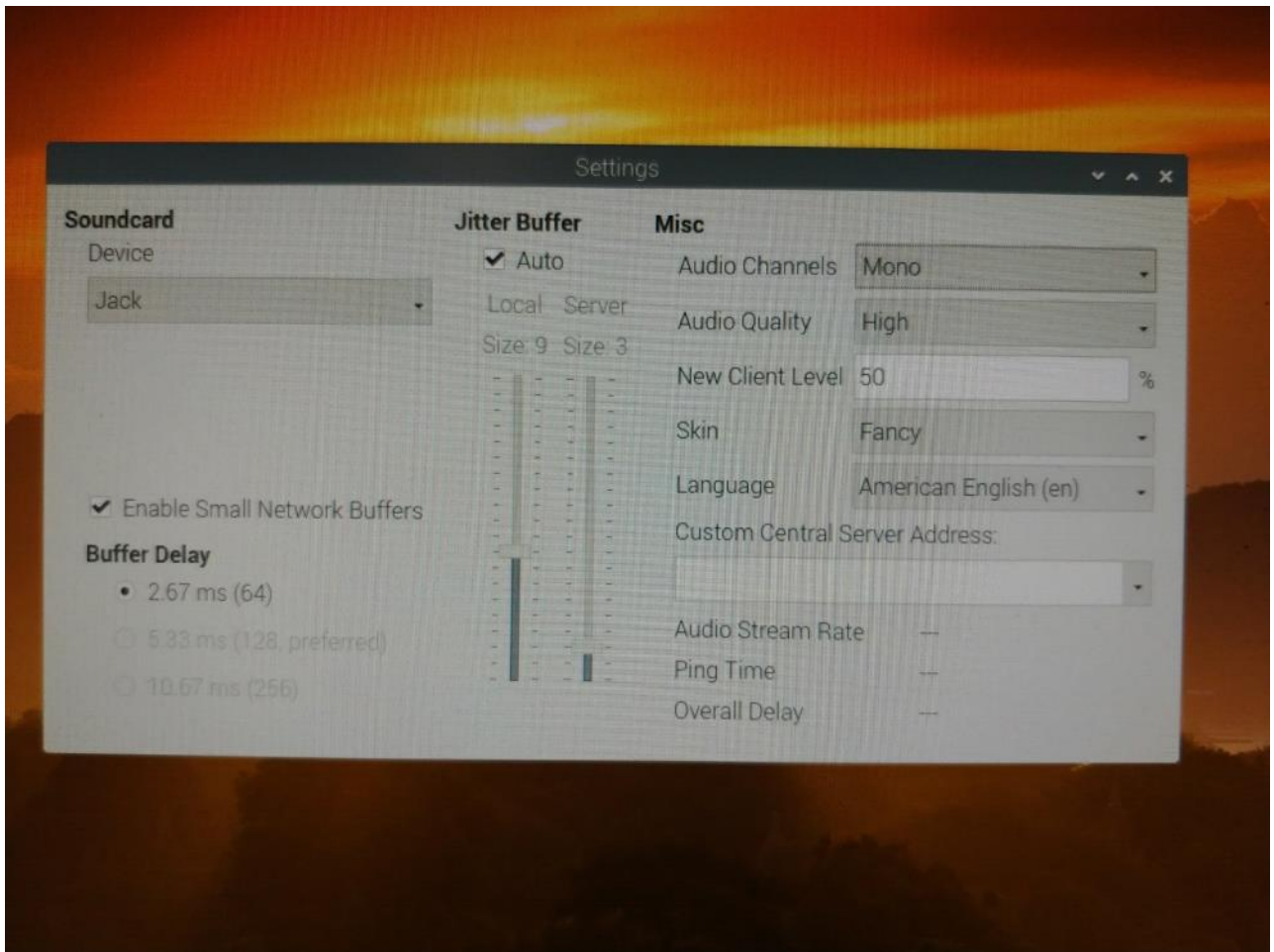
The main thing to watch out for is that you should see no error messages saying "couldn't run with real-time priority", "couldn't allocate memory", etc. These are indications that Jack isn't set up right.

The rest of the stuff there isn't too important. You shouldn't see any other errors, except possibly a single "socket err = No such file or directory". We don't know why this error occurs, but it seems to be harmless, and seems to only show up the first time you run Jamulus. Subsequent runs won't usually show this error.

NOTE: IF you see multiple "socket errors" it means that the SD card image wasn't set up right. Contact your System Administrator :)

Other types of errors you might see are "process error", or something indicating it couldn't process things in time. This is indicative of "xruns", and means you probably need to increase the buffer size. (Again, that's outside the scope of this document.) If you see a couple of these errors, it's no big deal, but a bunch of them indicates a problem. (And you'll hear it too when you actually use Jamulus)

21. Okay, then. Let's try out the Jamulus GUI. It should have popped up after you entered the Jamulus command, and it's in a window separate from the command line. The first thing to do is go into Settings (how to use Jamulus is out of scope of this document), and notice that the "Device" selected is "Jack". Also, you may wish to click the "Enable small network buffers" option, which will let Jamulus run as fast as possible with as small a latency as possible. But if later you get too many xruns, (pops and clicks") you might need to turn this off. Overall, here's what our settings dialog looks like:



Some optional things to do here: Set the audio quality to "high", and set the new client level to 50%. We like to use "high quality" audio, especially with things like fiddles (you'll hear a hissing sound when bowing the strings), unless we use high quality. I'm sure other instruments will sound better too (like cymbal crashes). Your Raspberry Pi 400 shouldn't have trouble handling the higher bandwidth (and your internet connection is likely also fine.)

23. Okay, then. It's time to connect to a server and start playing! Go back the main dialog, press Connect, select an server, and see what happens! Hopefully, if you are lucky, after things have settled down for a minute or two, you'll have reasonable ping times, (under 20ms is good), and reasonable overall delay (under 40 ms is good). And you'll have decent audio quality most of the time. One of the advantages to using a Raspberry Pi for Jamulus, like we've done here, is that you have a dedicated machine, just for Jamulus, with not much else running. So you should see good latencies with few drop outs.

NOTE: In case you forgot, BE SURE TO USE AN ETHERNET CABLE for a your internet connection, NOT WIFI! Otherwise, you'll likely experience nasty drop outs and high latencies.

24. Alrighty, then! Have happy days jamming with your friends on your new dedicated Jamulus player.