Step by Step: Using the Rpi400 Jamulus Image

Sunday, December 6, 2020   1:12 PM

1. **Introduction**
   The Rpi400 Jamulus SD card image is an normal Raspian OS image built for the Raspberry Pi 400 (we'll call this the Rpi400 throughout this document) that has been pre-configured with Jamulus installed and almost ready to go. (There's still one thing the user needs to do to connect their audio device of choice up to Jamulus.) It will work for any SD card that's 8 GB or larger. Upon first boot in the Rpi400, the OS installation will be expanded to the size of the card.
   NOTE #1: Though built for the Raspberry Pi 400, this image will probably also work with any of the Raspberry Pi 4 models as well, though that hasn't been tested yet.
   NOTE #2: The card image was set up with a locale of Phoenix, AZ, Mountain Standard Time, (-7 UMT), using American English and a US Keyboard. You may have to change some of those settings using the OS Preferences entry in the Raspberry Pi GUI. Also, the user name is set to "pi", and the password is set "music". You might want to change the password to something more secure.
2. The card image has a file name of **rpi400_jamulus.img.gz**, and can be downloaded using the link in Step #0 of the README.md file located at the following github site:

   Link to latest rpi400_jamulus image

   The README.md file also shows how to get a copy of this PDF file.

3. Next, we're going to show how to flash the card image onto an SD card, using the **Raspberry Pi Imager**. This program is available for Mac, PC, and Linux. If you don't already have the Raspberry Pi Imager installed, you can download it from the link below:

   Raspberry Pi Org download page

   NOTE: You can also use other tools to flash the image. One such tool is the popular **Balena Etcher**. However, the **Raspberry Pi Imager** is a bit simpler as it can work with images that haven't been downloaded to your computer, but instead, are accessed directly from the web, such as the different sanctioned OS's for the Raspberry Pi.

4. The main dialog for the Raspberry Pi Imager should look something like:

5. 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.
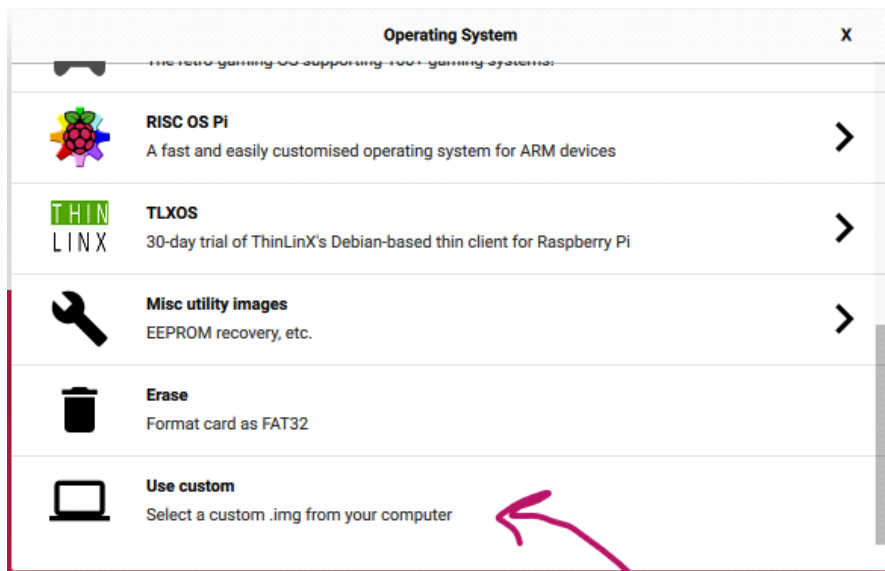
6. Click on the "Use custom" entry, which opens a browser dialog, and browse to where you stored the **rpi400_jamulus.img.gz** file and select it. Then press "Open". The browser dialog will close, and the Raspberry Pi Imager dialog should now show that file as the "operating system":
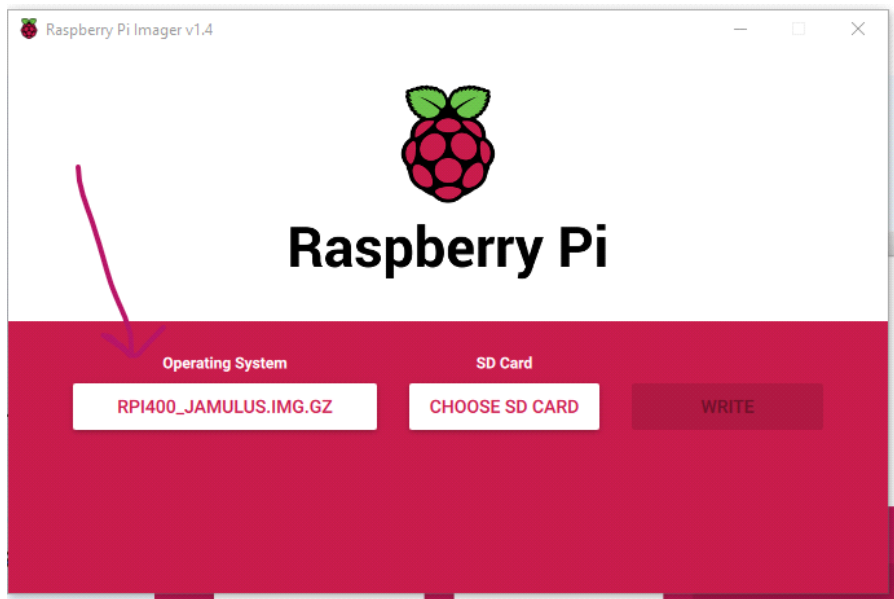
7. Next, insert the SD card that came with your Raspberry Pi 400 (or some other SD card at least 8 GB in size) into a card reader for your Mac or PC. Verify using your operating system that you can see the card. Then click on the "Choose SD Card" button on the Imager dialog, and select the appropriate drive.
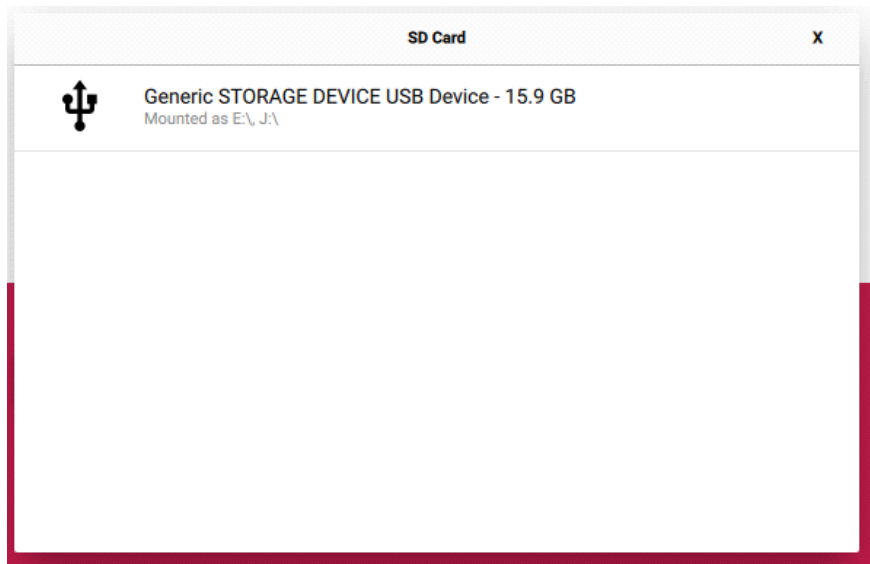
WARNING: Though the Imager is pretty good at only showing drives with SD cards in them, be careful. Be sure the drive you choose is in fact the one for your SD card, because the next step is going to wipe out the contents of whatever drive you pick! Note that in Windows the drive might show more than 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.

7. 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.

8. 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:
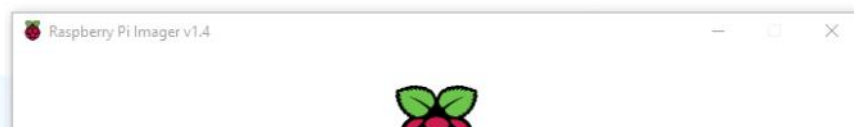
9. It will do a write pass and then a verify pass, and if all is well, it will eventually tell you that you can remove the card from the drive and you are good to go! You can close the imager program at this time.
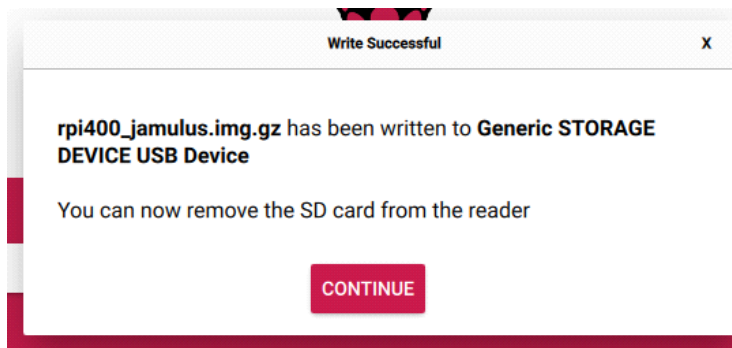
WARNING: After all this, Windows may pop up a dialog asking if you want to format the drive. Just ignore that dialog and close it. The reason for this dialog is that the image you just wrote is in Linux format, which Windows doesn't recognize and thinks it's a drive that needs formatting. Again, ignore 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 ones from FocusRite, such as Scarlett Solo, Scarlett 2i2, Scarlett 4i4, etc, will work just fine in Linux. So will the Behringer boxes such as the UMC202HD and UMC404HD. Basically, any USB audio device that is certified as a compliant USB audio device should work in Linux.

   NOTE: If you choose to use the cheaper Behringer boxes, such as the UM22, etc., please note we haven't had the best of experiences using it. The Behringer UMC202HD however, makes for a great device, as well as any of the Scarletts. These are admittedly about twice the price, but you get what you pay for.

14. Next, plug your USB audio device in to one of the USB slots.

15. To confirm that your audio device is recognized by Linux, open up a terminal window (using the mostly-black rectangular icon at the top menu) and type the following in at the command prompt.

   NOTE: In this guide, when illustrating command prompts we show the line starting with a $. We do this to help establish context. Don't actually type that $. It's part of the normal Linux command prompt.

   $ cat /proc/asound/cards

   You should get back a list of installed cards. For the 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:
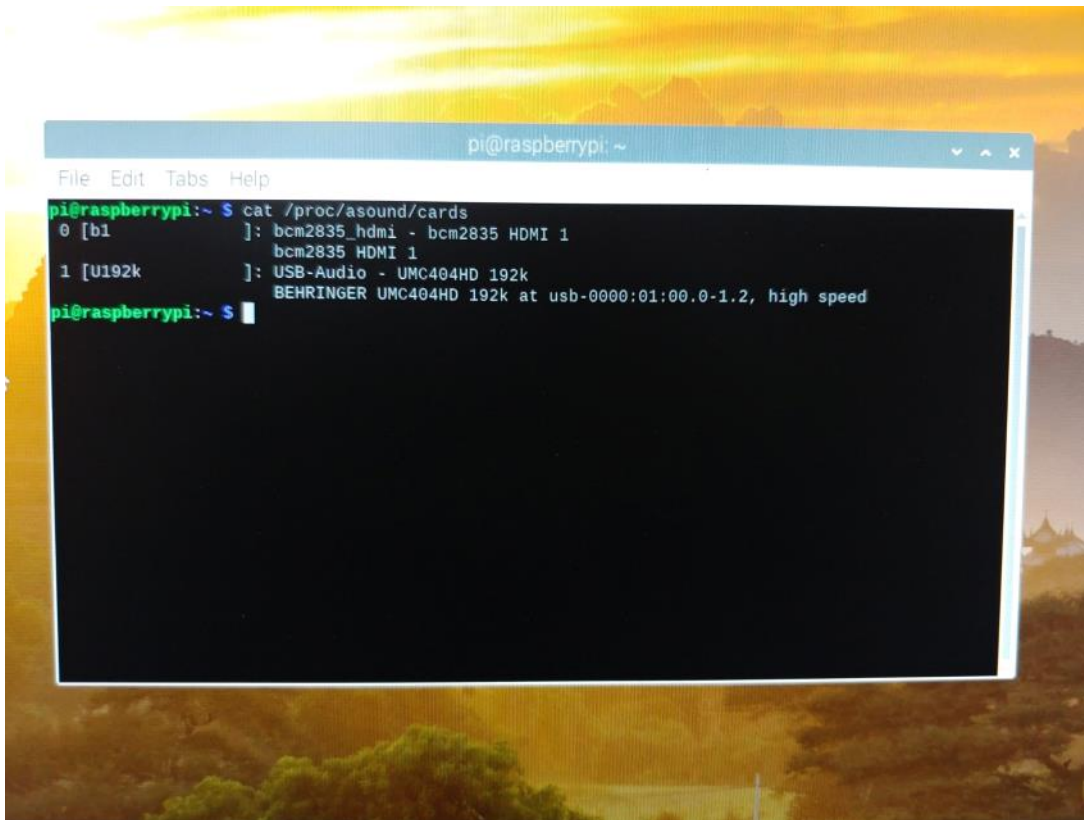
In our case we have a Behringer UMC404HD plugged in. The other "sound card" is the built-in device for HDMI audio.

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 to refer to the device. In our case, that name is "U192k". NOTE #1: Linux is case-sensitive, so pay attention to the capitalization of letters. Here, for example, the 'U' is upper case and the 'k' is lower case. NOTE #2: If you have one of the popular FocusRite Scarlett Solos, 2i2s, 4i4s, the shorthand name is most likely just "USB".

17. Next, we're going to set up the "Jack" audio service (which Jamulus uses to access audio), to refer to 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 the command line would look for the Behringer UMC404HD:

    $ jack_setup.sh U192k

    For many of the popular Scarlett's you would use:

    $ jack_setup.sh USB

18. What the jack_setup script actually does is create a hidden file called ".jackdrc" (notice the beginning dot) to your home directory. So it has a file path is "/home/pi/.jackdrc". You won't normally see this file if you do a simple listing, 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.
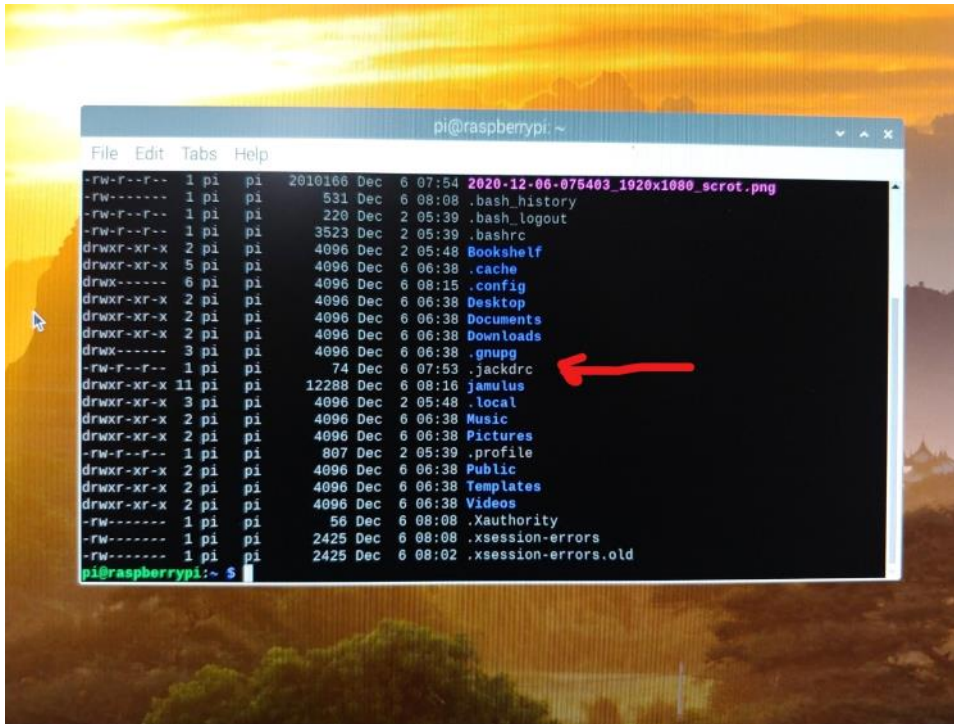
If you don't see this file, then something went wrong with the script. And then what? Well, "contact your system administrator". Ha! Don't you just love when user's guides tell you that?

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". The last three entries determine the latency. The "-p 64" entry says to use a period 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 a requirement of Jamulus.

The buffer parameters given here as defaults should work for all the decent quality USB audio devices, but they may not be appropriate for your particular setup. You'll know this because later, when running Jamulus, you might experience lots of drop outs, clicks, pops, etc (these are called xruns), so a bigger buffer might be needed. For example, yout 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 mostly out of scope of this document. "Contact

your system administrator :)". Hopefully, what we give here will work fine.

But we will mention two ways to edit the .jackdrc file. You can manually edit it using your favorite Linux text editor . We often use nano because it's simple, though awkward as it is cursor based only, or if you are familiar with qjackctl (the official graphical control for Jack) you can change the settings in that program and there is (conveniently for us!) an option to save the settings to the .jackdrc file.

WARNING: If you manually edit the .jackdrc file, do note that if you were to run jack_setup.sh again, your changes will be wiped out and you'll have to do your manual changes again.
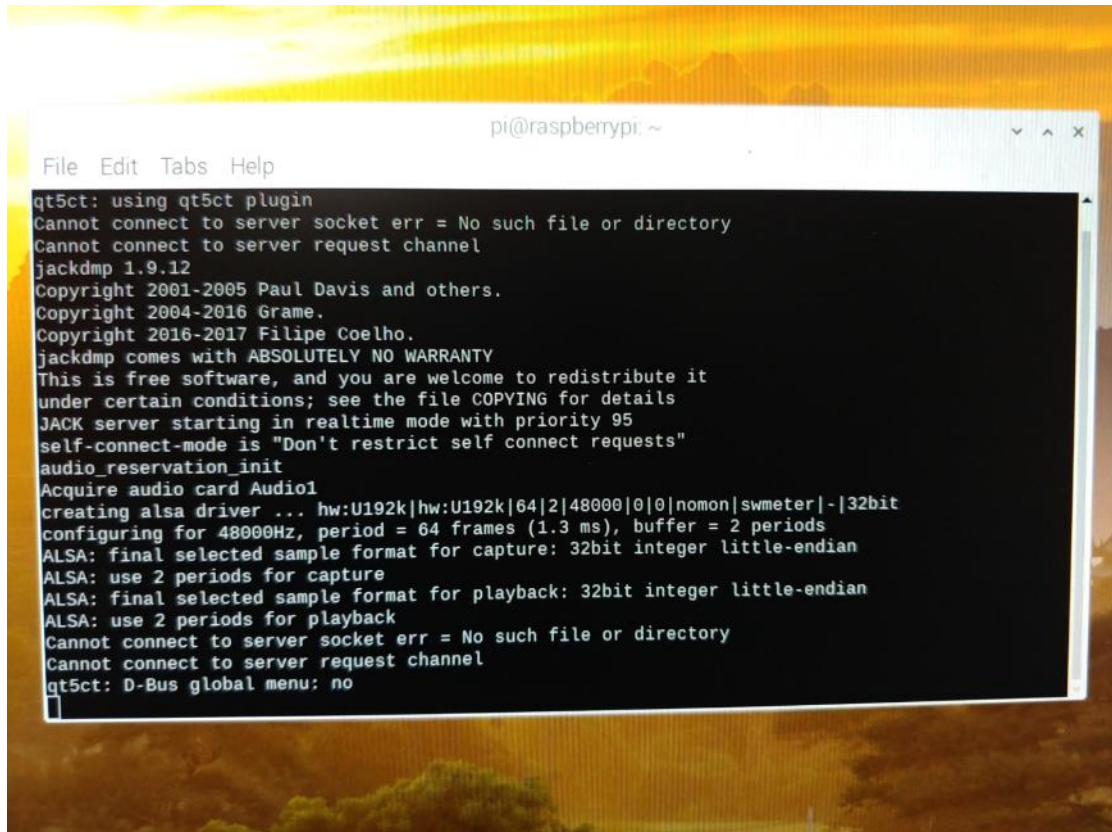
Oh and another thing: We've limited Jack to 16 channels here -- that's what the "-p 16" entry means. So no plugging in more than 16 microphones! Leave your choir back on the bus! Also, the "-T" parameter means "automatically terminate Jack after all Jack clients are finished. " That way, Jack won't be running unless Jamulus is running (or some other audio application.)

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:

$ 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:

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", or "couldn't allocate memory", etc. These are indications that Jack isn't set up right.

Most of the other stuff isn't too important. You shouldn't see many 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 Administator :)

Other types of errors you might see are "process error", or some message indicating the system couldn't process things in time. This is indicative of "xruns", and means you probably need to increase the buffer size. If you see just a few these errors, it's no big deal, but a bunch of them indicates a problem. (And you'll hear it too when running Jamulus)

21. Okay, then. Let's try out the Jamulus GUI. Note that it should have popped up after you entered the Jamulus command, and it resides in a window separate from the command line. The first thing to do is go into Settings and notice that the "Device" selected is "Jack". Also, you may wish to click the "Enable small network buffers" option, which will help Jamulus run with as small a latency as possible. However, if later you experience too many xruns, (pops and clicks) you might need to turn this off. Here's what our settings dialog looks like:
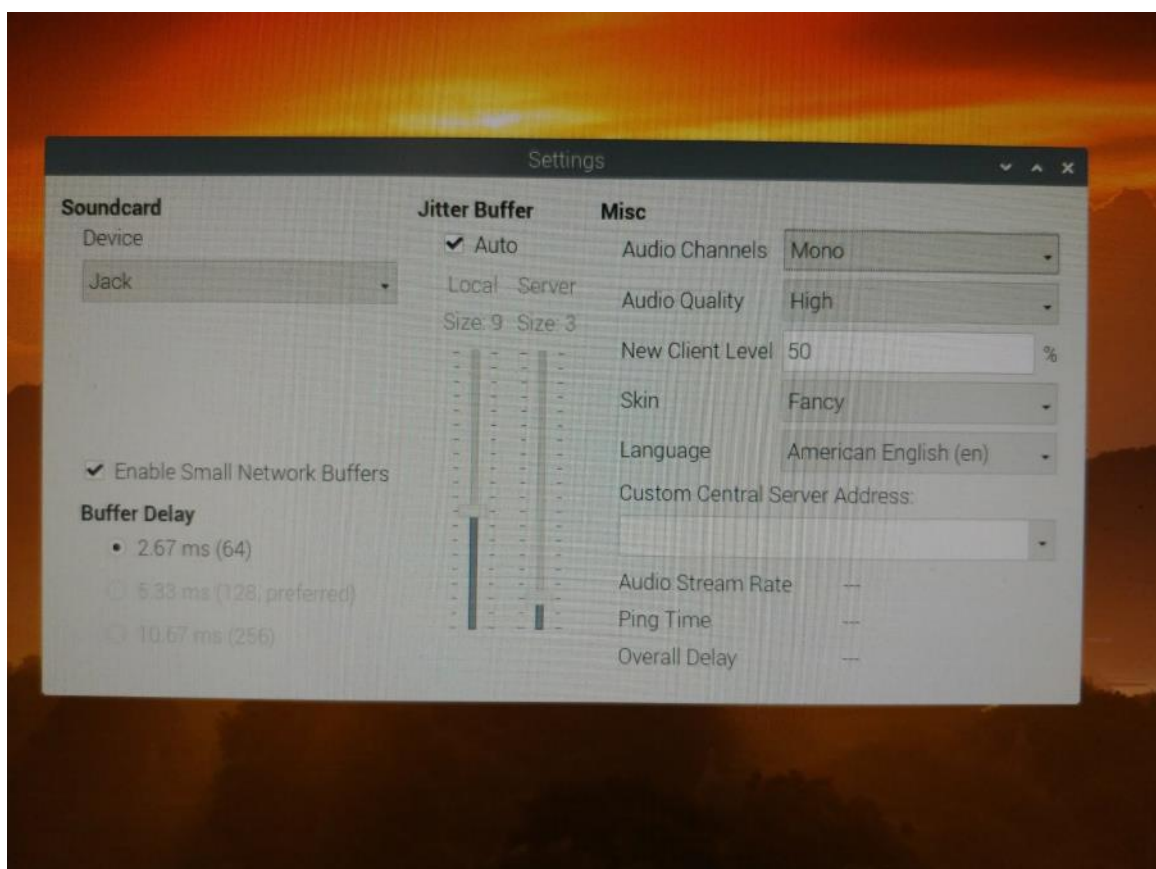
Some optional things to do here: Set the audio quality to "high", and set the new client level to 50%. We like to use "high quality" audio, especially with things like fiddles. We have noticed annoying high frequency artifacts on fiddles (you'll hear a distorted hissing sound when bowing the strings), unless high quality audio is selected. I'm sure other instruments will sound better too with this setting (like cymbal crashes). Your Raspberry Pi 400 shouldn't have trouble handling the higher bandwidth of the high quality audio setting, assuming your internet connection can handle it. Most likely, it'll be fine. Even at a high setting, audio streaming is much less intensive than, say, streaming video.

23. Okay, then. It's time to connect to a Jamulus server and start playing! Go back the main dialog, press Connect, select an server, and see what happens! If you are lucky, after things have settled down for a minute or two, you'll hopefully 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 with only occasional warbles. 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 and other warbles.

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

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

25. **Updating Jamulus**

    If a new version of Jamulus is released and you want to update your copy on your machine, you can use the provided jamulus_setup.sh script. You'll need to pass the release version string as a command line argument. For example, suppose you have version 3.6.1 installed and you want to update to version 3.6.2. You would need to pass the string r3_6_2 (that's how the Jamulus developers label their releases.) Run the script as follows:

    $ jamulus_update.sh r3_6_2

    This script will download the version specified, and then compile and install it. On the lowly Raspberry Pi, the compilation process takes a while (ten minutes or more -- we haven't timed it.)

26. **Running Jamulus as a Service (optional)**
    You may want run Jamulus as a service on your Raspberry Pi. The Rpi400Jam image has already been set up for this. But there are a couple of things you need to do manually, and this requires some technical skill (consult the Jamulus documentation for more information on server settings.)
    a. Change the default settings in the jamulus.service file. For example, to edit this file, use:
        i. $ sudo nano /etc/systemd/system/jamulus.service
        ii. The things you might want to change are in the places noted in the file where the parameters to Jamulus are defined. In particular, you might want to change the "server info" parameter, you probably want to change welcome message, and you may want to change the port (which we've defaulted to 22124, and as such no option is shown. Consult the Jamulus documentation for further information.
    b. **If wanting to use the server as a public Jamulus server**: You are good to go. Just skip to Step d below
    c. **If want to use the server as a private Jamulus server:** You need to set up port forwarding in your router to allow the port number set in Step a (such as 22124) to be forwarded to your Raspberry Pi. For this you need to know the IP address of your Pi, and you need to know

how to access the settings in your router. That's outside the scope of this document. Once you've setup the router, you can then start the Jamulus service as follows:

d. To start the Jamulus server as a service:
    i. $ sudo systemctl start jamulus

e. To stop the Jamulus server
    i. $ sudo systemctl stop jamulus

f. It's also possible to have the Jamulus server start up at boot time. For that you use:
    i. $ sudo systemctl enable jamulus
    ii. To disable this automatic startup, use:
    iii. $ sudo systemctl disable jamulus