FH AACHEN
UNIVERSITY OF APPLIED SCIENCES

Fachhochschule Aachen

Department of Electrical Engineering and Information Technology

MusiUeINet

Making Music over the Internet

Interdisciplinary Project

Nils Angelmann
An-Phong Pham Dinh
Marcel Lukas Paturej

Project Idea / Supervisor: Dipl.-Ing. Stephan Borucki

Aachen, Germany, July 2022

(English summary by Nils Angelmann, September 2024)

Contents

1. Introduction

This report summarizes the results of an interdisciplinary project that took place at *FH Aachen* in 2022.

The idea for the project came about when Stephan Borucki encountered numerous problems while trying to remotely rehearse with his choir during the pandemic in 2020 and 2021. Despite being an excellent option for remote rehearsals, the free and open source software *Jamulus*[1] presented some challenges.
Thus it was made the goal of this project to find ways of making *Jamulus* more user friendly for musicians with no background in technology and to determine which factors have the biggest influence on performance.

The initial plan included publishing the results shortly after the end of the project and writing instructions for recreating the test setups. Unfortunately, that never happened due to time constraints and a lack of interest in *Jamulus* after the pandemic had ended. Even though things may have already changed in the meantime, we would hereby like to publish at least our most important results. Maybe this proofs helpful to someone who uses *Jamulus* with a larger group of musicians or would like to reach the best possible performance.

All results of this project refer to the version of Jamulus that was up-to-date at the time of testing. It is therefore possible that some results may no longer be applicable to newer versions.

[1] jamulus.io

2. Optimizations

The first part of the project consisted of understanding the influence of all the settings inside *Jamulus* as well as the used components in general, in order to achieve the best possible performance.

It quickly became obvious, that latency is usually the biggest hindrance from a great musical performance over the internet and should therefore be minimized. This however turned out to be a trade-off between latency and stability.

2.1 Latency

In this context *latency* refers to the delay between a musician making a sound and then hearing that sound through their headphones.
A standout feature of Jamulus is synchronising those delays between all musicians, allowing all participants to be in time. This inevitably means that all other musicians have to "wait" for the one with the highest delay. However, in comparison to applications meant for video calls Jumulus' latency still is considerably lower.

From sound recording it is well known that latency can have a significant effect on a musician's performance. What amount of latency is tolerable is up to debate. But based on the following examples it can be assumed that there is no definitive answer:
A) For a drummer hearing the sound of a drum stick hitting the drum delayed by 10 milliseconds it might already be challenging to steadily keep the tempo while playing an up-tempo song.
B) The members of a big orchestra or choir could be up to 15 m apart, which equals a delay of 44 milliseconds given the comparatively low speed of sound (343 m/s).

As a first test it was decided to check whether the stated latency figure in Jamulus was accurate and could be relied on. Therefore, a burst signal was fed into an audio interface. By monitoring this input signal and the corresponding return signal on an oscilloscope, the latency reported in Jamulus could be confirmed (figure 2.1).

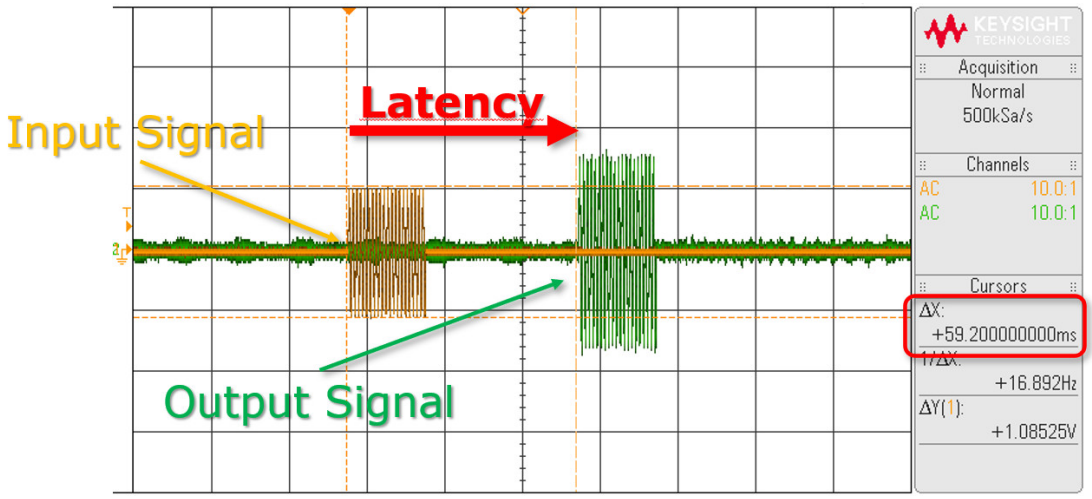

Figure 2.1: Latency measurement utilizing an oscilloscope and a signal generator

This overall latency through Jamulus is the sum of numerous factors. So, in order to minimize the latency it is key to understand the influence of each of those individual factors.

Operating System

A question that came up early on, was whether the amount of background tasks on Widnows (e.g. updates) might have an impact on Jamulus' performance/latency, which in return could be an advantage for Linux. However, a simple side by side test between a laptop with a fresh installation of *Windows 10* and a *Raspberry Pi 4* running the then latest version of *Raspberry Pi OS* showed no significant difference under otherwise identical conditions.
(This might not be true for a system with lots of bloatware and background tasks though.)

Audio Interface

Among musicians with an interest in recording their music, USB audio interfaces are widespread. Therefore a big selection of different models from various manufacturers exists. All these devices have in common that they require a buffer to ensure a steady stream of data over the USB connection. For most audio interfaces the size of this buffer can be set in their corresponding driver as a number of samples (usually a power of 2). Setting the buffer too high leads to unnecessary latency while decreasing the buffer size too far can cause dropouts in the audio signal and should therefore also be avoided.
The smallest possible buffer size for a stable operation can vary drastically depending on the model of audio interface. Once this ideal buffer size for a given audio interface has been found, its latency can be treated as a constant number.

$$Interface\ Latency = \frac{1}{Sample\ Rate} * Buffer\ Size * 2 \tag{2.1}$$

Example: At a common sample rate of 48,000 samples per second (48 kHz) a buffer size of 128 samples results in a latency of 2.67 ms per way or 5.33 ms round-trip.

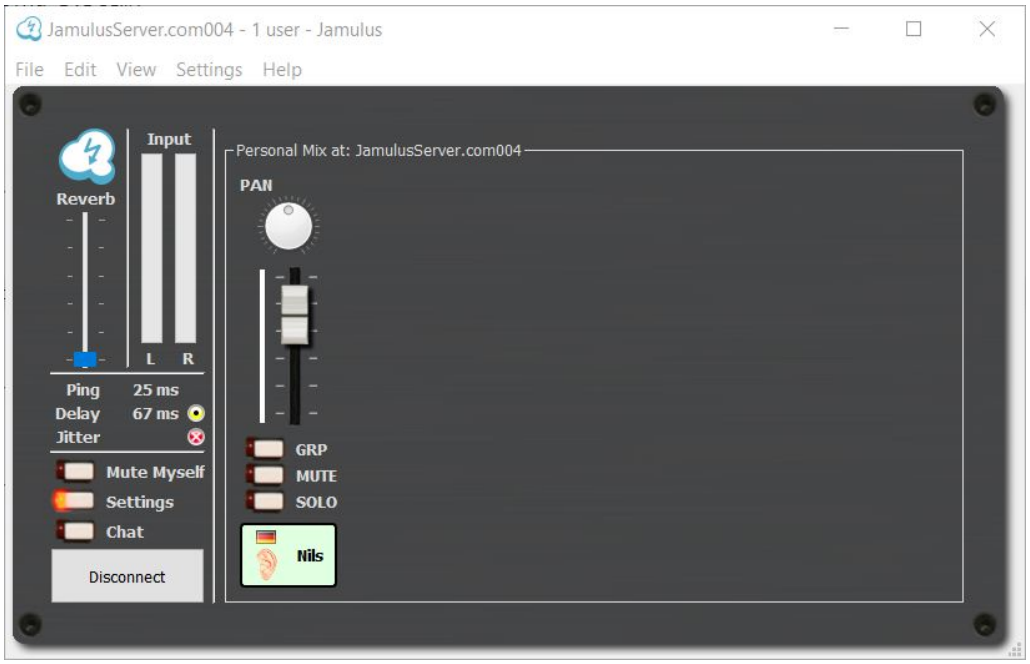

Figure 2.2: Jamulus user interface: (server) ping and overall delay

Server Location

A fairly obvious cause for latency is the server's ping as also displayed by Jamulus (figure 2.2). This latency is simply caused by the time it takes to transmit data to and from the server. Due to the finite speed of light a noticeable delay is induced by any long enough distance.
When using public servers, it thus makes sense to use one that is physically close to the musicians. Equally, when setting up a private server, e.g. for a remote choir rehearsal during the pandemic, a close-by server provider should be chosen - even if one half around the globe may be cheaper.
For achieving the best possible performance, the server's distance to the nearest internet hub should also be considered.

Latency Calculation

By treating both ping and the audio interface's latency as known values the remaining causes for latency can be estimated using the average values of some practical experiments (with a local server):

$$Latency = Interface\ Latency + Ping + Buffers\ (+\ Signal\ Processing) \tag{2.2}$$

$$20\ ms = 5\ ms + 0\ ms + x \tag{2.3}$$

$$\Rightarrow x = 20\ ms - 5\ ms - 0\ ms = 15\ ms \tag{2.4}$$

This suggests a latency of roughly 15 ms due to jitter/network buffers and signal processing.

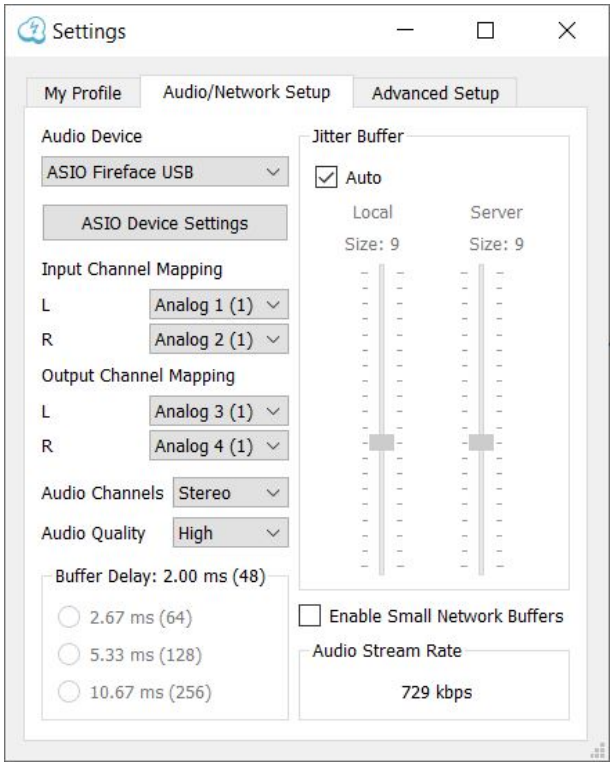

Figure 2.3: Settings menu in Jamulus: Jitter Buffer

Jitter/Network Buffer

While deactivating the *auto* function for the jitter buffer and manually lowering the buffer size does seem to result in a slightly lower latency, the connection will most likely be unstable - noticeable as dropouts in the audio signal. The same applies to the option for *small network buffers* - reduced latency at the cost of transmission errors. Like already mentioned one the Jamulus website, messing with these settings is absolutely not recommended.
One could question why those options are even available.

Server Utilization

A rather surprising discovery was made when the exact same setup was tested at different times throughout the day: During the day, when the university was crowded, an overall latency of 45 ms was measured. In the early evening, when most people had already left, this number dropped to 20 ms.
Since the server for those tests was a virtual machine, hosted on the university's server, this implies a big dependence on server utilization. A dedicated root server only for Jamulus should therefore provide a significant advantage over a shared or a virtual one.

2.2 Transmission Errors

Besides high latency, transmission errors can also worsen the user experience.
While feeding a continuous test signal into Jamulus and observing the returning signal on an oscilloscope, the following two different issues were found:

Clicks

The first one being short clicks that - depending on their rate of appearance - can sound similar to a Geiger counter. While those short clicking sounds may be annoying and can certainly render any recordings unusable, they should in general not present a hindrance in a rehearsal situation.

Dropouts

The second type of transmission errors is far more disrupting and could be observed in the form of short dropouts of the audio signal.

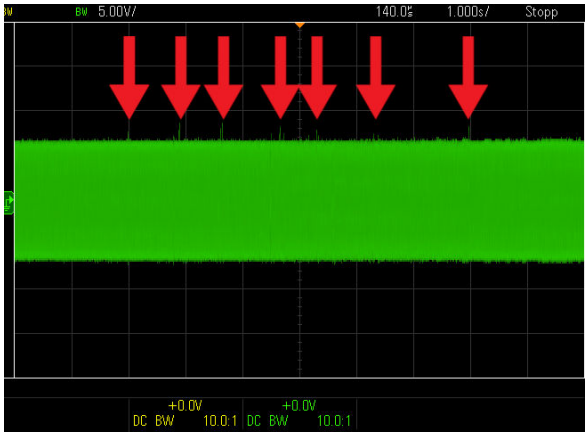

Figure 2.4: Short clicks

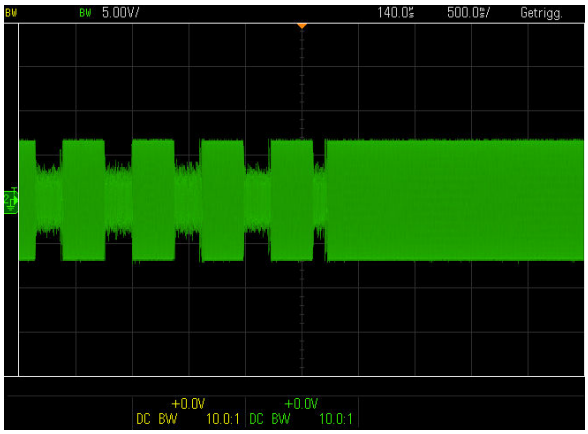

Figure 2.5: Obvious dropouts

Possible Explanation

While a definitive cause for the first type for transmission error could not be found, the longer dropouts were only observed on devices that were connected to the internet using WiFi. Seemingly, the dropouts are related to package losses of the WiFi connection. Therefore, a wired internet connection should be used whenever possible.

2.3 Test Setup

In order to make the aforementioned transmission errors quantifiable, a self-contained test device was developed. This allows for field testing without carrying around specialised tools, i.e. an oscilloscope and a function generator. Thereby different setups and internet connections can easily be compared.
Even though the primary goal consisted in measuring transmission errors and the latency reported by Jamulus was generally found to be correct, a way to measure latency was also implemented.

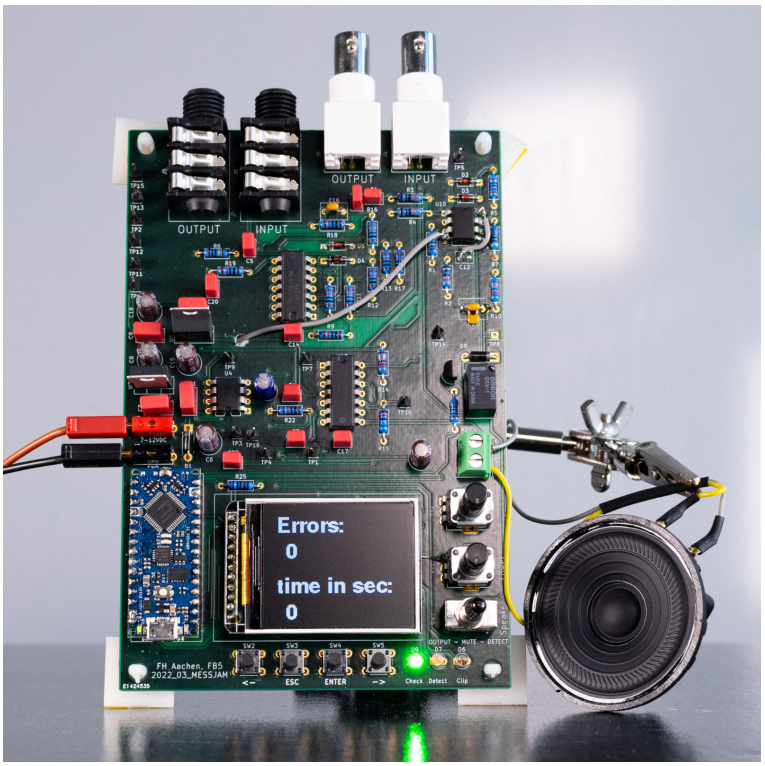

Figure 2.6: Prototype for a self-contained test device

The setup is based around an *Arduino Nano Every* microcontroller development board which is accompanied by some analog circuitry and can be connected to any regular audio interface. Similarly to the first test, latency is measured by sending a non-periodic signal (a simple step function in this case) and recording the time between sending and receiving this signal.
In order to detect transmission errors, a continuous sinusoidal signal, generated by an analog oscillator, gets fed through Jamulus. The returning signal gets rectified and filtered so that a window comparator can be used to decide whether the amplitude is within a carefully calibrated range. This circuit creates a simple (digital) high signal whenever a click or a dropout occurs. The Arduino is then used to count those detected errors.
Thereby, different setups can be compared on the basis of a *transmission errors per minute*. Due to the often high rate of clicks, this would be impossible to do manually, e.g. by watching an oscilloscope.
Note: Without wandering off into the field of psychoacoustics too much, it might actually be questionable whether multiple clicks occurring shortly after each other should be counted individually or as **one** *perceived* error.
Also note: The small speaker can be used to inject the test signals into devices that do not feature a line input.

3. Ease of Use

The second goal of this project was to find ways to make Jamulus more accessible to musicians who do not already own any special hardware and/or have limited technical knowledge.

3.1 Audio Interface

Relying on the built-in audio inputs and outputs of a computer is not a viable option as those cannot be directly accessed by Jamulus on a driver level.
The usual solution consists of using an external USB audio interface in conjunction with a microphone and a pair of headphones. While many musicians already own all the necessary equipment and know how to use it, this solution can present a challenge to anyone with no background in recording music.
Not only can the acquisition costs be problematic - especially when it comes to equipping an entire orchestra or choir - but setting everything up can also be a hindrance.

Therefore, we suggest the usage of USB microphones, besides which only a (probably already existing) pair of headphones is needed. Also and perhaps even more importantly, the setup process is reduced to plugging in a USB cable.

In order to compare both options, two exemplary test setups were compiled:

Setup A (figure 3.1)

- Audio interface: Behringer U-Phoria UM2 (50 €)
- Microphone: Behringer SL 85S (20 €)
- Headphones: Behringer HPM1100-BK (15 €)
- Table stand for microphone: Millenium DS-10 (10 €)
- Microphone cable: Cheap XLR cable (5 €)

Sum: 100 €

Setup B (figure 3.2)

- USB microphone: Mackie EM-USB[1] (60 €)
- Headphones: Behringer HPM1100-BK (15 €)

Sum: 75 €

Note: All prices were taken from Thomann.de (2024-09-17) and slightly rounded.

[1] Thomann's house brand has since released the t.bone SC 430 which (at least visually) seems to be a re-branded version of the same microphone while being around 10 € cheaper

Hinweis: Knöpfe zur manuellen Regelung wichtig ?

In our tests the performance of both setups (on Windows and on Linux) was absolutely comparable and sufficient for rehearsal duties. Regarding the important factor of latency, no difference could be observed as both solutions performed best at a buffer size of 128 samples.

Figure 3.1: Setup A: USB audio interface with headphones and a microphone

Figure 3.2: Setup B: USB microphone with headphones

A word of warning: While there is a big selection of USB microphones, only few models feature hardware controls, i.e. gain (microphone sensitivity) and headphone volume, in the form of knobs. Instead, most models rely on a specific software which may not be available for all operating systems and adds unnecessary complexity.

3.2 Closed Entity

Another idea that came up during the project was the possible development of a purpose-built, self-contained unit that would make using Jamulus as easy as possible. Those pre-configured Jamulus clients could then be handed out to a group of musicians by one administrator.

Raspberrry Pi

Based on the successful tests with Linux, using a single-board computer, i.e. a Raspberry Pi, seemed like a good starting point. Besides the small form factor and the competitive price, Linux allows for plentiful customisations, e.g. removing any unused features, and thereby improving the user experience. This would have been difficult on any device running Android. Windows meanwhile would require more powerful hardware due to a bigger overhead, making a bigger footprint necessary.

Even though most tests were performed with the then current *Raspberry Pi 4*, it could also be confirmed that the older *Raspberry Pi 3* with 1 GB of RAM is indeed powerful enough to run a Jamulus session on *Raspberry Pi OS*. These tests also showed that 1 GB of RAM is sufficient and that a version with 2 GB might be a good choice with future updates in mind while any more RAM will simply not be used.

Touchscreen

Considering Jamulus' simple user interface, a touchscreen makes for an elegant way of interacting with the channel sliders without the need of any external peripherals.

During tests an important disovery was made: Jamulus requires a vertical resolution of at least 600 pixels. Therefore, the *Waveshare 7inch HDMI LCD (C)*[2] with a resolution of 1024x600 px is a suitable option (figure 3.4), while the (B) variant[3] with a resolution of 800x480 px is not (figure 3.4).

Figure 3.3: Touchscreen (B): 800x480 px

Figure 3.4: Touchscreen (C): 1024x600 px

Since the idea involves an administrator doing the basic configuration beforehand, the user will only need to enter their name. For this a virtual on-screen keyboard should be sufficient (figure 3.5). The *Matchbox-Keyboard* worked well in our tests[4].

[2] waveshare.com/7inch-hdmi-lcd-c.htm
[3] waveshare.com/7inch-hdmi-lcd-b.htm
[4] github.com/mwilliams03/matchbox-keyboard

Figure 3.5: Virtual keyboard: Matchbox-Keyboard

Audio Interfaces

In addition to the already discussed USB audio interfaces and USB microphones (chapter 3.1) which can be used on any Linux device thanks to the *JACK Audio Connection Kit*[5], there are also specialised audio interfaces for the Raspberry Pi. An example for that would be the *HiFiBerry*[6] products. While the form factor would allow for an even tighter integration, there is currently no model that features a microphone input. This might be an opportunity for the development of a Jamulus specific module that features a headphone output and one XLR microphone input. However, when it comes to existing solutions we would still recommend an external USB microphone for most use cases.

Figure 3.6: Basic setup of a Jamulus client - Raspberry Pi 4 and touchscreen display

[5] jackaudio.org/
[6] hifiberry.com/

Teamviewer

To allow for remote administration of the clients, a tool like *TeamViewer*[7] could be used. In a test, remotely accessing a Raspberry Pi worked as intended and allowed for an easier configuration and setup than by using the touchscreen. However, to allow remote access *TeamViewer* would always need to be running on the client, which can pose a serious security risk. Alternatively a user would have to start the application manually if they needed assistance - again not ideal regarding the usability.
As a solution, every client could have a physical *"HELP!"* button that semi-automatically starts *TeamViewer* and connects to the responsible administrator.

Virtualization

Another possible security risk might be caused by not updating the software regularly. Simply installing every update could cause other issues, so a known stable version might be used over a prolonged period of time. Running Jamulus in a virtualized environment could theoretically reduce the risks. Unfortunately, our tests showed serious problems with accessing any connected audio interface from within a virtual installation, hence this idea was discarded.

Cost

Even though most features were working by the end of the project, a complete prototype has never been built due to time constraints. So, the cost per client can only be estimated:

- Raspberry Pi 4 (Model B 2 GB)[8]: 46 €
- Waveshare 7inch HDMI LCD (C)[9]: 48 €
- microSDHC card (16 GB Class10)[10]: 8 €
- Raspberry Pi USB-C power supply[11]: 8 €
- Custom case (probably 3D printed): 5 €

Sum: 105 €

Note: All prices were taken from rasppishop.de and eckstein-shop.de (2024-09-17) and slightly rounded.

In total each Jamulus client would cost around 190 € including a USB microphone and a pair of headphones (Setup B from chapter 3.1).

[7] teamviewer.com/
[8] rasppishop.de/Raspberry-Pi-4-Modell-B-2GB-SDRAM
[9] eckstein-shop.de/Waveshare7inch
[10] rasppishop.de/Sandisk-microSDHC-16GB-Class10-mit-Raspberry-Pi-OS
[11] rasppishop.de/Raspberry-Pi-15W-USB-C-Netzteil-Schwarz-EU

4. Single-board computer

4.1 Alternatives

In the aforementioned chapter only the Raspberry Pi (4 Model B) is discussed. However, since this project took place during the peak of the chip crisis and Raspberry Pis were extremely difficult and expensive to obtain, alternative single-board computers were also tested.

4.1.1 Raspberry Pi 3/4/5

As mentioned before, an older Raspberry Pi 3 with 1 GB of RAM was successfully tested. In conclusion, every model starting from generation 3 should work. The newer Raspberry Pi 5 was not yet released at the time of testing but can be expected to work just as well.[1]

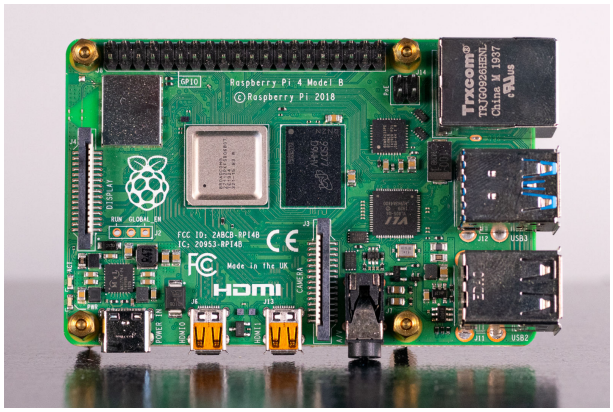

Figure 4.1: Raspberry Pi (4 Model B)

4.1.2 HardKernel ODROID XU4Q

The now discontinued *ODROID XU4Q* featured an octa-core processor (Samsung Exynos 5422) and 2 GB of RAM. As an operating system the manufacturer recommended Ubuntu. While Jamulus works on this device, the tested touchscreen does not seem to be supported.[2]

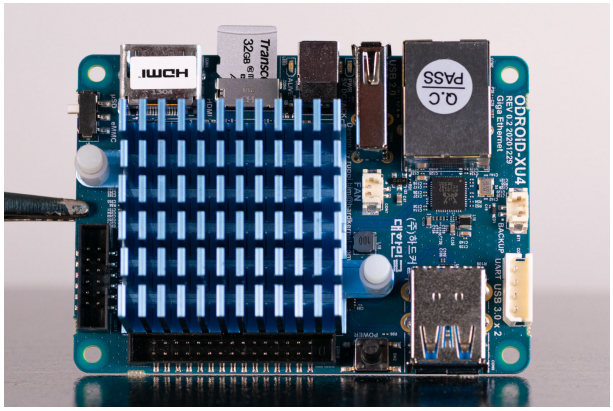

Figure 4.2: HardKernel ODROID XU4Q

[1] raspberrypi.com/products/
[2] hardkernel.com/shop/odroid-xu4q-special-price/

4.1.3 PINE A64-LTS

The slightly larger *PINE A64* (long term supply) features a quad-core cpu and 2 GB of RAM, making it roughly comparable to the Raspberry Pi 3. The recommended operating system *Armbian* is based on *Debian*. Unfortunately, the tested touchscreen does not seem to be compatible with this single-board computer either.

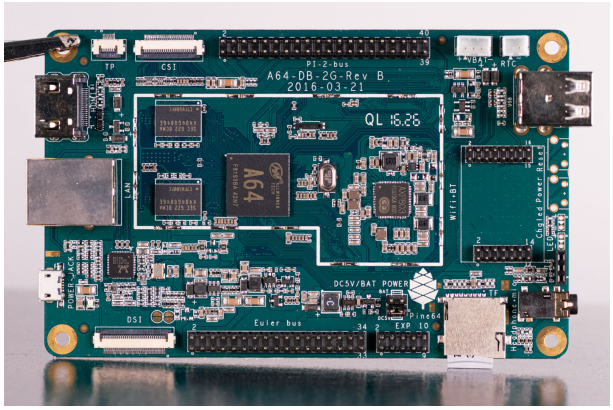

Figure 4.3: PINE A64-LTS

Since supply is no longer an issue, we do not recommend using any of the alternatives. None of those boards come close to the support of the Raspberry Pi's community and while some features worked fine for us, software support can be problematic and the installation process can be tedious.

4.2 Single-board server

After realizing that neither the ORDROID nor the PINE single-board computer could be used for the Jamulus client as planned and reading about Raspberry Pi's being used as Jamulus servers[3], we tried the same with the ODROID. Thanks to its powerful octa-core cpu it can in fact be used as a Jamulus server.
In general, using a single-board computer can be a cost effective and energy efficient option, when a private Jamulus server is needed.

[3] jamulus.io/de/kb/2020/03/28/Server-Rpi.html

5. Conclusion

Based on this project a number of recommendations can be given.

5.1 Ideal Jamulus Setup

In order to get the best performance from Jamulus...

- ...avoid using WiFi whenever possible.
- ...leave the network buffer settings inside Jamulus on *auto*.
- ...find the smallest, stable buffer size of your audio interface. A value of 128 samples has proven to be a good starting point.

If you need an audio interface...

- ...the Behringer UM2 is the cheapest recommendable option.
- ...consider a USB microphone for an easier setup.

When setting up your own server...

- ...avoid using a virtual server. A root server will most-likely perform significantly better.
- ...using a Rasbperry Pi can be cheaper than paying for a remote server.
- ...consider the phyisical distance between participating musicians and the server (ping).

5.2 Jamulus Client

If you like the idea of a self-contained Jamulus client...

- ...get the latest Raspberry Pi model. 2 GB of RAM are sufficient.
- ...do not buy a display with a vertical resolution of less than 600 pixels.
- ...consider ways of helping users remotely.

List of Figures