FH AACHEN
UNIVERSITY OF APPLIED SCIENCES

FH Aachen
Dipl. Ing. Stephan Borucki

MusiUelNet
Musizieren über das Internet

Projektbericht
für das
Interdisziplinäre Projekt

vorgelegt von
Nils Angelmann, 3286166
Marcel Lukas Paturej, 3551967
An-Phong Pham Dinh, 3115799

Abgabedatum:
22.07.2022

Inhaltsverzeichnis

1 Einleitung und Zielsetzung des Projektes 1

2 Theoretischer Hintergrund 1
2.1 Raspberry Pi 4 (Model B) . . . 1
2.2 ODROID XU4Q . . . 1
2.3 PINE A64-LTS . . . 2
2.4 Jamulus . . . 2
2.5 Vergleich verschiedener Optionen der Fernwartung . . . 2
2.5.1 Direkte Verbindung per SSH . . . 3
2.5.2 Verbindung per Desktop-Sharing (VNC, RDP) . . . 3
2.5.3 Verbindung per TeamViewer . . . 4
2.6 Touchdisplays . . . 4
2.6.1 Waveshare 7inch HDMI LCD (B) . . . 5
2.6.2 Waveshare 7inch HDMI LCD (C) . . . 5

3 Einrichtung des Jamulus-Servers 5
3.1 Mögliche Server-Typen . . . 5
3.1.1 Öffentlicher Server . . . 5
3.1.2 Privater Server . . . 6
3.2 Verwendete Server . . . 7
3.2.1 Eigener Server . . . 7
3.2.2 Server bei der DVZ . . . 7
3.2.3 Server bei einem Hosting Anbieter . . . 13
3.2.4 Einplatinen Computer als Server . . . 17
3.3 Installation der Jamulus-Server Software . . . 18
3.3.1 Installation auf einem Server mit x86 Architektur . . . 19
3.3.2 Installation auf einem Server mit ARM Architektur . . . 21

4 Einrichtung des Jamulus-Clients auf Raspberry Pi 4 23
4.1 Installation und Vorbereitung von Raspberry Pi OS . . . 24
4.2 Einrichtung von Jamulus . . . 25
4.3 Einrichtung von TeamViewer . . . 26
4.4 Einrichtung eines Touchdisplays . . . 28
4.4.1 Waveshare 7inch HDMI LCD (B) . . . 28
4.4.2 Waveshare 7inch HDMI LCD (C) . . . 29
4.5 Einrichtung einer virtuellen Tastatur . . . 30
4.6 Einrichtung eines Kurzbefehls zum Start von Jack und Jamulus . . . 33

5 Virtualisierung des Jamulus-Clients 35

6 Untersuchung der Verwendbarkeit der Raspberry Pis **38**
6.1 Raspberry Pi 3 vs. Raspberry Pi 4 38
6.2 Jamulus-Client auf Basis des Raspberry Pi 3 mit 1 GB RAM 38
6.3 TeamViewer auf dem Raspberry Pi 3 mit 1 GB RAM 39
6.4 Jamulus-Client auf Basis des Raspberry Pi 4 mit 2 GB RAM 40
6.5 Zusammenfassung und Wahl des Raspberry Pi - Modells 41

7 Alternative Hardware für den Jamulus-Client **42**
7.1 ODROID XU4Q 42
7.2 PINE A64-LTS 43

8 Audio-Interfaces **44**
8.1 Aufsteckplatinen 45
8.2 Externe USB-Interfaces 45
8.3 USB-Mikrofone 46

9 Messungen **46**
9.1 Frequenzgang (und Grundrauschen) 46
9.2 Latenz 48
9.2.1 Erste Messversuche 49
9.2.2 Einflüsse und Erklärungsansätze 49
9.3 Übertragungsfehler 50
9.4 Messplatine 52
9.4.1 Schaltung zur Detektion von Übertragungsfehlern 52
9.4.2 Schaltung zur Latenzmessung 55
9.5 Anforderungen und Ergänzungen 55
9.5.1 PCB 56
9.6 Messergebnisse und abgeleitete Erkenntnisse 59
9.6.1 Vergleich WLAN vs. LAN 59
9.6.2 Einfluss des Serverstandortes und der Serverauslastung 59
9.6.3 Minimal erreichbare Latenz 60
9.6.4 Einstellungen in Jamulus 60
9.6.5 Vergleich zu Windows 60

10 Fazit und Empfehlungen **61**
10.1 Jamulus-Client als Einplatinencomputer 61
10.2 Empfehlungen zur Hardwareauswahl 61
10.3 Empfehlungen zu den Einstellungen innerhalb von Jamulus 62
10.4 Empfehlungen zur Einrichtung eines Jamulus-Servers 62

11 Ausblick **62**
11.1 Jamulus-Client als geschlossene Entität 63
11.2 Weitere Messungen 63
11.3 Veröffentlichung der Ergebnisse 64

A Anhang **69**
A.1 Anfrage bei der DVZ 69
A.2 Jamulus Installations- bzw. Update-Script 70
A.3 Vollständiger Schaltplan der Messplatine 72
A.4 Bekannte Fehler der Messplatine 73

1 Einleitung und Zielsetzung des Projektes

Als bedingt durch die Coronapandemie nach einer Möglichkeit gesucht wurde, wie die Mitglieder des Hochschulchors der FH Aachen ihre gemeinsamen Proben online abhalten können, viel der Blick zunächst auf die gängigen Programme für Videokonferenzen. Schnell zeigte sich jedoch, dass diese zur synchronen Musikübertragung praktisch ungeeignet sind. Bei der Suche nach einer Alternative, die besser an den Anwendungsfall angepasst ist, stieß man dann schließlich auf die Open-Source-Software Jamulus. Zwar verliefen erste Tests mit Jamulus grundsätzlich recht vielversprechend, doch zeigte sich auch hier schnell Verbesserungspotenzial: Zum einen stellte die Konfiguration des jeweiligen Endgeräts viele Chormitglieder vor große Herausforderungen, zum anderen war unklar, welche Einstellungen für die bestmögliche Performance zu wählen waren. Auch wurde vermutet, dass im Hintergrund aktive Programme und Updates auf den von den meisten Chormitgliedern verwendeten Windows-Geräten zu erheblichen Einbußen in der Performance führten. Daraufhin rief Herr Borucki, selbst Chormitglied, schließlich das interdisziplinäre Projekt “MusiUeINet” ins Leben, im Rahmen dessen neben der Klärung möglichst vieler offener Fragen zu Auswirkungen von Einstellungen in Jamulus auch ein möglichst einfach zu bedienender und weitestgehend vorkonfigurierter Jamulus-Client auf Basis eines günstigen Einplatinencomputers mit Linux-Betriebssystem entwickelt werden sollte.

2 Theoretischer Hintergrund

2.1 Raspberry Pi 4 (Model B)

Beim Raspberry Pi 4 (Model B) handelt es sich um die aktuellste Version der beliebten Raspberry-Einplatinencomputer. Das Modell verfügt über einen Quadcore-Prozessor (4 x 1,5 GHz) und wahlweise 1 GB, 2 GB, 4 GB oder 8 GB RAM. Zur Videoausgabe steht neben zwei Micro-HDMI-Ports auch ein DSI-Anschluss zur Verfügung (Display Serial Interface). Außerdem verfügt das Board zusätzlich zur LAN-Schnittstelle auch über WLAN. (Raspberry Pi Foundation, o.J.) Als Betriebssystem kommt in der Regel Raspberry Pi OS zum Einsatz, welches auf der Linux-Distribution Debian basiert. (Wikipedia, 2022)

2.2 ODROID XU4Q

Der ODROID XU4Q verfügt mit seinem Octacore-Prozessor (4 x 2 GHz + 4 x 1,4 GHz) und 2 GB RAM über reichlich Leistung und ist entweder mit aktiver (XU4) oder passiver (XU4Q) Kühlung erhältlich. Im Vergleich zum Raspberry Pi fehlt ein DSI-Displayanschluss und WLAN ist auch nicht an Board. Außerdem arbeitet das Board mit einem Logikpegel von nur 1,8 V. (Hardkernel, 2021) Zum Zeitpunkt der Recherche (April 2022) war der XU4Q mit etwa 80€ deutlich teurer als ein Raspberry Pi 4 (2 GB) zum “normalen” Preis. Hinsichtlich der weiteren Verfügbarkeit ist zu erwähnen, dass der hier verbaute Prozessor (Samsung Exynos 5422), der ursprünglich in Samsungs Galaxy S5 zu finden war,

noch aus dem Jahr 2014 stammt, was die Ankündigung des Herstellers, den XU4(Q) noch mindestens bis Mitte 2023 produzieren zu wollen, etwas fragwürdig erscheinen lässt. Das Alter der 28nm-Architektur macht sich auch am mit 20 Watt vergleichsweise leistungsstarken Netzteil bemerkbar. (Samsung, o.J.)

2.3 PINE A64-LTS

Der PINE A64-LTS ist mit seinem Quadcore-Prozessor (4 x 1,2 GHz) eher mit einem Raspberry Pi 3 vergleichbar, hat mit 2 GB in der "Basisvariante" allerdings mehr RAM. Auch hier ist WLAN bei Bedarf nur über Zusatzhardware verfügbar. (PINE64, o.J.b) Zum Zeitpunkt der Recherche (April 2022) war der PINE A64-LTS für rund 40 € die mit Abstand günstigste Option. Bedenklich ist auch hier die langfristige Verfügbarkeit. Zwar handelt es sich bei der LTS-Version um den „Long Term Supply"-Nachfolger des ursprünglichen PINE A64, jedoch finden sich auf der Hersteller-Website widersprüchliche Angaben zum Support-Ende der LTS-Version: Entweder bis mindestens zum Jahr 2022 (PINE64, o.J.b) oder bis mindestens zum Jahr 2025 (PINE64, o.J.c).

2.4 Jamulus

Jamulus ist eine 2006 von Volker Fischer veröffentlichte, quelloffene, vernetzte Musik-Auftritt-Software, die Live-Proben, Jammen und Performen mit Musikern an beliebigen Orten im Internet ermöglicht (Fischer, o.J.a). Dabei wird ein öffentlicher oder privater Server von Nutzern bereitgestellt, auf denen sich dann ein oder mehrere Clients verbinden können, um darauf gemeinsam zu musizieren. Die Jamulus-Client-Software wird für verschiedene Betriebssysteme, darunter Windows, Mac und Debian/Ubuntu zur Verfügung gestellt. An die Client-Rechner werden dann, nachdem die Jamulus-Clientsoftware installiert ist, die entsprechenden Musikinstrumente (üblicherweise per externem USB-Audiointerface) angeschlossen. Damit Jamulus nun die Signale der Musikinstrumente verarbeiten kann, wird üblicherweise ein Audiotreiber, wie beispielsweise ASIO4ALL unter Windows oder JACK unter Linux, benötigt, der das musizieren unterstützt.

2.5 Vergleich verschiedener Optionen der Fernwartung

Im folgenden Abschnitt werden die verschiedenen Optionen einer Fernwartung diskutiert, die verwendet werden können, um einen Einplatinencomputer mit einem Linux-Betriebssystem aus der über das Internet steuern zu können. Dadurch soll insbesondere gewährleistet werden, dass wenn es zu Problemen mit dem Jamulus-Client kommt, diese über das Internet behoben werden können, ohne dass dafür ein Techniker vor Ort sein muss und direkten Zugriff auf den Einplatinencomputer nehmen muss.

2.5.1 Direkte Verbindung per SSH

Eine direkte Verbindung per SSH auf den Einplatinencomputer bietet den Vorteil, dass eine SSH Lösung bei den meisten in diesem Projekt verwendeten Betriebssystemen bereits vorinstalliert ist oder über einen Befehl nachinstalliert werden kann. Die Verwendung dieser SSH-Verbindungen ist bedingungslos kostenlos, da es zahlreiche Software gibt (beispielsweise OpenSSH), die ohne Einschränkung verwendet werden können. Die Einrichtung dieser SSH-Lösungen ist in allen Fällen dieses Projekts über wenige Kommandozeilenaufrufe möglich.

Allerdings haben diese SSH-Verbindungen den Nachteil, dass wenn Verbindungen von außerhalb des eigenen Heimnetzes auf den Einplatinencomputer vorgenommen werden sollen, dafür eine entsprechende Portfreigabe im Router des Nutzers vorausgesetzt ist. Durch diese Freigabe im Router, die zwangsläufig durch eine Fachkraft durchgeführt werden sollte, ergibt sich gleichzeitig ein erhöhtes Risiko eines Angriffs durch entweder fehlerhaft konfigurierte Freigaben im Router oder durch veraltete Software auf dem Raspberry Pi, die verwendet werden kann, um nicht nur den Einplatinencomputer zu übernehmen, sondern auch weitere Geräte im Netzwerk zu infiltrieren.
Ein weiterer Nachteil dieser Lösung ist, dass keine Übertragung der Desktop-Ansicht ohne weitere Softwareinstallation (siehe Kapitel 2.5.2) möglich ist. Dies kann aber unter Umständen zwingend nötig sein, um Einstellungen am Jamulus-Client oder an der Benutzeroberfläche des Betriebssystems vorzunehmen. Es ist in diesem Fall per SSH lediglich möglich, Kommandozeilenbefehle auszuführen.

2.5.2 Verbindung per Desktop-Sharing (VNC, RDP)

Wie bereits in Kapitel 2.5.1 beschrieben bietet SSH eine einfache Möglichkeit, Befehle über das Internet auf dem Einplatinencomputer auszuführen. Allerdings ergibt sich dabei das große Problem, dass eine Desktop-Ansicht benötigt wird, um Jamulus Einstellungen ändern zu können.
Eine Lösung dieses Problems ist die Verbindung per Desktop-Sharing (beispielsweise VNC oder RDP). Dieses Desktop-Sharing basiert in seinen Protokollen auf SSH, was bereits in Kapitel 2.5.1 beschrieben wurde. Durch diese Basieren auf SSH sind ebenfalls die Vorteile sowie die Nachteile dieser Lösung überwiegend identisch, mit der Ausnahme, dass nun die Übertragung und Steuerung der Desktop-Ansicht möglich ist. Es ist also mit dieser Lösung möglich, den Desktop zu sehen und darüber Veränderungen an Jamulus und co. durchführen zu können.
Es ist allerdings erneut zu betonen, dass das Problem der Portfreigabe im Router, die ebenfalls weiterhin benötigt wird, bleibt. Somit bleibt ebenfalls das Problem des Mehraufwandes, diese Portfreigabe durchführen zu müssen, und des daraus resultierenden Sicherheitsrisikos für die Nutzer des Netzes.

2.5.3 Verbindung per TeamViewer

Eine weitere mögliche Lösung für die Fernwartung ist die Verwendung von TeamViewer. TeamViewer ist eine Remote-Connectivity-Software, die dafür entwickelt wurde, den Fernzugriff auf andere Rechner durchführen zu können (TeamViewer, o.J.). Dafür wird lediglich eine Software auf beiden Rechnern installiert. Im Anschluss daran kann unmittelbar eine Verbindung aufgebaut werden.
TeamViewer bietet den Vorteil, dass die Installation und Konfiguration im Vergleich zu den Lösungen in den Kapiteln 2.5.1 und 2.5.2 ohne eine Portfreigabe im Router der Nutzer möglich ist. Gleichzeitig ist TeamViewer eine in Deutschland entwickelte Software aus dem Jahr 2005, die sich dadurch viele zusätzliche Funktionen, wie die Verbindung mehrerer Geräte über einen einzigen Account, die Steuerung der Geräte per Maus und Tastatur sowie den gehobenen EU-Datenschutzstandard, auszeichnet.
Allerdings hat die Verwendung von TeamViewer auch Nachteile. Beispielsweise ist TeamViewer nicht ohne weiteres kostenlos für die Verwendung im Business-Kontext. Dies schließt unter anderem das Verbinden in bzw. das Verbinden aus Firmennetzwerken mit ein. Dafür muss eine entsprechende TeamViewer-Lizenz erworben werden. Gleichzeitig kommt es durch TeamViewer zu einer höheren Last auf den Einplatinencomputern, die unter Umständen dazu führen kann, dass die Geräte bei einer aktiven Verbindung überlasten. Die Auslastung der Systeme unter Verwendung von TeamViewer wird im Kapitel 6 genauer untersucht.

2.6 Touchdisplays

Im Rahmen dieses Projekts soll auf die Verwendung von Touchdisplays zurückgegriffen werden, um Jamulus sowie andere Anwendungen (wie z.B. TeamViewer) ohne Peripheriegeräte nutzen zu können. Dafür werden zwei Touchdisplays vom Hersteller Waveshare daraufhin untersucht, ob und inwieweit diese für den Anwendungsfall innerhalb dieses Projekts verwendet werden können.
Um nicht an ein bestimmtes Modell von Einplatinencomputer gebunden zu sein, werden ausschließlich Displays in Betracht gezogen, die als Videoeingang eine HDMI- und zur Ausgabe der Nutzereingaben eine USB-Schnittstelle nutzen. Es finden sich zwar auch diverse Touchdisplays, die als Videoeingang auf DSI (Display Serial Interface) und zur Ausgabe der Nutzereingaben auf SPI (Serial Peripheral Interface) setzen, jedoch ist DSI deutlich weniger weit verbreitet als HDMI und Eingaben via SPI können je nach verwendetem Betriebssystem zu Komplikationen führen bzw. im Vergleich zu USB einen erheblichen Mehraufwand mit sich bringen.
Die bewusste Entscheidung für eine Displaygröße von 7 Zoll lässt sich damit begründen, dass die Jamulus-Benutzeroberfläche standardmäßig nicht auf Touchscreens ausgelegt ist, was die Bedienung an kleineren Displays schwierig macht und die Verwendung größerer Displays unweigerlich eine entsprechende Mindestgröße der Clients, welche möglichst kompakt ausfallen sollen, voraussetzen würde. Unter Berücksichtigung der Lieferbarkeit kommen die beiden im folgenden untersuchten Tochdisplays in die engere Auswahl.

2.6.1 Waveshare 7inch HDMI LCD (B)

Das Waveshare 7inch HDMI LCD (B) Display ist ein 7 Zoll Display mit integrierter HDMI-Schnittstelle und USB-Anschluss zur Ausgabe der Toucheingaben und gleichzeitigen Stromversorgung des Moduls. Es bietet eine Auflösung von 800 x 480 Pixeln und ist laut Hersteller explizit für die Nutzung mit dem Raspberry Pi (ohne die Installation von Treibern) vollständig nutzbar. (Waveshare, o.J.a)

2.6.2 Waveshare 7inch HDMI LCD (C)

Das Waveshare 7inch HDMI LCD (C) Display ist ebenfalls ein 7 Zoll Display mit integrierter HDMI-Schnittstelle und USB-Anschluss zur Ausgabe der Toucheingaben und gleichzeitigen Stromversorgung des Moduls. Es bietet eine Auflösung von 1024 x 600 Pixeln und ist laut Hersteller ebenfalls explizit für die Nutzung mit dem Raspberry Pi (ohne die Installation von Treibern) vollständig nutzbar. (Waveshare, o.J.b)

3 Einrichtung des Jamulus-Servers

In diesem Kapitel wird die Einrichtung des Jamulus-Servers betrachtet. Hierzu werden zunächst die verschiedenen Server Typen und die im Laufe des Projekts verwendeten Server vorgestellt. Anschließend werden mehrere Möglichkeiten betrachtet, um einen Jamulus-Server in Betrieb zu nehmen

3.1 Mögliche Server-Typen

Nachfolgend werden die zwei Server Typen vorgestellt, die beschreiben, wie ein eigener Jamulus-Server betrieben werden kann.

3.1.1 Öffentlicher Server

Der erste Typ ist ein öffentlicher Server. Bei einem öffentlichen Server können alle Nutzer eines Jamulus-Clients auf diesen zugreifen, da dieser in dem Jamulus-Verzeichnis als mögliche Verbindungsoption gelistet wird (siehe Abbildung 1).

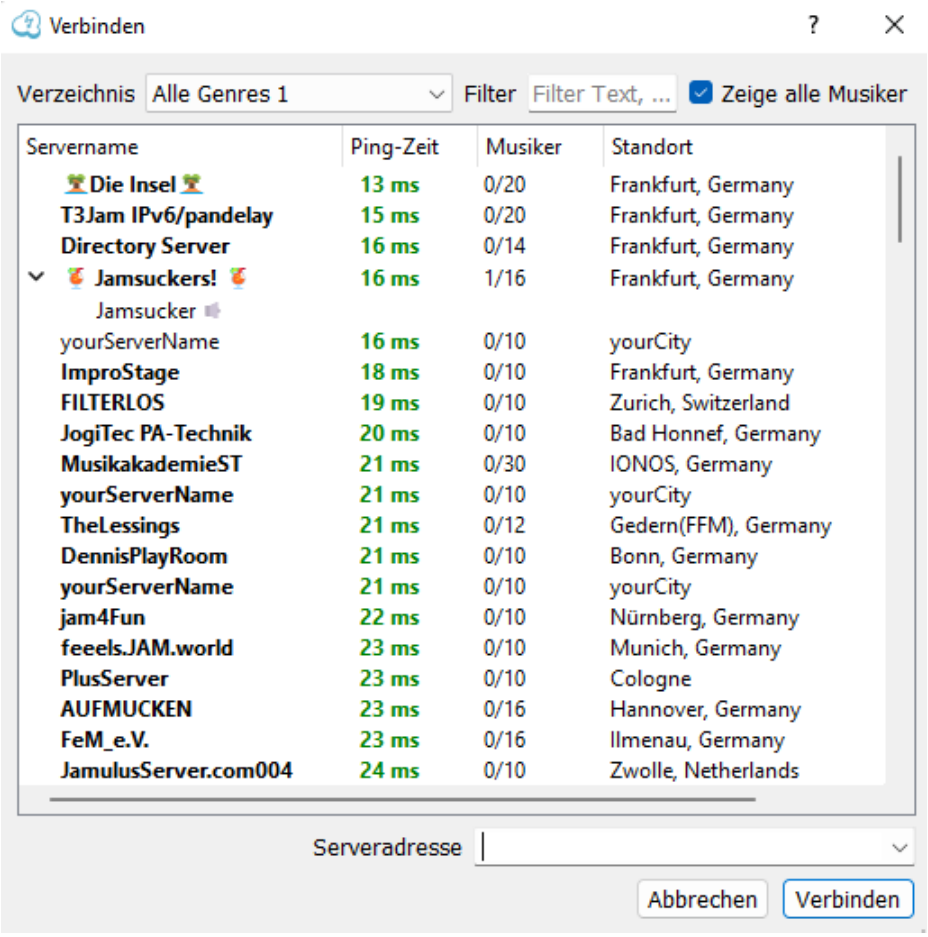


Abbildung 1: Öffentliche Server als Verbindungsoptionen im Jamulus Verzeichnis

Dieser Server Typ sollte nur verwendet werden, wenn der Zugriff von allen Jamulus-Nutzern erwünscht ist.

3.1.2 Privater Server

Der zweite Typ ist ein privater Server. Dieser Server-Typ zeichnet sich aus, dass dieser nicht im öffentlichen Jamulus-Server Verzeichnis gelistet wird. Dadurch ist dieser nur erreichbar, wenn ein Jamulus-Nutzer die Serveradresse dieses Servers kennt. Im Umkehrschluss heißt das wiederum, dass auch ein privater Server im eigentlichen Sinne ein öffentlicher Server ist, da jeder Jamulus-Nutzer in der Lage ist, sich zu einem privaten Server zu verbinden, solange dieser die Serveradresse besitzt. Um sich zu einem privaten Server zu verbinden, wird im Jamulus-Verzeichnis das Eingabefeld *Serveradresse* (siehe Abbildung 1) genutzt.
Jamulus bietet von sich aus keine Möglichkeiten an, den Zugriff zu einem Jamulus-Server zu beschränken, weshalb ein privater Jamulus-Server hierbei die einzige Möglichkeit ist, den Zugriff über die Serveradresse zumindest etwas zu beschränken. Der Typ privat ist die Default Einstellung.

3.2 Verwendete Server

Im Laufe des interdisziplinären Projektes wurden mehrere Jamulus-Server in Betrieb genommen, um herauszufinden, wie hoch die Latenz der Jamulus-Client Software je nach Server ist. Alle verwendeten Jamulus-Server wurden hierbei als *private Server* installiert. Nachfolgend werden die verwendeten Server und deren Hardware Komponenten vorgestellt.

3.2.1 Eigener Server

Als am Anfang des Projektes ein erster Server benötigt wurde, wurde aus Einfachheit ein Server aus dem privaten Bestand verwendet. Dieser war ein V-Server, der vom Hosting-Anbieter *Netcup* zur Verfügung gestellt wird. Unter einem V-Server versteht man einen virtualisierten Server d.h. dass von einem physischen Server ein Teil abgespalten wird, sodass dieser Teil als eigener Server fungiert. Dadurch das dieser virtualisierte Server von einem physischen Server abgespalten wird, nutzt es auch dessen Ressourcen, weshalb dieser einem echten physischen Server, auch Root-Server genannt unterlegen ist. Der eigene V-Server besitzt folgende Leistungsdaten:

- Betriebssystem: Ubuntu 20.04 LTS
- 2048 MiB RAM
- 20 GiB HDD
- Kosten 2,03€/Monat

Trotz schwacher Hardware, war die Nutzung von Jamulus dennoch ohne Probleme möglich. Dies zeigte sich spätestens umso deutlicher, als Herr Borucki am 19.04.2022 zu einer Probe von seinem Chor eingeladen hatte, bei dem unter anderem auch dieser Server zum Einsatz kam. Der Server funktionierte während der Chor Probe einwandfrei.

3.2.2 Server bei der DVZ

Nachdem ein erster Server durch den eigen Bestand bereitgestellt wurde, wurde nun ein Server benötigt, der über das interdisziplinäre Projekt hinaus zur Verfügung steht. Da die Datenverarbeitungszentrale der FH Aachen für FH interne Angelegenheiten auch als Hostinganbieter fungieren kann, wurde eine VM über `https://jira.fh-aachen.de` angefragt.
Da die Hardware Leistung des eigenen Servers bereits ausgereicht hat, um den Jamulus Server zu betreiben, wurden die Leistungsdaten der angefragten VM so gewählt, dass diese den Leistungsdaten des eigenen Servers etwas übersteigen, um leistungstechnisch einen Puffer zu haben. Folgende Leistungen wurden für die VM angefragt.

- Betriebssystem: Ubuntu 20.04 LTS
- CPU Kerne: 2

- RAM: 4GB
- Festplatte: 60GB
- Anzahl Netzwerkkarten: 1

Da ein Server von der DVZ nur innerhalb des FH Netzes oder mittels VPN erreichbar ist, das wiederum zu Latenzen bei der Nutzung von Jamulus führen würde, wurde zusätzlich angefragt, ob es möglich ist den Standard Port 22124/UDP vom Jamulus-Server freizugeben, sodass dieser Jamulus-Server auch von außerhalb des FH-Netzes und ohne VPN erreichbar ist. Diese Anfrage wurde jedoch abgelehnt. Die DVZ begründete dies damit, dass der Vertrag mit dem *Deutschen Forschungsnetzwerk*, das als Provider für die DVZ fungiert, aussagt, dass nur FH-Angehörige Zugriff auf dem Server haben dürfen und dieser Zugriff bei einer entsprechenden Port Freigabe nicht mehr kontrollierbar ist. Die VM wurde daher zwar zur Verfügung gestellt, jedoch ist diese nur im internen FH-Netz erreichbar. Als erste Laufzeit der VM wurden von der DVZ zunächst 4 Monate eingestellt. Gegen Ende des Projektes wurde bei der DVZ nochmal eine Anfrage gestellt, diese Laufzeit bis April 2023 zu verlängern.
Die zur Verfügung gestellte VM ist unter `https://vcenter.fh-aachen.de` einsehbar. Diese Seite ist jedoch nur im FH internen Netz oder mittels VPN erreichbar. Bei der Nutzung des Firefox Browser muss ebenfalls der private Modus ausgewählt werden, damit diese Seite aufrufbar ist. Nachfolgend wird beschrieben, wie der Server auf der Plattform vSphere einsehbar ist. Nachdem die Seite `https://vcenter.fh-aachen.de` wie oben beschrieben aufgerufen wurde, muss auf dieser Seite *vSphere (HTML5) Client Starten* ausgewählt werden (siehe rote Markierung in Abbildung 2).

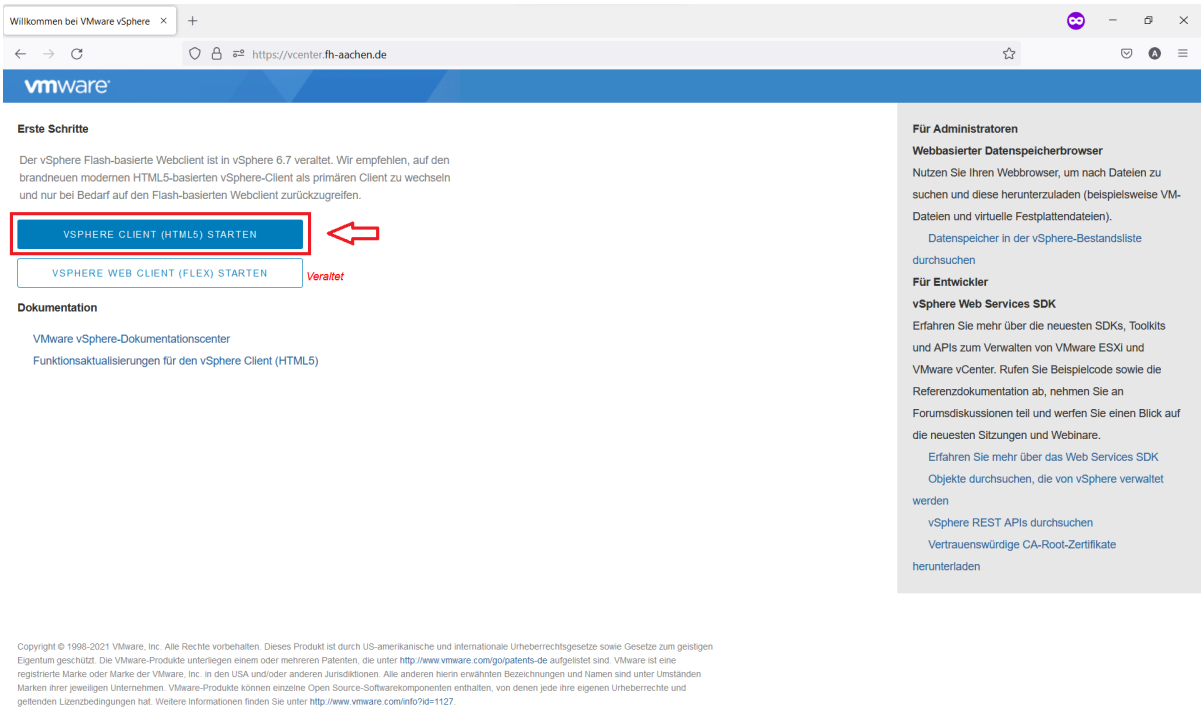


Abbildung 2: Aufruf von `https://vcenter.fh-aachen.de`

Nachdem *vSphere (HTML5) Client Starten* ausgewählt wurde, wird die Login Seite von vSphere aufgerufen. Auf dieser Login Seite wird die FH-Kennung zur Anmeldung benötigt (siehe Abbildung 3)

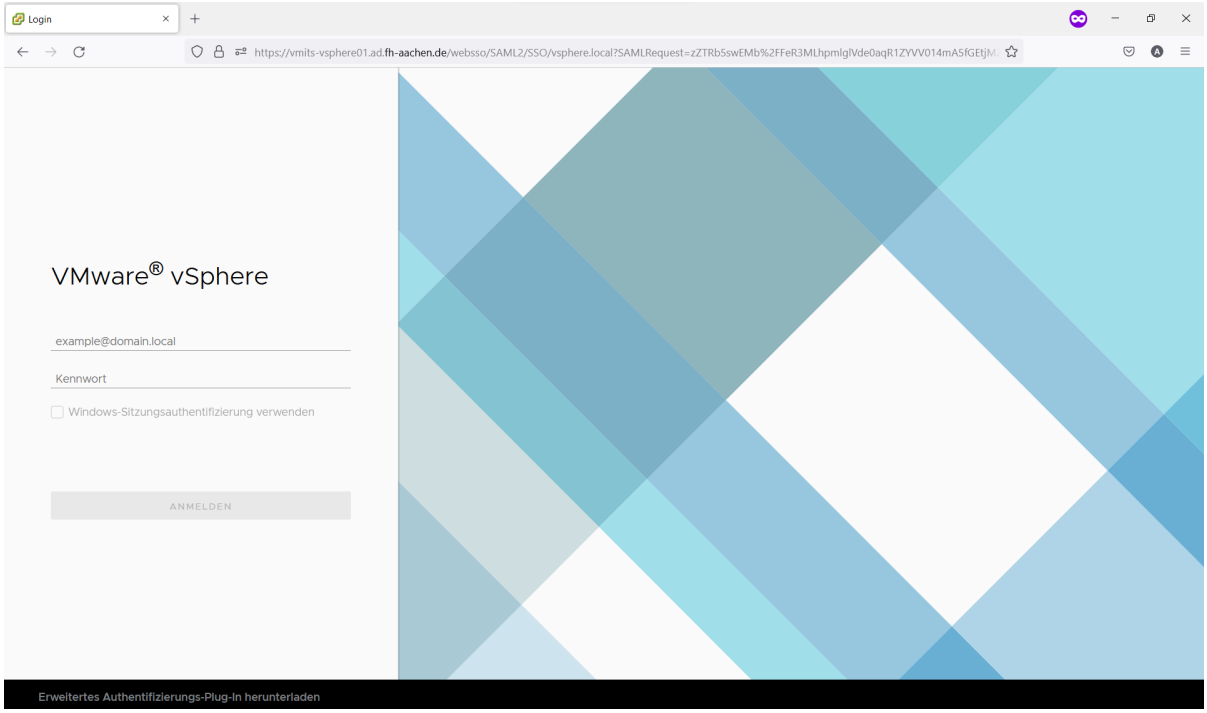


Abbildung 3: Login Seite von vSphere

Im nächsten Schritt wird die Startseite von vSphere nun angezeigt. Nun muss *VMs und Vorlagen* ausgewählt werden (siehe rote Markierung in Abbildung 4)

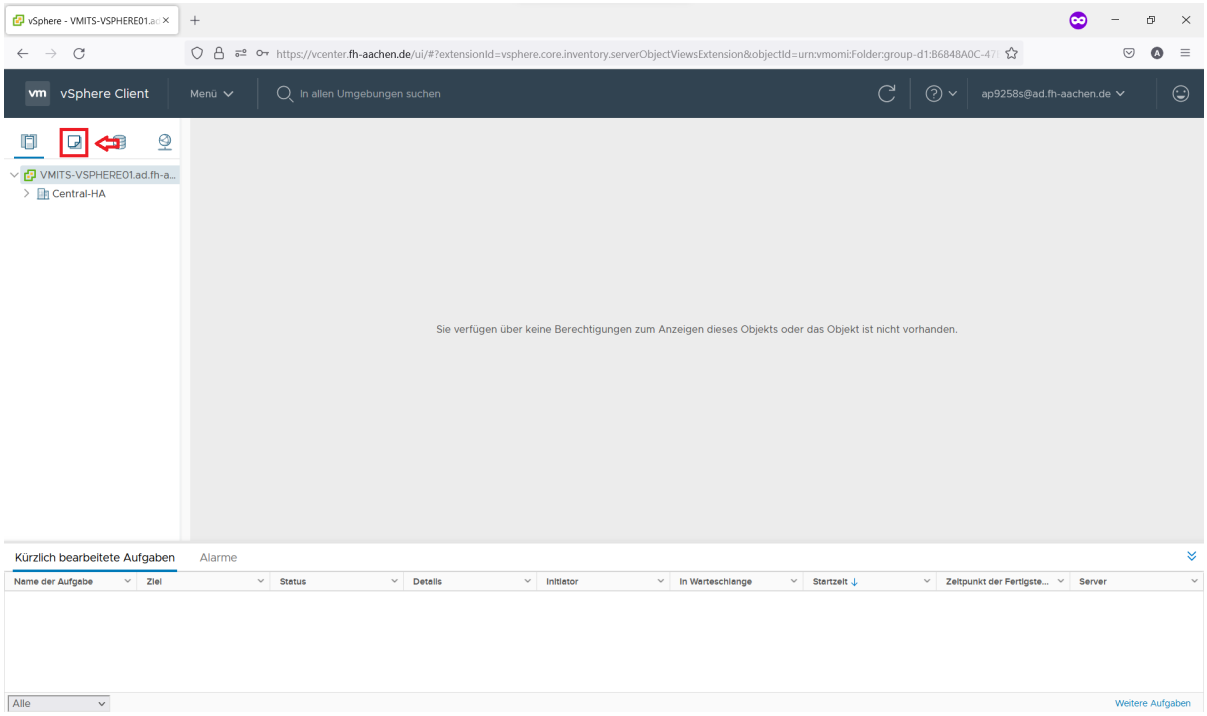


Abbildung 4: Startseite von vSphere

Anschließend muss die angezeigt Baumstruktur vollständig ausgeklappt werden. Siehe hierzu rote Markierung in der Abbildung 5.

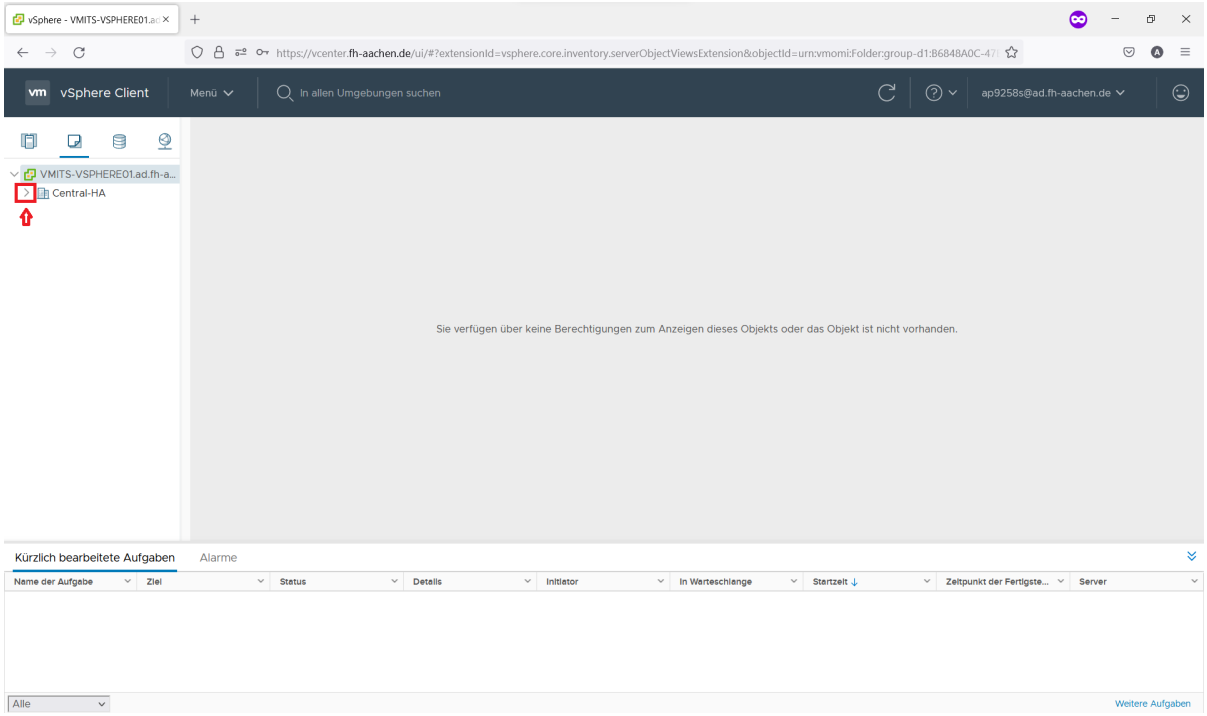


Abbildung 5: VMs und Vorlagen

Sobald die Baumstruktur vollständig ausgeklappt wurde, muss das letzte Element ausgewählt werden, um die angefragte VM einsehen zu können (siehe rote Markierung in Abbildung 6).

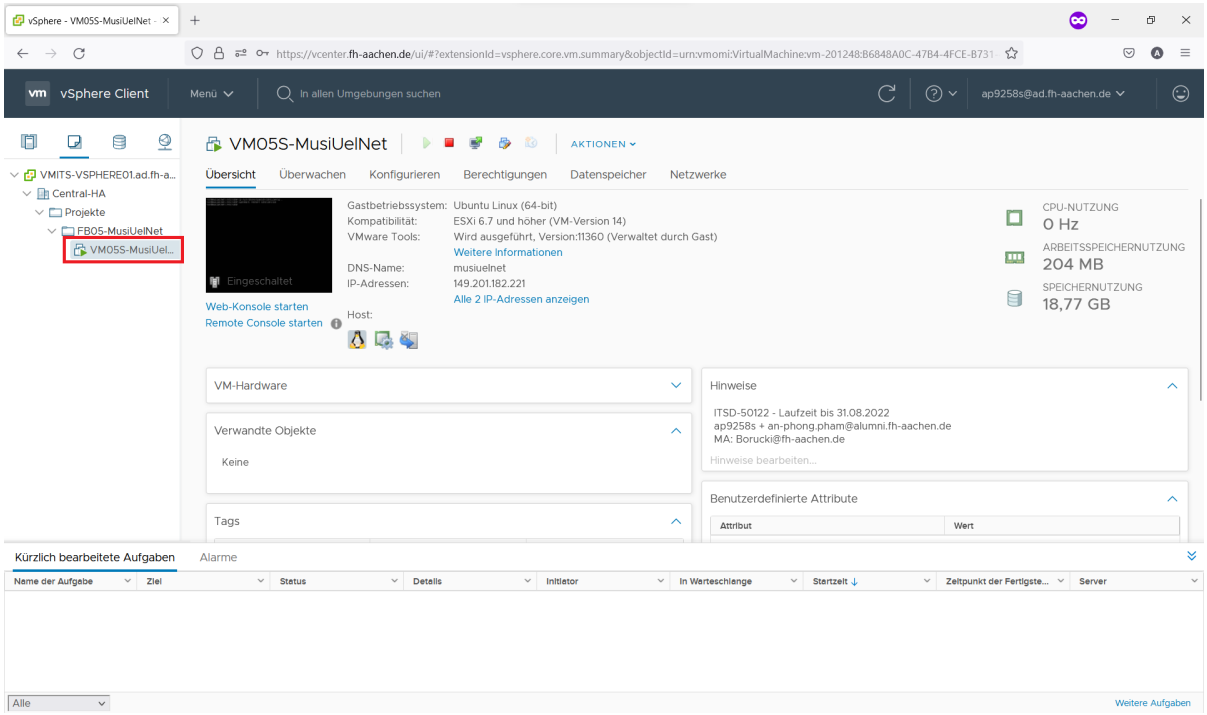


Abbildung 6: Anzeigen der angefragten VM

Die VM lässt sich nun über dieses Portal administrieren, wenn *Konsole starten* ausgewählt wird (siehe rote Markierung mit Pfeil in Abbildung 7).

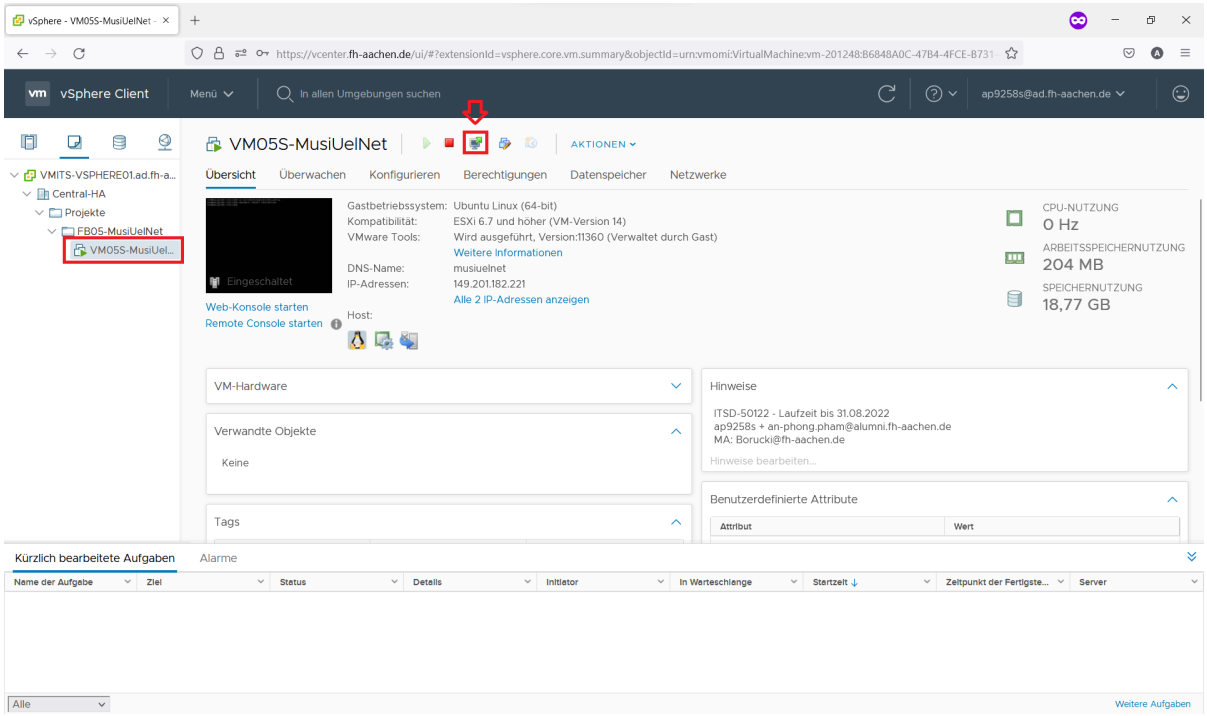


Abbildung 7: Administrieren der VM durch starten der Web Konsole

Nach der Auswahl muss zusätzlich noch die Art der Web Konsole ausgewählt werden. Hierbei kann die Vorauswahl bestätigt werden (siehe Abbildung 8).

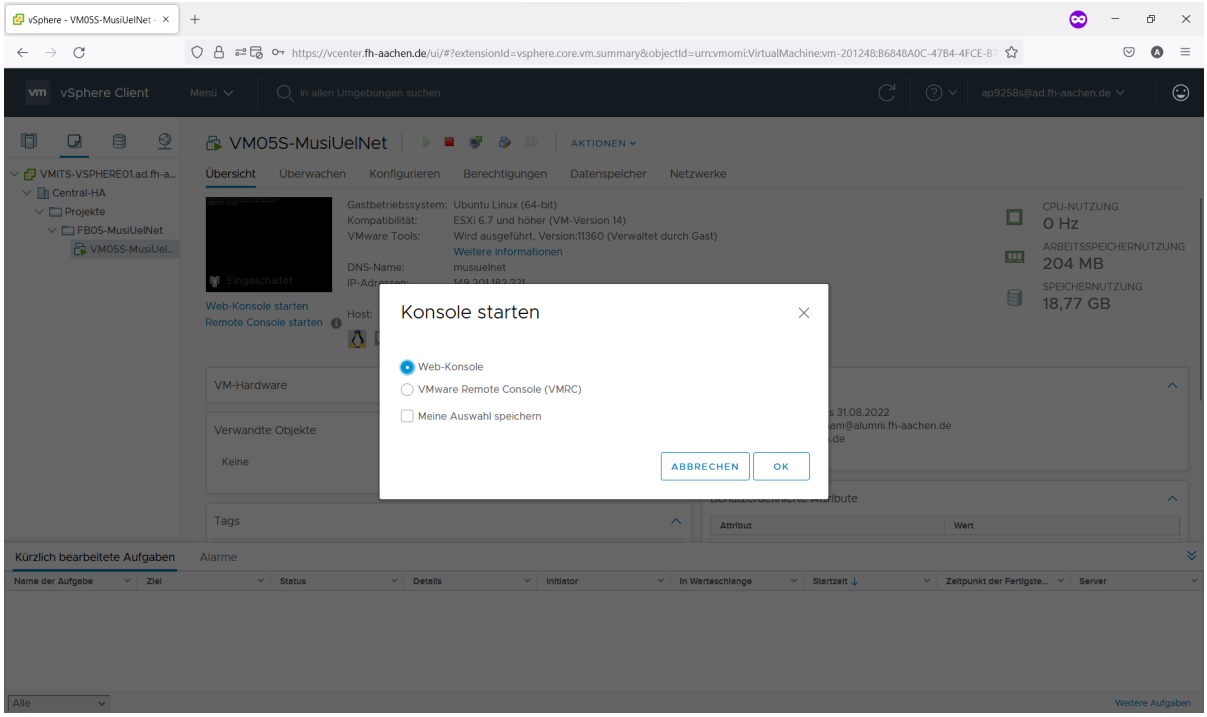


Abbildung 8: Web Konsole bestätigen und starten

Die Web Konsole wird als Popup gestartet. Bei einigen Browsern werden Popups standardmäßig blockiert weshalb dieses Popup zugelassen werden muss, damit die Web Konsole starten kann (siehe Abbildung 9).

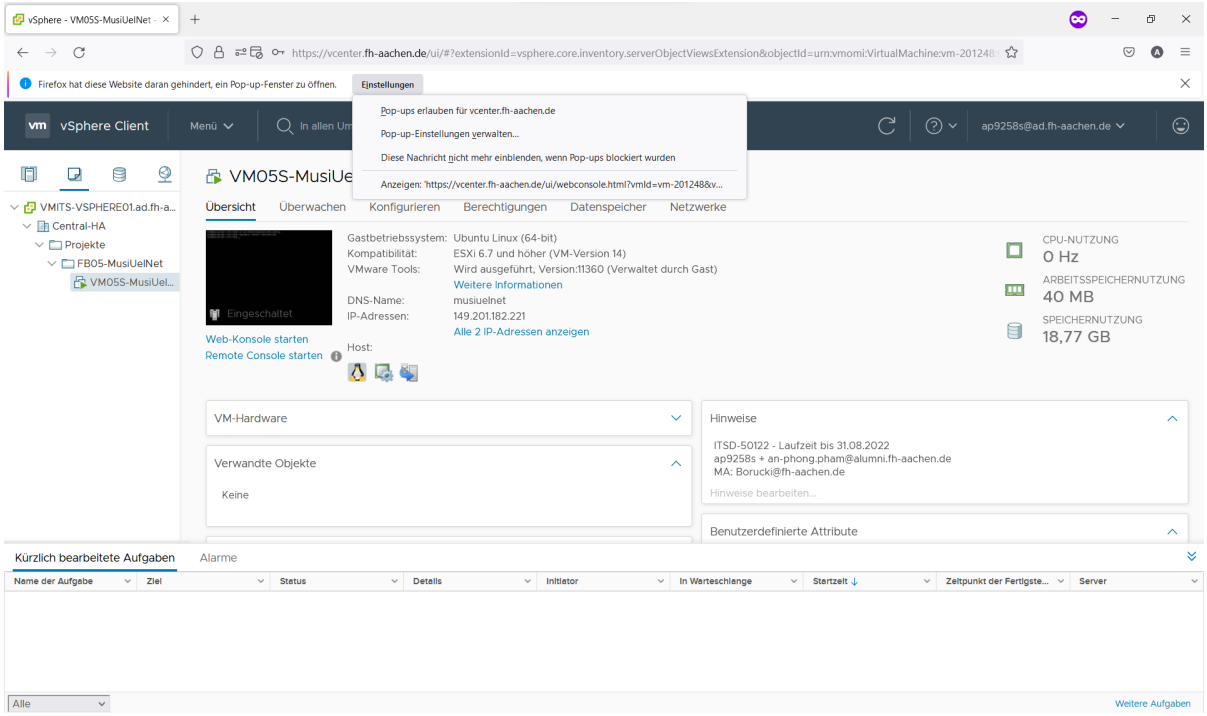


Abbildung 9: Popup der Webkonsole zulassen

Die gestartete Web Konsole als Popup wird in Abbildung 10 dargestellt.

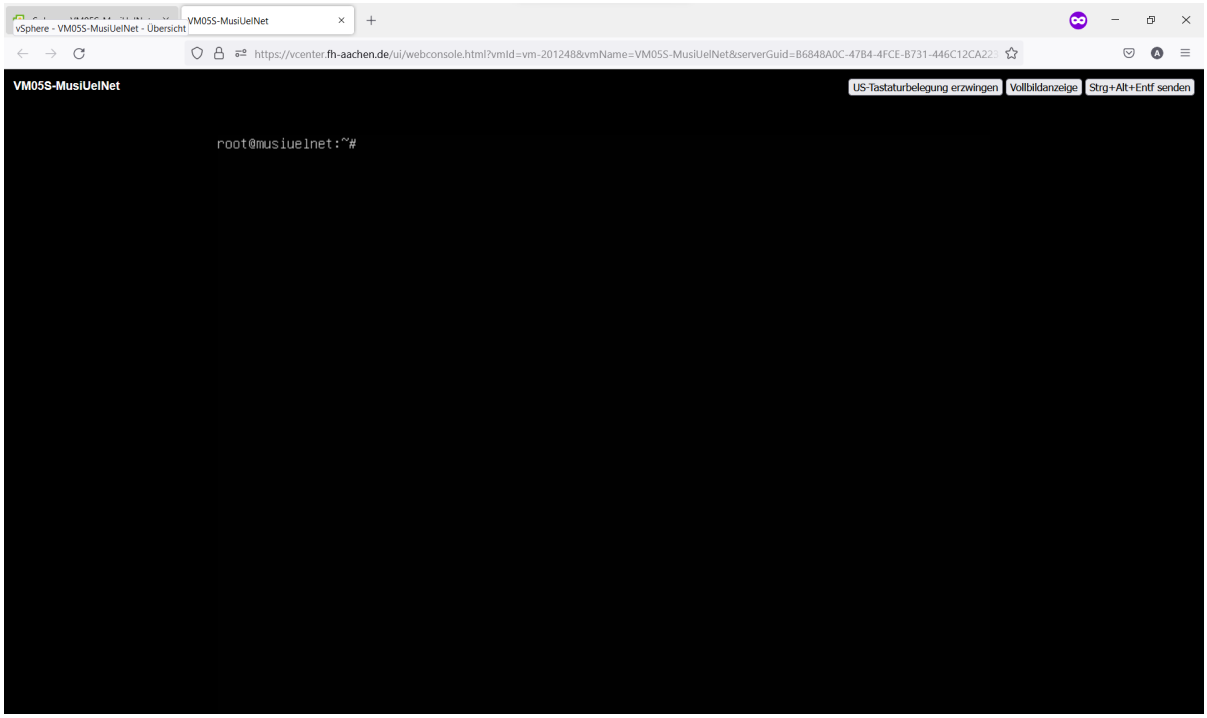


Abbildung 10: Gestartete Web Konsole

Alternativ kann zur Administration der VM auch ein SSH Client wie zum Beispiel PuTTY

verwendet werden. Hierbei muss die IP-Adresse, der Nutzername, sowie das Passwort der VM angegeben werden.

3.2.3 Server bei einem Hosting Anbieter

Da der Server von der DVZ nur im internen FH Netz oder mittels VPN erreichbar ist, musste zusätzlich eine Alternative gesucht werden, bei dem es möglich ist, diesen von jedem Netz aus zu erreichen. Ebenfalls würde auch dadurch getestet werden können, inwiefern die Latenz vom entsprechenden Server Standort abhängig ist. Als Messlatte wurde hierbei ebenfalls, der eigene private Server gewählt, um bei den verschiedenen Hosting Anbietern nach einem geeigneten Server zu suchen. Für die Server Recherche wurden die Hosting Anbieter Netcup, Hetzner und IONOS gewählt. Neben einem Deutschen Server Standort, sollte ebenso die vertragliche Laufzeit des Servers betrachtet werden. Bei Netcup kamen folgende V-Server in Frage:

VPS 200 G10s

- 2 CPU Kerne
- 2GB Arbeitsspeicher
- 40GB Festplattenspeicher
- 80TB Traffic
- 2,99€/Monat

VPS 500 G10s

- 4 CPU Kerne
- 4GB Arbeitsspeicher
- 80GB Festplattenspeicher
- 80TB Traffic
- 5,55€/Monat

Netcup gibt an, dass deren Server entweder in Nürnberg oder Wien stehen. Die vertragliche Mindestlaufzeit beträgt bei Netcup 0 Monate. Somit kann jederzeit der Server gekündigt werden.
Im V-Server Sortiment von Hetzner wurden hingegen folgende in Betracht gezogen:

CX11

- 1 CPU Kern
- 2GB Arbeitsspeicher
- 20GB Festplattenspeicher
- 20TB Traffic
- 4,15€/Monat

CPX11

- 2 CPU Kerne
- 2GB Arbeitsspeicher
- 40GB Festplattenspeicher
- 20TB Traffic
- 4,75€/Monat

CX21

- 2 CPU Kerne
- 4GB Arbeitsspeicher
- 40GB Festplattenspeicher
- 20TB Traffic
- 5,83€/Monat

Genauso wie bei Netcup, gibt es auch bei Hetzner keine Mindestlaufzeit. Bei Netcup gilt jedoch das Motto *Pay as you go*, das wiederum bedeutet, dass nur das bezahlt werden muss, das auch verwendet wird. Die Server Standorte gibt Hetzner mit Falkenstein, Nürnberg, Helsinki und Ashburn an.
Als letztes wurde bei IONOS nach einem geeigneten Server recherchiert. Hierbei kamen folgende in die engere Auswahl:

VPS S

- 1 CPU Kern
- 512MB Arbeitsspeicher
- 10GB Festplattenspeicher
- Unbegrenzter Traffic

- 1€ in den ersten 6 Monaten danach 2€/Monat

VPS M

- 2 CPU Kern
- 2GB Arbeitsspeicher
- 80GB Festplattenspeicher
- Unbegrenzter Traffic
- 2€ in den ersten 6 Monaten danach 5€/Monat

VPS L

- 2 CPU Kern
- 4GB Arbeitsspeicher
- 120GB Festplattenspeicher
- Unbegrenzter Traffic
- 5€ in den ersten 6 Monaten danach 10€/Monat

Bei IONOS besitzt ein Server die Mindestlaufzeit von 12 Monat. Vorteil hierbei ist jedoch, dass der Server Standort mit Baden-Baden angegeben wird, womit der Server Standort immer in Deutschland ist.
Nachdem die oben genannten möglichen Server Kandidaten vorgestellt wurden, wurde schnell entschieden, dass Netcup das beste Preis Leistungsverhältnis bietet, weshalb der *VPS 200 G10s* gebucht wurde, da dieser V-Server die Leistungsanforderungen schon erfüllt. Dieser bietet nämlich als aktuell schwächster V-Server im Vergleich zum eigenen privaten Server bereits eine stärkere Hardware.

Der gebuchte Netcup Server lässt sich auf der Server Control Panel Plattform unter `https://www.servercontrolpanel.de/SCP/Home` administrieren (siehe Abbildung 11).

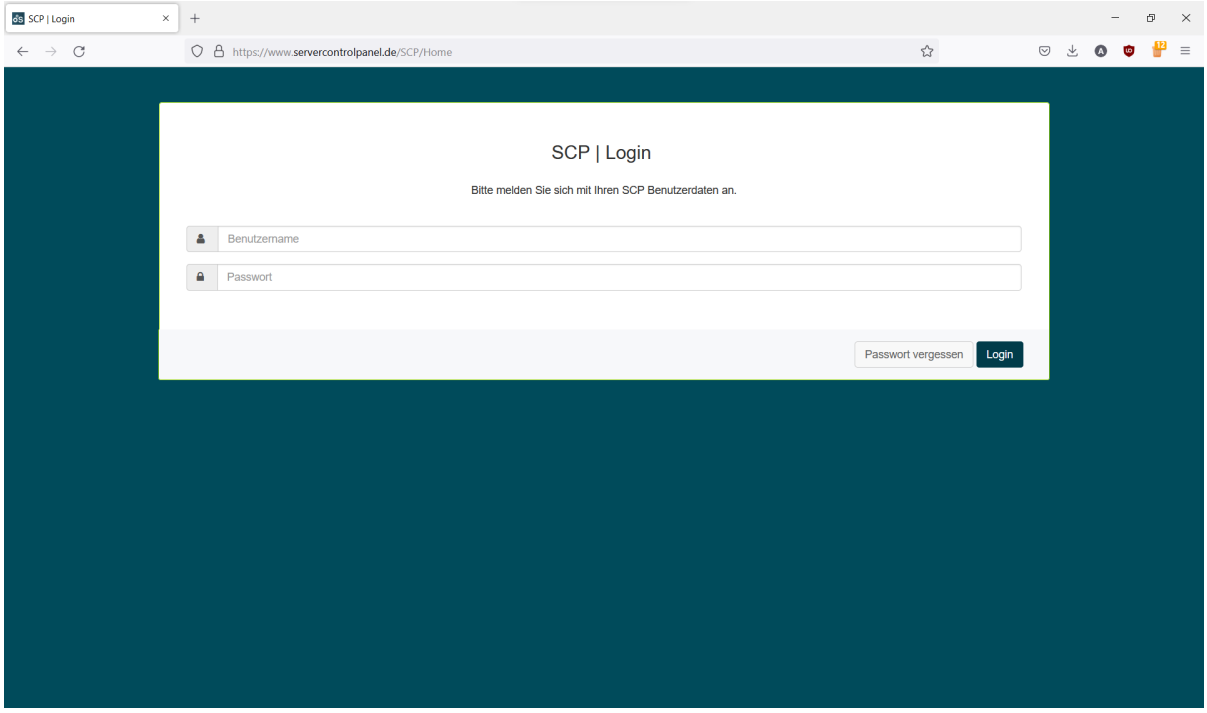


Abbildung 11: Login Seite des Server Control Panels

Nach der Anmeldung gelangt man die Startseite des Server Control Panels, bei dem alle Server aufgelistet werden, die gebucht werden. Da im Rahmen des interdisziplinären Projekts nur ein V-Server gebucht wurde, wird auch nur dieser gelistet. Dieser aufgelistete V-Server kann ausgewählt werden (siehe Abbildung 12).

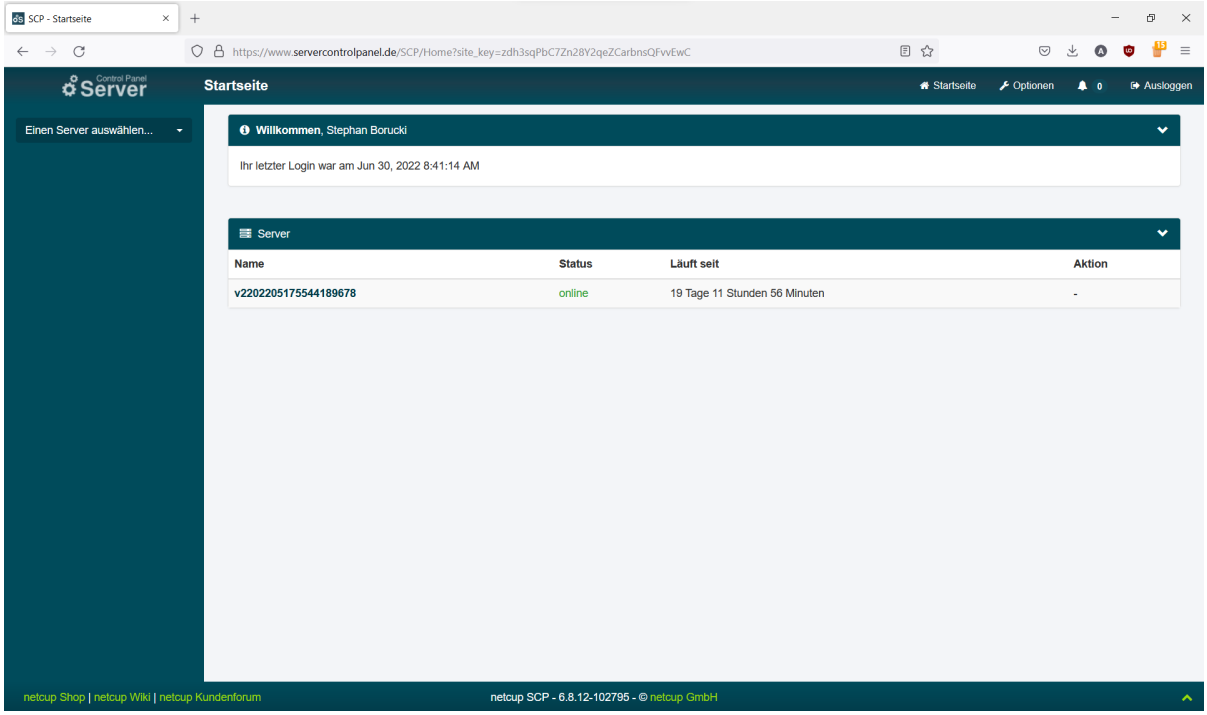


Abbildung 12: Startseite des Server Control Panels

Sobald dieser ausgewählt wurde gelangt man auf eine Seite mit Allgemeinen Infos über

den gebuchten Server (siehe Abbildung 13).

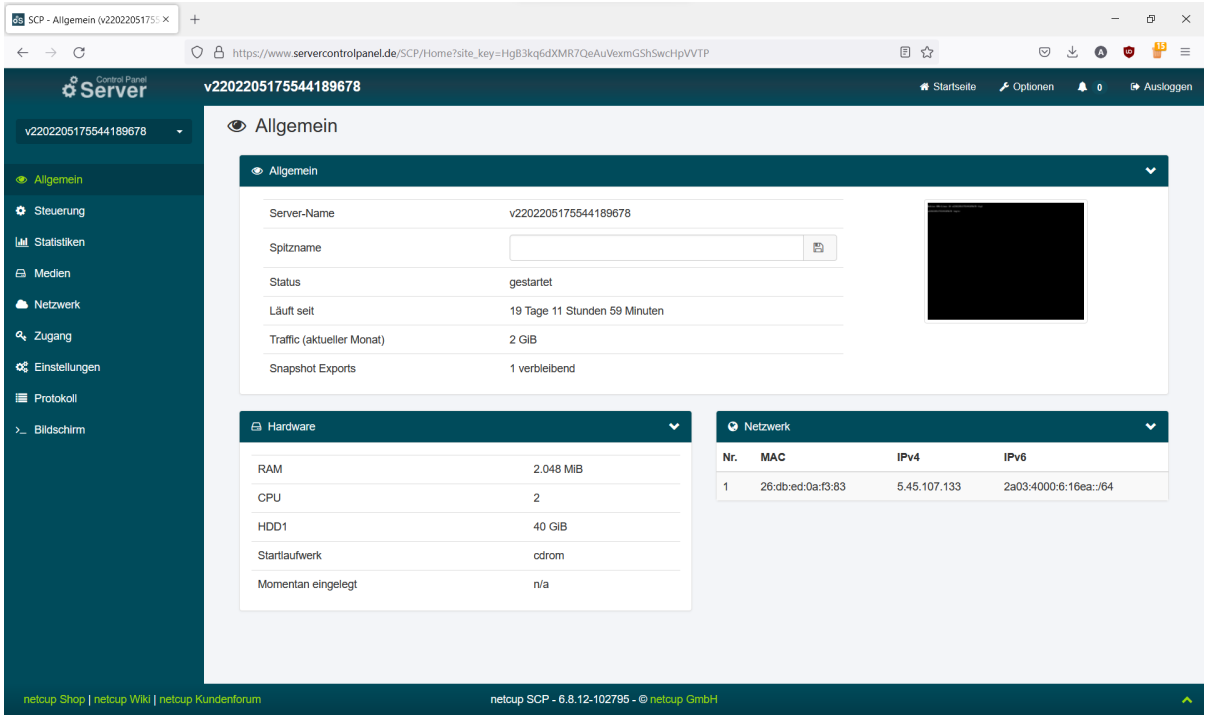


Abbildung 13: Allgemeine Infos des gebuchten Servers

Auf der linken Seite befindet sich eine Navigation. Die wichtigsten Reiter in der Navigation sind hierbei die Reiter:

- Steuerung, womit sich der Server Ein- und Ausschalten lässt
- Medien, womit sich die Festplatte des Servers komplett löschen lässt, sowie auch der Server komplett neu installiert werden kann
- >_ Bildschirm, womit direkt auf den Server über eine Web Konsole zugegriffen werden kann

Der Server wurde mit dem Betriebssystem Debian 10 Buster aufgesetzt.
Aus Sicherheitsgründen wurde der Root Zugang jedoch deaktiviert und ein neuer Nutzer mit dem Linux Befehl in Listing 1 angelegt, der genutzt werden soll, um sich zukünftigt zum mit dem Server via SSH-Client zu verbinden.

```
adduser
```

Listing 1: Befehl zum Erstellen eines neuen Nutzers

3.2.4 Einplatinen Computer als Server

In der späten Phase des Projektes wurde ein Versuchsaufbau erstellt, bei dem ein Jamulus Server lokal über einen Switch angesteuert werden sollte, um die Latenzen bei

einer quasi optimalen Umgebung zu messen. Hierfür wurde beim Versuchsaufbau auf den bereits vorgestellten ODROID XU4Q zurückgegriffen. Nähere Informationen zu diesem Einplatinen Computer können unter 2.2 nachgelesen werden.
Der für den Versuchsaufbau verwendete ODROID Server wurde mit dem Betriebssystem Ubuntu in der Version 22.04 Minimal geflasht. Nach dem Flash Vorgang ist das Betriebssystem bereits vollständig installiert. Bei der Minimal Version des Betriebssystem existiert ausschließlich ein root Nutzer mit dem Passwort odroid. Um nicht alle Befehle mit root Rechten azszuführen, wurde wie auf dem gebuchten Netcup Server, auch hier ein neuer Nutzer mit dem Befehl aus Listing 1 erstellt.
Damit der ODROID Server im Versuchsaufbau über den Switch erreichbar ist, musste dem ODROID Server eine feste IP-Adresse vergeben werden. Für diesen Zweck wurde das Programm Netplan verwendet. Dieses ist bereits auf dem ODROID installiert, weshalb im Verzeichnis /etc/netplan nur die bereits vorhandene config Datei 01-netcfg.yaml mit dem Befehl in Listing 2 angepasst werden musste, um die statische IP-Adresse 192.169.1.221 zu setzen.

```
sudo vim /etc/netplan/01-netcfg.yaml
```

Listing 2: Befehl zum Bearbeiten der vorhandenen Netplan config Datei

Der angepasste Inhalt dieser config Datei ist in Listing 3

```
network:
  version: 2
  renderer: networkd
  ethernets:
    ens3:
      dhcp4: no
      addresses:
        - 192.169.1.221/24
      gateway4: 192.169.1.1
      nameservers:
          addresses: [8.8.8.8, 1.1.1.1]
```

Listing 3: Angepasster Inhalt der 01-netcfg.yaml

3.3 Installation der Jamulus-Server Software

Jamulus liefert von Haus aus die Client-Software und die Server-Software in einem Paket aus. In diesem Paket besitzt nicht nur die Client-Software eine grafische Benutzeroberfläche, sondern auch der Jamulus-Server.
Um einen Jamulus-Server Ressourcen schonend zu betreiben bietet Jamulus, jedoch auch auch eine Headless-Version der Server-Software an. Bei der Headless Server Version, wird ausschließlich der Jamulus-Server angeboten. Hierbei bedeutet Headless ebenfalls, dass der ausgelieferte Jamulus-Server keine grafische Benutzeroberfläche

besitzt. Nachfolgend wird die Installation der Jamulus-Server Software als Headless-Version betrachtet. Dabei wird zwischen der Installation auf einem Server mit x86 Architektur und der Installation auf einem Server mit ARM Architektur unterschieden.

3.3.1 Installation auf einem Server mit x86 Architektur

Um Jamulus als Headless-Server zu installieren, muss zunächst die aktuellste Paketdatei der Jamulus Headless-Server Software heruntergeladen werden. Dies kann mit dem Befehl in Listing 4 durchgeführt werden.

```
wget https://github.com/jamulussoftware/jamulus/releases/download/r3_8_2/
    jamulus_headless_3.8.2-1_ubuntu_amd64.deb
```

Listing 4: Befehl zum Herunterladen der aktuellsten Paketdatei

Anschließend muss mit dem Befehl aus Listing 5 sichergestellt werden, dass die Paketquellen der Standardpakete aktuell sind.

```
sudo apt update
```

Listing 5: Auslesen der aktuellen Paketquellen

Im nächsten Schritt kann nun die heruntergeladene Paketdatei der Jamulus Headless-Server Software installiert werden (siehe Listing 6).

```
sudo apt install ./jamulus_headless_3.8.2-1_ubuntu_amd64.deb
```

Listing 6: Installation der Jamulus Paketdatei

Die Installation der heruntergeladenen Jamulus Paketdatei, erstellt gleichzeitig einen systemd Service für den Jamulus-Server. Systemd ist ein essentieller Bestandteil in aktuellen Linux Betriebssystemen, der unter anderem dafür sorgt, dass das Betriebssystem korrekt startet, indem es alle nötigen Prozesse startet. Das Anlegen eines Services für den Jamulus Headless-Server hat somit den Vorteil, dass das Betriebssystem selbst für das starten und überwachen der Jamulus Headless-Server Software verantwortlich ist. Dabei kann der Befehl in Listing 7 genutz werden, um den Betriebssystem mitzuteilen, dass dieser den Jamulus Server mit dem Starten des Betriebssystem starten soll.

```
sudo systemctl enable jamulus-headless
```

Listing 7: Jamulus mit Systemstart starten

Da es sich bei der Jamulus Installation um einen Service handelt, kann dieser ebenfalls mit den Befehlen in Listing 8 gestartet und gestoppt werden. Der Server ist nach der Installation bereits gestartet und muss deshalb nicht mehr gestartet werden.

```
# Starten des Servers
sudo systemctl start jamulus-headless

# Stopppen des Servers
sudo systemctl stop jamulus-headless
```

Listing 8: Starten und Stoppen des Servers

Wie der Server gestartet wird, kann in der von der Installation angelegten Service Datei betrachtet werden. Diese Datei kann mittels dem Befehl in Listing 9 eingesehen werden.

```
less /lib/systemd/system/jamulus-headless.service
```

Listing 9: Einsehen des Jamulus Headless-Server Services

Der Inhalt der Service Datei ist folgender (siehe Listing 10):

```
1  [Unit]
2  Description=Jamulus headless server
3  After=network.target
4  StartLimitIntervalSec=0
5
6  [Service]
7  Type=simple
8  User=jamulus
9  Group=nogroup
10 NoNewPrivileges=true
11 ProtectSystem=true
12 ProtectHome=true
13 Nice=-20
14 IOSchedulingClass=realtime
15 IOSchedulingPriority=0
16
17 #### Change this to publish this server, set genre, location and other
       parameters.
18 #### See https://jamulus.io/wiki/Command-Line-Options ####
19 ExecStart=/bin/sh -c 'exec /usr/bin/jamulus-headless -s -n'
20
21 Restart=on-failure
22 RestartSec=30
23 StandardOutput=journal
24 StandardError=inherit
25 SyslogIdentifier=jamulus
26
27 [Install]
28 WantedBy=multi-user.target
```

Listing 10: Inhalt der Service Datei

Wenn der Service gestartet wird, wird Zeile 19 der Service Datei in Listing 10 ausgeführt, um den Jamulus Service zu starten. Die Optionen -s sagt dabei aus, dass Jamulus im Server Modus gestartet werden soll, während die Option -n dafür sorgt, dass der Server ohne grafische Oberfläche gestartet wird. Jamulus bietet unter `https://jamulus.io/wiki/Running-a-Server#command-line-options` noch weitere Optionen an, die verwendet werden können. Im Rahmen des interdisziplinären Projektes wur-

de die Option –welcome-message verwendet, damit ein Chat Fenster mit einer Begrüßungsbotschaft geöffnet wird, sobald sich ein Jamulus-Client mit dem Server verbindet. Zeile 19 in Listing 10 wurde daher wie in Listing 11 dargestellt angepasst.

```
ExecStart=/bin/sh -c 'exec /usr/bin/jamulus-headless -s -n --welcome-
    message "Willkommen zum MusiUelNet IP-Server"'
```

Listing 11: Hinzufügen der Option –welcome-message

Damit die Änderungen der Service Datei aktiv werden, müssen die Befehle in Listing 12 nacheinander ausgeführt werden.

```
# 1.
sudo systemctl daemon-reload

# 2.
systemctl restart jamulus-headless
```

Listing 12: Änderungen der Service Datei aktiv werden lassen

3.3.2 Installation auf einem Server mit ARM Architektur

Da der für den erwähnten Versuchsaufbau verwendete ODROID eine ARM Architektur besitzt, kann der Installationsanleitung in 3.3 nicht gefolgt werden, um einen Jamulus-Server zu installieren, da die in 3.3 verwendete Paketdatei nur x86 kompatibel ist. Jamulus bietet leider keine Headless-Server Software Version von Jamulus für ARM Server an, weshalb beim ODROID auf das komplett Paket der Jamulus Software zurückgriffen werden muss. Komplettpaket bedeutet in diesem Fall, dass sowohl der Jamulus-Client samt grafischer Oberfläche als auch der gewünschte Jamulus-Server samt grafischer Oberfläche installiert werden muss. Für die Installation kann daher der Anleitung in 4.2 gefolgt werden. Hierbei muss aber angemerkt werden, dass die Anleitung in 4.2 nur gefolgt werden muss, bis erwähnt wird, dass Jamulus vollständig installiert wurde.
Diese Installation auf dem ODROID hat den Nachteil, dass die installierte Jamulus Version nicht ganz Ressourcen schonend ist, zudem muss der Server noch manuell gestartet werden, sowie eine Möglichkeit eingerichtet werden, dass dieser Server auch automatisch startet, wenn das Betriebssystem startet. Als Lösung für diese Probleme wird die Idee der Paketdatei aus 3.3 aufgegriffen, da die Paketdatei einen systemd Service für den Jamulus Server einrichtet. Dieser Service für den Jamulus-Server muss nun, manuell eingerichtet werden. Dazu wird zunächst ein Bash-Skript mit dem Befehl aus Listing 13 mit dem Inhalt aus Listing 14 erstellt.

```
sudo vim /usr/bin/start_jamulus.sh
```

Listing 13: Gleichzeitiges Erstellen und Editieren des jamulus Bash-Skript

```
#!/bin/bash
Jamulus -s -n --welcome-message "Willkommen zum MusiUelNet IP-Server"
```

Listing 14: Inhalt des erstellten Bash-Skriptes

Das Bash-Skript in Listing 14 startet Jamulus im Server Modus, deativiert die grafische Oberfläche und sorgt dafür, dass ein Chat Fenster mit einer Willkomensbotschaft angezeigt wird, sobald ein Jamulus-Client sich zum Server verbindet. Dieses Bash-Skript muss ebenfalls noch ausführbar gemacht werden, damit dieses von der entsprechenden Service Datei später ausgeführt werden kann. Der Befehl hierzu befindet sich in Listing 15.

```
sudo chmod +x /usr/bin/start_jamulus.sh
```

Listing 15: Befehl macht das Bash-Skript ausführbar

Im nächsten Schritt wird die entsprechende Service Datei erstellt, die dafür sorgt, dass das soeben erstellte Bash-Skript ausgeführt wird. Die Erstellung der Service Datei erfolgt mit dem Befehl aus Listing 16.

```
sudo vim /lib/systemd/system/jamulus-headless.service
```

Listing 16: Gleichzeitiges Erstellen und Editieren der Service Datei

Der Inhalt mit dem die Service Datei beschrieben wird, wird in Listing 17 dargestellt.

```
[Unit]
Description=Start Jamulus Server

[Service]
ExecStart=/usr/bin/start_jamulus.sh

[Install]
WantedBy=multi-user.target
```

Listing 17: Inhalt der erstellten Service Datei

Im nächsten Schritt muss der Befehl aus Listing 18 ausgeführt werden, damit der Service aktiv wird.

```
sudo systemctl daemon-reload
```

Listing 18: Änderungen der Service Datei aktiv werden lassen

Um den Jamulus Server zu starten, kann nun der Start Befehl aus Listing 8 verwendet werden. Um zu überprüfen, ob der der Jamulus-Server aktiv läuft, kann der Befehl in Listing 19.

```
sudo systemctl status jamulus-headless.service
```

Listing 19: Status des Jamulus-Server abfragen

Wenn der Server aktiv läuft, sollte die Ausgabe der Abbildung 14 entsprechen.

```
musiuelnet@v2202205175544189678:~$ sudo systemctl status jamulus-headless.service
[sudo] password for musiuelnet:
● jamulus-headless.service - Jamulus headless server
   Loaded: loaded (/lib/systemd/system/jamulus-headless.service; enabled; vendor preset: enabled)
   Active: active (running) since Thu 2022-06-30 10:41:26 CEST; 2 weeks 5 days ago
 Main PID: 477 (jamulus-headles)
    Tasks: 2 (limit: 2359)
   Memory: 49.7M
   CGroup: /system.slice/jamulus-headless.service
           └─477 /usr/bin/jamulus-headless -s -n --welcomemessage Willkommen zum MusiUelNet IP-Server

Warning: Journal has been rotated since unit was started. Log output is incomplete or unavailable.
musiuelnet@v2202205175544189678:~$
```

Abbildung 14: actvive Status als Indikator für aktiv laufenden Jamulus-Server

Damit der Jamulus-Server beim Systemstart mitstartet, kann der Befehl aus Listing 7 verwendet werden.

4 Einrichtung des Jamulus-Clients auf Raspberry Pi 4

In diesem Kapitel wird die Einrichtung des Jamulus Clients auf dem Raspberry Pi 4 (mit 2 GB RAM) beschreiben. Dafür wird zunächst auf die Vorgehensweise bei der Installation und der anschließenden Konfiguration von Raspberry Pi OS zur Nutzung als Jamulus-Client eingegangen. Im Anschluss daran wird die Installation des Jamulus-Clients, sowie der dafür benötigten Pakete beschrieben. Anschließend wird basierend auf der Diskussion in Kapitel 2.3 Teamviewer zur Fernwartung installiert und eingerichtet. Zuletzt wird dann noch ein Touchdisplay angeschlossen, um die Verwendung weiterer Peripheriegeräte obsolet zu machen.

4.1 Installation und Vorbereitung von Raspberry Pi OS

Für die Installation von Raspberry Pi OS wird von der Raspberry Pi Foundation eine Software, der sogenannte Raspberry Pi Imager, bereitgestellt, die auf einem Computer installiert wird. Nach der Installation der Software kann nun das Betriebssystem Raspberry Pi OS auf eine SD Karte geschrieben werden.

Abbildung 15: Raspberry Pi Imager v1.7.2 zur Übertragung des Raspberry Pi OS auf eine SD-Karte.

Im ersten Schritt wird dafür zunächst das Betriebssystem ausgewählt. Als erste Option empfiehlt die Software das System 'Raspberry Pi OS (32-bit)', die für dieses Projekt ausgewählt wird. Als nächstes muss die SD-Karte ausgewählt werden, auf die das Betriebssystem geschrieben werden soll. Im letzten Schritt wird dann die Option 'Schreiben' gedrückt. Damit beginnt das Schreiben des Betriebssystems auf die SD-Karte. Ist dieser Prozess abgeschlossen, kann die SD-Karte aus dem Computer entnommen werden und in den Raspberry Pi 4 gesteckt werden. Im Anschluss daran wird nun ein Netzteil angeschlossen und der Raspberry Pi gestartet.

Bevor nun weitere Installationen oder Konfigurationen vorgenommen werden, werden zunächst alle Pakete sowie das Betriebssystem aktualisiert. Hierfür ist es erforderlich, dass der Raspberry Pi mit dem Internet verbunden ist. Zur Aktualisierung der Pakete und des Betriebssystem wird folgender Befehl ausgeführt:

```
# Updaten aller Programme, die auf dem System installiert sind
sudo apt update && sudo apt upgrade -y
```

Im Anschluss daran ist der Raspberry Pi 4 mit Raspberry Pi OS bereit, um den Jamulus-Client zu installieren.

4.2 Einrichtung von Jamulus

An dieser Stelle wird nun der zuvor konfigurierte Raspberry Pi 4 (2 GB) aus Kapitel 4.1 verwendet, um darauf die Jamulus-Clientsoftware zu installieren und einzurichten. Dafür wird in Anlehnung an das offizielle Jamulus-COMPILING-Scripts von Jamulus (2022a) ein Bash-Shell-Script erstellt, das die Jamulus-Installation auf dem Raspberry Pi vollständig automatisiert und zeitgleich Jamulus für die Nutzung auf dem Raspberry Pi optimiert.

Da nun der Raspberry Pi 4 vollständig eingerichtet ist und alle Systempakete aktualisiert wurden, kann mit der Installation der für Jamulus benötigten Programme / Bibliotheken begonnen werden. Dabei wird sich überwiegend an die Empfehlungen / Anweisungen des Jamulus-COMPILING-Scripts gehalten. Die einzige Ausnahme betrifft das Paket 'qt5-default', da dieses in den neuen Ubuntu und Debian Versionen obsolet geworden ist (Langasek, 2021).

```
# Installiere die benötigten Pakete, um Jamulus und JACK zu verwenden.
## (qt5-default ist nun obsolet).
sudo apt-get install build-essential qt5-qmake qtdeclarative5-dev
      qttools5-dev-tools libjack-jackd2-dev qjackctl -y
```

Da nun alle von Jamulus benötigten Programme / Bibliotheken installiert wurden, kann nun mit der eigentlichen Installation von Jamulus begonnen werden. Jamulus empfielt im Jamulus-COMPILING-Script das Herunterladen des Source Codes von Github (Jamulus, 2022b) und dem anschließenden selbstständigen Kompilieren der Clientsoftware. Dadurch kann garantiert werden, dass die Jamulus Clientsoftware für das bestehende System optimiert ist (in diesem Fall der Raspberr Pi 4) und ebenfalls alle benötigten Pakete zum Betrieb von Jamulus vorhanden sind. Gleichzeitig ermöglicht das selbstständige Kompilieren von Jamulus das automatische Updaten in der Zukunft, sofern dies zukünftig (nach Beendigung des Projekts) benötigt wird. Beim Herunterladen des Jamulus Codes und dem Compiling wird sich, wie bereits erwähnt, überwiegend an dem Jamulus-COMPILING-Script gehalten, sodass sich folgende Kommandos ergeben:

```
# Herunterladen des aktuellen Jamulus Source-Codes von Github
# (WICHTIG: In den Home-Ordner des Nutzers pi)
wget https://github.com/jamulussoftware/jamulus/archive/latest.tar.gz -P ~/

# Entpacken des Jamulus Codes
tar -xvf latest.tar.gz

# Öffne den Ordner mit den entpackten Code-Dateien
cd jamulus-latest

# Installation des Jamulus Standard Desktop Builds
make distclean
```

```
qmake
make
sudo make install
```

Nun ist Jamulus vollständig installiert. Die Installation dauert üblicherweise zwischen 10 und 15 Minuten. Anschließend ist Jamulus bereits vollständig nutzbar.
Um nun auf einen Jamulus-Server zu verbinden, wird zunächst das externe Audiointerface an den Raspberry Pi angeschlossen und anschließend wird JACK gestartet. Dies erfolgt über den folgenden Befehl im Terminal:

```
# Starte JACK Virtual Audio Interface mit vordefinierten Einstellungen
# (WICHTIG: Parameter-Werte sind Platzhalter- bzw. Beispielwerte!)
## -dalsa: Treiber (Driver: ALSA)
## -dhw: Soundkartenauswahl (Driver: Hardware; siehe "aplay -l")
## -r: Sample rate (default: 48000)
## -p: Frames per period (default: 1024)
## -n: Network latency setting in no. of periods (default: 5)

/usr/bin/jackd -R -dalsa -dhw:1 -r48000 -p128 -n2
```

Da JACK nun funktionsbereit ist, wird anschließend der Jamulus Client gestartet. Dies kann entweder über den Befehl ‘Jamulus‘ oder über die Benutzeroberfläche von Raspberry Pi OS geschehen. Soll nun eine automatisierte Verbindung mit einem bestimmten Server erfolgen, wird der Parameter ‘–connect <IP>(:<Port>)‘ an den Jamulus-Befehl angehängt. Dafür wird folgender Befehl in einem weiteren Terminal ausgeführt:

```
# Starte Jamulus ohne GUI und verbinde zu dem Server mit der IP + Port
Jamulus --connect <IP>(:<Port>)
```

Jetzt ist Jamulus gestartet und der Client ist automatisch dem angegebenen Server beigetreten.

4.3 Einrichtung von TeamViewer

Die Einrichtung von TeamViewer auf dem Raspberry Pi 4 erfolgt ebenfalls über das Terminal. Da es hier (wie auch bei Jamulus) kein Package gibt, dass mithilfe des Advanced Packaging Tool (apt) installiert werden kann, wird eine von TeamViewer bereitgestellte deb-Datei heruntergeladen und installiert.
In Abhängigkeit von der aktuell vorliegenden Bit-Version des Betriebssystems muss hierbei eine Fallunterschiedung durchgeführt werden, ob nun das 64-bit oder das 32-bit Paket heruntergeladen und installiert werden soll. Für die 64-bit TeamViewer Installation wird

```
# TeamViewer von der offiziellen Website herunterladen (64-bit)
wget https://download.teamviewer.com/download/linux/teamviewer_arm64.deb
   -O /home/pi/teamviewer.deb
```

im Terminal ausgeführt, während für die 32-bit Version von TeamViewer folgender Befehl ausgeführt wird:

```
# TeamViewer von der offiziellen Website herunterladen (32-bit)
wget https://download.teamviewer.com/download/linux/teamviewer_armhf.deb
   -O /home/pi/teamviewer.deb
```

Die Angabe des Parameters '-O' ermöglicht anzugeben, unter welchem Pfad und Dateinamen die heruntergeladenen Datei gespeichert werden soll. Einfachheitshalber wird die Datei stets als 'teamviewer.deb' im Home-Verzeichnis des Nutzers pi gespeichert.
Als nächstes, nachdem die benötigte Datei nun heruntergeladen ist, kann mit der Installation von TeamViewer begonnen werden. Dafür wird der Befehl

```
# Entpacken und Installieren des heruntergeladenen TeamViewer Pakets:
sudo dpkg -i /home/pi/teamviewer.deb
```

im Terminal ausgeführt. Dadurch wird TeamViewer auf dem Raspberry Pi installiert.
Zum Abschluss muss nun noch dafür gesorgt werden, dass die Installation von TeamViewer auch funktioniert. Beispielsweise könnte an dieser Stelle nun das Problem auftreten, dass TeamViewer zwar installiert ist, jedoch ein oder mehrere andere Pakete, die zwangsläufig zum korrekten Betrieb benötigt werden (engl. dependencies), fehlen. Dafür wird noch der Befehl

```
# Installation der (potentiell) fehlenden Dependencies für TeamViewer:
sudo apt install -f
```

Nun ist TeamViewer vollständig installiert und kann im Raspberry Pi verwendet werden. Wichtig ist an dieser Stelle zu erwähnen, dass (aufgrund der aktuellen Standard-Konfiguration von TeamViewer) zwingend ein Display / Monitor benötigt wird, da die auf dem Nutzerinterface angezeigte ID und das Passwort eingesehen und ausgetauscht werden muss.

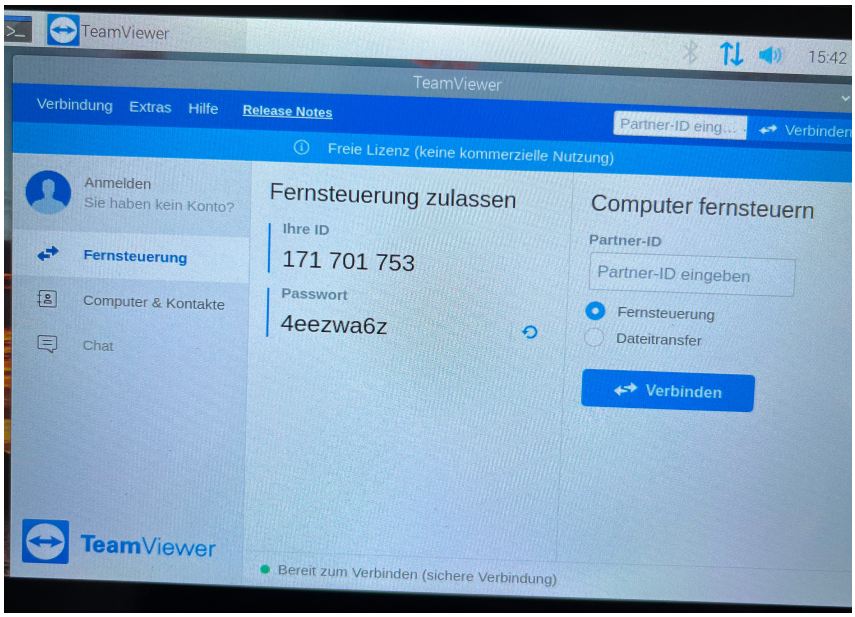


Abbildung 16: TeamViewer-Client auf dem Raspberry Pi 4 inkl. ID und Passwort.

4.4 Einrichtung eines Touchdisplays

In diesem Kapitel wird nun die Einrichtung sowie der Vergleich der zwei im Kapitel 2.6 beschriebenen Touchdisplays durchgeführt, um Jamulus sowie andere Anwendungen (wie z.B. TeamViewer) ohne Peripheriegeräte nutzen zu können. In Kapitel 4.4.1 wird zunächst die Konfiguration des Waveshare 7inch HDMI LCD (B) Display mit einer Auflösung von 800x480 beschrieben, während in Kapitel 4.4.2 die des Waveshare 7inch HDMI LCD (B) Display mit einer Auflösung von 1024x600 beschrieben wird.

4.4.1 Waveshare 7inch HDMI LCD (B)

Zu Beginn der Installation und Einrichtung des Waveshare 7inch HDMI LCD (B) wird zunächst (falls noch nicht geschehen) der Raspberry Pi vom Strom getrennt. Im Anschluss daran wird zunächst der Raspberry Pi per HDMI mit dem Waveshare 7inch HDMI LCD (B) verbunden. Im Anschluss daran wird dann noch der Raspberry Pi mit dem Waveshare 7inch HDMI LCD (B) über das mitgelieferte USB-Kabel verbunden, um die Kommunikation zwischen Raspberry Pi OS und dem Touchdisplay zu ermöglichen. Nun ist die hardwareseitig Installation des Displays bereits abgeschlossen.
Nun müssen noch Konfigurationen am Raspberry Pi durchgeführt werden, um die über das HDMI-Kabel übertragene Auflösung und Bildwiederholungsrate an das Touchdisplay anzupassen. Dafür wird entsprechend der Herstellerangaben die config.txt Datei, die sich auf der SD-Karte, in der sich die Boot-Dateien für den Raspberry Pi befinden, um folgende Werte ergänzt (bzw. die bestehenden Werte dahingehend abgeändert) (Waveshare, o.J.c):

```
hdmi_group=2
hdmi_mode=87
hdmi_cvt 800 480 60 6 0 0 0
hdmi_drive=1
```

Nun ist auch die softwareseitige Konfiguration des Touchdisplays abgeschlossen. Die SD-Karte kann nun wieder in den Raspberry Pi gesteckt werden. Danach wird der Raspberry Pi wieder mit dem Strom verbunden. Anschließend kann auf dem HDMI Display die Ausgabe des Raspberry Pis gesehen werden und das Display reagiert auf Berührungen.

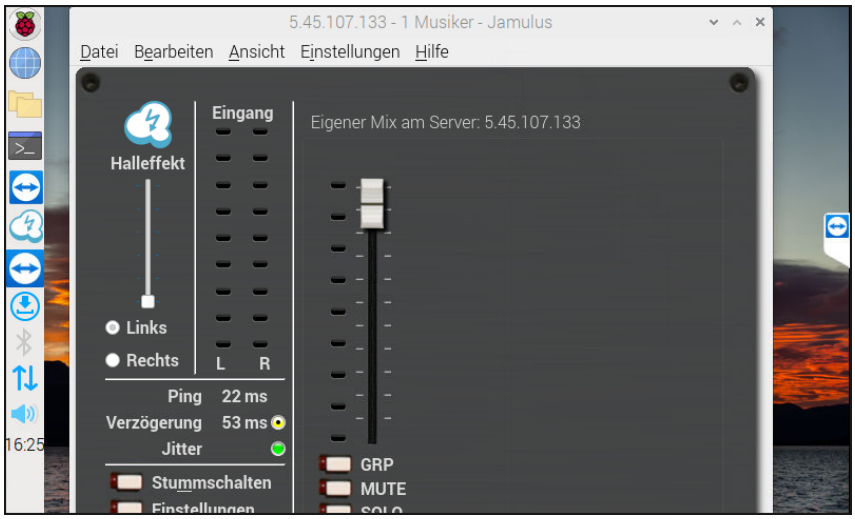


Abbildung 17: Jamulus auf dem Raspberry Pi 4 und Waveshare 7inch HDMI LCD (B).

Aus der Abbildung lässt sich nun ein Problem entnehmen, was die Verwendung dieses Touchdisplays für die Anwendung als Jamulus-Client erschwert bzw. unbrauchbar macht. Die Jamulus-Clientsoftware benötigt eine andere (bzw. höhere) Displayauflösung, um den gesamten Fensterinhalt darstellen zu können, als die verfügbare Auflösung von 800x480. Aus Kapitel 4.4.2 wird nun ein Touchdisplay mit einer Auflösung von 1024x600 herangezogen und auf Nutzbarkeit im Rahmen dieses Projekts untersucht.

4.4.2 Waveshare 7inch HDMI LCD (C)

Zu Beginn der Installation und Einrichtung des Waveshare 7inch HDMI LCD (C) wird, wie schon beim Waveshare 7inch HDMI LCD (B), zunächst der Raspberry Pi vom Strom getrennt. Im Anschluss daran wird der Raspberry Pi per HDMI mit dem Waveshare 7inch HDMI LCD (C) verbunden. Im Anschluss daran wird dann noch der Raspberry Pi mit dem Waveshare 7inch HDMI LCD (C) über das mitgelieferte USB-Kabel verbunden, um die Kommunikation zwischen Raspberry Pi OS und dem Touchdisplay zu ermöglichen. Nun ist die hardwareseitig Installation des Displays bereits abgeschlossen.
Nun müssen noch Konfigurationen am Raspberry Pi durchgeführt werden, um die über das HDMI-Kabel übertragene Auflösung und Bildwiederholungsrate an das Touchdisplay anzupassen. Dafür wird entsprechend der Herstellerangaben die config.txt Datei, die sich auf der SD-Karte, in der sich die Boot-Dateien für den Raspberry Pi befinden, um folgende Werte ergänzt (bzw. die bestehenden Werte dahingehend abgeändert) (Waveshare, o.J.d):

```
hdmi_group=2
hdmi_mode=87
hdmi_cvt 1024 600 60 6 0 0 0
hdmi_drive=1
```

Nun ist auch die softwareseitige Konfiguration des Touchdisplays abgeschlossen. Die SD-Karte kann nun wieder in den Raspberry Pi gesteckt werden. Danach wird der Raspber-

ry Pi wieder mit dem Strom verbunden. Anschließend kann auf dem HDMI Display die Ausgabe des Raspberry Pis gesehen werden und das Display reagiert auf Berührungen.

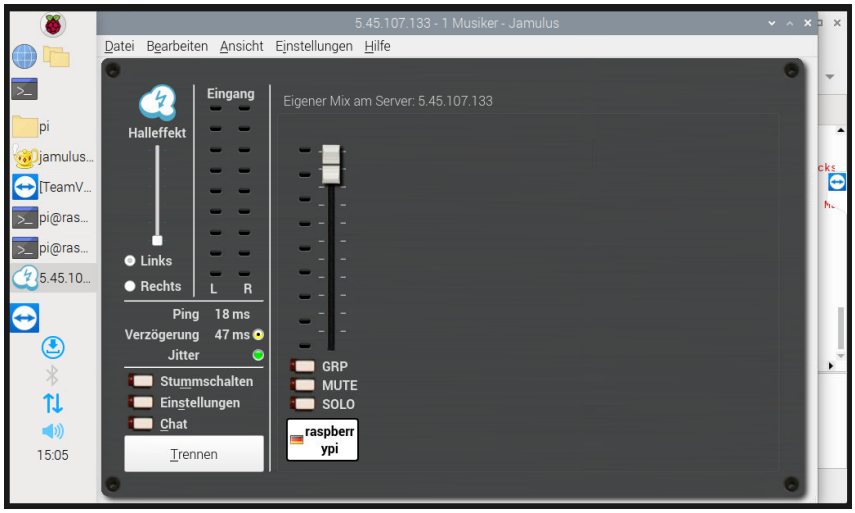


Abbildung 18: Jamulus auf dem Raspberry Pi 4 und Waveshare 7inch HDMI LCD (C).

Aus der Abbildung lässt sich nun entnehmen, dass Jamulus vollständig und korrekt angezeigt wird. Ebenfalls die Verwendung des Touchdisplays, um den Jamulus-Client, sowie die anderen Anwendungen zu steuern, funktionieren einwandfrei. Allerdings ist hierbei anzumerken, dass, um eine vollständige Darstellung des Jamulus-Clients zu ermöglichen, die vollen 600 Pixel in der Höhe benötigt werden. Somit ist es zwingend erforderlich, dass die Taskbar entweder auf die linke oder die rechte Seite verschoben wird. Zudem simuliert das Tippen auf das Touchdisplay lediglich die Aktion “Klicken mit einer Maus”. Tastatureingaben sind hiermit nicht ohne weiteres möglich. Für dieses Problem wird in Kapitel 4.5 die Einrichtung einer virtuellen Tastatur beschrieben.

4.5 Einrichtung einer virtuellen Tastatur

Die Verwendung des Touchdisplays als einziges Peripheriegerät sorgt für das Problem, dass keine Tastatureingaben ohne weiteres simuliert werden können, sondern lediglich Interaktionen, die üblicherweise per Maus durchgeführt werden würden, möglich sind. Aus diesem Grund wird in diesem Kapitel eine virtuelle Tastatur mithilfe des Pakets “matchbox-keyboard” eingerichtet, um nun Tastatureingaben per Touchscreen simulieren zu können.
Zu Beginn, falls noch nicht geschehen, werden die Paketbibliotheken sowie die bereits installierten Pakete aktualisiert.

```
sudo apt update
sudo apt upgrade
```

Im nächsten Schritt wird dann das Paket “matchbox-keyboard” installiert. Dieses Paket liefert die virtuelle Tastatur, die mit dem gleichnamigen Befehl über ein Terminal gestartet werden kann.

```
sudo apt install matchbox-keyboard
```

Nun ist bereits die virtuelle Tastatur installiert und voll funktionsfähig. Allerdings öffnet sich die Tastatur nicht automatisch, wenn ein Eingabefeld angewählt wird, sondern muss immer dann vom Nutzer manuell gestartet werden. Gleichzeitig kann es passieren, dass die Tastatur sich im Hintergrund eines anderen Programms befindet, sodass diese nicht mehr gesehen werden kann. Dafür muss die virtuelle Tastatur beendet werden, was ohne physische Tastatur nur sehr problematisch ist. Aus diesem Grund wird im folgenden ein Tastatur-Icon der Taskleiste hinzugefügt, mit dem man per Klick die virtuelle Tastatur aufrufen kann und mit einem erneuten Klick die Tastatur wieder beenden kann.
Zunächst wird ein Bash-Script erstellt, mit dessen Hilfe man bei Aufruf die virtuelle Tastatur starten kann, falls diese noch nicht gestartet ist. Ist diese allerdings bereits gestartet, wird der Prozess der virtuellen Tastatur beendet.

```
sudo nano /usr/bin/toggle-keyboard.sh
```

Dieses Bash-Script beinhaltet folgenden Code:

```
#!/bin/bash
PID="$(pidof matchbox-keyboard)"
if [  "$PID" != ""  ]; then
  kill $PID
else
  matchbox-keyboard &
fi
```

Danach, wenn die Datei erstellt ist und der Inhalt entsprechend gespeichert wurde, wird nun die benötigte Berechtigung erteilt, diese Datei auszuführen. Dafür wird folgender Befehl im Terminal ausgeführt.

```
sudo chmod +x /usr/bin/toggle-keyboard.sh
```

Nun ist, sofern die Datei “/usr/bin/toggle-keyboard.sh” aufgerufen wird, die gewünschte Funktionalität der Tastatur bereits gegeben. Nun wird mit der Einrichtung des Icons der Tastatur, das sich stets in der Taskleiste befinden soll, begonnen.
Als erstes wird die Datei “toggle-keyboard.desktop” erstellt, um den Betriebssystem mitzuteilen, dass das Tastaturicon stets angezeigt werden soll und dass das zuvor erstellte Script beim Antippen ausgeführt werden soll.

```
sudo nano /usr/share/raspi-ui-overrides/applications/toggle-keyboard.desktop
```

Als nächstes wird folgender Inhalt in die Datei geschrieben:

```
[Desktop Entry]
Name=Toggle Virtual Keyboard
Comment=Toggle Virtual Keyboard
```

```
Exec=/usr/bin/toggle-keyboard.sh
Type=Application
Icon=matchbox-keyboard.png
Categories=Panel;Utility;MB
X-MB-INPUT-MECHANISM=True
```

Somit ist die theoretische Einrichtung bereits dahingehend abgeschlossen. Jetzt müssen noch die nutzerspezifischen Taskleisteneinstellungen dahingehend geändert werden, dass sowohl das Tastaturicon zu sehen und zu nutzen ist, sowie die Tastatur entsprechend der Darstellung in Kapitel 4.4.2 auf der linken Seite des Desktops positioniert ist.
Dafür wird nun zunächst die Standardkonfiguration der Taskleiste in den nutzerspezifischen Konfigurationsordner kopiert, damit hier dann individuelle Einstellungen vorgenommen werden können.

```
cp /etc/xdg/lxpanel/LXDE-pi/panels/panel \
      /home/pi/.config/lxpanel/LXDE-pi/panels/panel
```

Als nächstes wird die kopierte Datei mit folgendem Befehl bearbeitet:

```
nano /home/pi/.config/lxpanel/LXDE-pi/panels/panel
```

Nun müssen in dieser Datei zwei Änderungen vorgenommen werden. Zum Einen wird nun die Zeile “edge” im Punkt “Global” zum Wert “left” geändert, um die Taskleiste auf der linken Seite anzeigen zu lassen:

```
Global {
  edge=left
  ...
```

Als nächstes wird nun am Ende der Datei folgender Code-Block eingefügt, um das Icon in der Taskleiste anzeigen zu lassen:

```
Plugin {
  type=launchbar
  Config {
    Button {
      id=toggle-keyboard.desktop
    }
  }
}
```

Nachdem diese Änderungen durchgeführt wurden und die Datei gespeichert wurde, wird nun folgender Befehl ausgeführt, um den Raspberry Pi neuzustarten:

```
sudo reboot
```

Nach dem Neustart ist die Einrichtung abgeschlossen und die virtuelle Tastatur kann über das Tastaturicon unten links in der folgenden Abbildung gestartet bzw. auch wieder beendet werden.

Abbildung 19: Startknopf der virtuellen Tastatur auf dem Raspberry Pi 4.

4.6 Einrichtung eines Kurzbefehls zum Start von Jack und Jamulus

Damit die Nutzer in der Lage sind, Jamulus (und Jack) ohne großen Aufwand starten und nutzen zu können, bedarf es, wie schon in Kapitel 4.5, einer Möglichkeit, per Tastendruck auf ein Icon in der Taskleiste sowohl Jack als auch Jamulus zu starten und den Client auf einen vordefinierten Server zu verbinden.
Zu Beginn, falls noch nicht geschehen, werden die Paketbibliotheken sowie die bereits installierten Pakete aktualisiert.

```
sudo apt update
sudo apt upgrade
```

Als nächstes wird dann ein Bash-Script erstellt, mit dessen Hilfe man bei Aufruf zunächst Jack (mit vordefinierten Parametern) sowie im Anschluss daran den Jamulus-Client, der dann auf eine vordefinierte IP-Adresse verbindet, falls beide Programme noch nicht gestartet sind. Sind sie allerdings bereits gestartet, wird zunächst der Jack-Prozess und im Anschluss daran der Jamulus-Prozess beendet.

```
sudo nano /usr/bin/toggle-jamulus.sh
```

Dieses Bash-Script beinhaltet folgenden Code:

```
#!/bin/bash
PIDJACK="$(pidof jackd)"
if [  "$PIDJACK" != ""  ]; then
  PIDJAMULUS="$(pidof Jamulus)"
  kill $PIDJAMULUS & kill $PIDJACK
else
```

```
  audioInterface=$(aplay -l | grep Audio | awk '{sub(":", "", $2); print $2}')
  jackd -R -dalsa -dhw:$audioInterface -r48000 -p128 -n2 & Jamulus --connect <IP>
fi
```

Danach, wenn die Datei erstellt ist und der Inhalt entsprechend gespeichert wurde, wird nun die benötigte Berechtigung erteilt, diese Datei auszuführen. Dafür wird folgender Befehl im Terminal ausgeführt.

```
sudo chmod +x /usr/bin/toggle-jamulus.sh
```

Nun ist, sofern die Datei "/usr/bin/toggle-jamulus.sh" aufgerufen wird, die gewünschte Funktionalität der Tastatur bereits gegeben. Nun wird mit der Einrichtung des Icons von Jamulus, das sich stets in der Taskleiste befinden soll, begonnen.
Als erstes wird die Datei "toggle-jamulus.desktop" erstellt, um den Betriebssystem mitzuteilen, dass das Jamulus-Icon stets angezeigt werden soll und dass das zuvor erstellte Script beim Antippen ausgeführt werden soll.

```
sudo nano /usr/share/raspi-ui-overrides/applications/toggle-jamulus.desktop
```

Als nächstes wird folgender Inhalt in die Datei geschrieben:

```
[Desktop Entry]
Name=Toggle Jamulus
Comment=Toggle Jamulus
Exec=/usr/bin/toggle-jamulus.sh
Type=Application
Icon=Jamulus.png
Categories=Panel;Utility;MB
X-MB-INPUT-MECHANISM=True
```

Somit ist die theoretische Einrichtung bereits dahingehend abgeschlossen. Jetzt müssen noch die nutzerspezifischen Taskleisteneinstellungen dahingehend geändert werden, dass das Jamulus-Icon zu sehen und zu nutzen ist.
Dafür wird wie bereits im Kapitel 4.5 die folgende Datei bearbeitet:

```
nano /home/pi/.config/lxpanel/LXDE-pi/panels/panel
```

Als nächstes wird nun am Ende der Datei folgender Code-Block eingefügt, um das Icon in der Taskleiste anzeigen zu lassen:

```
Plugin {
  type=launchbar
  Config {
    Button {
      id=toggle-jamulus.desktop
    }
  }
}
```

Nachdem diese Änderungen durchgeführt wurden und die Datei gespeichert wurde, wird nun folgender Befehl ausgeführt, um den Raspberry Pi neuzustarten:

```
sudo reboot
```

Nach dem Neustart ist die Einrichtung abgeschlossen und Jamulus (inklusive Jack) kann über das Icon unten links in der folgenden Abbildung gestartet bzw. auch wieder beendet werden.

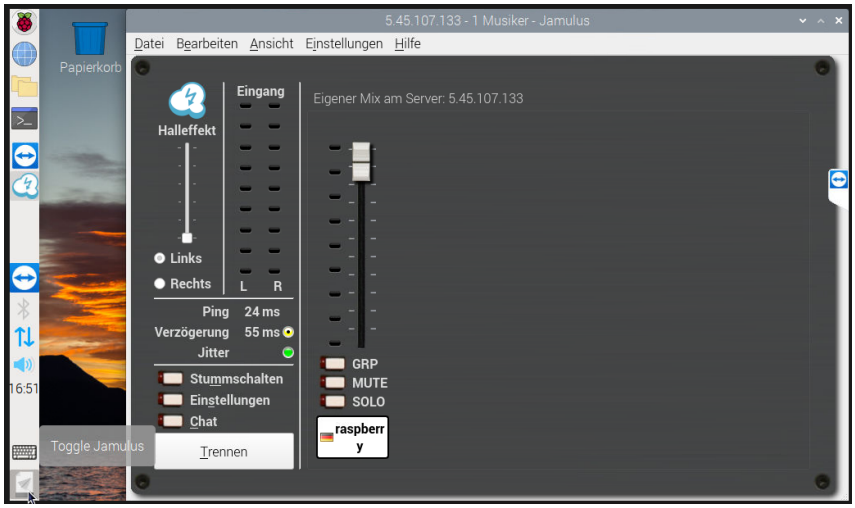


Abbildung 20: Jamulus-Client Startknopf auf dem Raspberry Pi 4.

5 Virtualisierung des Jamulus-Clients

Während am Anfang des Projektes nur auf Raspberry Pis gesetzt wurde, um auf diesen den Jamulus-Client zu installieren, wurde im Laufe des Projektes klar, dass auch nach alternativen Einplatinencomputern Ausschau gehalten werden muss, da aufgrund des Chip-Mangels zum einen die Verfügbarkeit der Raspberry Pis sehr gering ist und diese zum anderen dadurch auch sehr teuer geworden sind. Da es sich letztendlich bei den Alternativen um verschiedene Einplatinencomputer handelt, war die Idee die Jamulus-Client Software virtualisiert zu betreiben, um den Bereitstellungsprozess der Jamulus Software unabhängig von der Hardware zu machen, auf denen diese betrieben wird. Als Virtualisierungsart wurde hierbei auf die Containervirtualisierung zurückgegriffen, da dieser Ansatz der Virtualisierung leichtgewichtig ist, um die Latenz während der Verwendung des Jamulus-Clients möglichst gering zu halten. Nichtsdestotrotz war die Vermutung, dass die Latenz im Vergleich zum nativen Betrieb der Software deutlich höher ausfällt. Ein weiterer Grund, wieso die Idee der Virtualisierung aufkam, war ebenso der Aspekt der Sicherheit für den Endnutzer. Die Sicherheitsbedenken kamen dadurch auf, dass es erstmal nicht geplant ist, die Jamulus-Client-Software regelmäßig zu aktualisieren, wodurch Sicherheitslücken nicht geschlossen werden. Hier kommt die Virtualisierung ins Spiel, da bei der Virtualisierung die Client-Software in einem Bereich eingesperrt werden würde, in dem ein Angreifer ohne weiteres nicht an das gesamte System gelangt. Zu diesem Thema muss man jedoch anmerken, dass dieser Sicherheitsvorteil

mehr gewährleistet werden würde, wenn auf die Host-Virtualisierung zurück gegriffen worden wäre. Bei der Host-Virtualisierung wird ein kompletter Rechner virtualisiert, wohingegen bei der Containervirtualisierung auf Betriebssystemebene virtualisiert wird. Dies bedeutet, dass die verwendeten Container Teile des Betriebssystems, unter anderem den Kernel des Hosts, verwenden. Das bedeutet wiederum, dass eine Schwachstelle im Kernel des Betriebssystem im Container dazu führen kann, dass dies auf den Host überspringt und diesen zum Absturz bringen kann. Trotz dieses Problems wurde die Containervirtualisierung ausgewählt, da die Host-Virtualisierung durch das Virtualisieren eines kompletten Rechners viele Ressourcen erfordert, das sich wiederum auf die Latenz der Client-Software auswirken kann. Ein Vorteil hat die Containervirtualisierung jedoch trotzdem noch und zwar sorgt die Container Umgebung dafür, dass ein Absturz des Programms nicht ohne weiteres auf andere laufende Programme negativ überspringen kann, wodurch gewährleistet werden kann, dass zum Beispiel die Fernwartung über das Programm TeamViewer nicht beeinträchtigt wird. Als Containervirtualisierungssoftware wurde im Rahmen des Projekts Docker gewählt, da dieses das weitverbreitetste Programm in diesem Gebiet ist.
Docker muss zunächst auf der entsprechenden Hardware installiert werden. Hierzu kann das von Docker zur Verfügung gestellte Installationsskript genutzt werden (siehe Listing 20).

```
# Herunterladen des Skripts
curl -fsSL https://get.docker.com -o get-docker.sh

# Ausführen des Skripts
sudo sh get-docker.sh
```

Listing 20: Installation von Docker

Um die Jamulus-Client Software zu virtualisieren, wurde folgendes Dockerfile erstellt (siehe Listing 21).

```
FROM debian:latest

RUN \
 apt update

RUN \
   apt install -y build-essential \
   devscripts \
   debhelper \
   fakeroot \
   libjack-jackd2-dev \
   qtbase5-dev \
   qttools5-dev-tools \
   wget \
   tar
```

```
16
17 WORKDIR /tmp
18 RUN \
19    wget https://github.com/jamulussoftware/jamulus/archive/latest.tar.gz
20
21 RUN tar -zxf /tmp/latest.tar.gz
22
23 WORKDIR /tmp/jamulus-latest
24
25 RUN qmake
26 RUN make
27 RUN make install
28 RUN rm -rf /tmp/*
29
30 ENTRYPOINT ["Jamulus"]
```

Listing 21: Dockerfile für die Jamulus-Client Software

Das Dockerfile wird mit dem Befehl in Listing 22 gebaut und mit dem Befehl in Listing 23 in einem Container gestartet.

```
1 docker build -t jamulus:latest .
```

Listing 22: Bauen des Docker Images

```
1 docker run -d -it --net=host -e DISPLAY -v /tmp/.X11-unix -n jamulus
    jamulus:latest
```

Listing 23: Starten des Containers

Leider muss an dieser Stelle erwähnt werden, dass das Starten der Jamulus-Client Software in einem Docker Container nicht funktioniert hat. Hierbei ist das Problem, dass die grafische Oberfläche nicht an den Host weitergegeben werden kann und Jamulus das Öffnen einer grafischen Oberfläche voraussetzt, um starten zu können. Die Ursache hierfür ist nach einer Recherche scheinbar, dass Programme, dessen grafische Oberfläche auf der Bibliothek QT basieren nicht ohne komplexe Einstellungen in einem Docker Container gestartet werden können. Ein weiteres Problem bei der Thematik den Jamulus-Client in einem Docker Container laufen zu lassen, wäre zudem, dass dafür gesorgt werden müsste, dass neben der grafischen Oberfläche auch der Ton vom Host zur virtualisierten Anwendung weitergeben wird und andersherum. Dadurch wäre der Virtualisierungsansatz sehr komplex und auch sehr fehleranfällig, weshalb dieser Ansatz verworfen wurde.

6 Untersuchung der Verwendbarkeit der Raspberry Pis

Im diesem Kapitel wird die Auslastungen der Jamulus-Clients auf Basis des Raspberry Pi 3 mit 1GB RAM und des Raspberry Pi 4 mit 2 GB RAM untersucht und miteinander verglichen. Das Ziel soll es dabei sein, die günstigste, noch gut verfügbare Hardware, auf denen Jamulus sowie TeamViewer zur Fernwartung betrieben werden können, herauszuarbeiten.

6.1 Raspberry Pi 3 vs. Raspberry Pi 4

Der Raspberry Pi 4 ist die neueste Generation der Raspberry Pis, die Ende 2019 erschienen ist, und den Vorgänger, den Raspberry Pi 3, abgelöst hat (Speed, 2020). Der Raspberry Pi 4 existierte zu release in den Varianten mit 1GB, 2GB und 4GB RAM. Mitte 2020 wurde dann allerdings die 1GB Variante des Raspberry Pis für Endkunden eingestellt und wird seitdem ausschließlich an Unternehmenskunden vertrieben. Dafür wurde im selben Zeitraum der Raspberry Pi 4 mit 8GB RAM vorgestellt, mit dem ebenfalls eine neue Version des Betriebssystems Raspberry Pi OS als 64-bit Version veröffentlicht wurde (Mantel, 2020).
Daraus lässt sich nun ableiten, dass im Punkt der Verfügbarkeit der Raspberry Pi 4 dadurch, dass dieser aktuell noch produziert wird, im Vergleich zum Raspberry Pi 3 bevorzugt verwendet werden sollte. Dennoch wird in den folgenden Abschnitten untersucht, ob der Raspberry Pi 3 mit 1GB aus funktioneller Sicht dennoch für die Verwendung im Rahmen dieses Projekts geeignet wäre, da damit eine Untergrenze an RAM ermittelt werden kann, die für den Jamulus-Client benötigt wird.

6.2 Jamulus-Client auf Basis des Raspberry Pi 3 mit 1 GB RAM

Zunächst werden Raspberry Pi OS sowie Jamulus auf dem Raspberry Pi 3 mit 1 GB RAM nach der im Kapitel 4 beschriebenen Vorgehensweise installiert und konfiguriert. An dieser Stelle wird sich lediglich auf das Testen des Jamulus-Clients auf dem Raspberry Pi 3 konzentriert, sodass Kapitel 4.3 Einrichtung von TeamViewer an dieser Stelle keine Anwendung findet, sondern erst in Kapitel 6.3 verwendet und genauer untersucht wird. Nach der Installation und Konfiguration des Jamulus Clients auf dem Raspberry Pi 3 wird sich mit einem öffentlichen Server aus der Serverübersicht verbunden und es wird den dort verbundenen acht Musikern zugehört.

Abbildung 21: Auslastung des Raspberry Pi 3 - 1 GB RAM mit aktivem Jamulus-Client.

Die CPU-Auslastung des Raspberry Pis liegt bei der Untersuchung bei etwa 14%, während die RAM-Auslastung bei etwa 240 MB von rund 923 MB liegt. Aus dieser Beobachtung lässt sich nun die Erkenntnis ziehen, dass der Raspberry Pi 3 mit 1 GB durchaus in der Lage ist, an einer Jamulus-Session teilzunehmen, ohne dass der Raspberry Pi 3 bezüglich der CPU- oder der RAM-Auslastung an seine Grenzen kommt.

6.3 TeamViewer auf dem Raspberry Pi 3 mit 1 GB RAM

Zunächst werden Raspberry Pi OS sowie Jamulus auf dem Raspberry Pi 3 mit 1 GB RAM nach der im Kapitel 4 beschriebenen Vorgehensweise installiert und konfiguriert. An dieser Stelle wird sich lediglich auf das Testen des TeamViewer-Clients auf dem Raspberry Pi 3 konzentriert, sodass Kapitel 4.2 an dieser Stelle keine Anwendung findet.

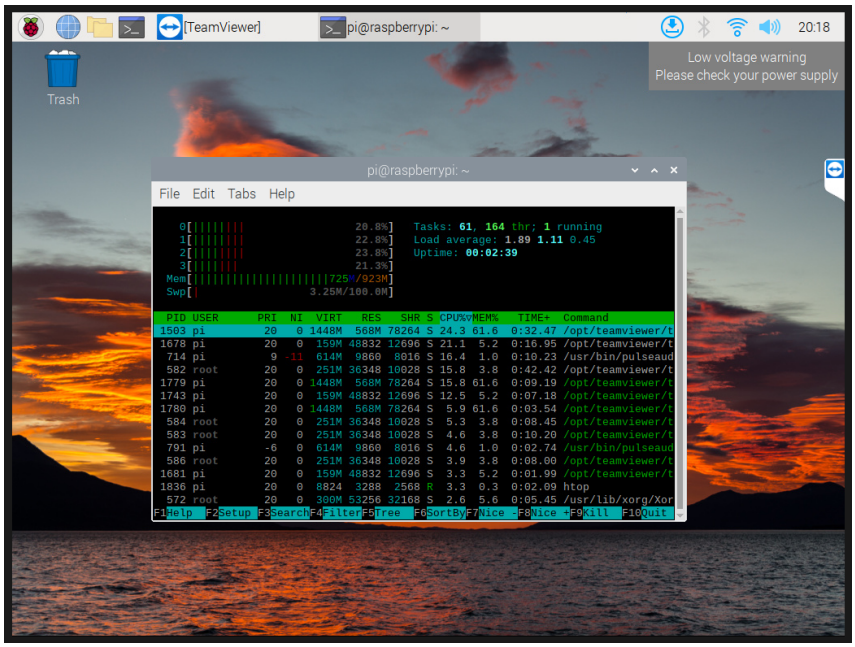


Abbildung 22: Auslastung des Raspberry Pi 3 mit aktiver TeamViewer-Verbindung.

Die CPU-Auslastung des Raspberry Pis liegt bei der Untersuchung bei etwa 20%, während die RAM-Auslastung bei etwa 725 MB von rund 923 MB liegt. Aus dieser Beobachtung lässt sich nun die Erkenntnis ziehen, dass der Raspberry Pi 3 mit 1 GB durchaus in der Lage ist, den TeamViewer-Client auszuführen, wobei hier allerdings gesagt werden muss, dass die 1GB RAM hier bereits an ihre Grenzen stoßen, würde beispielsweise wie in Kapitel 6.2 zusätzlich noch der Jamulus-Client gestartet werden.

6.4 Jamulus-Client auf Basis des Raspberry Pi 4 mit 2 GB RAM

Zunächst werden erneut Raspberry Pi OS sowie Jamulus auf dem Raspberry Pi 4 mit 2 GB RAM nach der im Kapitel 4 beschriebenen Vorgehensweise installiert und konfiguriert. Es werden also sowohl der Jamulus-Client als auch der TeamViewer-Client installiert und gestartet.
Nach der Installation und Konfiguration des Jamulus Clients auf dem Raspberry Pi 4 wird sich mit einem privaten Server verbunden, während gleichzeitig eine Verbindung über den TeamViewer-Client aktiv ist.

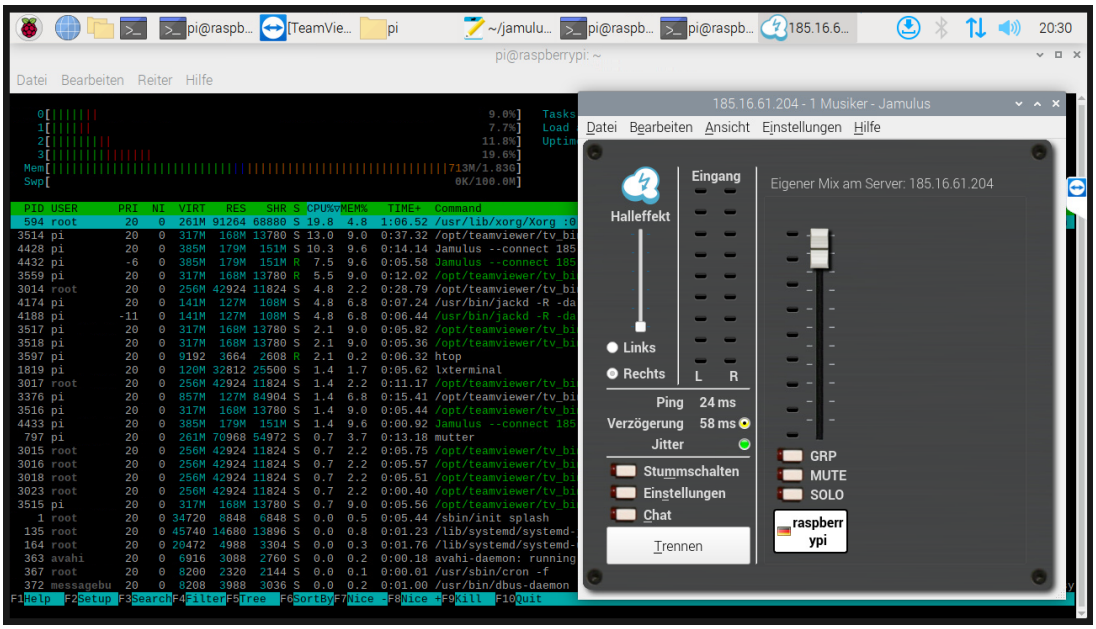


Abbildung 23: Auslastung des Raspberry Pi 4 - 2 GB RAM mit aktiver Jamulus- und TeamViewer-Verbindung.

Aus der Abbildung ist nun zu entnehmen, dass die CPU-Auslastung des Raspberry Pis bei der Untersuchung bei etwa 13% liegt, während die RAM-Auslastung bei etwa 710 MB von rund 923 MB liegt. Aus dieser Beobachtung lässt sich nun die Erkenntnis ziehen, dass der Raspberry Pi 3 mit 2 GB bezogen auf die Auslastung bzw. Leistung geeignet ist, die Aufgaben als Jamulus-Client mit TeamViewer als Fernwartung zu erfüllen.

6.5 Zusammenfassung und Wahl des Raspberry Pi - Modells

An dieser Stelle werden nun nocheinmal die Ergebnisse der Untersuchungen der vorherigen Kapitel zusammengefasst, sowie im Anschluss die Wahl des Raspberry Pi - Modell, dass für dieses Projekt verwendet wird, begründet. Die Kriterien der Wahl des Raspberry Pi-Modells sind dabei die Verfügbarkeit, sowie die Leistungs- bzw. die Auslastungswerte, die im Rahmen der Experimente ermittelt wurden.
Der Raspberry Pi 3 (in diesem Fall mit 1 GB RAM) ist aufgrund der Ablösung durch das Nachfolgemodell (der Raspberry Pi 4) nicht mehr, oder nur mit großem Aufwand, zu erwerben, da diese nicht mehr produziert werden. Zudem ist eine 1GB Variante der Raspberry Pis (unabhängig von der Version) nicht empfehlenswert, da Jamulus in Kombination mit TeamViewer stets eine RAM-Auslastung von über 700MB aufweisen, sodass der Raspberry Pi da an seine Grenzen kommt, was sich negativ auf das Nutzererlebnis auswirken kann. Bei der CPU-Auslastung gibt es weder beim Raspberry Pi 3, noch beim Raspberry Pi 4, nennenswert hohe Werte, sodass hier beide Versionen gut geeignet sind.

Der Raspberry Pi 4 mit 2GB ist für dieses Projekt die geeignete Hardware, um sowohl den Jamulus-Client als auch den TeamViewer-Client zu verwenden. Der Raspberry Pi 4 wird weiterhin produziert und bietet in der 2GB RAM Variante genug Arbeitsspeicher

für die benötigten Anwendungen. Alternativ kann sich für eine Raspberry Pi 4 Variante mit mehr RAM entschieden werden, falls diese bereits vorhanden bzw. zugänglich sind. Im Rahmen dieses Projekts wird sich allerdings nun explizit für den Raspberry Pi 4 mit 2GB RAM entschieden.
Der Vollständigkeit halber werden nun allerdings im Kapitel 7 noch alternative Einplatinencomputer für die Verwendung als Jamulus-Client herangezogen und genauer untersucht. Der Grund hierfür liegt bei während dieses Projekts vorherrschenden Problemen in der Verfügbarkeit und den hohen Kosten, von denen unter anderem der Raspberry Pi betroffen ist.

7 Alternative Hardware für den Jamulus-Client

In diesem Kapitel werden zwei alternative Einplatinencomputer untersucht, die als Alternative zum Raspberry Pi zur Anwendung kommen können. Der Grund dafür, dass nach Alternativen für die Verwendung des Raspberry Pis gesucht wird, ist die "Chip-Krise" in 2022, die sich auch auf die für dieses Projekt benötigten Einplatinencomputer auswirkt und deshalb insbesondere die Raspberry Pi Modelle kaum bzw. nur zu Preisen weit jenseits der Hersteller-UVP erhältlich sind (Costa, o.J.). Im Folgenden sollen zwei alternative Einplatinencomputer näher untersucht werden, deren Spezifikationen denen der Raspberry Pi Modelle ähnlich sind. In Kapitel 7.1 wird ODROID XU4Q als mögliche Alternative diskutiert, während in Kapitel 7.2 der PINE A64-LTS als mögliche Alternative untersucht wird.

7.1 ODROID XU4Q

Die offizielle Webseite des Herstellers HardKernel empfiehlt für die Nutzung das Betriebssystem Ubuntu, für das der Hersteller auf der Seite eigene, offiziell unterstützte Images zur Verfügung stellt (odroid, o.J.). Dieses Image wird dann auf eine SD-Karte übertragen und der ODROID XU4Q wird gestartet.
Als nächstes wird hier, wie in Kapitel 4.2 der Jamulus-Client installiert und gestartet. Im Anschluss daran sollen die in Kapitel 2.6 beschriebenen Touchdisplays angeschlossen und konfiguriert werden, sodass die Nutzung ohne weitere Peripheriegeräte möglich ist. Dabei ist allerdings das Problem aufgetreten, dass weder das angezeigte Bild auf dem Display korrekt dargestellt wird, noch dass die Touchfunktionalität nutzbar ist. Der Cursor reagiert mit "sprüngen" in die verschiedenen Ecken des Displays, nicht allerdings dort hin, wo der Nutzer auf das Display tippt. Eine Recherche auf der Webseite des Herstellers (Waveshare, o.J.b) bzw. der Dokumentationsseite des Herstellers (Waveshare, o.J.d) legt nahe, dass die Displays ausschließlich für den Einplatinencomputer Raspberry Pi ausgelegt und nur mit diesen kompatibel sind. Andere Nutzer, wie beispielsweise szafran (2017), berichten von ähnlichen Problem.

7.2 PINE A64-LTS

Die offizielle Webseite des Herstellers PINE64 empfiehlt für die Nutzung der unterschiedlichen PINE64-Modelle das Betriebssystem Armbian (PINE64, o.J.a), das eine leichtgewichtige, einfache, optimierte und schnelle Anpassung des Debian-Betriebssystems auf die ARM-Prozessoren der in diesem Projekt untersuchten Einplatinencomputer bietet (armbian, o.J.).
Armbian wird in zwei Versionen (anhängig vom Anwendungsfall) zum Download angeboten: der Desktop-Version für die Anwendung als Desktopcomputer und die CLI-Version als Anwendung als Server (armbian, o.J.). Die Desktop-Version von Armbian enthält eine große Menge an Programmen, die der Verwendung als Desktopcomputer dienen soll. Im Rahmen dieses Projekts wird sich nun allerdings dafür entschieden, auf die CLI-Version als Betriebssystem zu setzen, da das Ziel ist, möglichst wenig Software auf dem Betriebssystem zu haben, um den Overhead zu reduzieren, der dadurch potentiell entstehen kann. Da der Jamulus-Client per Dokumentation allerdings eine Desktopoberfläche voraussetzt, muss diese in der CLI-Version mit den folgenden Befehlen noch zusätzlich installiert werden.
Zunächst werden alle Paketbibliotheken sowie die bereits installierten Pakete aktualisiert. Dafür werden die folgenden Befehle verwendet:

```
sudo apt update
sudo apt upgrade
```

Als nächstes wird dann die Desktop-Software XFCE installiert. Dafür wird der folgende Befehl verwendet:

```
sudo apt install task-xfce-desktop
```

Während der Konfiguration der zu installierenden Pakete hat man die Wahl, entweder den “GNOME display manager (gdm)” oder den “Light Display Manager (LightDM)” zu installieren. Für den Verlauf in diesem Projekt wird sich dafür entschieden, den “Light Display Manager (LightDM)” zu verwenden, wobei beide ähnlich gut für dieses Projekt funktionieren sollten.
Als nächstes muss nun geprüft werden, ob die Desktop-Software XFCE nun auch standardmäßig beim Systemstart ausgeführt wird. Dafür wird der folgende Befehl ausgeführt. Das Ergebnis soll, sofern alles funktioniert hat, dann “graphical.target” sein.

```
sudo systemctl get-default
```

Sollte die Ausgabe des Befehls nun nicht “graphical.target” entsprechen, so ist folgender Befehl auszuführen:

```
sudo systemctl set-default graphical.target
```

Nun ist die Desktop-Software XFCE vollständig installiert und konfiguriert. Damit nun der Desktop verwendet werden kann, muss der PINE A64-LTS mit folgendem Befehl neugestartet werden:

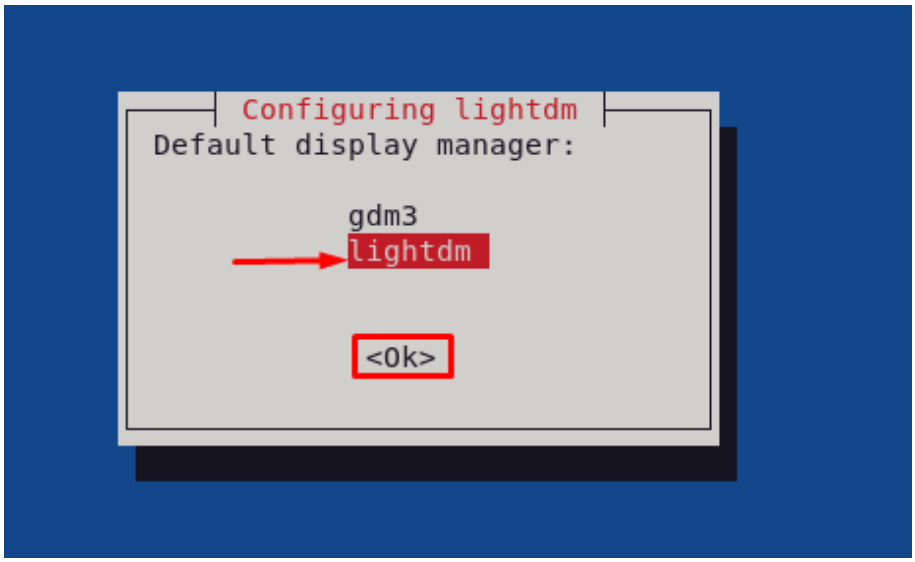


Abbildung 24: Auswahl des Display-Managers bei der XFCE-Installation.

```
sudo reboot
```

Als nächstes wird, wie in Kapitel 4.2 beschrieben, der Jamulus-Client installiert und gestartet. Im Anschluss daran sollen die in Kapitel 2.6 beschriebenen Touchdisplays angeschlossen und konfiguriert werden, sodass die Nutzung ohne weitere Peripheriegeräte möglich ist. Dabei ist allerdings, wie schon beim ODROID XU4Q in Kapitel 7.1, das Problem aufgetreten, dass weder das angezeigte Bild auf dem Display korrekt dargestellt wird, noch dass die Touchfunktionalität nutzbar ist. Der Cursor reagiert mit “sprüngen” in die verschiedenen Ecken des Displays, nicht allerdings dort hin, wo der Nutzer auf das Display tippt. Eine Recherche auf der Webseite des Herstellers (Waveshare, o.J.b) bzw. der Dokumentationsseite des Herstellers (Waveshare, o.J.d) legt nahe, dass die Displays ausschließlich für den Einplatinencomputer Raspberry Pi ausgelegt und nur mit diesen kompatibel sind. Andere Nutzer, wie beispielsweise iPhyse (2016), berichten von ähnlichen Problem.

8 Audio-Interfaces

Da die verwendeten Einplatinencomputer entweder gar keine oder für die Anwendungen der Jamulus-Clients schlecht geeignete Audioanschlüsse besitzen, war schnell klar, dass separate Audio-Interfaces zur Ein- und Ausgabe der übertragenen Tonsignale notwendig sein würden. Die Mindestanforderungen an Audio-Interfaces waren dabei, die Möglichkeit, ein Mikrofon zur Toneingabe und Kopfhörer zur Tonausgabe anschließen zu können sowie eine möglichst geringe Latenz bei der Übertragung zwischen Interface und Computer. (Mehr zum Thema Latenz in Kapitel 9 „Messungen“.) Zunächst kamen drei verschiedene Ansätze in Betracht, welche nachfolgend jeweils kurz erläutert werden sollen:

8.1 Aufsteckplatinen

Wie Herr Borucki in einer Vorabrecherche herausgefunden hatte, gibt es Audio-Interfaces in Form von Erweiterungsplatinen, die einfach auf den Einplatinencomputer aufgesteckt werden können. Der klare Vorteil einer solchen Lösung besteht in ihrer Kompaktheit, welche ein hohes Maß an Integration für die Jamulus-Clients erlauben würde. Bei weitergehenden Recherchen stellten sich dann allerdings zahlreiche Probleme mit diesen Platinen heraus. Zum einem sind alle gefundenen Lösungen ausschließlich auf die Verwendung am GPIO-Header der Raspberry Pi Modelle ausgelegt (siehe Namen: „PiSound“ (Vilniaus Blokas UAB, o.J.), „HIFIBERRY“ (Modul 9 GmbH, 2020), „Raspiaudio Ultra+“ (Sertronics GmbH, o.J.b)), was die Auswahl der Einplatinencomputer erheblich eingeschränkt hätte. Zum anderen gibt es kein Modul mit den für unsere Zwecke benötigten Anschlüssen. Zwar verfügen fast alle Modelle über Digital-Analog-Wandler und Analog-Digital-Wandler, wobei letzteres bei manchen Modellen bereits fehlt (z.B. „IQaudio DAC+“ (Sertronics GmbH, o.J.a)), doch sind die Audio-Eingänge in allen Fällen auf unsymmetrische Signale mit Line-Pegel ausgelegt, was für die Verwendung mit einem Mikrofon weitere Hardware in Form eines Mikrofonvorverstärkers notwendig machen würde. Auch verfügen manche Modelle anstelle eines Kopfhörerausgangs lediglich über Ausgänge mit Line-Pegel, was wiederum einen externen Kopfhörerverstärker voraussetzen würde. Insgesamt wäre also zusätzliche Hardware nötig, welche mangels Verfügbarkeit entsprechender Module in kompakter Bauform entweder den Vorteil der Kompaktheit zunichtemachen oder entsprechende Eigenentwicklungen voraussetzen würde. Hinzu kommt, dass auch hier zum Zeitpunkt der Recherche (April 2022) enorme Probleme bezüglich der Verfügbarkeit festzustellen waren und manche Modelle durch eine sehr dürftige Dokumentation und den Anschein einer eher fragwürdigen Qualität auffielen.

8.2 Externe USB-Interfaces

Eine andere Möglichkeit, die verwendeten Einplatinencomputer um Audio-Eingänge und -Ausgänge ergänzen zu können, stellen weit verbreitete, externe Audio-Interfaces mit USB-Schnittstelle dar. Aufgrund der weiten Verbreitung der USB-Schnittstelle an PCs haben sich Audio-Interfaces mit USB-Anbindung in den letzten 10 Jahren (zumindest im nicht-professionellen Bereich) als Standard etabliert, wenn es um die Aufnahme von Audio geht. Dementsprechend gibt es ein enormes Angebot verschiedener Modelle von diversen Herstellern, die sich vor allem in der Anzahl der Ein- und Ausgänge, der Qualität (Audio- und Fertigungsqualität, Treiberstabilität etc.) und dem Preis unterscheiden. Diese große Auswahl verfügbarer Modelle, die nahezu alle mit Linux-Betriebssystemen kompatibel sind, stellt den großen Vorteil von USB-Audio-Interfaces dar. So lässt sich einerseits immer ein zu den Anforderungen des jeweiligen Musikers passendes Interface finden und andererseits kann theoretisch auch auf bei vielen Musikern bereits vorhandene Geräte zurückgegriffen werden. Für unsere Zwecke wurden mehrere preisgünstige Modelle des Herstellers Behringer (UM2, UMC202, UMC404) getestet, die sich lediglich in der Anzahl der Ein- und Ausgänge unterscheiden. (Music Tribe Global Brands Ltd, o.J.)

8.3 USB-Mikrofone

Eine Sonderform der USB-Audio-Interfaces stellen USB-Mikrofone dar. Hierbei handelt es sich um ein Mikrofon mit integriertem Audio-Interface, das neben dem Mikrofon als Eingangskanal häufig über einen Kopfhöreranschluss zur Ausgabe verfügt. Zwar mindert das Fehlen von zusätzlichen Eingängen gegenüber „normalen“ USB-Interfaces mit externen Mikrofonen die Flexibilität, doch reduziert sich damit auch der Verkabelungsaufwand enorm, sodass nur ein USB-Kabel eingesteckt werden muss. Dadurch stellen USB-Mikrofone eine sehr nutzerfreundliche Lösung dar. Beachtet werden muss allerdings, dass viele Modelle nur mittels Software, bedienbar sind was die Verwendung für die Jamulus-Clients ausschließt. Ein Modell, das mit einem Drehregler für die Eingangsempfindlichkeit und einem Drehregler für die Kopfhörerlautstärke sehr einfach und direkt am Mikrofon bedient werden kann, ist das Mackie EM-USB, welches wir in unsere Tests einbezogen haben. (LOUD Audio, LLC, o.J.)

9 Messungen

Schon zu Beginn des Projektes war klar, dass es nötig sein würde, Methoden zur quantitativen Bewertung einzelner Faktoren bzw. deren Auswirkungen zu finden. Gemessen werden sollten dabei vor allem die Latenz der Audioübertragung, das Auftreten von Übertragungsfehlern und der Frequenzgang der Jamulus-Verbindung.
Wie die jeweiligen Messmethoden gefunden und teilweise schaltungstechnisch so umgesetzt wurden, dass Messungen einfach und unter reproduzierbaren Bedingungen möglich sind, soll in diesem Kapitel näher erläutert werden. Außerdem soll, soweit möglich, versucht werden, die beobachteten Einflussfaktoren zu erklären.

9.1 Frequenzgang (und Grundrauschen)

Der Frequenzgang der Audioübertragung mit Jamulus wurde versucht, mittels der kostenlosen Software Room EQ Wizard (REW) zu messen. (Mulcahy, 2022)
Diese Software war zwar ursprünglich zur Einmessung von Lautsprechern in Räumen gedacht (daher der Name), wird inzwischen aber häufig in der DIY-Community, also von Hobby-Bastlern, in Verbindung mit einem Audio-Interface mit guten Ergebnissen als günstiger Ersatz für einen Audio-Analyzer genutzt. (DIY Recording Equipment, 2020)
Zur Aufnahme des Frequenzgangs von Jamulus wird das Ausgangssignal von REW, ein Frequenzsweep, auf Jamulus gegeben und das zurückkommende Signal wieder in REW geroutet. Da jeder noch so kurze Übertragungsfehler während des Sweeps die Messung unbrauchbar macht, ist es unter Umständen erforderlich, die Messung einige Male zu wiederholen.
Der sich ergebende Graph im Reiter *All SPL* von REW zeigt nun den Amplitudengang von Jamulus im Bereich von 20 Hz bis 20 kHz (siehe Abbildung 25).

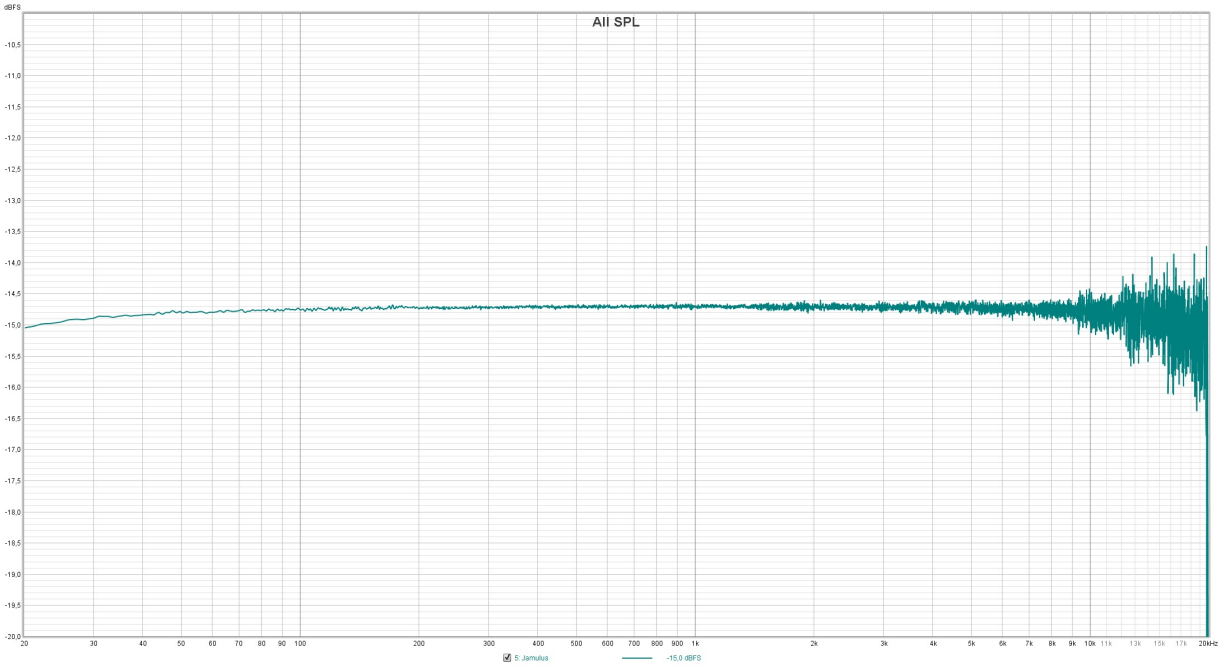


Abbildung 25: Mit REW gemessener Amplitudengang

Auf den ersten Blick zeigt sich ein unterhalb von 1 kHz sehr konstanter Frequenzgang, der lediglich unterhalb von 100 Hz geringfügig abfällt (etwa -0,3 dB) und oberhalb von 1 kHz mit zunehmender Frequenz immer stärker vom Ideal abweicht.

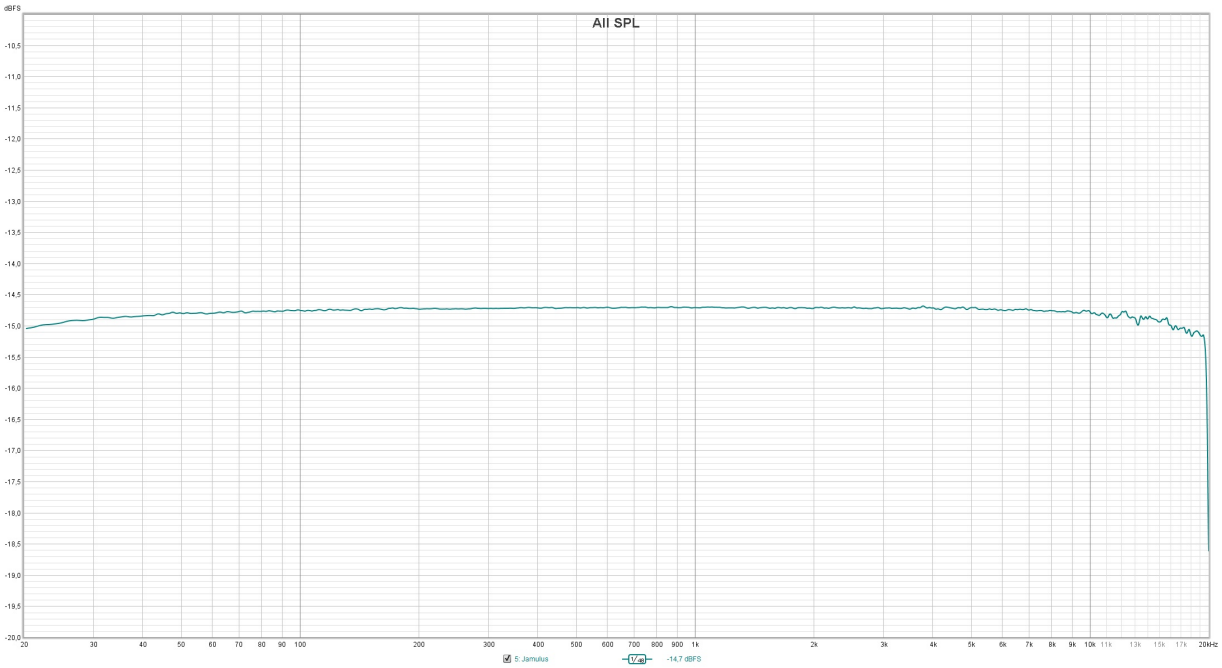


Abbildung 26: Mit REW gemessener Amplitudengang – geglättet (1/48 Oktave)

Wählt man in REW allerdings eine sehr feine Glättung (1/48 Oktave), so zeigt der Graph des Amplitudengangs auch im Bereich von 1 kHz bis 10 kHz einen nahezu linearen Verlauf und fällt zwischen 10 kHz und 19 kHz leicht ab (etwa -0,5 dB), bevor er oberhalb von 19 kHz rapide abfällt.

Während sich der starke Abfall der Amplitude oberhalb von 19 kHz recht einfach durch die Anti-Aliasing-Filter der Analog-Digital-Wandler und Digital-Analog-Wandler des verwendeten Audio-Interfaces erklären lässt, lassen sich für die starken Ausschläge oberhalb von 10 kHz im nicht geglätteten Graphen nur verschiedene Erklärungsansätze finden: Eine Möglichkeit wäre, dass die Ausschläge durch hochfrequentes Rauschen hervorgerufen werden.
Dies würde auch zu Beobachtungen des Ausgangssignals von Jamulus am Oszilloskop bei gleichzeitiger Einspeisung eines 1 kHz Sinus als Testsignal passen, wobei immer eine Überlagerung des Ausgangssignals mit mehr oder weniger starkem Rauschen auffiel. Die Ursache des Rauschens konnte bislang jedoch nicht geklärt werden.
Festhalten lässt sich aber, dass die Audio-Übertragung via Jamulus einen relativ geradlinigen Amplitudengang aufweist und die Audio-Qualität zumindest zu Probezwecken ausreichen sollte.

9.2 Latenz

Eine weitere wichtige Kenngröße der Übertragung ist die Latenz, also die Zeitverzögerung, die zwischen eingespeistem Signal und zurückkommendem Signal auftritt.
Im Gegensatz zu anderen Anwendungen, die vorrangig für Videokonferenzen ausgelegt sind (Zoom, Webex, Sype, ...), zeichnet sich Jamulus dadurch aus, dass die Latenz möglichst minimal gehalten wird (siehe Kapitel 2.4).
Zur maximal akzeptablen Latenz gibt es zwei verschiedene Sichtweisen:
Erstens die Faustregel aus der Musikindustrie, der zufolge eine Roundtrip-Latenz (Umlaufzeit), also die die Latenz zwischen Eingang und Ausgang des Audio-Interfaces, von etwa 10 ms angepeilt werden sollte, weil es sonst vor allem bei Schlaginstrumenten zu einer für den Musiker spürbaren Verzögerung zwischen (zum Beispiel) dem Aufschlag eines Trommelstocks und dem entsprechenden Klang im Kopfhörer käme, was es schwierig machen könnte, im Takt zu bleiben. (PreSonus Audio Electronics, Inc, o.J.)
Zweitens das Beispiel eines Sinfonieorchesters, dessen Musiker durchaus bis zu 15 m weit voneinander entfernt sein können, was bei einer Schallgeschwindigkeit von etwa 343 m/s einer Verzögerung von knapp 44 ms entspricht.
Der entscheidende Unterschied der beiden Sichtweisen besteht darin, dass die Mitglieder des Sinfonieorchesters sich selbst ohne Verzögerung hören und sie lediglich der Schall der Mitmusiker mit zunehmender Entfernung immer verzögerter erreicht. Daher widersprechen sich diese beiden Sichtweisen auch nicht.
Um jedoch das eigene Instrument oder den eigenen Gesang synchron zu den Mitmusikern hören und möglichst im Takt musizieren zu können, wird auf der Jamulus-Website empfohlen, geschlossene Kopfhörer zu tragen und sich selbst auch über Jamulus, also im Mix mit den anderen Musikern, zu hören, was wiederum eher dem ersten Fall entspricht. Daher sollte die Latenz so weit wie möglich reduziert werden. (Fischer, o.J.b)

9.2.1 Erste Messversuche

Als erste Messung sollte die integrierte Latenzanzeige der Jamulus-Software überprüft werden. Als Messaufbau kam dazu eine Kombination aus Oszilloskop und Signalgenerator zum Einsatz.
Dazu wird mittels Signalgenerator ein Burst-Signal (einige Sinusschwingungen gefolgt von einer Pause) in das verwendete Audio-Interface eingespeist und sowohl Eingangs- als auch Ausgangssignal werden auf einem zweikanaligen Oszilloskop dargestellt, so lässt sich leicht die Verzögerung zwischen den beiden Signalen messen (siehe Abbildung 27).

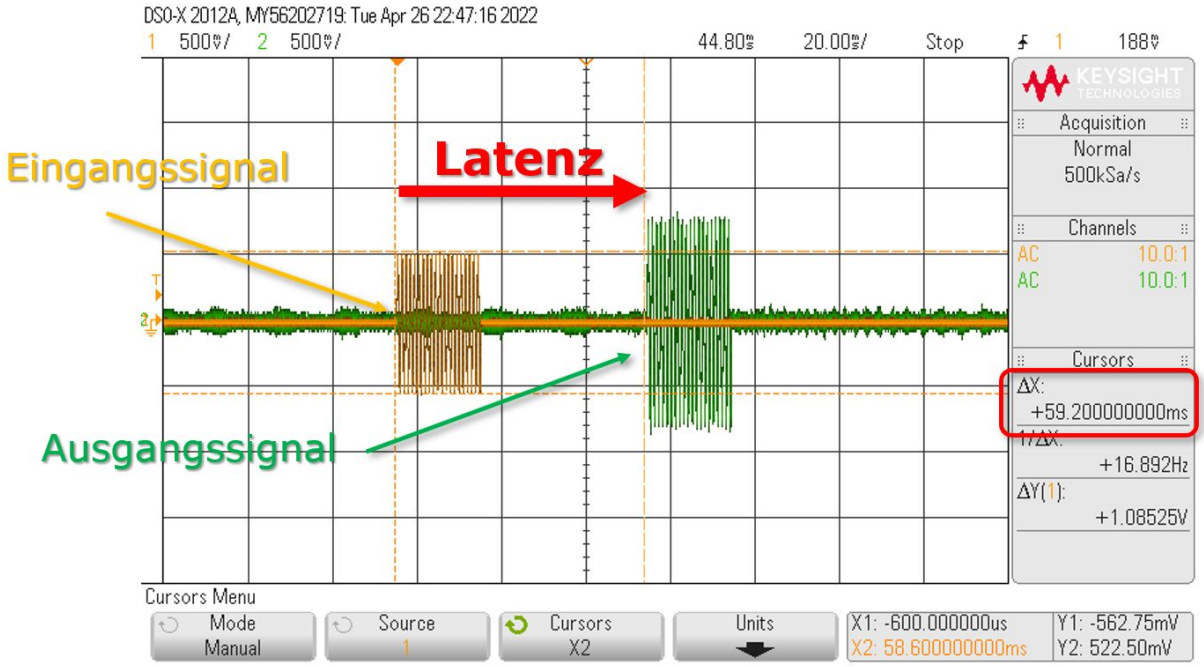


Abbildung 27: Oszilloskop-Screenshot - Latenzmessung mit Burst-Signal

Mittels dieser Messmethode konnten die in Jamulus angezeigten Werte verifiziert werden, was bedeutet, dass für weitere Versuche zur Verringerung der Latenz auf spezielle Versuchsaufbauten verzichtet werden kann.

9.2.2 Einflüsse und Erklärungsansätze

In Jamulus gibt es gleich mehrere Faktoren, die die Latenz beeinflussen:
Ein gut berechenbarer Faktor ist die Puffergröße (Buffer Size) des Audio-Interfaces, welche meist als Zweierpotenz in Samples eingestellt wird. Wird dieser Puffer zu klein gewählt, kann es bei der USB-Übertragung zu Aussetzern kommen, die sich als deutlich hörbare *Knackser* bemerkbar machen. Eine zu hohe Puffergröße vergrößert dagegen ohne jeglichen Nutzen die Latenz. Mit welcher minimalen Puffergröße ein Audio-Interface ohne Aussetzer funktioniert, hängt maßgeblich von den verwendeten Treibern und dem Betriebssystem ab. Für die hierdurch verursachte Latenz gilt:

$$Latenz[ms] = 2 \cdot Puffergrösse[Samples] \cdot 1/Samplerate[Samples/s]$$

(Der Faktor 2 kommt zustande, weil ein- und ausgehendes Signal den Puffer durchlaufen.)
Bei einer üblichen Puffergröße von 128 Samples und einer Samplerate von 48 kHz entspricht dies einer Latenz von etwa 5,3 ms.
Ein weiterer leicht nachvollziehbarer Faktor ist der Ping, also die Laufzeit zwischen Jamulus-Client und Jamulus-Server. Dieser Wert wird beim Verbinden eines Servers in Jamulus angezeigt.
Welche Auswirkungen der ebenfalls in Jamulus einstellbare Netzwerk-Puffer und die serverseitige Signalverarbeitung haben, lässt sich nicht so leicht abschätzen, weshalb zusätzliche Messungen und Versuche durchgeführt werden mussten. Dass die beiden Werte in Summe einen nicht zu verachtenden Teil der Latenz ausmachen zeigt folgende beispielhafte Rechnung anhand von Werten aus einem konkreten Test:

$$Gesamtlatenz = Interfacepuffer + Ping + Netzwerkpuffer + Signalverarbeitung$$

$$\Rightarrow 57ms - 4ms - 30ms = 23ms = Netzwerkpuffer + Signalverarbeitung$$

9.3 Übertragungsfehler

Ein Problem, das von Jamulus-Nutzern als extrem störend beschrieben wurde, stellen Übertragungsfehler dar, welche es daher ebenfalls zu quantifizieren galt, um sie dann möglichst weit reduzieren zu können.
Hier wurde für einen ersten Test ein 1 kHz Sinussignal wahlweise mittels zweitem Audio-Interface oder Signalgenerator in Jamulus eingespeist und das Ausgangssignal mittels Oszilloskop (idealerweise im Modus *Rollen*) beobachtet und gleichzeitig über Lautsprecher oder Kopfhörer angehört.
Dabei konnten gleich mehrere Arten von Übertragungsfehlern festgestellt werden: In scheinbar zufälligen Abständen auftretende, hörbare *Knackser*, welche sich auf dem Oszilloskop als Phasensprünge, Peaks oder Kombination aus beidem darstellen (siehe Abbildung 28 und 29).

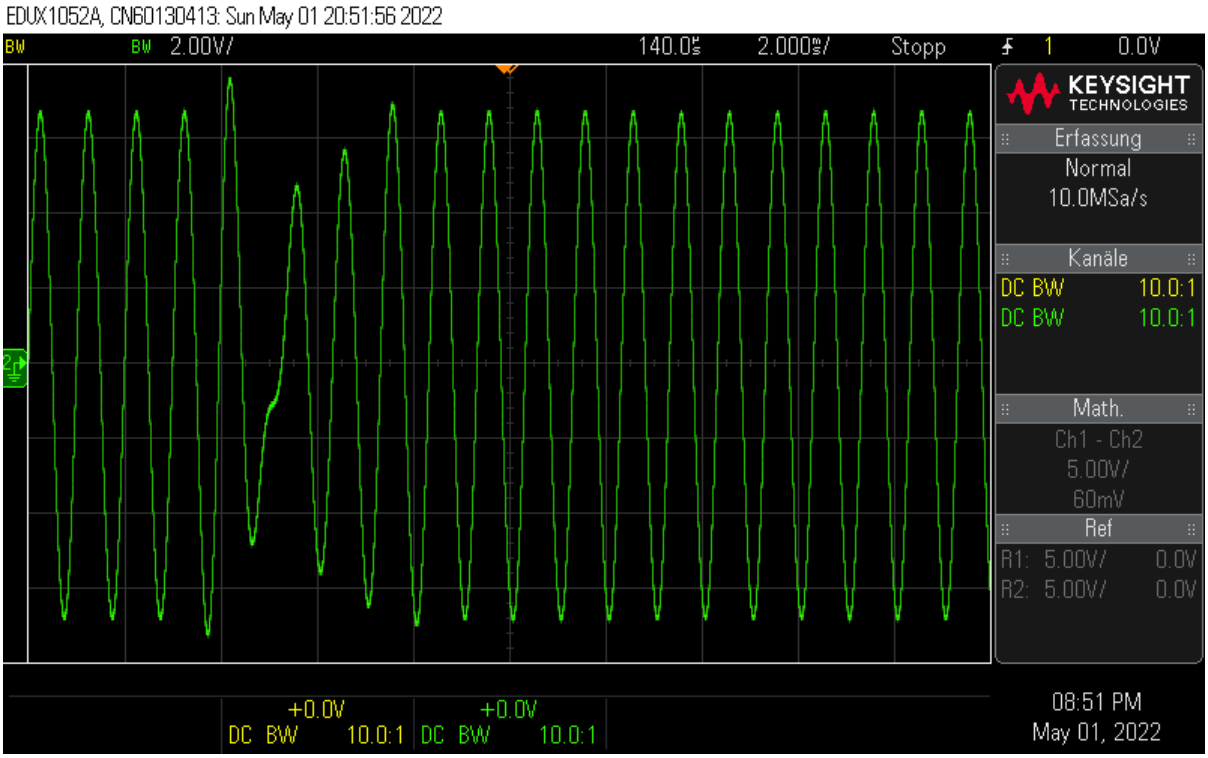


Abbildung 28: Deutlich sichtbarer Phasensprung

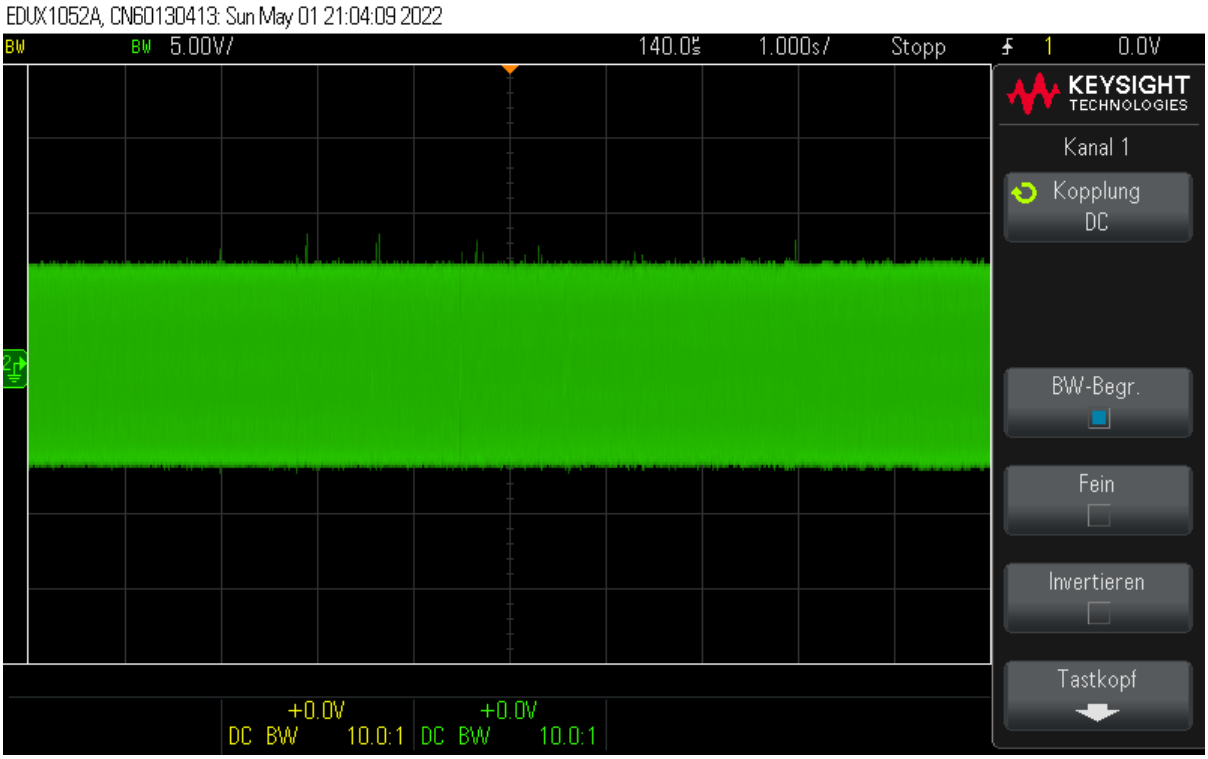


Abbildung 29: *Knackser* (als kleine Überhöhungen sichtbar)

Und Aussetzer, während denen kurzzeitig kein Ausgangssignal bzw. nur Rauschen am Audio-Interface anliegt und die zeitlich immer gehäuft auftreten (siehe Abbildung 30).

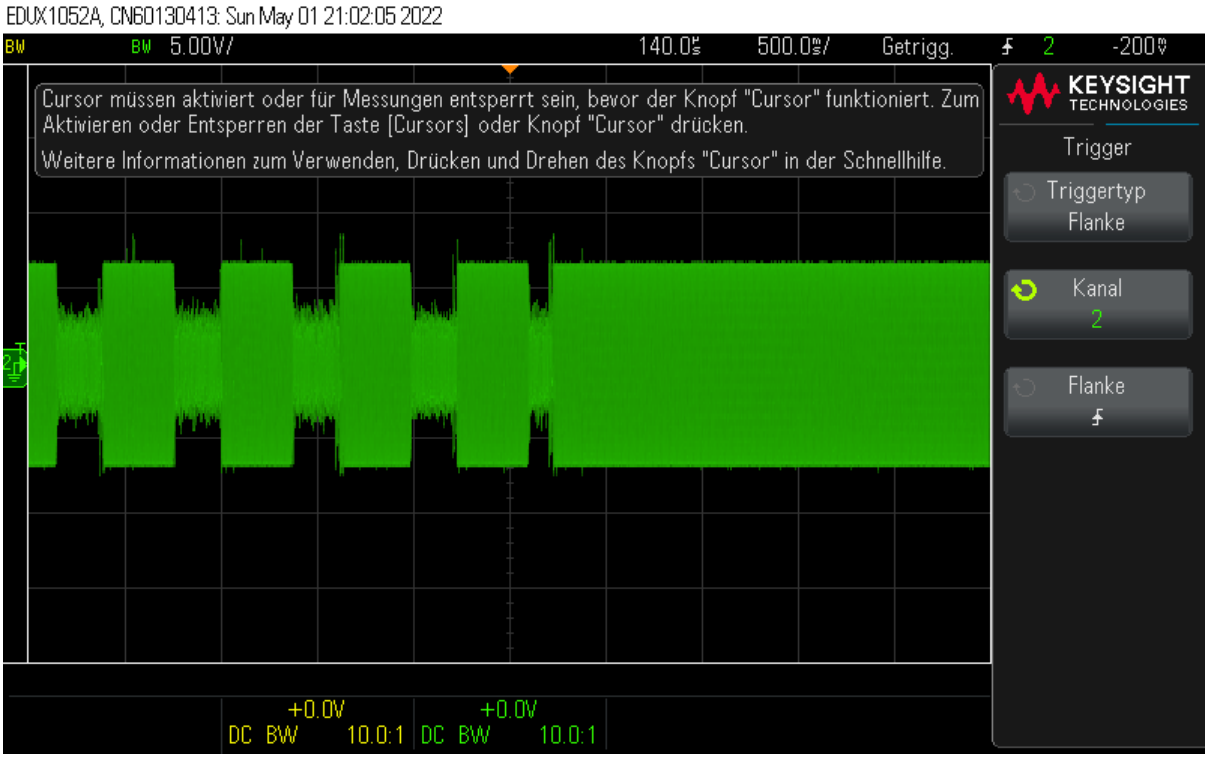


Abbildung 30: Deutlich sichtbare Signalaussetzer

9.4 Messplatine

Da diese teilweise nur sehr kurzen Übertragungsfehler von Hand und mittels Oszilloskop nicht zuverlässig zählbar sind, was einen objektiven Vergleich verschiedener Setups unmöglich machen würde, kam die Idee auf, eine Schaltung zu entwickeln, mit deren Hilfe Übertragungsfehler aller Art detektiert und automatisiert gezählt werden können.

9.4.1 Schaltung zur Detektion von Übertragungsfehlern

Die Schaltung zur automatisierten Fehlerdetektion besteht aus 4 großen Blöcken:

1.) Einem Wien-Oszillator mit einer Frequenz von ungefähr 1 kHz, dem ein (invertierender) Tiefpass ($fg = 1,38kHz$) zur Klirrfaktorreduzierung und Verstärkung nachgeschaltet ist, wodurch insgesamt ein näherungsweise sinusförmiges Testsignal erzeugt wird, welches dann in Jamulus eingespeist wird (siehe Abbildung 31). (Die Dioden dienen als rudimentäre Amplitudenregelung und ersetzen, die oftmals in dieser Schaltung verwendete Glühlampe. (Wikipedia, 2019))

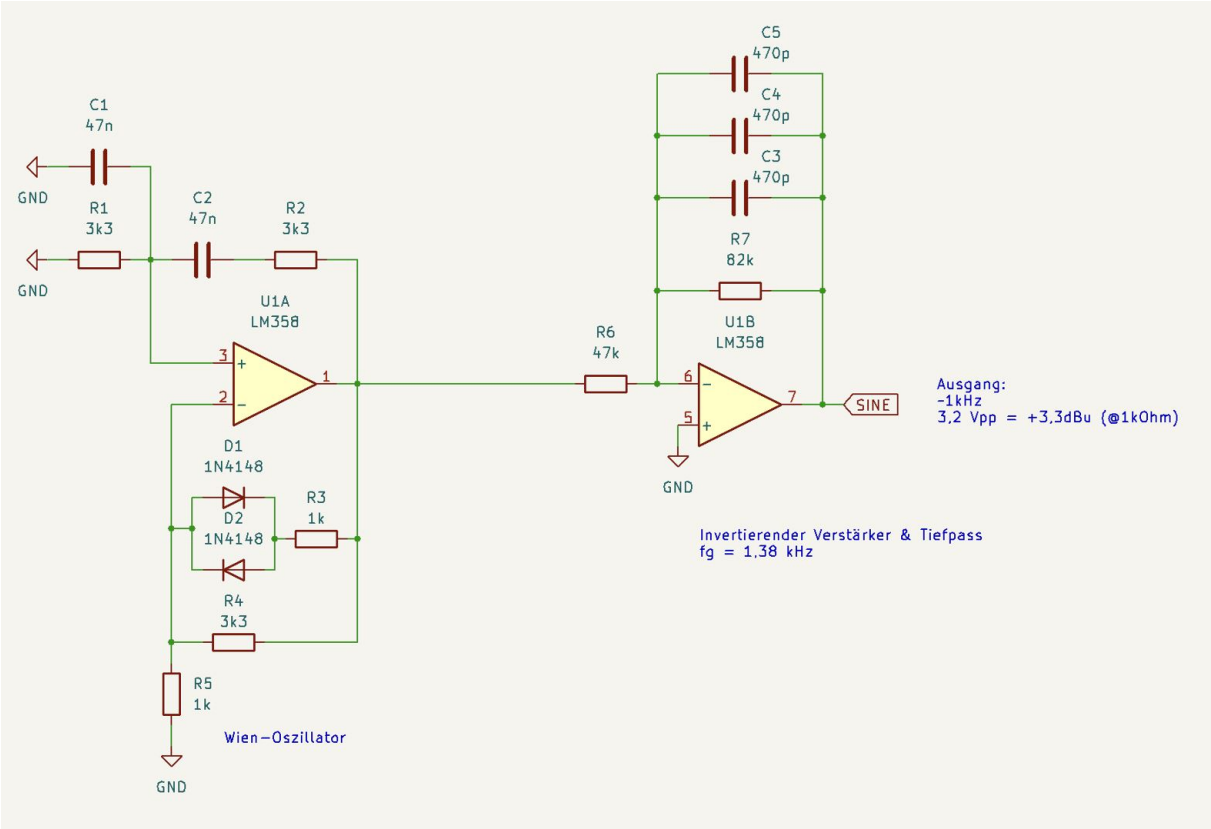


Abbildung 31: Wien-Oszillator und Tiefpass/Verstärker

2.) Einer Kombination aus einem Eingangsverstärker (Gain: 2) und einem Präzisionsgleichrichter (Elliott, 2021) mit nachgeschaltetem RC-Glied zur Umwandlung des Ausgangssignals von Jamulus in eine Gleichspannung (siehe Abbildung 32). (Der Eingangskondensator dient der gleichspannungsmäßigen Entkopplung von Audio-Interface und Messschaltung.)

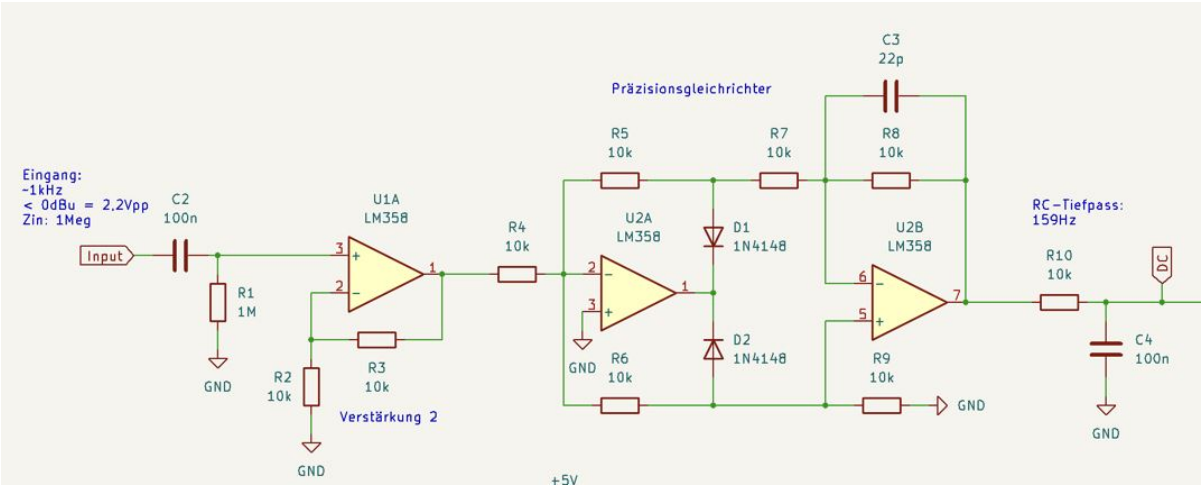


Abbildung 32: Verstärker, Gleichrichter und RC-Glied

3.) Einem Fensterkomparator (AspenCore, Inc, o.J.), bestehend aus zwei Komparatoren, deren Referenzspannungen über Potentiometer eingestellt werden können und de-

ren Ausgänge mittels eines Pull-Down-Widerstands verknüpft sind, was zu einem High-Pegel am Ausgang führt, wenn die gleichgerichtete Spannung den eingestellten Bereich verlässt. Dabei leuchtet die LED „Detect“ auf (siehe Abbildung 33).

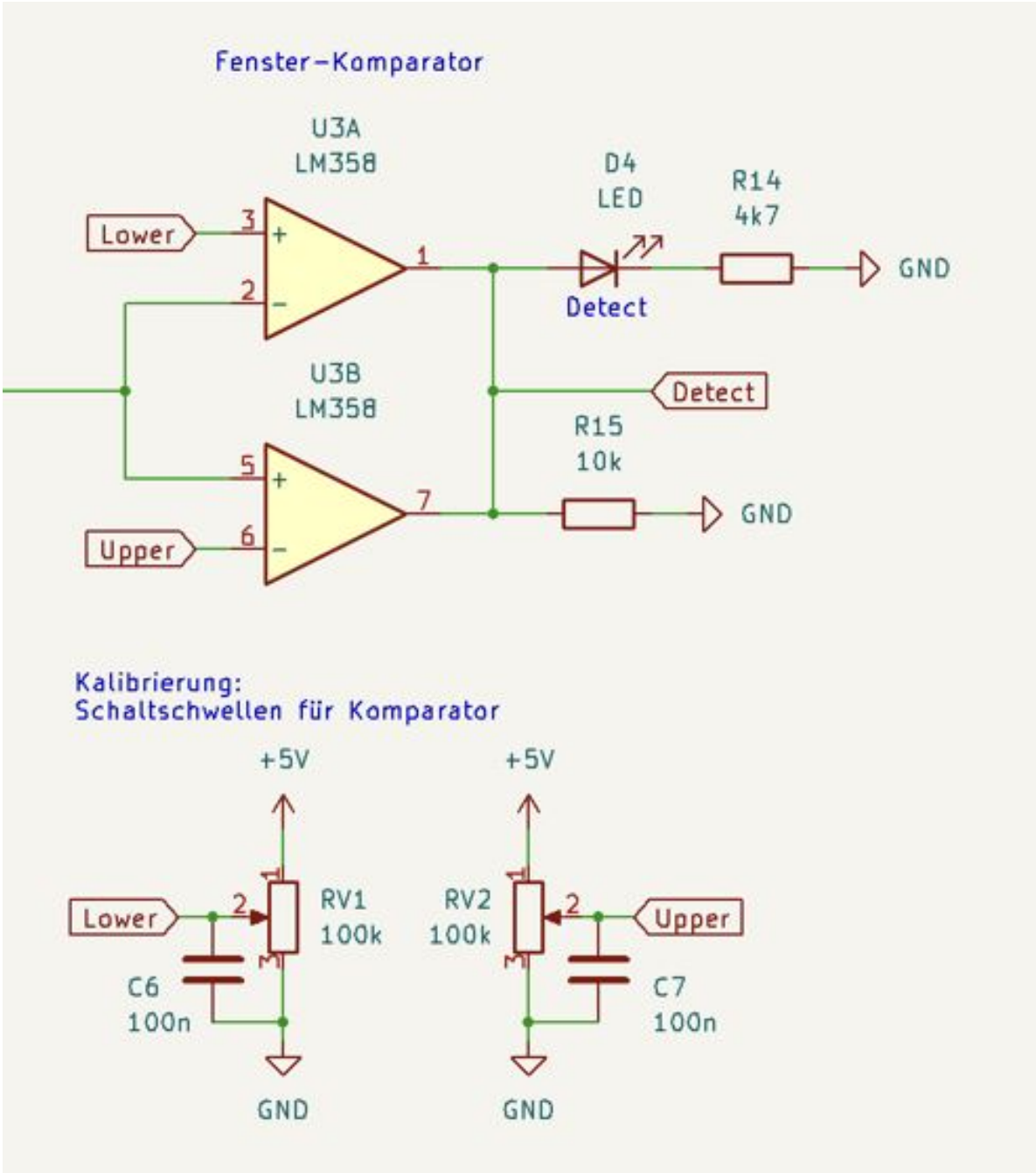


Abbildung 33: Fensterkomparator

4.) Einem Microcontroller (Arduino Nano Every (Arduino, o.J.)) mit TFT-Display (Simac Electronics GmbH, o.J.) zur Erkennung positiver Flanken am Ausgang des Komparators und Anzeige des Zählerstandes.

Solange keine Übertragungsfehler auftreten, liegt durch den kontinuierlichen Sinus als Testsignal am Komparator eine konstante Gleichspannung an. Tritt aber ein Übertra-

gungsfehler auf, z.B. ein kurzer Aussetzer, dann verringert (oder erhöht) sich die Gleichspannung am Ausgang des Gleichrichters kurzzeitig geringfügig, was durch den Komparator bei entsprechend eng eingestellten Schranken zu einem deutlichen High-Pegel umgesetzt wird, dessen positive Flanke der Microcontroller mittels Hardware-Interrupt erkennen und zählen kann. Ob dabei die Zahl der fehlerfreien Sekunden innerhalb der letzten Minute oder die Anzahl der Fehler seit Programmstart angezeigt wird, lässt sich softwareseitig frei definieren – auch andere Statistiken sind denkbar.

9.4.2 Schaltung zur Latenzmessung

Wie im Unterkapitel zum Thema Latenz bereits erläutert wurde, kann zwar eigentlich die Latenzanzeige in Jamulus als Vergleichswert herangezogen werden, da es mittels Microcontroller aber einfach zu realisieren ist, wurde die Messplatine um eine Möglichkeit zur Latenzmessung erweitert. So kann auch ohne Oszilloskop und Funktionsgenerator schnell und einfach eine Latenzmessung durchgeführt werden, sollte der Anzeigewert in Jamulus angezweifelt werden.
Dazu wird ähnlich dem bereits aufgezeigten Messverfahren ein Burst-Signal erzeugt und dessen Laufzeit gemessen.
Das Burst-Signal wird aus einem vom Microcontroller mittels PWM generierten Rechtecksignal mit einer Frequenz von ungefähr 1 kHz und einem Duty-Cycle von 50% erzeugt. Nach Tiefpassfilterung ($fg = 1,6kHz$) ist das Signal näherungsweise sinusförmig.
Das rücklaufende Signal wird anhand der Ausgangsspannung des Gleichrichters, welche digitalisiert und mit einer zuvor gemessenen Referenzspannung verglichen wird, erkannt.

9.5 Anforderungen und Ergänzungen

Zusätzlich gab es noch einige weitere Anforderungen an die Messplatine:

Die Eingangsimpedanz der Messschaltung wird durch den vorgeschalteten Eingangsverstärker auf etwa $1M\Omega$ erhöht, was es erlaubt, auch Kopfhörerausgänge anzuschließen.
Die Ausgangsimpedanz des Wien-Oszillators bzw. des Burst-Signals ist durch den nachgeschalteten aktiven Tiefpass bzw. durch Spannungsfolger vergleichsweise niederohmig.
Da alle Audio-Interfaces über Möglichkeiten verfügen, sowohl die Eingangsempfindlichkeit als auch den Ausgangspegel zu regulieren, konnte auf einstellbare Dämpfungen und Verstärkungen verzichtet werden. Bei einem Eingangspegel von mehr als 2,2 Vpp – ein Pegel den alle getesteten Interfaces erreichen können – leuchtet eine *Clipping*-LED auf. Die eigentliche Clip-Grenze der Schaltung liegt zwar höher, jedoch ist ein Pegel von etwas unter 2,2 Vpp ausreichend. Der Ausgangspegel des kontinuierlichen Sinus-Testsignals beträgt ca. 3,2 Vpp, der des Burst-Signals etwas weniger, was ebenfalls mit allen getesteten Interfaces kompatibel ist.

Welches Testsignal (Burst oder Sinus) gerade am Ausgang anliegt, wird vom Microcontroller zur jeweiligen Messung passend per Relais umgeschaltet.
Die Ein- und Ausgänge sind als 6,3 mm TRS-Klinkenbuchsen (Tip, Ring, Sleeve) ausgeführt, wobei der Ring-Kontakt des Eingangs nicht angeschlossen ist und der des Ausgangs auf Masse liegt. Somit können (mit den richtigen Kabeln) auch bedenkenlos symmetrische Ein- und Ausgänge von Audio-Interfaces angeschlossen werden.
Zusätzlich gibt es BNC-Buchsen, mittels derer sich das Eingangssignal (hinter dem Eingangsverstärker) und das Ausgangssignal am Oszilloskop darstellen lassen. Hinzu kommen diverse Testpunkte an allen *interessanten* Stellen der Schaltung, sodass sich die korrekte Funktion der Schaltung kontrollieren lässt.
Um auch mit dem erwähnten USB-Mikrofon Tests durchführen zu können, wurde ein kleiner Lautsprecher (reichelt elektronik GmbH & Co. KG, o.J.) ergänzt, mit dem Testsignale in das USB-Mikrofon eingespeist werden können. Über einen Kippschalter lässt sich umschalten, ob der Lautsprecher ein Testsignal ausgeben soll, stummgeschaltet ist oder am Ausgang des Komparators angeschlossen ist, was bei jedem erkannten Übertragungsfehler zu einem Knacken des Lautsprechers, ähnlich einem Geigerzähler, führt und somit auch eine akustische Überwachung der Messung ermöglicht.
Das verwendete (Farb-)Display besitzt eine Auflösung von 160 x 128 Pixeln, was theoretisch auch die grafische Ausgabe von Messergebissen erlauben würde. Zusätzlich ist eine weitere LED (*Check*) am Microcontroller angeschlossen, mit der (frei programmierbar) zum Beispiel der Status einer Messung angezeigt werden kann.
Zur Eingabe und Funktionsauswahl stehen vier Taster zur Verfügung.
Die Spannungsversorgung der Messplatine benötigt via 2 mm Laborsteckern eine externe Spannung von etwa 7 VDC. Diese wird intern zunächst mittels Schaltregler (ICL7660S) invertiert. Aus diesen beiden Spannungen ($\pm 7V$) werden dann mittels Linearreglern (7805/7905) die eigentlichen Betriebsspannungen der Analogschaltung ($\pm 5V$) generiert.
Die Komponenten zur Steuerung (Arduino, Display und Relais) werden vom integrierten 5V-Spannungswandler des Arduinos versorgt, um den Ripple an den Messschaltungen möglichst gering zu halten. (Vollständiger Schaltplan im Anhang A.3)

9.5.1 PCB

Nach ausgiebigen Tests aller kritischen Teile der Messchaltung auf einem Breadboard, wurden alle Komponenten auf einer professionell hergestellten Platine zusammengefasst (siehe Abbildung 34).
Das Platinenlayout wurde dabei ebenso wie die Schaltpläne mit dem kostenlosen Programm KiCAD erstellt. (Charras, o.J.)

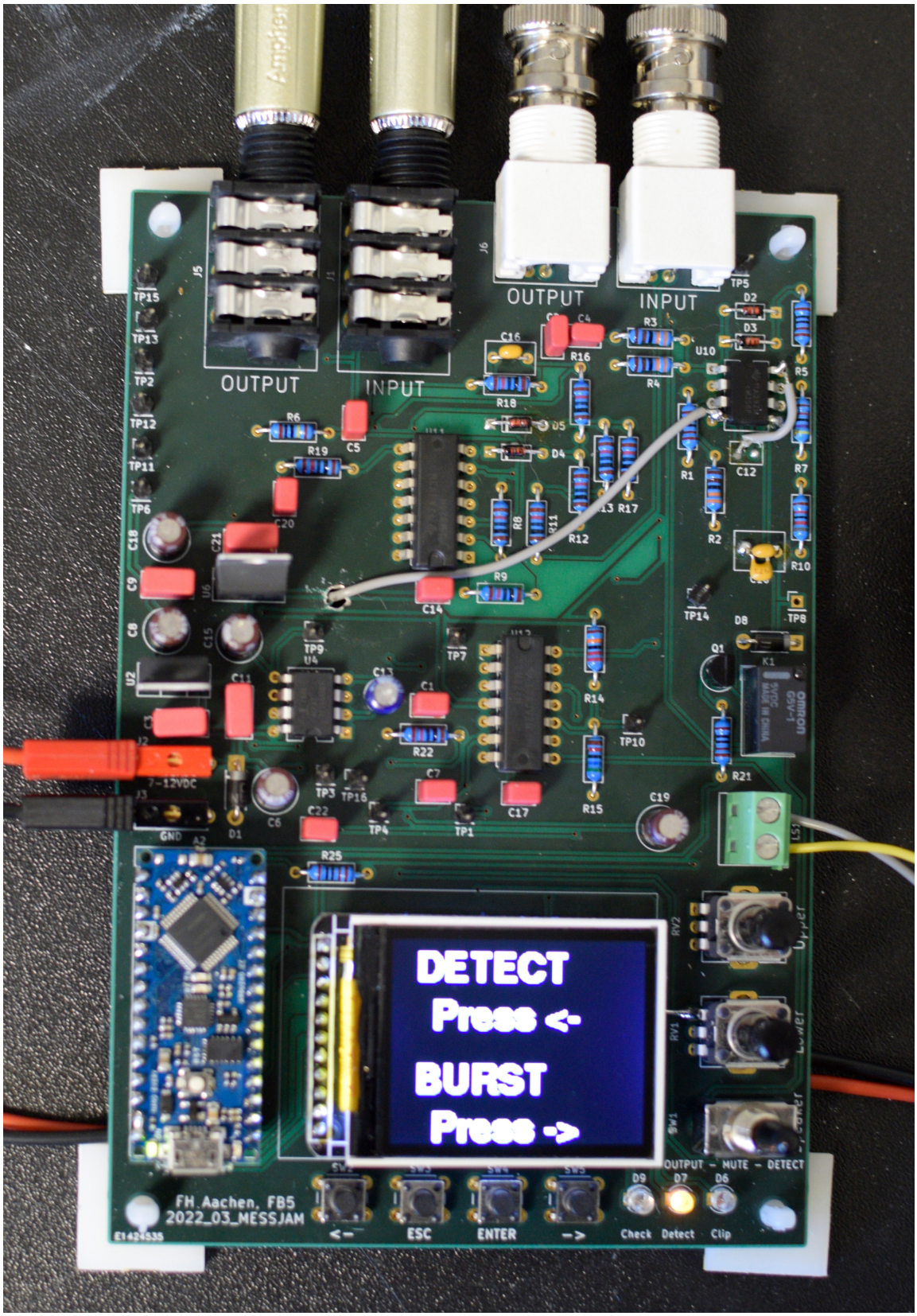


Abbildung 34: Messplatine

Leider enthält das aktuelle Platinenlayout ein paar Fehler, die beim Nachbau unbedingt beachtet werden müssen (siehe A.4). Für eine Überarbeitung der Patine blieb leider keine Zeit mehr; allerdings sollten diese Punkte ausgebessert werden, bevor das die Da-

teien gegebenenfalls online zum Nachbau zur Verfügung gestellt werden.
Trotz dieser Fehler funktioniert die Messplatine wie geplant und es konnten in den letzten Tagen des Projektes Messungen durchgeführt werden, die zuvor nicht möglich waren.
Nachfolgende Oszilloskop-Screenshots zeigen die an einigen Testpunkten anliegenden Signale:

Test der Schaltung zur Detektion von Übertragungsfehlern:

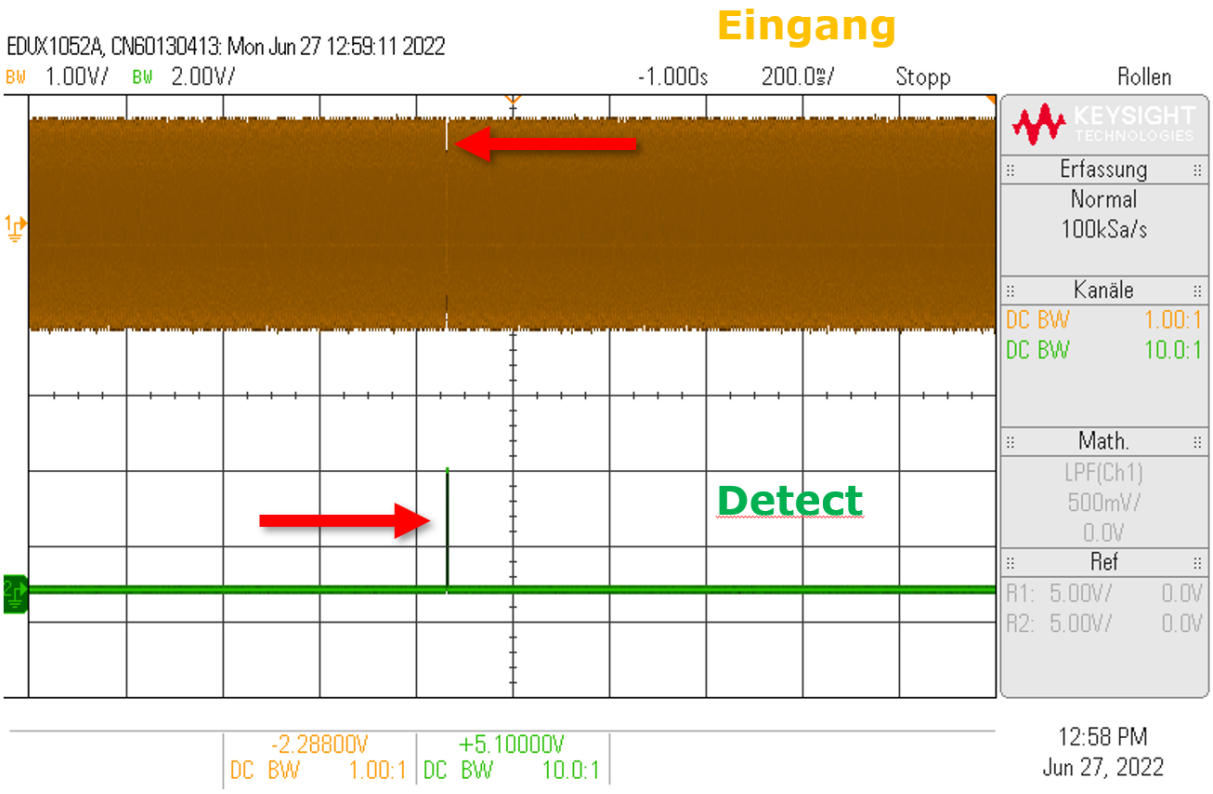


Abbildung 35: Eingangssignal mit erkennbarem Aussetzer und Detect-Signal am Komparator-Ausgang

Test der Latenzmessung:

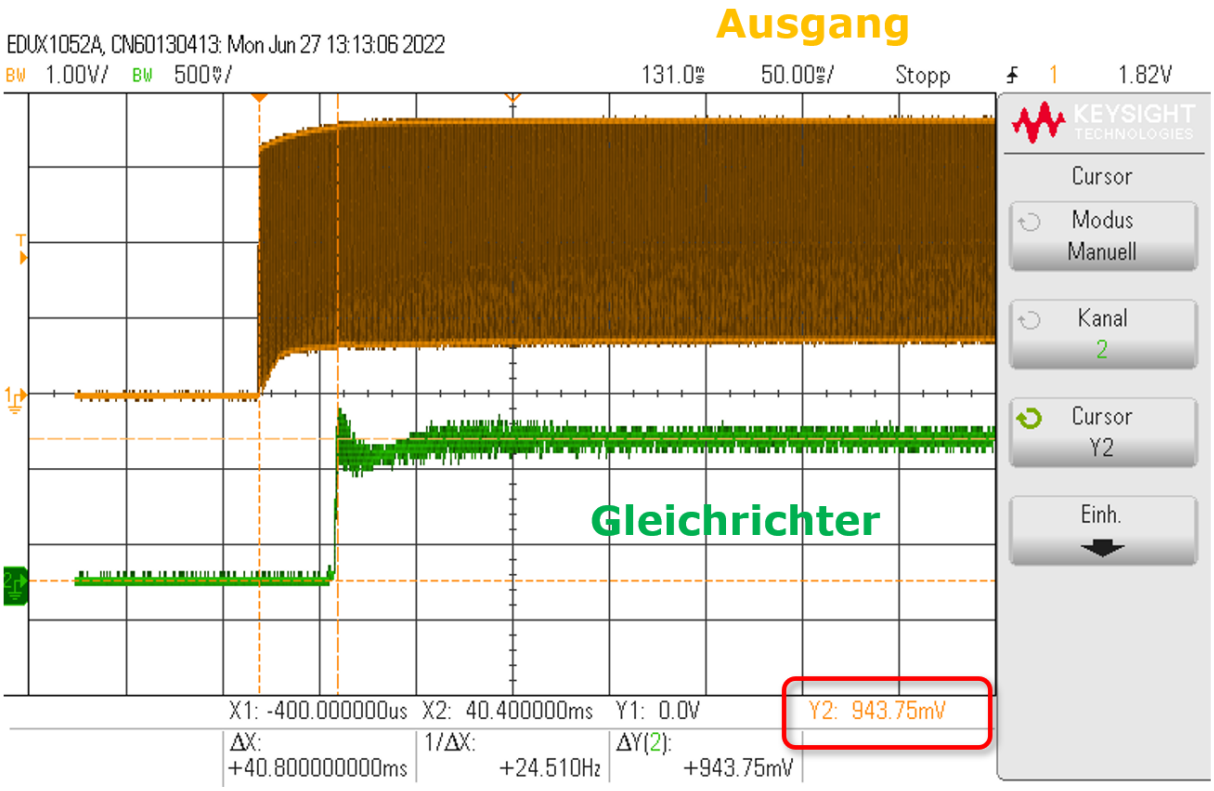


Abbildung 36: Gesendetes Burst-Signal und Spannungsanstieg am Gleichrichter durch rückkehrendes Signal

9.6 Messergebnisse und abgeleitete Erkenntnisse

9.6.1 Vergleich WLAN vs. LAN

In der offiziellen Jamulus-Dokumentation wird dringend geraten, das verwendete Endgerät via LAN und nicht via WLAN mit dem Internet zu verbinden. (Fischer, o.J.c) Diese Empfehlung können wir klar unterstützen. Zwar zeigtes ich in unseren Tests kein nennenswerter Unterschied hinsichtlich der Latenz, doch traten die längeren Übertragungsaussetzer (s.o.) ausschließlich über WLAN-Verbindungen auf, weshalb wir Paketverluste als Ursache für diese Art der Übertragungsfehler vermuten. Grundsätzlich war die gemessene Fehlerrate bei LAN-Verbindungen signifikant geringer.

9.6.2 Einfluss des Serverstandortes und der Serverauslastung

Da der Serverstandort maßgeblich zum Ping beiträgt, welcher wiederum einen Summanden der Gesamtlatenz darstellt, ist es bei Nutzung öffentlicher Server ratsam, die Liste nach aufsteigendem Ping zu sortieren. Soll ein eigener Sever eingerichtet werden, kann es irreführend sein, einen möglichst nahegelegenen Server zu verwenden. Wichtiger ist die schnelle Anbindung des Servers an einen Internet-Knoten.
Auch die Verwendung von lokalen Servern wurde im Rahmen des Projektes untersucht, was aus unserer Sicht aber (mit Ausnahme von Testzwecken) keinen sinnvollen Anwen-

dungszweck haben dürfte. Genutzt wurden dafür der bereits vorgestellte (virtualisierte) Server der DVZ der FH Aachen (Kapitel 3.2.2) sowie die vorhandenen Einplatinencomputer (Kapitel 3.2.4).
Bei Verwendung des DVZ-Servers konnte sehr gut beobachtet werden, wie sehr sich die Auslastung des Servers bemerkbar macht. So sank die Latenz eines aus Raspberry Pi und Behringer UM2 bestehenden Test-Setups, nachdem sich der Campus nachmittags geleert hatte, von etwa 45 ms auf etwa 20 ms.
Der externe Zugriff auf lokale Server mittels VPN erwies sich erwartungsgemäß als nicht sinnvoll, da die Latenz durch die Weiterleitung deutlich ansteigt.

9.6.3 Minimal erreichbare Latenz

Mittels lokalem Server und dadurch vernachlässigbarem Ping ($< 1ms$), ließ sich wiederholt eine minimale Latenz von ungefähr 20 ms messen.
Unter Berücksichtigung des Interface-Puffers bleibt für die Kombination aus Netzwerk-Puffer und Signalverarbeitung also ein Beitrag von etwa 15 ms zur Gesamtlatenz.

9.6.4 Einstellungen in Jamulus

Bei zahlreichen Tests konnte festgestellt werden, dass alle getesteten USB-Interfaces und auch das getestete USB-Mikrofon bei einer Puffergröße von 128 Samples (unter Linux) fehlerfrei funktionieren. Bei Umstellung auf 64 Samples Puffergröße kommt es jedoch teilweise zu Aussetzern.
Außerdem konnte beobachtet werden, dass sich bei Aktivierung der Option *Aktiviere kleine Netzwerkpuffer* in den Netzwerkeinstellungen der Jamulus-Applikation zwar eine minimale Gesamtlatenz von nur 16 ms erreichen lässt, allerdings in Verbindung mit erheblichen Übertragungsfehlern.
Grundsätzlich kann in Bezug auf die Netzwerkeinstellungen in Jamulus festgehalten werden, dass die Option „Auto“ gemäß unserer Tests eigentlich in allen Fällen den besten Kompromiss aus geringer Latenz und stabiler Übertragung liefert.

9.6.5 Vergleich zu Windows

Ein Vergleichest zwischen einem Raspberry Pi 4 (2 GB) und einem frisch installierten Windows-PC lieferte mit einer Puffergröße von 64 Samples und Verbindung zum selben lokalen Server nahezu identische Latenzwerte von in beiden Fällen etwa 19 ms.
Zumindest im Fall eines Windows-Gerätes, auf dem kaum Software installiert ist und dementsprechend wenige Programme im Hintergrund laufen, ist die Performance zwischen Linux und Windows also absolut vergleichbar.

10 Fazit und Empfehlungen

Abschließend werden noch einmal die Ergebnisse zusammengefasst und basierend darauf einige Empfehlungen zur Konfiguration von Jamulus-Clients und des benötigten Servers abgeben.

10.1 Jamulus-Client als Einplatinencomputer

Der im Rahmen des Projektes entwickelte Jamulus-Client basiert auf dem Raspberry Pi 4 mit 2 GB RAM und dem Waveshare 7inch HDMI LCD (C) Touchdisplay, wie in Kapitel 4 beschrieben. Dieser Client bietet Nutzern die Möglichkeit, unabhängig der vorliegenden technischen Versiertheit, mit einem Minimum an Aufwand an einer Jamulus-Session teilzunehmen. Dabei wird gleichzeitig garantiert, dass das System weder durch andere Programme, noch durch Updates oder Nutzereingaben dahingehend negativen Einfluss auf die eigene Nutzererfahrung nehmen kann, noch auf die Erfahrungen der anderen Nutzer in der Session. Durch das in Kapitel 4.4.2 vorgestellte Touchdisplay wird den Nutzern die Möglichkeit geboten, vollständig auf die Verwendung weiterer Peripheriegeräte zu verzichten, während gleichzeitig unter Ausnutzung der virtuellen Tastatur aus Kapitel 4.5 alle Eingaben möglich sind, um den Jamulus-Client vollständig zu bedienen. Darüber hinaus ermöglicht das System, durch die in Kapitel 4.3 vorgestellte Installation von TeamViewer auf dem Raspberry Pi 4, die Fernwartung (bei Bedarf). Dadurch kann hier von einer hohen Verfügbarkeit der Clients gesprochen werden, da diese in der Regel per Fernwartung gesteuert bzw. aktualisiert/ geändert werden können, ohne, dass ein Techniker den Client vor Ort verändern muss.

10.2 Empfehlungen zur Hardwareauswahl

Gegenüber Konkurrenzprodukten bieten die Raspberry Pi Modelle den Vorteil einer deutlich größeren Community, was sich vor allem bei der erheblich besseren Dokumentation bemerkbar macht, weshalb wir, aller Lieferschwierigkeiten zum Trotz, ganz klar zur Verwendung des Raspberry Pi 4, idealerweise mit 2 GB RAM, raten. Sofern ein Raspberry Pi Modell verwendet wird, muss die Auswahl an verfügbaren Touchdisplays hinsichtlich der Anschlüsse eigentlich nicht weiter eingeschränkt werden. Wir müssen jedoch dringend zur Verwendung eines Displays mit einer Auflösung von mindestens 1024 x 600 Pixeln raten, da die Jamulus-Benutzeroberfläche, wie in Kapitel 4.4.1 gezeigt, sonst nicht dargestellt werden kann. Dies geht aus der offiziellen Dokumentation von Jamulus leider nicht hervor. Bei der Auswahl des Audio-Interfaces können wir lediglich zur Verwendung eines USB-Interfaces raten, das zu den individuellen Bedürfnissen des jeweiligen Musikers passt. Die Performance aller getesteten Interfaces unterschied sich in unseren Tests allenfalls marginal, sodass bereits vorhandene USB-Interfaces problemlos in das Setup integriert werden können. Sollen neue oder zusätzliche Interfaces angeschafft werden, so können wir die vergleichsweise sehr preisgünstigen Interfaces des Herstellers Behringer empfehlen – vor allem das Modell UM2 weißt ein nahezu

unschlagbar gutes Preisleistungsverhältnis und eine sehr einfache Bedienbarkeit auf. Allerdings müssen hierfür zusätzlich externe Mikrofone samt Kabel und Ständer vorhanden sein bzw. angeschafft werden. Als vollintegrierte und dadurch komfortablere, aber auch etwas teurere Alternative können wir das getestete USB-Mikrofon Mackie EM-USB uneingeschränkt empfehlen. Um eine stabile Verbindung zwischen Client und Server sowie eine geringe Fehlerrate zu gewährleisten, raten wir dringend zur Internetanbindung des Clients via LAN. Eine WLAN-Verbindung sollte allenfalls eine Notlösung darstellen.

10.3 Empfehlungen zu den Einstellungen innerhalb von Jamulus

Hinsichtlich der Einstellungen in Jamulus können wir anhand unserer Tests und Messungen dazu raten, dass in den Netzwerkeinstellungen von Jamulus die Option „Auto“ stets aktiviert und die Option „Aktiviere kleine Netzwerkpuffer“ stets deaktiviert sein sollte, da es sonst fast immer zu einem erheblichen Anstieg der Fehlerrate kommt. Für die Puffergröße des Audio-Interfaces erwies sich mit allen getesteten Modellen (inkl. USB-Mikrofon) ein Wert von 128 Samples als völlig ausreichend, um Aussetzer bei der USB-Übertragung zuverlässig zu vermeiden. Für die verschiedenen Auswahlmöglichkeiten der Option „Audioqualität“ konnte hingegen kein messbarer Unterschied festgestellt werden.

10.4 Empfehlungen zur Einrichtung eines Jamulus-Servers

Bei der Einrichtung eines eigenen Jamulus-Servers ist ein Root-Server einem virtualisierten Server vorzuziehen, da die Serverauslastung, wie im Test des DVZ-Servers zu verschiedenen Uhrzeiten (siehe Kapitel 9.6.2) gezeigt, großen Einfluss auf die Latenz des Systems hat. Außerdem sollte der Server nicht zu weit von den Nutzern entfernt sein und über eine schnelle Anbindung an einen Internet-Knoten verfügen, um den Ping möglichst gering zu halten.

11 Ausblick

In den folgenden Abschnitten werden nun noch Zukunftsaussichten geboten, inwiefern das Projekt weitergeführt werden kann. Dafür wird in Kapitel 11.1 der Jamulus-Client als geschlossene Entität bzw. als fertiges Produkt, dass von den Nutzern ohne weiteren Konfigurationsaufwand verwendet werden kann, vorgestellt. In Kapitel 11.2 werden dann weitere Messexperimente vorgeschlagen und Fragen gestellt, die im Anschluss an dieses Projekt weiter zu beantworten sind. In Kapitel 11.3 werden dann zum Abschluss noch Vorschläge zur Öffentlichkeitsarbeit bzw. zum Marketing geboten, um den Jamulus-Client inklusive der gebotenen Vorzüge der Allgemeinheit zu präsentieren und bei dieser zu bewerben.

11.1 Jamulus-Client als geschlossene Entität

Der in Kapitel 10.1 als Projektabschluss beschriebene Jamulus-Client ist in seiner aktuellen Form voll funktionsfähig und nutzbar. Allerdings sind noch einige Optimierungsmöglichkeiten offen, die in diesem Abschnitt im Detail beschrieben werden.
Um den Jamulus-Client als fertiges Produkt zur Verfügung zu haben, bedarf es eines Gehäuses, in dem sowohl der Einplatinencomputer, als auch das Touchdisplay fest verbaut sind. Dies ist dahingehend wichtig, um eventuelle Schäden an Display oder Einplatinencomputer zu verhindern, die entstehen könnten, da das Hardwaresystem aktuell sehr instabil ist. Auch aus softwareseitigem Hintergrund ist es wichtig, eventuelle Zugriffe per USB oder anderen Ports zu verhindern, um den Zugriff auf die geschlossene Entität zu verhindern. Dabei soll (nach Möglichkeit) lediglich ein USB-Port, in das dann ein USB-Audiointerface angeschlossen werden kann, sowie der LAN-Port, zur Verbindung mit dem Internet, für die Nutzer zugreifbar sein.
Der aktuelle Stand des Jamulus-Clients setzt voraus, dass ein Touchdisplay oder ein normales Display mit zusätzlichen Peripheriegeräten angeschlossen ist. Dies wird besonders dafür benötigt, um Jamulus per Touch auf das Icon in der Taskleiste zu starten (siehe Kapitel 4.6). Vorstellbar ist es allerdings nun auch, falls der Client als fertiges Produkt an Nutzer unterschiedlicher technologischer Versiertheit verteilt werden soll, ebenfalls eine Version des Clients zur Verfügung zu stellen, bei der stets dann, wenn der Jamulus-Client (bzw. der Einplatinencomputer) mit Strom versorgt wird und das Betriebssystem vollständig gestartet ist, automatisch eine Verbindung zu einem vordefinierten Server erfolgen soll. Die Dokumentation in Kapitel 4.6 bietet dafür eine solide Grundlage, da hierbei beschrieben ist, wie ein Shell-Script auszusehen hat, mit dem sich ein Client automatisch auf einen Server verbindet. Entsprechend muss dann dieses Shell-Script (beispielsweise über einen Service) automatisch ausgeführt werden. Darüber hinaus kann es sinnvoll sein, einen physischen Knopf für das direkte Aktivieren einer TeamViewer-Verbindung bzw. dem automatisieren Übermitteln der TeamViewer Daten an einen Empfänger an den Jamulus-Client anzuschließen. Dadurch können ebenfalls technisch weniger versierte Nutzer im Falle technischer Probleme unmittelbar Hilfe anfordern, ohne das Touchdisplay oder andere Peripheriegeräte zu verwenden.

11.2 Weitere Messungen

Zwar konnte weitestgehend geklärt werden, wie die Gesamtlatenz im Jamulus zustande kommt und wie sich diese minimieren lässt, doch ist zum aktuellen Zeitpunkt immer noch nicht ganz klar, welchen Einfluss Latenz-Jitter, also Schwankungen der Latenz, auf die Übertragung haben und ob sich dadurch möglicherweise auch das manchmal recht hohe Grundrauschen der Übertragung erklären lässt. Ferner bietet die Messplatine in ihrer aktuellen Form Raum für Verbesserungen. So sollte zum einen das Platinenlayout korrigiert werden, zum anderen könnten in der Programmierung des Arduinos noch zahlreiche weitere Messstatistiken implementiert werden. Um letzteren Punkt möglichst aussagekräftig gestalten zu können, wäre auch eine Studie unter Mitwirkung von Hochschulchor und FH Big Band denkbar. Mögliche Fragestellungen dafür wären

zum Beispiel: Wie lang muss ein Übertragungsfehler sein, um als störend wahrgenommen zu werden? Wie groß muss der zeitliche Abstand zwischen zwei Übertragungsfehlern sein, damit diese getrennt wahrgenommen werden? (Gerade bei Burst-Fehlern könnte dies sehr interessant sein.) Wie viel Latenz ist maximal akzeptabel? Welcher minimale Latenzwert kann von Instrumentalisten, welcher von Sängern wahrgenommen werden?

11.3 Veröffentlichung der Ergebnisse

Wie in der ursprünglichen Ausschreibung des interdisziplinären Projektes durch Herrn Borucki vorgesehen, wäre es gut, wenn durch MCD-Studierende Öffentlichkeitsarbeit für das Projekt betrieben würde. Dies würde neben einer detaillierten Bedienungsanleitung bzw. einem Tutorial-Video auch die hochschulinterne und externe Bekanntmachung des Projektes beinhalten.

Literatur

Adafruit Industries (2022). Adafruit GFX Graphics Library. Verfügbar unter: <https://learn.adafruit.com/adafruit-gfx-graphics-library> [aufgerufen am 19. Juli 2022].

Arduino (o.J.). Arduino Nano Every. Verfügbar unter: <https://docs.arduino.cc/hardware/nano-every> [aufgerufen am 19. Juli 2022].

armbian (o.J.). armbian - Linux for ARM development boards. Verfügbar unter: <https://www.armbian.com/> [aufgerufen am 17. Juli 2022].

AspenCore, Inc (o.J.). Op-amp Comparator: Window Comparator. Verfügbar unter: <https://www.electronics-tutorials.ws/opamp/op-amp-comparator.html> [aufgerufen am 19. Juli 2022].

Charras, J.-P. (o.J.). KiCad. Verfügbar unter: <https://www.kicad.org/> [aufgerufen am 19. Juli 2022].

Costa, A. (o.J.). rpilocator - Find Raspberry Pi Computers in Stock. Verfügbar unter: <https://rpilocator.com/> [aufgerufen am 18. Juli 2022].

DIY Recording Equipment (2020). Measure Your Audio Gear with REW: Pt. 2 Frequency Response. Verfügbar unter: <https://www.youtube.com/watch?v=nCXfol7oTg4> [aufgerufen am 19. Juli 2022].

Elliott, R. (2021). Precision Rectifiers: Another Version (Figure 6A). Verfügbar unter: <https://sound-au.com/appnotes/an001.htm> [aufgerufen am 19. Juli 2022].

Fischer, V. (o.J.a). Case Study: Performing Band Rehearsals on the Internet With Jamulus. Verfügbar unter: <https://jamulus.io/PerformingBandRehearsalsontheInternetWithJamulus.pdf> [aufgerufen am 19. Juli 2022].

Fischer, V. (o.J.b). Jamulus FAQ: Why should I not listen to my own signal? Verfügbar unter: <https://jamulus.io/de/wiki/FAQ> [aufgerufen am 19. Juli 2022].

Fischer, V. (o.J.c). Jamulus FAQ: Why shouldn't I use wireless equipment? Verfügbar unter: <https://jamulus.io/de/wiki/FAQ> [aufgerufen am 19. Juli 2022].

Hardkernel (2021). ODROID-XU4. Verfügbar unter: <https://wiki.odroid.com/odroid-xu4/odroid-xu4> [aufgerufen am 19. Juli 2022].

iPhyse (2016). Pine64 + waveshare display? Verfügbar unter: <https://forum.pine64.org/showthread.php?tid=1414> [aufgerufen am 18. Juli 2022].

Jamulus (2022a). Compiling Jamulus. Verfügbar unter: <https://github.com/jamulussoftware/jamulus/blob/51709ebdae2ba6b2bb2157647824545f5ad6fa6d/COMPILING.md> [aufgerufen am 23. Juni 2022].

Jamulus (2022b). Jamulus - Internet Jam Session Software. Verfügbar unter: <https://github.com/jamulussoftware/jamulus/tree/392e02a8cb1ba805e48d500d4f6a7d9146c4af29> [aufgerufen am 23. Juni 2022].

Langasek, S. (2021). [Bug 1921781] [NEW] Build-depends on qt5-default which is obsolete and dropped in hirsute. Verfügbar unter: <https://www.mail-archive.com/ubuntu-bugs@lists.ubuntu.com/msg5902890.html> [aufgerufen am 23. Juni 2022].

LOUD Audio, LLC (o.J.). Mackie - EM-USB. Verfügbar unter: <https://mackie.com/en/products/microphones/em-series-usb-microphones/EM_USB.html> [aufgerufen am 19. Juli 2022].

Mantel, M. (2020). Einplatinencomputer: Raspberry Pi 4 mit 8 GByte RAM verfügbar. Verfügbar unter: <https://www.heise.de/news/Einplatinencomputer-Raspberry-Pi-4-mit-8-GByte-RAM-verfuegbar-4767659.html> [aufgerufen am 17. Juli 2022].

Modul 9 GmbH (2020). DATASHEET DAC+ ADC. Verfügbar unter: <https://www.hifiberry.com/docs/data-sheets/datasheet-dac-adc/> [aufgerufen am 19. Juli 2022].

Mulcahy, J. (2022). REW - Room EQ Wizard Room Acoustics Software. Verfügbar unter: <https://www.roomeqwizard.com/> [aufgerufen am 19. Juli 2022].

Music Tribe Global Brands Ltd (o.J.). Behringer - Audio Interfaces. Verfügbar unter: <https://www.behringer.com/catalog.html?brandName=behringer&catalog=Application&category=C-BEHRINGER-AUDIOINTERFACES&application=P-BEHRINGER-RECORDINGANDBROADCAST&itemsPerPage=15&pageNumber=1&sort=salesRevenue> [aufgerufen am 19. Juli 2022].

odroid (o.J.). odroid - Linux Release note. Verfügbar unter: <https://wiki.odroid.com/odroid-xu4/os_images/linux/start> [aufgerufen am 18. Juli 2022].

PINE64 (o.J.a). Getting Started | Pine64 - Welcome. Verfügbar unter: <https://www.pine64.org/gettings-started/> [aufgerufen am 17. Juli 2022].

PINE64 (o.J.b). PINE A64-LTS. Verfügbar unter: <https://www.pine64.org/devices/single-board-computers/pine-a64-lts/> [aufgerufen am 19. Juli 2022].

PINE64 (o.J.c). PINE A64-LTS/SOPine. Verfügbar unter: <https://wiki.pine64.org/wiki/PINE_A64-LTS/SOPine> [aufgerufen am 19. Juli 2022].

PreSonus Audio Electronics, Inc (o.J.). Digital Audio Latency Explained. Verfügbar unter: <https://www.presonus.com/learn/technical-articles/Digital-Audio-Latency-Explained> [aufgerufen am 19. Juli 2022].

Raspberry Pi Foundation (o.J.). Raspberry Pi 4 Tech Specs. Verfügbar unter: <https://www.raspberrypi.com/products/raspberry-pi-4-model-b/specifications/> [aufgerufen am 19. Juli 2022].

reichelt elektronik GmbH & Co. KG (o.J.). LSM-50M/K/F Kleinlautsprecher LSM-50M/K/F, 0,5W, 50Ohm. Verfügbar unter: <https://www.reichelt.de/kleinlautsprecher-lsm-50m-k-f-0-5w-50ohm-lsm-50m-k-f-p145865.html?&trstct=pol_6&nbc=1> [aufgerufen am 19. Juli 2022].

Renesas Electronics K.K. (2020). ICL7660S. Verfügbar unter: <https://www.renesas.com/us/en/document/dst/icl7660s-icl7660a-datasheet?language=en&r=501856> [aufgerufen am 19. Juli 2022].

Samsung (o.J.). Exynos 5422. Verfügbar unter: <https://semiconductor.samsung.com/processor/mobile-processor/exynos-5-octa-5422/> [aufgerufen am 19. Juli 2022].

Sertronics GmbH (o.J.a). IQaudio DAC+. Verfügbar unter: <https://www.berrybase.de/iqaudio-dac-43> [aufgerufen am 19. Juli 2022].

Sertronics GmbH (o.J.b). RaspiAudio ULTRA+. Verfügbar unter: <https://www.berrybase.de/raspiaudio-ultra-43> [aufgerufen am 19. Juli 2022].

Simac Electronics GmbH (o.J.). D1.8“ DISPLAY. Verfügbar unter: <https://joy-it.net/de/products/RB-TFT1.8> [aufgerufen am 19. Juli 2022].

Speed, R. (2020). Get in the C: Raspberry Pi 4 can handle a wider range of USB adapters thanks to revised design’s silent arrival. Verfügbar unter: <https://www.theregister.com/2020/02/21/pi_4_fixed/> [aufgerufen am 17. Juli 2022].

STMicroelectronics N.V. (2018). L7805. Verfügbar unter: <https://www.st.com/resource/en/datasheet/l78.pdf> [aufgerufen am 19. Juli 2022].

STMicroelectronics N.V. (2019). L7905. Verfügbar unter: <https://www.st.com/resource/en/datasheet/l79.pdf> [aufgerufen am 19. Juli 2022].

szafran (2017). Waveshare 10.1"1280x800 Multitouch Display on Odroid-XU4. Verfügbar unter: <https://forum.odroid.com/viewtopic.php?t=25891> [aufgerufen am 18. Juli 2022].

TeamViewer (o.J.). TeamViewer - Remote-Connectivity-Software. Verfügbar unter: <https://www.teamviewer.com/de/> [aufgerufen am 23. Juni 2022].

Texas Instruments Inc (2022a). LM324. Verfügbar unter: <https://www.ti.com/lit/ds/symlink/lm324.pdf> [aufgerufen am 19. Juli 2022].

Texas Instruments Inc (2022b). LM358. Verfügbar unter: <https://www.ti.com/lit/ds/symlink/lm358.pdf> [aufgerufen am 19. Juli 2022].

Vilniaus Blokas UAB (o.J.). Pisound. Verfügbar unter: <https://blokas.io/pisound/> [aufgerufen am 19. Juli 2022].

Waveshare (o.J.a). 7inch Capacitive Touch Screen LCD (B), 800×480, HDMI, Low Power. Verfügbar unter: <https://www.waveshare.com/7inch-HDMI-LCD-B.htm> [aufgerufen am 15. Juli 2022].

Waveshare (o.J.b). 7inch Capacitive Touch Screen LCD (C), 1024×600, HDMI, IPS, Low Power. Verfügbar unter: <https://www.waveshare.com/7inch-HDMI-LCD-C.htm> [aufgerufen am 15. Juli 2022].

Waveshare (o.J.c). 7inch HDMI LCD (B). Verfügbar unter: <https://www.waveshare.com/wiki/7inch_HDMI_LCD_(B)> [aufgerufen am 15. Juli 2022].

Waveshare (o.J.d). 7inch HDMI LCD (C). Verfügbar unter: <https://www.waveshare.com/wiki/7inch_HDMI_LCD_(C)> [aufgerufen am 15. Juli 2022].

Wikipedia (2019). Wien-Robinson-Brücke. Verfügbar unter: <https://de.wikipedia.org/wiki/Wien-Robinson-Br%C3%BCcke> [aufgerufen am 19. Juli 2022].

Wikipedia (2022). Raspberry Pi OS. Verfügbar unter: <https://en.wikipedia.org/wiki/Raspberry_Pi_OS> [aufgerufen am 19. Juli 2022].

A Anhang

A.1 Anfrage bei der DVZ

Hallo liebe DVZ,

im Rahmen des interdisziplinären Projekts MusiUelNet geht es darum das musizieren über das Internet mit der Client/Server Software Jamulus möglich zu machen. Die Server Version der Jamulus Software ermöglicht hierbei die Synchronisation von Tonsignalen, wodurch das msuizieren über das Internet möglich wird.

Da es sich bei dem interdiziplinären Projekt um ein Projekt handelt, dass im Rahmen des Studiums an der FH durchgeführt werden muss, hätten wir gerne eine VM auf dem DVZ Cluster. Falls es möglich ist hätten wir uns eine VM mit folgenden Leistungen vorgestellt:

Betriebssystem: Ubuntu 20.04 LTS
CPU Kerne: 2
RAM: 4GB
Festplatte: 60GB
Anzahl Netzwerkkarten: 1
Verwendungszweck: s.O.

Zudem bräuchten wir für die VM SSH Zugang aus dem Netz der FH Aachen. Damit alle IP Teilnehmer optimal im Home Office an dem Projekt arbeiten können, benötigen wir ebenfalls die Portfreigabe des Ports 22124/UDP. Dies ist der Standardport auf dem die Jamulus Server Software arbeitet.

Bei weiteren Fragen rund um das Interdisplinäre Projekt steht Ihnen unser IP Verantwortlicher Dipl.-Ing. Stephan Borucki unter borucki@fh-aachen.de gerne zur Verfügung.

Wir bedanken uns im Voraus.

Beste Grüße
MusiUelNet IP-Team
An-Phong Pham Dinh

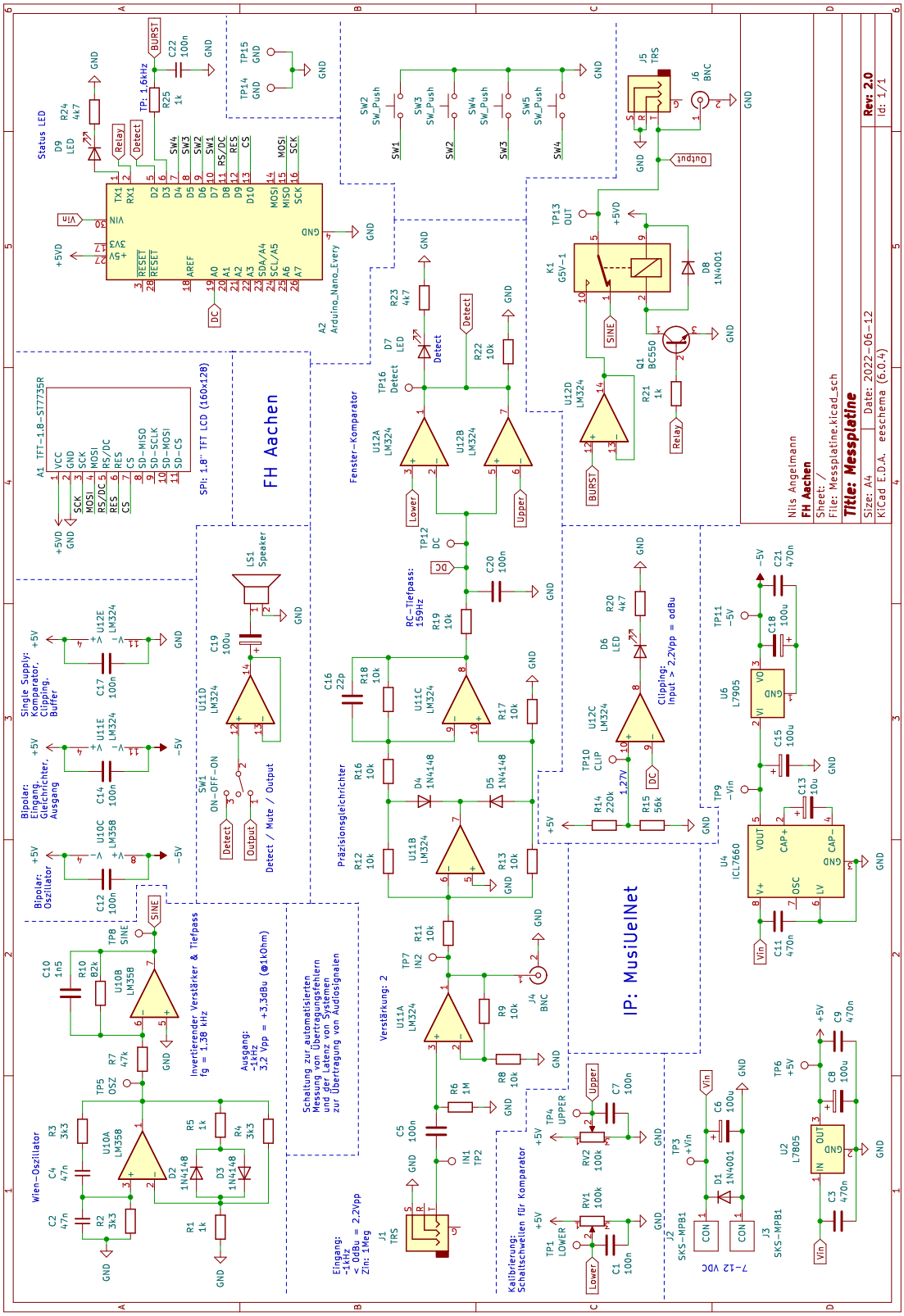
Abbildung 37: DVZ Anfrage

A.2 Jamulus Installations- bzw. Update-Script

```
#############################################################
### AKTUALISIEREN VON RASPBERRY PI OS INKL. ALLER PAKETE ###
#############################################################

# Updaten aller Programme, die auf dem System installiert sind
sudo apt update
sudo apt upgrade -y

# Entfernen aller Programme, die nicht mehr benötigt werden
sudo apt autoremove -y


#########################################
### INSTALLATION DES JAMULUS CLIENTS ###
#########################################

# Installiere die benötigten Pakete, um Jamulus und JACK zu verwenden.
## (qt5-default existiert nicht mehr in den neuesten Versionen von Linux
    und wird wohl auch nicht mehr benötigt.)
sudo apt install build-essential qt5-qmake qtdeclarative5-dev qttools5-
    dev-tools libjack-jackd2-dev qjackctl -y

# Lösche bereits vorhandene Jamulus Source-Codes, falls diese existieren
sudo rm -f /home/pi/latest.tar.gz

# Herunterladen des aktuellen Jamulus Source-Codes von Github (In den
    Home-Ordner des Nutzers)
sudo wget https://github.com/jamulussoftware/jamulus/archive/latest.tar.
    gz -P /home/pi/

# Benötigte Berechtigungen, die Jamulus-Sourcedatei auszuführen, erteilen
sudo chmod 755 /home/pi/latest.tar.gz

# Lösche bereits vorhandenes Jamulus Code Verzeichnis, falls dieses
    existiert
sudo rm -rf /home/pi/jamulus-latest

# Entpacken des Jamulus Codes
sudo tar -xvf /home/pi/latest.tar.gz -C /home/pi/

# Benötigte Berechtigungen, die Jamulus-Source auszuführen, erteilen
```

```
37 sudo chmod -R 755 /home/pi/jamulus-latest
38
39 # Öffne den Ordner mit den entpackten Code-Dateien
40 cd /home/pi/jamulus-latest
41
42 # Installation des Jamulus Standard Desktop Builds
43 make distclean
44 qmake
45 make
46 sudo make install
47
48 # Benötigte Berechtigungen, die Jamulus-Source auszuführen, auf das
     Minimum zurücksetzen
49 sudo chmod -R 755 /home/pi/
50
51 # Springe zurück zum Home-Verzeichnis des aktuellen Nutzers
52 cd ~/
53
54
55 ##########################################
56 ### INSTALLATION DES TEAMVIEWER CLIENTS ###
57 ##########################################
58
59 # Teamviewer von der offiziellen Website herunterladen (abhängig davon,
     ob ein 32-bit oder 64-bit System vorliegt)
60 if [ $(sudo getconf LONG_BIT) == "64" ]
61 then
62   wget https://download.teamviewer.com/download/linux/teamviewer_arm64.
     deb -O /home/pi/teamviewer.deb
63 else
64   wget https://download.teamviewer.com/download/linux/teamviewer_armhf.
     deb -O /home/pi/teamviewer.deb
65 fi
66
67 # Entpacken des heruntergeladenen Teamviewer Pakets
68 sudo dpkg -i /home/pi/teamviewer.deb
69
70 # Installation des entpackten Teamviewer Pakets
71 sudo apt install -f
```

A.3 Vollständiger Schaltplan der Messplatine



A.4 Bekannte Fehler der Messplatine

- Vin: Statt 7...12 VDC nur 7...7,5 VDC
- Beschriftung des Schalters SW1 vertauscht (detect/burst)
- Fehlende Markierung der Polarität von Elektrolytkondensatoren -> Erkennung der Kathode an Verbindung zur Ground-Plane
- Keine funktionsbezogene Bezeichnung der Testpunkte -> Schaltplan (A.3)
- Fehlender, ungenutzter Pin am Display
- Footprints aller DIP-Gehäuse zu breit -> ICs vorsichtig zurecht biegen
- V+ und V- an U10 (Sinus-Generator) vertauscht -> mit Litze anlöten
- D5: Falsche Polarität
- Vertauschte Eingänge am Clipping-Komparator -> LED drehen und gegen +5V anlöten
- Ggfs. notwendiger Austausch der Widerstände zur Anpassung der Helligkeit der LEDs
- MOSI und MISO an Display vertauscht -> Pins an Arduino brücken (nur MOSI benötigt)