Halvor Schjelderup

Amp over IP

Analog fjernprosessering av audio over IP-nettverk

Masteroppgave i Musikkteknologi
Veileder: Sigurd Saue, Trond Engum

Januar 2019

Masteroppgave

NTNU
Norges teknisk-naturvitenskapelige universitet
Det humanistiske fakultet
Institutt for musikk

NTNU
Kunnskap for en bedre verden

Halvor Schjelderup

Amp over IP

Analog fjernprosessering av audio over IP-nettverk

Masteroppgave i Musikkteknologi
Veileder: Sigurd Saue, Trond Engum
Januar 2019

Norges teknisk-naturvitenskapelige universitet
Det humanistiske fakultet
Institutt for musikk

NTNU
Kunnskap for en bedre verden

Sammendrag

Dette masterprosjektet har hatt som mål å lage en gitarforsterker som kan brukes i sanntid og fjernstyres over internett fra bunnen av. Med utgangspunkt i et rammeverk for C++ kalt Juce og mikrokontrolleren Bela har jeg klart å oppnå et fungerende resultat. Fra brukerens perspektiv er alt som trengs for å spille gjennom denne forsterkeren en datamaskin, et lydkort, monitoreringsmuligheter og internettilgang, i tillegg til en gitar og en jack-kabel. Denne teknologien muliggjør blant annet at musikkutstyrsentusiaster kan dele, oppleve og oppdage nye utstyrsenheter og lydestetiske uttrykk. Videre kan slik teknologi bidra som en helhetlig generell løsning for sanntids analog fjernprosessering med anvendelsesområder i flere konvensjonelle musiseringsarenaer.

Rapporten søker å kartlegge de ulike utfordringene man kan støte på i utviklingen av et slikt system, som for eksempel latencyproblematikk, sanntidskrav og fysisk fjernmanipulasjon av parametere på en ekstern enhet. Videre diskuteres fjernstyring i et designteoretisk lys, og hvorvidt det er likhetstrekk mellom forholdene rundt bruk av digitale replikasjoner av analogt utstyr og virtuell tilgang til de faktiske enhetene.

Abstract

The aim of this thesis has been to build a guitar amplifier which can be remotely controlled and played in real time over the internet from scratch. I set out on this task equipped with the framework Juce for C++ and the micro controller unit Bela. The project has resulted in a working implementation. From the user's perspective, all that is needed to play through this amplifier remotely is a computer, a sound interface, a monitoring solution, an internet connection and the ability to type in the correct IP Address and port number of the amplifier unit, with the addition of a guitar and a jack cable. This entails the possibility for gear enthusiasts all over the world to share, experience and discover new equipment and possibilities for musical expression. Furthermore, this technology can be seen as a general real time remote processing service with usages within conventional arenas for music making.

The following report seeks to address the different challenges one can face in the development process of similar systems, e.g. latency, real time requirements and the remote physical manipulation of parameters on a geographically separated unit. Over the following pages I will discuss remote control from a design theoretical view, if there are similarities between the use of digital emulations of analogue hardware and virtual access to the unit itself.

Forord

Det er med både vemod og lettelse jeg nå leverer dette masterprosjektet. Vemodig fordi prosjektet for meg representerer en svært lærerik periode i livet mitt og en lettelse fordi jeg føler jeg nå har nådd en viktig milepæl etter en svært arbeidsintensiv tid. Selv med «nordnorsk blod» i årene og sans for kraftuttrykk synes jeg det er vanskelig å finne en formulering som overdriver den innsatsen prosjektet har krevd. Resultatet av innsatsen er heldigvis en følelse av tilfredshet – jeg har skapt noe som virker. Dryppene med gode erfaringer og aha-opplevelser underveis har kommet som perler på en snor.

Musikkteknologi ved NTNU har vært det riktige studievalget for meg. Ved siden av å være et studieprogram som tilbyr en stor kompetansebredde innenfor musikkteknologifaget og flotte fasiliteter, er det særlig et aspekt som jeg har lyst til å dra frem i forordet; det entusiastiske miljøet som eksisterer der og aktivt opprettholdes av engasjerte ansatte og studenter. I et slikt miljø er det enkelt å finne inspirasjon til å gå mer i dybden på deler av faget man ikke nødvendigvis ville turt å begi seg på innenfor tids- og ressursrammene man har som student. Den første takken rettes dit.

Dernest ønsker jeg å takke veilederne jeg har hatt i masterprosjektet, Sigurd Saue og Trond Engum. Tilbakemeldingene og rådene jeg har fått har vært uvurderlige.

Jeg vil også takke mine foreldre som har støttet meg gjennom studiene, og en ekstra stor takk til faren min som har bidratt masse under jobbingen med dette masterprosjektet.

Til slutt ønsker jeg å takke min kjæreste som har vært en god støttespiller gjennom de verste kodeknutene og bidratt til å bedre restitusjonen mellom arbeidsøktene med sitt vennlige vesen og talent på kjøkkenet.

Innholdsfortegnelse

Del 1: IP-basert fjernprosessering av audio, problemstilling og grunnlag

1.1 Innledning

Dette masterprosjektet har hatt som mål å lage en gitarforsterker som kan brukes i sanntid og fjernstyres over internett. Gitarforsterkeren, i tillegg til nødvendig programvare for lydoverføring og fjernstyring, skal lages fra scratch. De siste årene har Audio-Over-IP-teknologi (AoIP) blitt mer tilgjengelig, og i økende grad blitt integrert i flere konvensjonelle arenaer for musisering. Eksisterende AoIP-løsninger kan sies å være tilpasset og formet av behovene i disse arenaene. Jeg finner det derfor naturlig å utforske hvorvidt AoIP-teknologi kan implementeres direkte inn i utstyrsenhetene vi som musikere bruker. For å finne ut dette har jeg valgt å ta for meg mikrokontrolleren Bela, et rammeverk for programmeringsspråket C++ kalt Juce, et gitarforsterkerbyggesett og en hjemmestudiorigg. Arbeidet med prosjektet har resultert i en fungerende implementasjon. Vedlegg 1 og 2 viser filmopptak av systemet i aksjon under testing, og det kan være til fordel å se disse før resten av rapporten leses.

Som problemstillinger for prosjektet har jeg valgt ut følgende:

1. Hvilke tekniske utfordringer oppstår under utviklingen av en gitarforsterker som kan fjernstyres og brukes over nett?
2. Hvordan kan en Audio-over-IP-applikasjon bygges fra bunnen av ved hjelp av åpne digitale utviklingsverktøy?
3. Er rammeverket Juce og mikrokontrolleren Bela egnede verktøy for denne oppgaven?

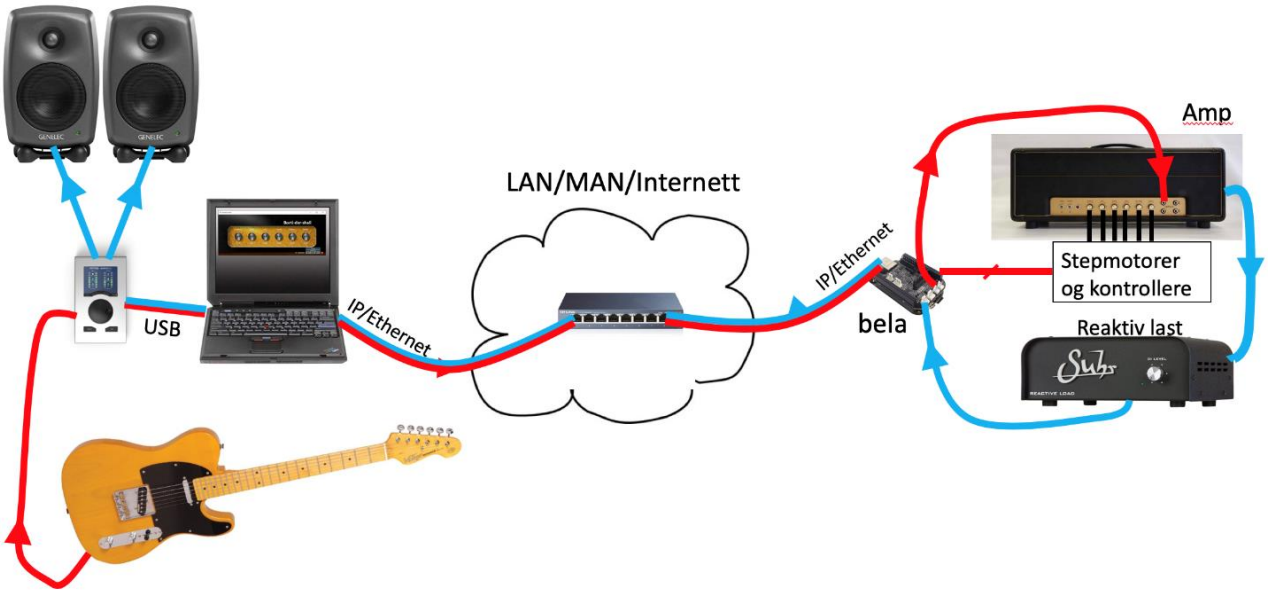
Videre har jeg satt meg noen mål for det praktiske arbeidet. Disse målene er:

1. Det skal ikke være nødvendig for brukeren å ha annet utstyr enn hva man kan finne i et typisk hjemmestudio. Det vil si: datamaskin, lydkort og høyttalere (pluss gitar og jack-kabel).
2. Kompetansegulvet for bruk skal ikke gå utover det å justere inn en gitarforsterker til ønsket lydlig resultat, bruk av enkle, digitale lydprosesseringsverktøy (av typen plugin for DAW-er) og å skrive inn korrekt IP-adresse og portnummer for enheten.
3. Det grafiske brukergrensesnittet skal være oversiktlig og til en viss grad gjenspeile hvordan den faktiske gitarforsterkeren ser ut.
4. En gitarforsterker brukes til vanlig i kombinasjon med et høyttalerkabinett, noe som kan være veldig høylytt. Av sikkerhetsmessige grunner bør forsterkeren kunne innrettes slik at den ikke er i stand til å forårsake en hørselskade der den er fysisk plassert under bruk.
5. Teknologien som muliggjør fjernstyringen og bruken skal helst bygges inn i forsterkeren for å minimere tid på oppsett.

Med denne rapporten ønsker jeg å redegjøre for flere ting. Det første er å opplyse om tenkte anvendelsesområder for gitarforsterkeren jeg har bygd. Dette skal sees opp mot hvilke løsninger som allerede eksisterer i dag og deres anvendelsesområder, samt to beslektede prosjekter. Rapportens mest omfattende del søker å sette lys på de generelle utfordringene man gjerne støter på i implementasjonen av AoIP for sanntidssystemer. Jeg har valgt å la dette ta stor plass i rapporten av to grunner; overføringsverdi til andre beslektede anvendelser og fordi jeg i det praktiske arbeidet støtte på disse selv. Videre har jeg drøftet gitarforsterkeren i et designteoretisk lys med utgangspunkt i begrepene Donald Norman har domestisert og benyttet innen designfeltet, noe som er relevant i utviklingen av grafiske brukergrensesnitt – der teknologien møter brukeren. Rapporten inneholder også en beskrivelse og gjennomgang av teknologien som muliggjør dette, som eksempelvis internett, fjernstyring og mikrokontrollere.

I første del av rapporten har jeg lagt vekt på prosjektobjektets relevans innen musikkpraksis, både i kommersiell og akademisk sammenheng, og har derfor innledningsvis valgt å gi en kort systembeskrivelse fremfor å hoppe rett på teorien. Inspirasjon og idéen til prosjektet kom først og fremst av tanken på dets praktiske anvendelsesmuligheter.

1.2 Overordnet systemoppsett



Figur 1.1: Systemoppsett. Viser alle fysiske komponenter som inngår i systemet. Signalgang er angitt med piler. Fargen på pilene, rød og blå, angir henholdsvis om signalet er på vei til gitarforsterkeren eller på vei tilbake. Figuren er en kompilasjon med bilder fra Fender, RME, Genelec, Marshallforum, Suhr og Bela.

Figur 1.1 viser systemoppsettet. Hardware som inngår i systemet under bruk er gitar, lydkort, monitorer, datamaskin, mikrokontroller (Bela), gitarforsterker, reaktiv last og nødvendige kabler. Lydoverføring, fjernstyring og tilgjengeliggjøring av parametere gjennom grafisk brukergrensesnitt muliggjøres av over 4000 egenutviklede kodelinjer i C++, fordelt mellom fjernstyringsprogram utviklet i Juce og mikrokontrolleren. Koden beskrives på overordnet nivå i kapittel 2.4, og kan utleveres ved forespørsel.

1.3 Rammer for gjennomføring av prosjektet

Implementasjonen av et system slik som i dette prosjektet er nokså omfattende og krever at utvikler behersker flere områder innen programmering og bruk og design av elektromekaniske innretninger. De viktigste utfordringene er fremlagt i kapittel 2.5. Rammene for prosjektet er derfor satt til hva veilederne i prosjektet og jeg har ansett som gjennomførbart innen tidsrammen. Dette masterprosjektet er dermed et grunnlag for videre arbeid. Derfor ønsker jeg nedenfor å gi en pekepinn hvordan jeg ville anvendt ytterlige tidsmessige ressurser i masterperioden. Disse forslagene vil settes opp mot det arbeidet jeg tidsmessig har hatt mulighet til å gjennomføre.

Prototypen på et produkt, som dette masterprosjektet har resultert i, er et svært interessant objekt for utprøving i en rekke settinger der kravene for reliabilitet og robusthet gir en indikasjon på systemets anvendelsespotensial. En slik setting kan eksempelvis være en studioproduksjon, en konsert eller en øvingsøkt med et band. Det kunne også vært interessant og relevant å prøve ut systemet gjennom en rekke brukstester med forsterker og bruker plassert langt unna hverandre geografisk. Slike tester har dog ikke vært mulig å sette opp innen tidsrammen, men er en naturlig fortsettelse av utprøvingen. Utprøvingen som er foretatt er en vurdering av systemdelenes hovedkomponenter (mikrokontroller, rammeverk og gitarforsterker) i en utviklingsprosess for prosjekter av denne typen, samt en vurdering av opplevelsen av systemet i bruk, uformelt i hjemmestudioet med bruker og forsterker i samme lokalnett.

Videre kunne en utredning/analyse om ønsker og behov hos eventuelle brukere vært interessant. Dette for å se på prosjektet i et markedsperspektiv – særlig som forløper til en eventuell kommersialiserings-/ eller tilgjengeliggjøringsprosess av systemet. Som en forberedelse på noe slikt har jeg i rapporten lagt fram en del forhold rundt praksis hos musikere og studioteknikere som argumenter for at behovet kan tenkes å enten finnes eller oppstå på ett eller annet tidspunkt. Dette alene resulterer dog kun i en antakelse, og kan ikke svare på hvorvidt slik teknologi vil bli omfavnet. Domestisering er betinget på en rekke forhold, både av kulturell og teknologisk natur (Aalen, 2015).

1.4 Prosjektets relevans i lys av generell lydstudiopraksis

I min rolle som studiotekniker søker jeg ofte å anvende ulike teknologier for å oppnå et tenkt lydlig resultat. Disse teknologiene er alt fra instrumenter til mikrofoner og prosesseringsteknologi i både analogt og digitalt domene. Mange av disse teknologiene er oppfunnet på ulike tidspunkt, og lar seg ofte sammenkoble og benyttes til å gjøre en produksjon (Katz, 2010). Teknologiens muligheter og begrensninger åpner for dannelsen av nye teknikker innen generering og prosessering av lydlig materiale, hvorav noen av disse teknikkene blir standard innen studiopraksis (Emerick, 2013). På grunn av samspillet mellom teknologien og brukerne kan rollen som studiotekniker og idéen om hva et studio er derfor sies å være i en konstant metamorfose.

I lys av studiopraksis kan prosjektet mitt ses på som teknologi som gir en ny type tilgang til prosesseringsutstyr. Ettersom en studioproduksjon kan kreve mange prosesseringsenheter og instrumenter for å realisere musikkprosjektet har dermed store lydproduksjoner historisk sett, fra slutten av 1800-tallet til relativt nylig, dels vært forbehold komponister, musikere og teknikere med tilgang til institusjoner med dedikerte lokaler for blant annet oppbevaring av utstyr samt deres bruks- og vedlikeholdskompetanse. Grunnet delingen av ressurser og høy bemanning av teknisk personell krevde også disse institusjonene flere ansatte som hadde rent administrative roller (Emerick, 2013; Katz, 2010). Gradvis og frem til i dag har studioet vært i forandring grunnet teknologiske nyvinninger. Et av de virkelig store paradigmeskiftene innen studiopraksis er utviklingen av teknologi som muliggjorde å gjennomføre en studioproduksjon på en hjemmedatamaskin.

Emericks bok "*Here, There And Everywhere"*, referert ovenfor, gir et tydelig bilde på hvor mye ressurser som måtte til for å realisere forskjellige Beatles-produksjoner på 60-tallet han var deltakende i. I kontrast vil jeg dra fram en kjent, norsk artist som etablerte seg for få år siden. I perioden desember 2015 til oktober 2016 ble Alan Walkers låt «*Faded*» spilt 700 millioner ganger på tjenesten Youtube, og låten var produsert på en soverom i Fana, Bergen (Valderhaug, 2016). Hjemmestudioet som musikkproduksjonsarena er realisert av blant annet ny teknologi som tillater ressurser for generering, prosessering og strukturering (gjennom DAW-er) av lydlig materiale å eksistere internt i en datamaskin, fremfor i form av dedikerte og separate enheter. Prosjektet mitt, sett opp mot teknologien som muliggjør hjemmestudioet, gir tilgang til en bestemt dedikert prosesseringsenhet, en gitarforsterker, og krever kun tilgang til en hjemmestudiorigg og lokalnett/internettilkobling.

1.5 Motivasjon

Musikkteknologifagfeltet er svært bredt, noe som gjenspeiler seg i mangfoldet av fag som inngår i studieprogrammet både på bachelor- og masternivå. Dette mangfoldet strekker seg også til mulighetene studentene har for hva de kan gjøre i masterprosjektet sitt. For å unngå å la meg hemme av valgmulighetene valgte jeg å sikte meg inn på noe som omfatter verktøyet jeg bruker mest i det daglige; den elektriske gitaren.

Motivasjonen for å ta for meg akkurat dette prosjektet, å bygge en Marshall-forsterker som kan fjernstyres og brukes over nett, er todelt: både personlig og rent faglig. Jeg velger å begynne med det personlige.

Musikkinteressen min begynte for alvor, passende nok, da jeg hørte en Marshall-forsterker i bruk på Guns n' Roses-plata Appetite For Destruction (utgitt 1987), og hovedinstrumentet mitt gikk fra å være tuba til gitar det påfølgende året. Etter en stund overtok Jimi Hendrix plass som største gitarforbilde, som også ofte benyttet Marshall-forsterkere. Lydlandskapene som males på Electric Ladyland-plata (utgitt 1968) står fremdeles veldig høyt for meg den dag i dag og inspirerer meg både som gitarist og studiotekniker. Det å etterstrebe oppnåelse av en tilsvarende musikalsk kompetanse som heltene sine kan være nært umulig, men med dette prosjektet får jeg kanskje muligheten til å ha gjort noe til felles med dem; å skape noe skikkelig kult med en Marshall-forsterker.

Rent faglig, har jeg gjennom mine fem år på musikkteknologi plukket ut valgfag innen IT som virket relevante innenfor musikkteknologisk praksis. Blant disse har særlig fagene Kommunikasjon – Tjenester og Nett, Prosedyre- og Objektorientert Programmering, Datamaskiner og Digitalteknikk, og Algoritmer og Datastrukturer vært de som jeg har hatt mest nytte av. Min musikkbakgrunn gav naturligvis et ønske om å kunne bruke kompetansen disse fagene gav i en musikksammenheng, noe dette prosjektet mer eller mindre er et resultat av.

Som nevnt ovenfor har jeg også en stor interesse for studioarbeid. En teknikers jobb i studio kan beskrives på mange måter, men om jeg skulle gi en beskrivelse selv ville det vært anvendelse av ulike teknologier med den hensikt å skape noe nytt. For meg er det derfor ikke nødvendigvis et klart skille mellom studiorelatert arbeid og teknologisk utviklingsarbeid.

Det virker passende for meg å avslutte fem års studier med et prosjekt der jeg får benyttet kompetansen fra de ulike fagdisiplinene i studietiden til å skape noe nytt. Tidlig i prosjektplanleggingen skjønte jeg at dette var et dristig prosjekt med høy risiko for å ikke lykkes grunnet omfanget og ukjente tekniske hinder og overraskelser man kunne støte på. Det er ikke å legge skjul på at prosjektet innebar en stor, men også lærerik dose slike tilfeller.

1.6 Bruksområder for systemet

Audio-over-IP er allerede i bruk innen mange områder. En av kjente de anvendelsene er såkalt IP-telefoni der eksempelvis Skype (Microsoft), FaceTime (Facebook) og Discord benyttes av mange. Innen musikk er sanntids lydoverføring blitt benyttet til blant annet samspill over avstand både i konserter og studioinnspillinger, samt til strømming av konserter.

Systemet som jeg har laget er en toveis én-til-én forbindelse mellom bruker og gitarforsterker som krever en datamaskin, et lydkort og internettilgang for å kunne brukes. Systemet er ment for å gi brukeren fleksibilitet for en rekke anvendelsesområder. Nedenfor drøftes 5 tenkte scenarier der systemet kan tenkes integrert.

1) Studioproduksjon. Ettersom utstyrskravet ovenfor som regel er dekket internt i et vanlig lydstudio er dette et naturlig område å bruke systemet i. Gjennom analog eller virtuell ruting av lyd (for eksempel gjennom SoundFlower eller Jack) kan en DAW kobles opp mot fjernstyringssystemet. Systemet er dermed like godt egnet i et stort dedikert studiolokale som et lite hjemmestudio.
2) Asynkront gjennom re-amping. Re-amping er en teknikk der eksisterende råmateriale blir prosessert gjennom prosesseringsenheter, som eksempelvis gitarforsterkere, i etterkant av innspillingen. Slik som ovenfor kan ruting av lyd benyttes for å koble fjernstyringssystemet sammen med opptaksprogramvare. Eventuelt kan den utviklede programvaren for fjernstyringssystemet og lydoverføringen tilpasses og kompileres som plugin (VST, AU, AAX).
3) Konsert. Gitt at konsertlokalet har nettverkstilkobling vil systemet kunne tas i bruk her. Det plassmessige fotavtrykket til en laptop og et lydkort er i noen tilfeller lavere enn i et konvensjonell gitarforsterkeroppsett med mikrofon.
4) Egenøving. Brukeren kan benytte en ekte gitarforsterker til egenøving dersom utstyrskravet for systemet dekkes. Muligheten for monitorering gjennom høretelefoner åpner også for å holde lydnivået i området rundt svært lavt.
5) Utleie. Forsterkeren trenger ikke nødvendigvis å være eid av brukeren. Dette gjør at systemet både kan tas i bruk av et utleieselskap eller være del av en delingsøkonomi eller som en nettjeneste.

For sanntidsanvendelser er det i de fleste sammenhenger viktig at den totale tidsforsinkelsen i ende-til-ende-kommunikasjonen er tilstrekkelig lav for musisering (Jack, Stockman, & McPherson, 2016).

1.7 Audio-over-IP: Hvilke løsninger tilbys i dag og egne erfaringer

I dette kapittelet ønsker jeg å gi en rask oversikt over to løsninger som er tilgjengelige innen AoIP. Blant eksemplene jeg drar fram finnes både kommersielle løsninger og åpen kildekode som kan brukes i sanntids lydoverføring.

Dante er en kommersiell løsning for AoIP utviklet av Audinate. Dante tilbyr overføring av opp til 512 høyoppløselige lydkanaler, og krever dedikert hardware for å kunne brukes. Den spesialiserte Dante-hardwaren kan i kombinasjon med et lokalnett av høy kvalitet levere svært lav latency, ned til under et millisekund i optimale omgivelser. Løsningen er beregnet for alt fra lydstudio og konsertlokaler til konferansesentre. Oppsett og ruting gjøres gjennom et brukergrensesnitt som leveres med hardware-en (Audinate, 2019).

Jacktrip er et resultatet av forskningsarbeid ved Stanford University. I motsetning til Dante er Jacktrip åpen kildekode, som vil si at hvem som helst kan hente og endre programkoden. For at Jacktrip skal kjøre kreves at programmet JACK er installert, som brukes til virtuell ruting av lyd. Under bruk krever Jacktrip oppsett i tilhørende grafisk brukergrensesnitt og i kommandovinduet i operativsystemet. Brukeren trenger ikke dedikert hardware for å bruke Jacktrip, og grunnet dens mulighet for flere kanaler gjør den også egnet for de samme anvendelsesområdene som Dante. I dag kjører Jacktrip kun på Mac og noen Linux-distribusjoner, og fantes tidligere for Windows (Jacktrip, 2019).

Begge disse fungerer utmerket under bruk, men krever mye trening for å foreta et vellykket oppsett på egenhånd. Kompleksiteten i oppsett skyldes dels hvor fleksible de er – full kontroll over samplingsrate, kanalmengde og gode rutingmuligheter. Kompleksiteten i oppsett gjør at jeg, personlig, noen ganger søker å unngå å ta det i bruk om analog ruting er en mulighet. Dante ble benyttet under mitt arbeid med bachelorprosjektet, da til lydoverføring parallelt med et separat fjernstyringssystem. Lokalnettet som ble brukt i utprøving kunne kjøre disse systemene hver for seg helt fint. Dog oppsto ofte konnektivitetsproblemer med en gang de skulle kjøre sammen. Hypotesen om årsak den gang var at Dante sin bruk av multicast gjorde at fjernstyringssystemet også mottok massive mengder AoIP-pakker og dermed gikk i metning. Dette førte til at fjernstyringssystemet tidvis ikke fikk kontakt med kontrollersystemet. Selv om det kanskje fantes en løsning på kompatibilitetsproblematikken, anser jeg muligheten for at de kan oppstå i det hele tatt som en klar indikasjon på at AoIP- og fjernstyringsteknologi i noen tilfeller har godt av å implementeres og testes sammen, til et felles formål.

1.8 Beslektede prosjekter, sanntids analog fjernprosessering

Under arbeidet med dette prosjektet har det vært to pågående og beslektede eksterne prosjekter som jeg ønsker å dra frem. Disse er «Stompenberg FX», en avdeling hos musikkutstyrsforhandlingen Thomann, og Ph.d.-prosjektet «Innovative Music Production Approaches Empowered by Internet of Things Technologies».

Stompenberg FX som sanntids fjernprosesseringsprosjekt ble åpnet for utprøving i overgangen til 2019. Prosjektet gir brukere fra hele verden mulighet til å prøve ut gitareffektpedaler i sanntid over nett gjennom AoIP-teknologi. Dette kan gjøres med tre ulike moduser: sanntids lydoverføring frem og tilbake, opptak etterfulgt av lydoverføring tilbake og forhåndsinnspilte snutter med gitarspill. Tanken er at gjennom å rute lyden ut til egen gitarforsterker, kan brukerne høre pedalen i bruk sammen med sin egen rigg. Jeg har testet prosjektet selv og er imponert over brukervennligheten, men det virker til å ha problemer med dropouts og høy latency som kan være resultat av avstand og variabel nettkvalitet. Prosjektet ser ut til å ha en funksjon som en markedsføring av pedalene som kan prøves, da man enkelt kan navigere seg frem til kjøpssiden hos Thomann (Stompenberg, 2019).

Ph.d.-prosjektet «Innovative Music Production Approaches Empowered by Internet of Things Technologies» utføres av Marques J. Hardin under veiledning av Rob Toulson. Prosjektet tar for seg en rekke prosesseringsressurser, deriblant akustisk romklang, synthesizer med analog input, en analog equalizer som alle kan brukes og styres over nett. For overføring av lyd og kontrollsignal tar prosjektet utgangspunkt i henholdsvis JackTrip og WebRTC. For fysisk manipulasjon benyttes blant annet beltedrift av potentiometre ved hjelp av motorer (Hardin, 2019).

Disse to prosjektene tyder på at det finnes interesse for slik teknologi både innen den kommersielle og den akademiske sfæren. Utenom disse eksemplene er det liten synlighet av konkrete og pågående prosjekter som rører ved idéen om at AoIP-teknologi kan implementeres direkte inn i prosesseringsutstyr (som for eksempel gitarforsterkere, ekkomaskiner osv.).

1.9 Fjernstyring

Cambridge Dictionary (2019) definerer fjernstyring (remote control) på denne måten: «*a system for controlling something such as a machine or vehicle from a distance, by using electrical or radio signals*». Fjernstyring kan ut fra dette tolkes å dekke et bredt spekter av ulike systemer. En elegant side av denne definisjonen er at den viser til tre distinkte deler av fjernstyringsteknologi. De tre delene er som følger:

1. Fjernstyringssystem, eller kontroller
2. Medium
3. Apparat eller maskin som blir manipulert

For å eksemplifisere disse delene ønsker jeg å ta i bruk et tenkt moderne fjernsynsapparat. Kontrolleren vil her være fjernkontrolleren. Fjernkontrolleren tilbyr de viktigste funksjonene til fjernsynsapparatet, som en av/på-knapp, kanalskifte, lydtrykksnivåkontroll osv. Apparatet som blir manipulert her er fjernsynsapparatet. Denne manipulasjonen over avstand muliggjøres, og begrenses, av medieteknologien som her ofte er infrarødt lys eller radiosignaler.

Systemet jeg har utviklet i oppgaven kan også deles inn i disse tre delene. Det grafiske brukergrensesnittet fungerer som kontroller og kan gjennom nettverksteknologi koble seg opp mot internett, som er mediet. Forsterkeren er apparatet som blir manipulert. Her regnes mikrokontrolleren naturligvis som del av forsterkeren. Spesielt for dette tilfellet er at mikrokontrolleren har funksjonalitet for å overføre lyd begge veier.

1.10 Hvorfor fjernstyre musikkutstyr?

Det å legge til et fjernstyringssystem i eksisterende teknologi uten å erstatte noen originale komponenter vil naturligvis øke mengden ting som kan gå i stykker, samt øke produksjonskostnaden. Dette er også tilfelle med forsterkeren jeg bygger i prosjektet. Likevel finnes en rekke behov som rettferdiggjør en slik utvidelse av forsterkerens funksjonalitet.

I et lydstudio har man mulighet til å jobbe med mange ulike innspillingsprosjekter, som hver kan stille ulike utstyrskrav for å oppnå sin ønskede lydestetikk. Utstyrets rolle i det estetiske arbeidet er å transformere et lydforløp fra et lydestetisk uttrykk til et annet etter brukerens ønske. Dette kan være alt fra kompresjon til etterklang, ekko, forvrengning eller en form for modulasjon, for eksempel chorus-effekt. I lang tid krevde bruken av disse effektene umiddelbar fysisk tilgang til analoge kretser og tilstrekkelige lokaliteter for å realisere denne transformasjonen (Emerick, 2013). I dag finnes digitale varianter av mange populære utstyrsenheter tilgjengelig i form av eksempelvis plugins og digitale multieffektenheter, da også i tillegg til prosesseringsteknologi som utelukkende finnes digitalt. Software-produsentene Waves og Slate Digital leverer mange digitale etterlikninger. Hvorvidt det er det digitale eller analoge/akustiske som «er best» debatteres støtt og stadig i forskjellige fora (eksempelvis på www.gearslutz.com). Jeg skal ikke påta meg rollen som oppmann i debatten, men ønsker å dra fram noen paralleller mellom fordelene ved digitale gjengivelser av analogt prosesseringsutstyr . Disse kan også gjenspeile behovet for fjernstyring og bruk over nett.

1) Analogt musikkutstyr tar plass. Som eksempel krever gitarforsterkeren jeg bygger et høyttalerkabinett for konvensjonell bruk. Når gitarforsterkeren kombineres med kabinettet kan den ikke lengre stables i høyden og har et plassmessig fotavtrykk på 33 cm * 77 cm - altså omtrent en kvart kvadratmeter og høyde på over en meter. Dette vil dog kun være oppbevaringsplass, da den i bruk vil kreve et mye større areal for å ikke få faserelaterte problemer i lydutbredelsen.
2) Utstyr som for eksempel bass- og gitarforsterkere må ofte utstyres veldig høyt for å gi ønsket lydkarakter. Dette er ikke bestandig forenelig med at mennesker oppholder seg i umiddelbar nærhet.
3) Analogt utstyr kan være dyrt. Her ønsker jeg å ta utgangspunkt i min yndlingskompressor: Teletronix LA2A. Ett eksemplar av den analoge enheten har en listepris på 3799 dollar. I sanntidsbruk kan en slik kompressor dekke 1 kanal om gangen. En av de dyrere digitale variantene som leveres av Universal Audio, har i forhold en listepris på 299 dollar og leveres som pakke med tre modeller som hver kan brukes så mange ganger man vil i sanntid på en gang.
4) Analogt utstyr må vedlikeholdes. Dersom eieren av utstyrsenheten ikke har kompetanse, nødvendige verktøy og måleinstrumenter, samt lokalene for å foreta vedlikeholdsarbeidet på egen hånd må dette overlates til en ekstern, egnet tekniker. Verdt å merke seg er at tilstanden til en

utstyrsenhet ikke nødvendigvis er binær, i form av enten fungerende eller ikke, men kan også bestå av en gradvis prosess der enkeltkomponenter forringer over tid. Typiske eksempler på slike komponenter er rør og kondensatorer.

De fire overnevnte punktene betyr ikke at digitale emuleringer nødvendigvis er overlegne. Blant områdene der de kan komme til kort kan det nevnes at de digitale verktøyene kun eksisterer virtuelt på en datamaskin – skulle datamaskinen av hvilken som helst grunn ikke kunne kjøre dem, av eksempelvis driver/systeminkompatibilitet (for eksempel som følge av operativsystemoppdatering) eller feil i hardware, kan ingen av de digitale verktøyene brukes før systemoppsettet er satt i stand igjen. Videre kan det nevnes at ikke alle lydlige fenomener som oppstår under bruk av enkelte analoge enheter lar seg gjenskape eksakt i det digitale domenet uten enorme mengder prosesseringsressurser. Mange brukere kombinerer imidlertid analog og digital teknologi etter tilgjengelighet, ønsker og behov.

Her mener jeg det finnes en klar analogi, hva gjelder argumenter for digitalt distribuert fjernprosessering. Gjennom eksempelvis en lav tilgangskostnad, vil disse fire punktene møtes. Brukeren av fjernprosesseringstjenesten vil slippe å ofre areal/volumplass for bruk av den aktuelle enheten, en høy økonomisk barriere for tilgang, og eieren av den fysiske enheten kan få vedlikeholdet finansiert i bytte mot utleie.

1.11 Hvorfor ta utgangspunkt i en gitarforsterker?

Det at valget av prosjektobjekt er gitarforsterker er naturligvis farget av min rolle som gitarist - gitarforsterkere er av interesse for meg i utgangspunktet. Likevel føler jeg gitarforsterkeren representerer noe som står sentralt i dette prosjektet: det å «flytte lydkilden». Elektriske gitarer finnes i mange varianter, noen der slektskapet til den akustiske gitaren kommer godt fram. Likheten mellom instrumentene kan gjenspeile seg i utforming, antall strenger og hvordan man interagerer med instrumentet for å generere lyd. En av de mest sentrale forskjellene mellom instrumentene er hva som er den endelige lydkilden. Der hvor den akustiske gitaren selv har et hulrom som farger og omvandler den mekaniske energien i strengene til lyd, brukes en elektrisk gitar konvensjonelt i kombinasjon med en gitarforsterker med tilhørende høyttalere. Disse kobles sammen ved hjelp av et overføringsmedium – oftest en ubalansert jack-kabel. Dersom bruken av elektrisk gitar er forbundet med å flytte lydkilden, til gitarforsterkeren, vil det kanskje føles mer naturlig å benytte et system slik som i dette prosjektet – det er bare mediet for overføring og styring som endres. Noen flerkanals gitarforsterkere leveres også med fotbrytere for kanalskifte og annen funksjonalitet (som eksempelvis klang eller signalforsterkning), noe som etter definisjonen gitt tidligere i rapporten vil kunne kalles fjernstyring.

Figur 1.2: Fotbryter til Marshall-forsterker. De tre bryterne til venstre velger kanal, nest til høyre skrur av eller på en klangeffekt og sistnevnte legger en effektsløyfe inn i kretsen. Fotbryteren kan sees på som en form for fjernstyring. Bildet er hentet fra www.Sweetwater.com.

1.12 Fjernstyring og bruk over nett i et designteoretisk lys

Hvordan ting anvendes, og hvordan man designer ting med konkrete anvendelser som formål er også vitenskap. Under prosjektet har jeg sett til Donald Normans arbeid og bruk av designterminologi i boken «The Design of Everyday Things». Relevansen var ikke bare innen utviklingen av det grafiske brukergrensesnittet, men gir også et viktig perspektiv på hva prosjektets resultat kan tilby av funksjonalitet sett opp mot en konvensjonell gitarforsterker.

Det første begrepet jeg ønsker å dra inn er «affordances». Norman (2013) introduserer begrepet slik:

> *«The term affordance refers to the relationship between a physical object and a person (or for that matter, any interacting agent, whether animal or human, or even machines and robots). An affordance is a relationship between the properties of an object and the capabilities of the agent that determine just how the object could possibly be used. A chair affords (is for) support, and therefore, affords sitting. Most chairs can also be carried by a single person (they afford lifting), but some can only be lifted by a strong person or by a team of people. If young or relatively weak people cannot lift a chair, then for these people, the chair does not have that affordance, it does not afford lifting.»*

Affordances er altså en relasjon som beskriver muligheten for situasjonsbetinget samhandling mellom bruker og objekt der egenskapen «affording» tillegges objektet når kriteriene møtes hos begge parter. En viktig nyanse av begrepet er at en affordance ikke nødvendigvis er tiltenkt av skaperen i utgangspunktet. Eksempelvis innen musikkpraksis kan en kopphåndduk benyttes til å dempe høye frekvenser og resonans i trommer (Emerick, 2013), og medisinglass kan benyttes som *slide* til gitar.

Det andre begrepet jeg ønsker å dra fram fra Normans bok er «*signifiers*». Begrepet introduseres i The Design of Everyday Things først gjennom hvordan affordances har blitt brukt til å beskrive en helt annen egenskap ved objekter; hvordan affordances oppdages (Norman, 2013). Norman eksemplifiserer dette gjennom en hypotetisk dialog mellom en designer og dens mentor. I dialogen kommer det fram at designeren har brukt piler i det grafiske brukergrensesnittet til å vise mulighet for sveiping i en mobilapplikasjon. Designeren velger å kalle pilene for en affordance, men mentoren korrigerer med å si:

> «*… you called them affordances even though they afford nothing new: they signify what to do and where to do it. So call them by their right name: signifiers*».

Signifiers og affordances er altså tett forbundet, men beskriver ulike aspekter ved objekter. Distinksjonen mellom disse begrepene kan være noe vrien å få grep på i noen tilfeller. En grei huskeregel er at affordances benyttes om mulighet for interaksjon, mens signifiers benyttes for å beskrive egenskaper ved objektet som viser frem muligheten for interaksjon samt hvordan interaksjonen skal foregå. Videre er ikke en affordance avhengig av en signifier for å eksistere – et tiltenkt bruksområde kan være tilstede uten at brukeren får informasjon om det. En signifier, derimot, skal henvise til en tiltenkt affordance og er dermed avhengig av den for å eksistere. I de neste avsnittene skal jeg gå gjennom noen eksempler på affordances og signifiers på gitarforsterkeren som anvendes i prosjektet.

Figur 1.3: Marshall 1959. Bildet er hentet fra Thomann.

Figuren ovenfor viser den originale utgaven av gitarforsterkermodellen som er bygget i dette prosjektet. Denne gitarforsterkeren kan forbindes med mange ting, som for eksempel merkevaren Marshall eller musikken som er skapt med den. Som verktøy, fremfor et symbol, er den antakelig mest kjent for den lydlige estetikken den skaper i kombinasjon med en gitar og et høyttalerkabinett. En affordance som er tilstede ved gitarforsterkeren vil da være denne lydestetikken, mens dens visuelle fremstilling og utforming fungerer dels som en signifier gjennom forsterkerdesignets gjenkjennelighet.

Frontpanelet på forsterkeren har 6 ratt som kan skrus for å endre det lydlige resultatet av prosesseringen. Innstillingene av rattene er gradert med heltall fra 0 til 10, og en uthevning i rattet peker til disse for å angi posisjonen til rattet i dens dreieområde. Her fungerer både rotasjonen av rattene og den påfølgende lydlige endringen som to ulike affordances. Signifierne her er heltallene rundt rattene i kombinasjon med rattets uthevning, samt rattenes friksjonsgivende, rillete overflate.

Øverst på bildet av gitarforsterkeren ser vi et håndtak. Dette håndtaket tilbyr (affords) et fast grep med en hånd under manuell transport, noe som ville krevd minst én veldig stor hånd foruten håndtaket. Håndtaket kan antas å være lagt til under designprosessen med dette i tankene. Den fysiske utformingen til håndtaket kan også minne om tilsvarende løsninger på andre objekter, som eksempelvis på håndkurver og bøtter, og har dermed også en bifunksjon som signifier utenom dens affordance-aspekt.

I en gitarforsterkeren som kan fjernstyres og brukes over nett kan systemets hensikt beskrives ut fra disse begrepene. Funksjonaliteten i fjernstyringssystemet kan sees på som tilgjengeliggjøring av affordances på et eksternt objekt.

Ettersom gitarforsterkeren ikke nødvendigvis er i samme rom som brukeren er det også noen aspekter som forsvinner. Designet av fjernstyringssystemet dreier seg da til en viss grad om å velge ut hvilke affordances som skal være tilgjengelig, samt sørge for at brukeren er klar over funksjonaliteten gjennom signifiers. De affordancene jeg primært ønsker å bevare i fjernstyringssystemet er:

1) Prosesseringen av råsignal som sendes inn, eller den lydestetiske karakteren til forsterkeren.
2) Tilgang på parameterne på forsterkerens frontpanel, eller justering av ratt.

I tillegg til dette ønsker jeg også å bevare opplevelsen av å spille gjennom en analog gitarforsterker. Hvorvidt dette oppnås vil utelukke avhenge av brukerens subjektive opplevelse. Denne formen for bruk av gitarforsterker er tross alt ikke konvensjonell, og likner heller på hvordan man benytter digitale emuleringer av gitarforsterkere både i hvordan den styres gjennom et grafisk brukergrensesnitt og i signalgang på brukersiden.

1.13 Fjernprosessering

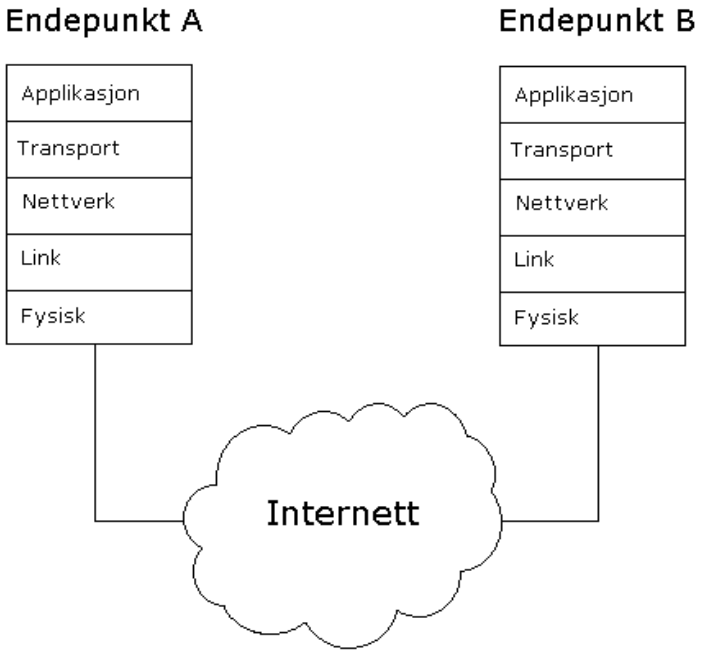
Begrepet fjernprosessering refererer til bruken av eksterne ressurser for prosessering av informasjon. Selv om dette ofte benyttes i anledning digital databehandling velger jeg å ta det i bruk som en nyansering av hva prosjektet kan benyttes til.

Når jeg i rapporten omtaler målet med prosjektet som å bygge en forsterker som kan «fjernstyres og brukes over nett» henviser jeg til relasjonen mellom brukerens handlinger opp mot forsterkeren. Gjennom fjernstyring foretar brukeren en fysisk manipulasjon av rattene på forsterkerens kontrollpanel, og bruken av forsterkeren skjer over et IP-basert nettverk. Fjernprosessering i prosjektets kontekst henviser til gitarforsterkeren som prosesseringsressurs sin relasjon til andre prosesseringsressurser, som for eksempel en lokalt oppbevart gitarpedal.

1.14 IP-baserte nettverk

Internett er et verdensomspennende kommunikasjonsnettverk bestående av sambandslinjer, rutere og endepunkter. Sambandslinjer kan være: fiberoptisk kabel, radio og tvinnede kobberledninger. Nodene og endepunktene som inngår i nettverket benytter seg av en rekke protokoller for å tilby ende-til-endekommunikasjon. Dette kapitlet tar utgangspunkt i boken til Kurose og Ross (2017).

Kommunikasjon som går over internett benytter ofte det vi kaller for TCP/IP-modellen, som er en lagvis struktur som viser til hvordan- og i hvilken rekkefølge informasjon pakkes inn i protokoller og hentes ut mellom nodene og endepunktene i nettet.



Figur 1.4: TCP/IP-modellen med fem lag.

Figuren ovenfor viser TCP/IP-modellen jeg tar utgangspunkt i forklaringen. Her vises to tenkte endepunkter, A og B. Under tabellen finner vi en tabell med de ulike nettverkslagene; applikasjonslaget, transportlaget, nettverkslaget, linklaget og det fysiske laget. For at en applikasjon hos A skal kunne kommunisere med en applikasjon hos B er informasjonen nødt til å pakkes inn i protokoller innen lagene under i rekkefølge før det så sendes ut på nettverket. Videre i kapitlet skal jeg gå gjennom noen kjente protokoller innen hvert lag og forklare hvordan innpakkingen foregår.

Applikasjonslaget ligger øverst i stakken på figur 1.4. Et kjent eksempel på en slik applikasjonsprotokoll er HTTP som benyttes i nettlesere for web-innhold.

Protokollen tilbyr logikk for forespørsler og svar mellom klienter og tjenester. Tilkoblingen over HTTP avsluttes idet en forespørsel er behandlet og godtatt mellom endepunktene, og er derfor egnet til websider med forskjellig mengder trafikk. FTP en annen protokoll på applikasjonslaget og bruker flerkanals kommunikasjon i de underliggende lagene for filoverføring og kontrollsignal. FTP leverer ikke ende-til-endekryptering og brukes derfor ofte i kombinasjon med annen teknologi for å tilby sikker overføring.

Transportlaget i stakken befinner seg mellom applikasjonslaget og nettverkslaget. Når informasjon skal sendes fra applikasjonslaget hos et endepunkt til applikasjonslaget i et annet pakkes det inn i en transportlagspakke. De to mest kjente transportlagsprotokollene er TCP og UDP. Begge disse sørger for at applikasjonslaget vet at pakken er ankommet riktig endepunkt i nettverket, men de er likevel nokså forskjellige og har ulike formål. TCP tilbyr en rekke funksjonaliteter for både endepunktene og nettverket. Blant disse finner man metningskontroll dersom en node i nettet overbelastes, reliabilitet i overføring gjennom retransmisjon av tapte pakker og sekvensnummer på pakkene slik at dataen kan behandles i riktig rekkefølge. TCP «garanterer» derfor overføring på bekostning av varierende overføringsrate og tidsforsinkelse. UDP, på den andre siden, tilbyr kun transmisjon over nettet og har derfor potensiale for å ha et lavere fotavtrykk på nettverkskapasiteten gjennom sin minimalisme. En annen fordel er at UDP også kan utnytte nettverkskapasiteten maksimalt med minimal ressursbruk i endesystemene, men uten garantier mot pakketap.

Nettverkslaget sørger for at alle nodene mellom endepunktene vet hvor pakkene skal videresendes for å nå aktuell destinasjon. Dette gjøres med en IP-adresse og et portnummer. IP-adressen benyttes til å finne frem til lokalnettet der endepunktet befinner seg og portnummeret brukes til å finne prosessen (internt i applikasjonen) som skal motta pakken.

Link- og fysisk lag sørger for at datapakkene overføres til neste node i nettverket, enten fra et endepunkt eller mellom to noder. Kjente linklagsprotokoller er Ethernet og WiFi. For Ethernet fins en rekke ulike fysiske medier (tvinnet par CAT-5 og oppover, ulike varianter optisk fiber osv.).

Informasjon som skal over nettet blir altså først pakket inn med en applikasjonlagsprotokoll før den deretter pakkes inn med en transportlagsprotokoll, så med en nettverksprotokoll før pakken til slutt pakkes inn med en linklagsprotokoll. Det vi sitter igjen med er et datagram, som lagvis er pakket inn for at kommunikasjonen skal være mulig. Blant ulike typer noder i nettverket vil pakken kunne pakkes ut av noen av disse skallene før den på nytt pakkes inn og sendes videre. Når pakken ankommer endepunktet vil den pakkes ut, lag for lag, til applikasjonen sitter igjen med informasjonen den skulle ha.

1.15 Mikrokontrollere

I dag omgis vi av datamaskiner, og datamaskiner dekker et mye brede område enn bærbare- og stasjonære datamaskiner og mobiltelefoner. Etter hvert som produksjonen av enkle chipsett har blitt billigere å masseprodusere har de overtatt mange av oppgavene som tidligere ble løst av enklere kretser. Blant produkter som benytter seg av små datamaskiner finner vi digitale klokker, stekeovner, høyttalersystemer og biler. Disse typene datamaskiner kalles mikrokontrollere. Mikrokontrollere er små generelle datamaskiner, som vil si at gitt en egnet applikasjon og nok tid, vil den kunne løse en oppgave. Mikrokontrollere har ofte et sett med I/O den bruker for å hente informasjon fra omverdenen enten i form av sensorer eller knapper som brukes til å utføre oppgavene den blir gitt. I tillegg til å være billige dersom masseprodusert kan mikrokontrollere ha en annen stor fordel; brukerbasen og ressurskravet til de mest anvendte kommersielle operativsystemene på personlige datamaskiner har økt for blant annet å sikre brukeropplevelsen, og hardwareteknologien har møtt de økte kravene dette stiller. Mikrokontrollere kan lages for å brukes til veldig spesifikke ting, med enkle operativsystemer og operasjonssett (diskrete operasjoner utført av CPU-en) som gjør at de kan utføre jobben effektivt (Tanenbaum, 2016).

Oppgavene en mikrokontroller kan gjennomføre trenger ikke bare være begrenset til å holde styr på hvor mye tid som har gått eller å sjekke om en knapp på en bilnøkkel har blitt trykt inn – de kan også gjøre langt mer sofistikerte oppgaver. Raspberry Pi, for eksempel, kan kjøre en fullverdig Linux-distribusjon og kjøre flere grafiske applikasjoner som spill, nettlesere og avspilling av HD-film (P. Foundation, 2019). Dersom mikrokontrollere er i stand til å behandle store filstrømmer fortløpende kan de også være egnede verktøy for AoIP-implementasjoner. Dette er tilfelle for Belaen, som er brukt i dette prosjektet. Bela, som i sine tekniske spesifikasjoner er sammenlignbar med Raspberry Pi, er i tillegg utstyrt med høykvalitets lydporter og er spesiallaget med lav-latency lydbehandling som formål (Bela, 2019).

1.16 Latency

I systemer som foretar konvertering til- og fra digital lyd vil det i utgangspunktet oppstå en tidsforsinkelse (McPherson, Jack, & Moro, 2016). Dette kan påvirke brukerens evne til å spille instrumentet sitt dersom monitorkilde hentes fra dette systemet Musikeres evne til å spille på instrumentet med tidsforsinkelse i monitor varierer, men rundt 10 millisekunder virker til å være passende i mange sammenhenger. Det finnes også variasjon mellom instrumentgrupper for generell forsinkelsestoleranse (Lester & Boley, 2007). I systemet som utvikles i dette prosjektet vil kildene til tidsforsinkelse være flere enn den som følger av ren lydkonvertering. Eksempelvis vil det i tillegg være latency som følge av kommunikasjon mellom operativsystem og driver for lydkort, bufring i alle ledd, tidsforsinkelse i nettverkskommunikasjonen og tidsforsinkelse som følge av digital prosessering. Denne tidsforsinkelsen kan hope seg opp og bli problematisk for brukeren. Jeg skal senere gå inn på hvordan jeg har gått frem for å minimere tidsforsinkelsen.

1.17 Internet of things og relevans for musikkteknologien

Forsterkeren jeg bygger i prosjektet kan sies å inngå i begrepet Internet of Things, ofte forkortet IoT. Internet of Things refererer til den raskt økende andelen ulike ting som kobles opp mot internett. Eksempler kan være; baderomsvekter som oppbevarer målinger i skyen, fjernsynsapparater med smartfunksjonalitet, termostater som kan styres og overvåkes over nett, klokker, leketøy, til og med kjøretøy osv. Ett anslag på antall ting oppkoblet mot nettet i 2015 landet på 5 milliarder, og det er forventet at dette tallet kan øke til 25 milliarder innen 2020 (Kurose & Ross, 2017).

Det å bygge inn små generelle datamaskiner/mikrokontrollere i forbrukervarer har vært gjort lenge, og har muliggjort mye. Apparater som tidligere var avhengig av komplekse og spesiallagde kretser med strenge krav til plassbruk og layout for å utføre oppgavene sine kan i noen tilfeller lages med små, billige generelle datamaskiner til en brøkdel av prisen (Tanenbaum, 2016). Med disse datamaskinene åpner også muligheten for å innhente, generere og strukturere data til alle formål. Når dette kombineres med tilgang til det største kommunikasjonsnettverket i verden åpnes en hel verden av muligheter, både innen forbrukervarer, automasjon og musikkteknologi (Kurose & Ross, 2017).

1.18 Bygge ny gitarforsterker kontra bruke eksisterende

Jeg har, som nevnt, i dette prosjektet valgt å bygge gitarforsterkeren ut fra et byggesett i stedet for å ta i bruk en eksisterende gitarforsterker. Dette er av flere grunner. Foruten læringsutbytte av selve byggingen, er muligheten for å underveis vurdere innplassering av mikrokontrollere og ekstra elektronikk betydelig høyere. Videre medfører modifikasjon av en eksisterende forsterker en risiko for en irreversibel skade (og eventuelt tap av originalitet/affeksjonsverdi). Til sist vil kjøp av et byggesett i noen tilfeller bety en reduksjon i kostnad sett opp mot kjøp av en ferdigbygd gitarforsterker.

Del 2: Implementasjon og utfordringer

2.1 Innledende til del 2

Denne delen av rapporten vil ta for seg de tekniske aspektene ved systemet jeg har utviklet. Jeg vil gjennomgå hvilke utfordringer jeg har støtt på og hvordan jeg har valgt å løse disse. Dette har hatt en stor personlig læringseffekt, og jeg tror overføringsverdien til liknende systemer er høy.

2.2 Utstyrsliste

Hardware:

- Lenovo Thinkpad T430, laptop.
- RME Babyface Pro, lydkort.
- M-Audio Fast Track Pro, lydkort (etter hvert forkastet grunnet dårlig støttet driver for Windows 10).
- Edirol MA-15D, studiomonitorer.
- KRK Rokit 5, singel studiomonitor (for mono).
- Enkel ethernet-svitsj.
- Marshall 1959, byggesett fra The Tube Amp Doctor.
- Suhr Reactive Load, reaktiv lastenhet med DI-utgang.
- Bela, mikrokontroller.
- Raspberry Pi, mikrokontroller
- 28BYJ-48, stepper motor.
- ULN2003, drivkrets til stepper motor.
- Nødvendige kabler.
- Diverse andre stepper motorer, servoer og motoriserte potentiometre vurdert.

Software:

- Bela IDE, nettleserbasert integrert utviklingsmiljø for Bela.
- Juce, rammeverk til C++ for design av GUI og lydprosessering.
- Microsoft Windows 10, operativsystem.
- Projucer, integrert utviklingsmiljø for Juce.
- Visual Studio 2017, integrert utviklingsmiljø.
- Pro-Tools, DAW til enkel latencymåling.

2.3 Fremgangsmåte

I et prosjekt som dette finnes det mange mulige fremgangsmåter som vil føre til et fungerende sluttprodukt. For gjennomføringen har de tidsmessige- og økonomiske rammene vært førende. Videre har også prosjektet vært gjennomført på egen hånd og dekker mange teknologiområder.

Jeg valgte å sette av lang tid til forberedelse som første steg i prosjektarbeidet. Tidlig i prosjektet måtte jeg identifisere og velge ut mulig hardware som kunne passe inn i løsningen. Dette var en opportunistisk strategi, og en rekke stepper motorer, servoer, enkelt- og dobbeltpotentiometre ble bestilt på Ebay, da prisene var innenfor rammene og med håp om at de kanskje kunne passe. Disse kan typisk ha en til to måneder leveringstid. Leveringstidens største ulempe er at tiden mellom ankomst av eventuell uegnet komponent og erstatning er høy. Alternativet ville vært bestille komponenter etter spesifikasjon ville på den andre siden blitt dyrt, og krevd nøye spesifisering (og dermed nært ferdigbygd gitarforsterker) på forhånd.

For programvare har jeg gjennom masterstudiet valgt ut fag som jeg antok var relevante, for denne typen maskinnær utvikling, nettverksprogrammering og databehandling. Denne kom til å bli veldig sentralt for prosjektets suksess, særlig kunnskap om ringbufferalgoritmer var viktig for effektiv datahåndtering på tvers av datamaskin og mikrokontroller. I tillegg kom socketprogrammering som viktig ingrediens for den IP-baserte kommunikasjonen.

Etter forberedelsene gikk jeg i gang med utvikling av det grafiske brukergrensesnittet, en enkel selvlagd og integrert feilsøkingskonsoll og den tilhørende applikasjonen. Tanken bak å begynne det praktiske arbeidet her, var at et selvlagd feilsøkingskonsoll kunne benyttes aktivt under hele utviklingsprosessen.

Siste del av arbeidet har bestått i å veksle mellom utvikling og feilsøking av systemet. Rekkefølgen dette ble gjort i var med utgangspunkt i signalgangen fra brukerperspektivet.

2.4 Overordnet kodebeskrivelse

Nedenfor følger en rask oversikt over det egenutviklede programsystemets to hovedkomponenter for å illustrere omfang og kompleksitet. Noen av objektene nedenfor vil bli referert til videre i rapporten. Eierskap av objektene angis ved innrykk etter linjeskift i oversikten.

Program skrevet i Juce har følgende objekter:

- Maincomponent: Objektet all funksjonalitet relatert til lyd og grafisk brukergrensesnitt, samt underklasser eksisterer i.
    - TCPThread: Tråd med TCP-implementasjon. Henter informasjon fra GUI-et som sendes over til fjernstyringssystemets motpart (Bela).
    - AudioSendThread: Tråd med UDP-implementasjon. Forsynes med lydpakker fra Maincomponent som sendes over til Bela gjennom en socket.
    - AudioReceiveThread: Tråd med UDP-implementasjon. Mottar lyd fra Bela over socket, og oppbevarer den i et ringbuffer lokalt. Maincomponent-objektet henter lyden fra ringbufferet med eksterne funksjonskall. Intern logikk sørger for at pakkene settes inn i ringbufferet i riktig rekkefølge.
    - RingBuffer: Klasse med implementasjon av et ringbuffer. Brukes av Maincomponent til å formattere lyd i en datapakke før det sendes til Belaen gjennom AudioSendThread.
    - Images: Implementasjon av Juce sin Image-klasse. Oppbevarer bilderessurser til GUI-et og definerer størrelser og posisjoner på bildene.

Objektene TCPThread, AudioSend/ReceiveThread kjøres ved hjelp av Juce sin implementasjon av såkalt tråding. Dette betyr i praksis at Juce fordeler prosessortid mellom objektene slik at de kan samkjøre.

Overordnet kodebeskrivelse av program i Bela begynner på neste side.

Program skrevet for Bela:

- Render: Hovedprogrammet med alle objekter, variabler, initsialiseringsfunksjon og prosesseringsfunksjon.
    - TCPThread: Tråd med TCP-implementasjon. Mottar kontrolldata fra Juce sin TCP-tråd. Oppbevarer posisjonen til de virtuelle rattene for videre formattering og bruk av objektene Render og DeepState.
    - AudioSendThread: Tråd med UDP-implementasjon. Forsynes med lydpakker fra prosesseringsfunksjonen i Render.
    - AudioReceiveThread: Tråd med UDP-implementasjon. Forsynes med lydpakker fra Juce over socket.
        - RecvBuffer: Ringbuffer-implementasjon for AudioReceiveThread. Logikk mellom disse objektene sørger for at pakker settes inn i og hentes ut av bufferet i riktig rekkefølge.
    - SendBuffer: Implementasjon av ringbuffer for formattering av lydpakker før sending gjennom AudioSendThread.
    - DeepState: Tilstandsmaskin som holder styr på nåværende og fremtidig posisjon til steppermotorene, logikk som sikrer at kommunikasjon mellom Render og DeepState ikke fører til posisjonsforskyving av ratt og logikk for koordinert posisjonsoppdatering med objektet Render.

Objektene TCPThread, AudioSend/ReceiveThread og DeepState kan samkjøre ved hjelp av standardbibliotek for tråding levert med C++.

Felles for Juce og Bela er at behandling av lyd skjer gjennom kall til deres prosesseringsfunksjon (implementert funksjon som kalles for å informasjon fra og til I/O). I Juce kalles funksjonen idet bufferet internt i lydkortet er fullt, før det så hentes inn i programmet. Hos Bela er det tilsvarende, men direkte implementert mot en kjernemodul i Linux. Belaen benytter også prosesseringsfunksjonen til å skrive til de fysiske I/O-portene som kreves for fysisk manipulasjon av gitarforsterkerens ratt. Spenningsnivået disse I/O-portene leverer er egnet for signalering mot steppermotorenes drivkretser (tilsvarende mikrokontrollerne Raspberry Pi og Arduino). Disse prosesseringsfunksjonene kan modifiseres etter eget ønske, men må gjennomløpes før neste kall for å unngå dropouts.

2.5 Oversikt over forventede og erfarte utfordringer i prosjektet

I dette kapitlet oppsummerer og detaljerer jeg forventede og erfarte utfordringer i utviklingen av systemet. Noen er av generell karakter, mens andre er spesifikt for systemer med latency- og sanntidskrav og der systemet skal foreta fysisk manipulasjon over avstand. Dette kapitlet inneholder underkapitler som tar for seg hver enkelt i detalj.

Oppsummert:

1. Valg av hardware, programmeringsspråk og integrert utviklingsmiljø. Her er kompatibilitet og tilgjengelige ressurser (kapasitet, ytelse, funksjonalitet) kritiske områder å ta i betraktning.
2. Utviklingen av grafisk brukergrensesnitt. Utforming, brukervennlighet og funksjonalitet kan ha store innvirkninger på hvordan systemet blir tatt i bruk.
3. Kommunikasjon. Ulike problemstillinger knyttet til kommunikasjon: avstandsbegrensninger, latency, pakketap og jitter (variasjon i latency).
4. Parallellisering. Noen funksjoner krever at systemet aktivt venter, som uten såkalt tråding gjør at alle andre deler av programmet må settes på vent.
5. Minneforbruk. Ulike datastrukturer tilbyr ulikt forbruk av minneplass. I lydsammenheng, der minnebruken øker proporsjonalt med tid, kan dette by på utfordringer. Minimering av minnebruk og gjenbruk av minneposisjoner for å unngå flytting av data kan skape et mer effektivt system.
6. Kjøretid. Sanntidsprosessering stiller krav til maksimal kjøretid. Dersom man ikke kan forutse ressursforbruket kan man potensielt ende opp med at prosesseringen ikke er ferdig i tide.
7. Fjernmanipulasjon. Fjernstyring innebærer at noe manipuleres. Dersom manipulasjonen er mekanisk må digitale signaler omvandles til fysisk bevegelse. Dette fordrer spesialisert hardware (motorer, aktuatorer og drivkretser).

2.5.1 Valg av hardware, programmeringsspråk og integrert utviklingsmiljø

Det fins mange gitarforsterkere som kunne vært aktuelle som utgangspunkt for prosjektet. Som nevnt var det viktig for meg å kunne bygge forsterkeren ut fra et byggesett av flere grunner, blant annet risiko for irreversibel skade, læringseffekt og muligheter for modifikasjon allerede i byggeprosessen. Byggesettet jeg valgte er inspirert av designet til Marshall 1959-modellen som leveres av Tube Amp Doctor. Den relativt romslige plassen på innsiden av denne såkalte «hand wired» forsterkermodellen ga et antatt potensiale for eventuell reorganisering av enkeltkomponenter i forkant av montasje. Videre er potentiometrene plassert på rekke med jevn avstand, som åpner for enkel uniform løsning for mekanisk manipulasjon.

Det er lett å få inntrykk av at det finnes mikrokontrollere tilpasset et hvert formål til en overkommelig pris i dag. En av de mer populære, Arduino (2019), er svært egnet til hobbyelektronikk og ble benyttet i mitt bachelorprosjekt. Overgangen fra å kun fjernmanipulere ratt i bacheloren til å i tillegg skulle kjøre en fullverdig audio-over-IP-implementasjon vil kreve betydelig mer prosesseringskraft enn det nåværende generasjon av Arduino har tilgjengelig. Gjennom søk på forumet ved arduino.cc har jeg ikke klart å finne noen som har lyktes i å implementere et system med liknende sanntidskrav som dette prosjektet. Etter tips valgte jeg å sjekke ut mikrokontrolleren Bela. Dette er en mikrokontroller designet for lav-latency lydprosessering som har tilstrekkelig I/O til fjernstyring av ratt og en noenlunde fullverdig Linux-distribusjon som kjører i bunn. Ut fra de tekniske spesifikasjonene var det også andre aspekter som tydet på at Bela var egnet: prosessoren er på hele 1 GHz (flere titalls ganger raskere enn Arduinoen) og 512 MB med RAM (opp mot Arduinoens 8 KB). Denne mikrokontrolleren kunne også programmeres i ren C++, som jeg er kjent med. Gjennom lesing på diverse nettfora ble det klart at det er en rekke ting som må tas hensyn til i utviklingen av programvare til Belaen. Blant disse er at Belaen kjører best med en samplingsrate på 44100 Hz og bitdybde på 16. Denne samplingsraten må også benyttes i andre enden av systemet (her: Juce). Videre gjøres alle kall til funksjoner i hovedtråden gjennom prosesseringsfunksjonen som kalles med jevne intervaller angitt av samplingsrate/bufferstørrelse i sekundet. Ekstra prosesser må gjøres i parallelle tråder.

Valget av programmeringsspråk var først og fremst gjort på grunnlag av hvilke jeg kan fra før og hva jeg antok var egnet til det brede spekteret av funksjoner jeg forutså ville være nødvendig for systemet. De tre kandidatene jeg endte opp med var Python, C++ og Javascript. Avgjørelsen ble gjort ut fra hvilken av disse jeg ønsket å få mer kompetanse i, C++, som fremdeles er svært aktuell innenfor sanntids lydprosessering. I tillegg, som nevnt ovenfor, legger Bela-en opp til at den skal skrives i dette språket fra dens integrerte utviklingsmiljø. C++ var dermed et enkelt valg.

For utviklingen av grafisk brukergrensesnitt og system som kan snakke opp mot Belaen valgte jeg å ta i bruk rammeverket Juce for C++ (ROLI, 2019). Dette rammeverket er skapt for å enkelt kunne gjøre tre av de viktigste tingene som prosjektet mitt krever: håndtere lydstrømmer opp mot lydkort, prosessere lyd og utvikle grafiske brukergrensesnitt. Juce kan kombineres med flere biblioteker til C++ og stiller selv med gode klasser og datastrukturer for å utføre diverse oppgaver. Blant disse finnes klasser for tråder, TCP- og UDP-klienter, konvertering av datatyper osv. Det å sette opp et blankt prosjekt i Juce krever noen hundre linjer koder med funksjoner som ligger godt gjemt i biblioteket, så å bruke Juce sitt eget integrerte utviklingsmiljø, Projucer, til automatisk oppsett kan spare mye unødvendig tidsforbruk. Projucer tilbyr også funksjoner for å konvertere filer med rådata til et format som kan benyttes som interne ressurser i programmet, slik som lyd- og bildefiler. For rask kompilering og fargekategorisering av syntaks har jeg valgt å bruke Visual Studio, som viste seg å bli svært nyttig etter hvert som programmet vokste.

2.5.2 Utvikling av grafisk brukergrensesnitt

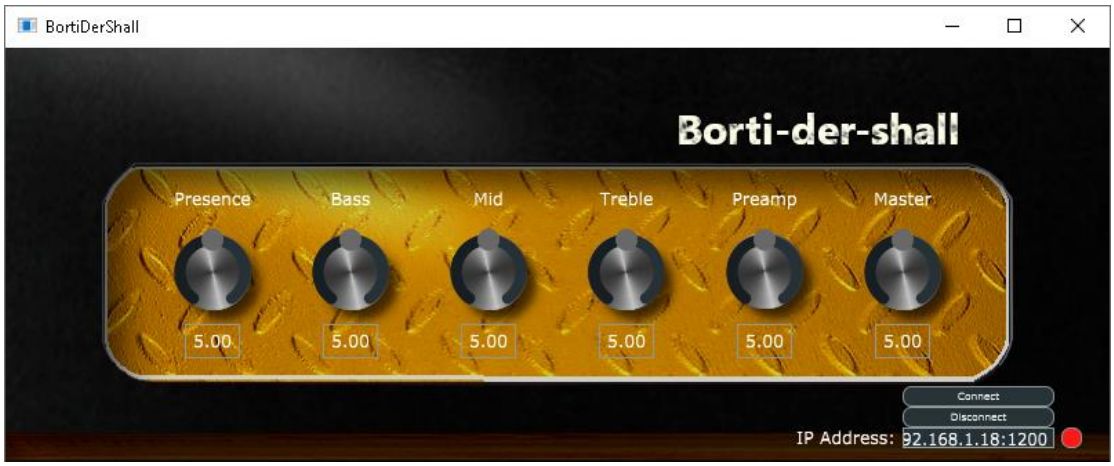
Behovet for et grafisk brukergrensesnitt oppstår når det kan være ønskelig for brukeren å få fremstilt informasjon på en oversiktlig, eller gjenkjennelig måte. For et fjernstyringsprogram kan denne informasjonen eksempelvis være virtuelle representasjoner av parameterne man styrer. Videre kan det også være fordelaktig for brukeren å få feedback på om prosessene som inngår i fjernstyringen er aktive eller ikke. Dette kan eksempelvis gjøres gjennom å imitere lyspærer fra elektrisk utstyr som indikerer om kretsen tilføres strøm.

Effekten av et grafisk brukergrensesnitt trenger likevel ikke å øke brukervennligheten – den kan også svekke den. Utfordringene ved å utvikle et grafisk brukergrensesnitt kan oppstå av mange årsaker. En av dem er hvis antall funksjoner som skal vises for brukeren blir så høyt at de ikke lar seg fremvises eller plasseres på en ryddig og intuitiv måte. Et lurt grep her kan være å ta utgangspunkt i tidligere vellykkede brukergrensesnitt. Det er dette jeg har forsøkt å gjøre i mitt system.

Figur 2.1: Frontpanel, Marshall 1959. Hentet fra www.Marshallforum.com

Bildet ovenfor viser framsiden av forsterkermodellen jeg har bygget. Brukergrensesnittet som befinner seg på det gullfargede frontpanelet har vært standard for Marshall-forsterkere siden den første kommersielle forsterkeren deres kom på markedet i 1963 under navnet Marshall JTM 45. Delen av brukergrensesnittdesignet jeg ønsker å fokusere på her er hvordan parameterne er stilt opp – alle står på rekke. Videre finnes også en slags inndeling i kategorier. Lengst til venstre (utenfor bildet) er en bryter for å forsyne forsterkeren med strøm, etterfulgt av en bryter for å «aktivere rørene» med anodespenning (standby). Etter dette kommer fire ratt som man kan vri for å tilpasse forholdet mellom spektralt innhold, inndelt i fire bånd. Til slutt kommer to ratt for utstyring av henholdsvis forforsterker og effektforsterker og inngang for TS-kabel (jack-kabel, utenfor bildet). Uansett hva som har ført til at brukergrensesnittets design har forblitt nært uforandret, enten det er tradisjon eller en designmessig genistrek, er det en annen faktor som kanskje er enda mer relevant: At layout i grafisk brukergrensesnitt gjenspeiler dét fra enheten som fjernstyres kan, gjort riktig, fungere som en signifier. Det er nemlig mange gitarforsterkere designes med liknende kontrollflater.

Dette betyr at mange gitarister kan være tilvendt denne måten å innjustere en gitarforsterker på. Hvis det å gjenskape layoutet i best mulig grad har denne fordelen burde et alternativt layout være minst like intuitivt. Videre håper jeg at gjenspeiling av layout kan skape et slags skille mellom parameterne som styrer rattene og de som er nødvendige for nettverkskommunikasjon. Løsningen min kan ses på neste side.



Figur 2.2: Grafisk brukergrensesnitt til fjernstyringssystem. Virtuelle ratt langs rad på midten. Nettverksfunksjonalitet er plassert nederst i høyre hjørne.

Her er de virtuelle rattene stilt opp på en rad langs frontpanelet, omtrent midt i programvinduet. Posisjonen til hvert ratt representeres som et flyttall mellom 0 og 10. I et forsøk på å ikke forstyrre gjenspeilingens funksjon som signifier har jeg plassert alt nettverksrelatert funksjonalitet nede i høyre hjørne. Her er en felt for visning og innskriving av IP-adresse og portnummer, samt knapper for til- og

frakobling. Til høyre for dette feltet er en virtuell lampe som lyser enten rødt, gult eller grønt for å angi henholdsvis om den er frakoblet forsterkeren (dvs. Belaen), forsøker å koble seg til eller om den er koblet til. Lampens farge oppdateres i sanntid.

2.5.3 Hvordan transportere lyd over lengre avstander

For at gitarforsterkeren skal kunne bruker til fjernprosessering kreves det i tillegg til fjernstyring at et lydsignal fra en elektrisk gitar kan oppstå ett sted og flyttes til der hvor forsterkeren måtte befinne seg. Overføring av lyd over avstand med lite til ingen signalforringelse gjøres på mange måter og i mange ulike musiseringsscenarier.

Et logisk første eksempel er kabelen som oftest brukes for å transportere signalet fra gitaren til forsterkeren: TS-kabel, eller populært kalt jack-kabel. Jack-kabelen bærer et ubalansert analogt signal og kan medføre signaltap over korte avstander avhengig av kabelens kvaliteter. Blant de vanligste jack-kablene kan man forvente en rekkevidde på 5 meter før signaltapet har en merkbar lydlig innvirkning.

Som eksempel på en balansert kabeltype for audio vil jeg dra fram XLR. Denne kabelen kansellerer støy gjennom å sende signalet langs to tråder med motsatt polaritet. Kanselleringen av støy skjer idet polariteten snus tilbake og signalet summeres. Dette er en av grunnene til at XLR kan frakte et signal betydelig lengre enn jack-kabelen. Dette er dog fremdeles betydelig kortere enn de avstandene vi mennesker kommuniserer over i dag – internett kan brukes for å frakte digital informasjon uforandret rundt hele kloden.

Dersom forsterkeren kan nås gjennom internett er ikke avstand lengre et problem gitt at både bruker og gitarforsterker kan tilkobles der de er plassert. Likevel er ikke denne fordelen ved internett uten bekostning. Etter noen ti-talls mil er tidsforsinkelse som følge av lyshastigheten allerede være betydelig merkbar i en musikalsk sammenheng.

2.5.4 Hvordan overføre informasjon over nett

I dette prosjektet har jeg behov for å overføre to hovedkategorier av informasjon mellom Juce og Bela: lyd og kontrollsignal, og dét over samme medium – et IP-basert nettverk, og gjerne over etablerte protokoller oppå IP. Disse protokollene

kan potensielt dekke flere nødvendige funksjoner i kommunikasjonen og kan drastisk redusere utviklingstiden om de er kompatible med kravene til systemet. Jeg skal i dette kapittelet gjøre rede for valgene jeg har gjort i henhold til standardiserte nettverksprotokoller og litt om hvordan jeg har designet systemet på applikasjonsnivå for å motvirke feil som kan oppstå i alle ledd i kommunikasjonen.

For overføring av lyd finnes det flere nettverksprotokoller som er aktuelle. Den første jeg vurderte var RTP (Real-Time Protocol), og årsaken til det var at navnet hentyder til sanntidsfunksjonalitet som kunne være nyttig. RTP består av minimum to kommunikasjonskanaler: minst én for media (lyd eller video) og en for kontrollsignal med økning av førstnevnte ved behov. Kontrollsignalkommunikasjonen brukes til å holde oversikt over pakketap og pakker som kommer frem i feil rekkefølge (nettkvalitet). Denne oversikten brukes til å regulere bitraten til signalet som sendes på bekostning av kvaliteten. For mediaoverføringen benyttes UDP, der ekstra informasjon pakkes inn ved siden av rådataen for å muliggjøre analysen av nettkvaliteten. En rask analyse av andres implementasjoner av RTP (og utfordringene de møtte på) konkluderte jeg med at RTP var uegnet innen tidsrammen for prosjektet – RTP lar seg ikke implementere uten å prege designet av hele systemet, noe som virket risikabelt da dette er mitt første prosjekt både på Juce og Bela. Etter RTP ble lagt vekk begynte jeg å drøfte noen design som involverte UDP og ringbufre. Første skisserte utkast ble ferdig samme dag som jeg ble tilsendt en epost fra Otto Wittner vedlagt mange artikler om nettmusikk, deriblant én om Jacktrip. Det viste seg at designet mitt var svært likt det Jacktrip sin AoIP-funksjonalitet er implementert. Den eneste store forskjellen i design er at min løsning sender hver UDP-pakke to ganger, men Jacktrip sine pakker inneholder første halvdel av samplene i den etterfølgende pakken (Cáceres & Chafe, 2010). Likheten i designet med Jacktrip, som jeg har erfart at fungerer godt, gjorde at jeg valgte å gjennomføre designet. Idéen bak doble UDP-pakker var med tanke på mindre følsomhet for enkeltstående pakketap. Gitt at mediehastigheten er høyere enn bitraten for audio, vil ikke latencybidraget fra dette øke dramatisk, dog er dette en mekanisme som kan endres til bare én pakke.

Kravene som stilles til kontrollsignalet og overføringen av den er at den representerer alle parameternes oppløsning i den grad at den gjenspeiler behovene brukeren kan ha for finjustering. Jeg har i dette prosjektet valgt å gå for 1000 diskrete punkter langs hele dreieområdet til hvert potentiometer. Dette var et enkelt og håndterbart tall for konvertering til steppermotorbevegelse (steppermotorene i prosjektet har 4096 trinn for full rotasjon). Dette er langt utenfor det jeg anser som mulig å justere inn med fingertuppene. Overføringen av kontrollsignalene over nett bør helst skje med en gang en endring er gjort slik at brukeren får vurdert det lydlige resultatet av justeringen så fort som mulig.

Jeg har valgt å formattere kontrolldataene på følgende vis: hvert virtuelle ratt i brukergrensesnittet representeres av fire karakter-datatyper, eksempelvis slik: «10.0», «4.91» eller «0.55». Representasjonene av rattene settes så inn i en streng med mellomrom mellom hverandre. En slik streng for eksempel kan se slik ut: «5.00 7.85 9.50 3.22 10.0 10.0».

Den faktiske overføringen av kontrolldata over nett er i prosjektet gjort gjennom TCP. En kunne også benyttet UDP til dette, men ettersom UDP er «connectionless», altså at endepunktene ikke tar hensyn til hverandres eksistens, vil verken brukeren eller fjernstyringsprogrammet kunne fastslå at kontrolldataen faktisk blir mottatt. Dette er derimot funksjonalitet som TCP leverer. Tilstanden til TCP-socketen angir fargen på en lampe i brukergrensesnittet i sanntid, som gir en indikasjon på om tilkoblingen var vellykket og om Belaen av en eller annen grunn ikke kommuniserer lengre. Denne funksjonaliteten benytter jeg også i lydoverføringer. Lydoverføringen over UDP begynner ikke før TCP har bekreftet at IP-adressen som angis av brukeren er «korrekt».

2.5.5 Latency i nettverk

Som nevnt tidligere, kan en økning i latency mellom en musikalsk handling blir utført til lyden høres av utøveren (utover instrumentets naturlige treghet) kan påvirke utøverens evne til å musisere. I dette prosjektet fins flere kilder til latency, konvertering mellom analog og digital lyd, bufring i alle ledd, prosessering og til en viss grad operativsystem. Disse latency-kildene eksisterer i systemet uavhengig av fysisk avstand mellom brukeren og forsterkeren. Ettersom dette systemet er tiltenkt å brukes over IP-baserte nettverk oppstår andre former for latency som kommer i tillegg til de overnevnte kildene. IP-baserte nett består, som nevnt tidligere, av flere noder i et nettverk. Hver av disse nodene skal gjøre flere oppgaver som hver vil tilføre latency. Disse formene for latency kalles:

- Prosesseringsforsinkelse (Processing delay)
- Køforsinkelse (Queueing delay)
- Transmisjonsforsinkelse (Transmission delay)
- Propagasjonsforsinkelse (Propagation delay)

Summen av disse fire latency-kildene kalles «total nodal delay». Jeg skal i de påfølgende avsnittene gå punktvis gjennom disse og belyse utfordringen med latency i en audio-over-IP-implementasjon. Informasjon hentet fra Kurose og Ross (2017).

Prosesseringsforsinkelse er en tidsforsinkelse som oppstår når en node i et nettverk analyserer innholdet i en nettverkspakke. Noen noder kan også sjekke pakken for bit-feil som kan oppstå underveis i sendingen. Dette prosessen går i gang etter noden har mottatt hele pakken og flyttet den ut av bufferet og avsluttes i det pakken igjen bufres opp for transmisjon på utgående link.

Køforsinkelse oppstår dersom en node har ventende pakker på utgående link. Dette betyr at dersom denne køen er tom vil forsinkelsen være lik null, samtidig som tidsforsinkelsen øker lineært med antall pakker i køen. I praksis er denne tidsforsinkelsen ofte mellom noen mikrosekunder til noen millisekunder.

Transmisjonsforsinkelse er tiden det tar å flytte hele pakken fra køen til ut på linken. Dette forsinkelsen kan regnes ut fra pakkens størrelse i bit og kjennskap til nodens transmisjonsrate på denne måten: pakkestørrelse i bit/transmisjonsrate, eller L(ength)/R(ate). Nodene i nettet (typisk rutere) har typisk alt mellom 10 Mbps og flere gigabit på forbindelsen mellom dem, og pakker sendt over nettverk er vanligvis opp til 1500 byte, som er maksimalt for standard type ethernet-linker. I praksis vil denne tidsforsinkelsen, i likhet med køforsinkelse, være mellom noen mikrosekund og millisekund.

Propagasjonsforsinkelse kan forklares som tiden det tar før en pakke som er flyttet i sin helhet over på en link før det siste bitet har ankommet neste node i nettverket. Hastigheten på propagasjonen er nært lysets hastighet og tidsforsinkelsen kan regnes ut slik: avstand/hastighet. Årsaken til at det skilles mellom propagasjonsforsinkelse og transmisjonsforsinkelse er fordi propagasjonstiden ikke har noe med noden å gjøre og transmisjonstiden ikke har noe med avstanden å gjøre. Ping fra NTNU til UIO gav mellom 7 og 8 millisekund. Her er propagasjonsforsinkelsen største bidragsyter (lyshastighet).

2.5.6 Sanntidskrav og behovet for parallellisering og tråding i implementasjon

Parallellisering (her av engelske *concurrency*) innebærer å dele opp prosesser i mindre deler som kan jobbe uavhengig av hverandre på én og samme prosessorkjerne. Det er i praksis det samme som skjer om man surfer på nettet og lytter til musikk fra en medieavspiller på samme datamaskin samtidig. I dette scenarioet er det operativsystemet som fordeler kjøretid mellom de ulike programmene. Parallellisering internt i et program kan fungere på samme vis, men kan også gjøres med prosesser som deler og forsyner hverandre med ressurser i sanntid. Å designe et parallellisert program krever at utvikleren strukturerer programmet for å effektivisere tildelingen av kjøretid, samt sikre at kommunikasjonen mellom prosessene ikke forårsaker noen feil. På sett og vis «adopteres» da en del av operativsystemets funksjon, men da for å få mer kontroll på sanntidsegenskapene internt i programmets kjøretidsvindu (Dvecko, 2019).

I prosjektet mitt oppsto behovet for parallellisering idet jeg skulle implementere TCP for overføring av kontrollsignal i Juce. Når TCP-delen av programmet ventet

på kvittering for pakker den hadde sendt måtte oppdatering av grafisk brukergrensesnitt og behandling av lydstrømmen fra lydkortet stå på vent. Dermed måtte tråding implementeres, noe som Juce tilbyr i sin Thread-class. Til sammen har jeg implementert 3 parallelle tråden i Juce.

Belaen gjør som nevnt tidligere eksplisitte kall til funksjoner gjennom sin prosesseringsfunksjon. Dersom alle Belaens funksjoner skulle blitt kalt fra denne, ville det gått på bekostning av at prioriteten til lydbehandling. Jeg har derfor strukturert programmet i Belaen til å ta i bruk tråder for all funksjonalitet utenom å sende lyd og kontrollsignal til rattene.

2.5.7 Håndtering av varierende nettkvalitet

Tidligere har jeg nevnt at UDP ikke garanterer mot pakketap. Pakketap kan oppstå av flere grunner, men de mest vanlige årsakene er at køen er full hos en node i nettverket, dårlig linjekvalitet eller andre forstyrrelser i nettet. Rutere i et pakkesvitsjet nett forkaster pakker som ikke får plass i bufferet. Dette gjøres for å oppnå to ting. Den første er, naturligvis, at pakker som allerede er i køen skal prioriteres. Den andre er for å regulere trafikk som benytter seg av metningskontroll, enten gjennom eksempelvis TCP-protokollen i transportlaget på nettverksmodellen eller på applikasjonsnivå i applikasjonslaget. All trafikk med metningskontroll vil derfor senke pakkeutsendelsesfrekvensen dersom den passerer en node som får metning i køen. På dette viset kan metningskontroll sies å ha en demokratiserende effekt på nettverkskapasiteten. Konsekvensen av pakketap for kommunikasjon over TCP vil være midlertidig lavere pakkeutsendelsesfrekvens inntil trafikken har stabilisert seg, men for UDP vil det være tapt informasjon (Kurose & Ross, 2017). Så hvorfor har jeg likevel valgt å bruke UDP fremfor TCP for overføring av lyd?

Bitstrømmen for høyoppløselig lyd (PCM) er konstant. Over TCP vil metningskontroll kunne gi utslag i form av at pakker ikke sendes ut i høy nok hastighet og medføre dropouts for en typisk lydstrøm. Retransmisjon i møte med pakketap over TCP vil også medføre en forskyvelse av utgående pakker, som øker behov for bufring, og dermed latency. Alternativt kunne en kompresjonsalgoritme blitt brukt for å forhindre dropouts, men dette ville økt latency på grunn av konvertering og eventuell oppbufring for sømløs overgang mellom bitrater. Ettersom jeg ønsker å holde latency så lavt som mulig og lydkvaliteten høy, var det av interesse for meg å i hvert fall gjøre et forsøk med UDP.

2.5.8 Jitter

Når en lydstrøm sendes over nettet er den som nevnt tidligere delt opp i pakker. Hver pakke i mitt system inneholder 256 samples og lydstrømmen er på 44100 samples i sekundet. Det vil si at hvert sekund sendes omtrent 172 pakker, og dette pågår helt til kommunikasjonen opphører. Ettersom internett er et asynkront kommunikasjonssystem er det små variasjoner i tiden mellom pakker sendes og mottas. Dette fenomenet kalles jitter, og forekommer i alle systemer der en tidsbestemt repeterende hendelse skal skje, blant annet selv i klokkesystemer som brukes i synkrone systemer.

Figur 2.3: Figur som viser Jitter.

Konsekvensen av jitter for sanntids lydsystemer kan være de samme som ved pakketap: dropouts. Dette skyldes at pakkene som sendes oppstår og brukes i samme tempo, angitt av overnevnte pakkestørrelse og samplingsrate. Dersom pakkene hadde blitt spilt av med en gang de ankom vil en økning av tid mellom to pakker resultere i et dropout av like stor størrelse i tid mellom dem.

Løsningen på jitter er å bufre opp pakkene med sampler før det avspilles. *Headroom*-et som gis av bufferet skal i teorien gjenspeile hvor mye jitter som tolereres. Jeg har satt *headroom*-et til å være 512 samples, som er omtrent 11 millisekund med samplerate på 44100. Dette er i overkant for et lokalnett, og antakelig godt innenfor flere anvendelser internt i Norge. I videre utprøving bør dette eksperimenteres med, med intensjon om å minimere latency. Lav bufferstørrelse fordrer god nettkvalitet og godt «*headroom*» i nettverket.

2.5.9 Minnebruk

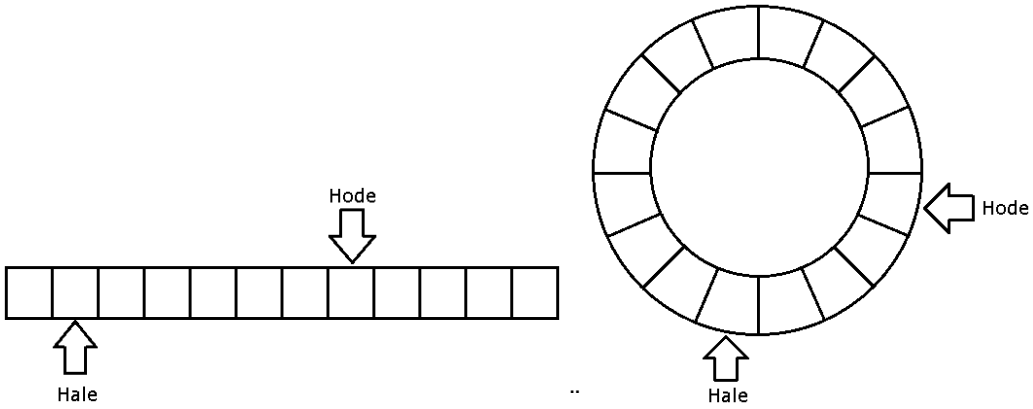
Når utfordringen med jitter er møtt vil man i sanntidsprosessering umiddelbart støte på et annet problem: minnebruk. En enkanals lydstrøm med samplingsrate på 44.1 kHz og 24 bit samplingsdybde vil eksempelvis ta opp i overkant av 1058 kilobit per sekund hver vei, og videre over 476 MB per time. For et rockeband med trommer, to elektriske gitarer, en elektrisk bass, tangenter og vokal vil kanalbehovet fort kunne overstige 20 kanaler, og tidsbruken kan være flere timer. Det å gå tom for lagringsplass under prosessering vil resultere i at

prosesseringen nødvendigvis må slutte. Selv om lagringsplass blir stadig billigere vil det ikke opphøre å være en begrenset ressurs. Hvordan kan dette problemet møtes?

En løsning kan ligge i en effektiv datastruktur. En datastruktur defineres ut fra hvordan informasjonen struktureres i minne og hvordan den manipuleres. Manipulasjonen av datastrukturen skjer gjennom flere funksjoner som hver sørger for at manipulasjonen blir gjennomført uten at integriteten til datastrukturen går tapt. Innen datastrukturer refererer begrepet integritet til blant annet datastrukturens korrekthet, brukbarhet og konsistens over hele dens livstid – altså fra datastrukturen opprettes til den slettes. Eksempler på slike funksjoner kan være søk, innsettelse, sletting, bygging og flytting. Datastrukturer kan dannes ut fra forskjellige noder som hver inneholder informasjon som er nyttig for programmet eller oss, samt informasjon som trengs for datastrukturen (Cormen, Leiserson, Rivest, & Stein, 2009).

Både Belaen og fjernstyringsapplikasjonen må benytte seg av datastrukturer for oppbufring av samples ettersom det er lydoverføring begge veier mellom disse. Kravene som stilles til datastrukturen er at informasjon må kunne settes inn fra en ekstern prosess og tas ut i riktig rekkefølge av prosesseringsfunksjonen. Videre er det til fordel at uttak av informasjon til datastrukturen ikke krever et betinget valg av typen if/else statement, da dette vil trigge et såkalt mode switch hos Belaen. Et mode switch innebærer at en prosess krever ressurser fra en annen prosess med lavere prioritet og dermed må senke sin egen prioritet for å utføre handlingen. Dette kan gå bra i det korte løp, men Belaen vil bruke ekstra ressurser på å dokumentere alle forekomster og årsaker til mode switcher fortløpende, som spiser verdifull prosesseringstid. Verdt å nevne er at Belaen kan gjennomføre betingede valg av typen if statements i prosesseringsfunksjonen – det er else-delen som forårsaker mode switcher.

Etter å ha vurdert flere typer datastrukturer for oppgaven, deriblant lenkede lister (med hyppig sletting av informasjon som ikke lengre er relevant), landet jeg på ringbuffer. Et ringbuffer kan kort forklares som et statisk avsatt minneområde som gjenbrukes der informasjon som settes vil skrives over etter et gitt antall innsettelser. Når det skrives til ringbufferet vil all rådataen havne sekvensielt etter hverandre med utgangspunkt i en angitt posisjon (bufferhode), og uttak vil skje fra en annen angitt posisjon (bufferhale). Når innsettelse og uttak er gjort, vil inkrementering av disse posisjonene gjøres etter hvert uttak og hver innsettelse. Bufferhode og bufferhale oppbevares eksternt fra selve bufferet, enten i form av pekere til spesifikke noder eller heltallsvariabler som angir nodenummeret.



Figur 2.4: Figuren viser to vanlige måter å representere et ringbuffer på. Til venstre er minneområde angitt langs én dimensjon, og til høyre er minneområde representert i en ring der det kommer tydelig frem hvordan bufferet fungerer i praksis.

Etter ringbufferet er opprettet i systemet, gjennom å angi antall noder og størrelsen på nodene, henviser både bufferhode og bufferhale til første node. Funksjonene som benyttes etter opprettelsen av ringbufferet er insert() og extract(). Pseudokoden under vil forklare hvordan de fungerer.

```
insert(*data)
    For i = 0; i < sizeOfNode; i++
        ringbuffer[bufferHead + i] = data [i]
    bufferHead = (bufferHead + sizeOfNode) % totalSizeOfBuffer


extract(*data)
    For i = 0; i < sizeOfNode; i++
        *data [i] = ringbuffer[bufferTail + i]
    bufferTail = (bufferTail + sizeOfNode) % totalSizeOfBuffer
```

Dette tilfredsstiller kravet om konstant kjøretid og forårsaker ingen mode switches hos Belaen.

2.5.10 Fjernstyring og fysisk manipulasjon over avstand

Dette underkapitlet vil ta for seg hvordan jeg har løst utfordringen med fysisk manipulasjon av forsterkeren over avstand både i hardware og software. Målet med fjernstyringen er at kontrollpanelet på fremsiden av forsterkeren skal

gjenspeile parameterinnstillingene i det grafiske brukergrensesnittet til enhver tid. Som nevnt overføres parameterinnstillingene over TCP til Belaen. Når kontrolldataen er framme må den reformateres slik at Belaen kan bruke den til den ønskede manipulasjonen.

Valget av steppermotorer for fysisk manipulasjon legger føringer for både formateringen og håndtering av styringssignal. En stepper motor roteres av at sirkulært og jevnt fordelte spoler internt i motorhuset forsynes med spenning og roterer ett gir som er koblet til akslingen. Spolene må forsynes med spenning i en rekkefølge som gjenspeiler rotasjonen til giret. For stepper motoren jeg har valgt, 28byj-48, er det fire slike spoler, og rekkefølgen for rotasjon kan sees ut fra tabellen under. Her er spenningstilførsel markert med 1.

Spole 1	Spole 2	Spole 3	Spole 4
1			
1	1		
	1		
	1	1	
		1	
		1	1
			1
1			1

Figur 2.5: Tabell over rekkefølge på spenningstilførsel til steppermotor for rotasjon.

Dersom spolene forsynes i rekkefølgen angitt loddrett nedover vil giret rotere en runde mot klokken, og med klokken dersom rekkefølgen snus. En full rotasjon (360 grader) krever 4096 steg, eller 512 fulle girrotasjoner, i én retning. Potentiometrene i forsterkeren krever omtrent 3400 steg for å rotere fra en ende til annen. Spenningen i spolene forsynes i dette prosjektet av en drivkrets, som her er et mellomledd mellom Belaen og stepper motorene. Det kan nevnes at dersom man absolutt måtte, kunne spenningen også blitt forsynt direkte fra Belaen, men det å forsyne 5-12V kan skape store problemer for flere mikrokontrollere – så jeg har latt være å i det hele tatt prøve. Drivkretsen forsynes med strøm fra en dedikert strømskinne med egen strømforsyning og mottar kontrollsignal i form av fire digitale inputs (binært spenningssignal). Dette kontrollsignalet oversettes til spenningsforsyning av spoler i samme mønster som signalet inn. Fjernstyringssystemet i Belaen sin jobb blir derfor å oversette en flyttallsrepresentasjon av alle rattene mellom 0 og 10 til antall steg i ulike retninger. For å sende dette kontrollsignalet ut må Belaen skrive til utgangene sine gjennom prosesseringsfunksjonen.

Det første utkastet av fjernstyringssystem jeg programmerte til Belaen var rent algoritmisk implementert direkte i prosesseringsfunksjonen og strakk seg over 40 linjer kode. Designet var avhengig av å benytte if/else-statements og resulterte derfor i mode switches som gav tap av rotasjoner og dropouts i lyd. For å unngå dette måtte jeg skrive en tilstandsmaskin som kunne gå parallelt med prosesseringsfunksjonen, der prosesseringsfunksjonen kan hente informasjon uten å forholde seg til tilstandsmaskinens logikk. Tilstandsmaskinen har jeg gitt tilnavnet DeepState.

DeepState inneholder en rekke variabler for hver av stepper motorene. Disse er som følger:

1) Nåværende posisjon fra 0-3400 steg fra null.
2) Målposisjon fra 0-3400 steg fra null.
3) Fire boolske variabler i en liste som representerer kontrollsignalet (en til hver output).
4) En peker til overnevnte liste med boolske verdier.
5) Heltall som angir endring etter neste uthenting av posisjoner (enten -1, 0 eller 1).
6) En liste med boolske variabler som alle er satt til «false». Denne listen deles av alle rattene, og vil herfra refereres til som «den tomme listen».

Funksjonene som muliggjør tilstandsendringer internt i DeepState er som følger:

1) getCurrentPosition(). Henter nåværende posisjon til rattene.
2) getFuturePosition(). Henter målposisjonen til rattene.
3) setFuturePosition(). Setter ny målposisjon for rattene.
4) changeStates(). Endrer de boolske variablene i listen med kontrollsignalet.
5) getStates(). Henter ut verdiene fra listen med kontrollverdier til rattene.

Tilstandsmaskinen manipuleres fra to parallelle tråder som kjører på Belaen. Disse prosessene henter ut rattposisjoner etter hvert som de kommer over TCP og oversetter dette til antall steg fra null gjennom å avrunde resultatet av flyttallet som angir rattposisjon etter det er multiplisert med 340. Dette nye tallet settes så inn i DeepState gjennom setFuturePosition.

Uttak av kontrolldataverdier fra DeepState gjøres som nevnt i prosesseringsfunksjonen i Belaen. Prosedyren for uttak er som følger:

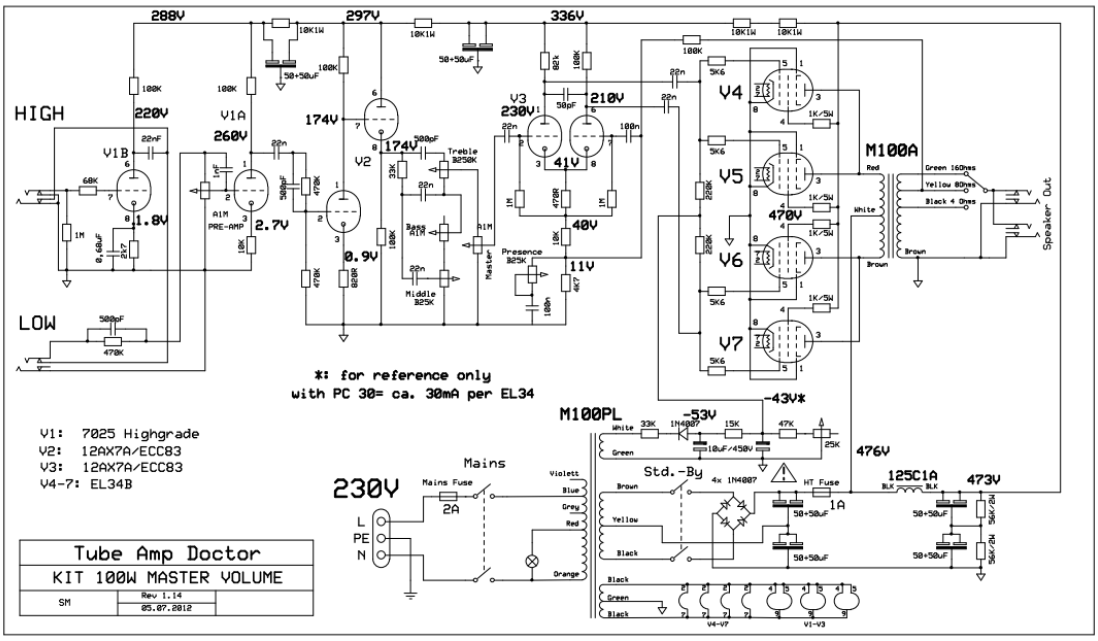
1) Kopier ut de boolske verdiene gjennom pekeren til rattet i DeepState.
2) Påfør endringen i nåværende posisjon for rattet (-1, 0 eller 1).
3) Sett pekeren til de boolske verdiene til å peke mot den tomme listen.
4) Sett heltallet over neste endring til å være lik 0.
5) Skriv de boolske verdiene til output.

Denne prosedyren er utformet av hensyn til det faktum at manipulering av DeepState og uthenting av data går parallelt og uten kommunikasjon mellom DeepState og prosesseringsfunksjon. Dersom prosesseringsfunksjonen skulle rekke å hente ut data fra DeepState uten at den eksterne manipulasjonen har blitt gjort ville stepper motoren beholdt samme fysiske posisjon samtidig som nåværende posisjon i DeepState ville endret seg. Over tid kunne da avviket mellom faktiske posisjon og ønskede posisjon på forsterkerens ratt blitt større – uten mulighet for hverken brukeren eller Belaen å vite om, eller hvor stor forskyvelsen er.

Når den eksterne tråden skal endre tilstanden til de boolske verdiene i DeepState gjøres dét gjennom å sjekke to ting. Det første er om nåværende posisjon er ulik målposisjonen, og den andre er om DeepState sin peker til de boolske verdiene peker til den tomme listen. På dette viset er Belaen sikker på at stepper motoren er flyttet, før den går gjennom en prosedyre som sikrer at de boolske verdiene oppdateres på riktig vis, før den til slutt setter korrekt verdi på heltallet som angir neste endring til enten -1 eller 1 avhengig av nødvendig rotasjonsretning.

2.6 Bygging av gitarforsterkeren og påmontering av fjernstyringssystem

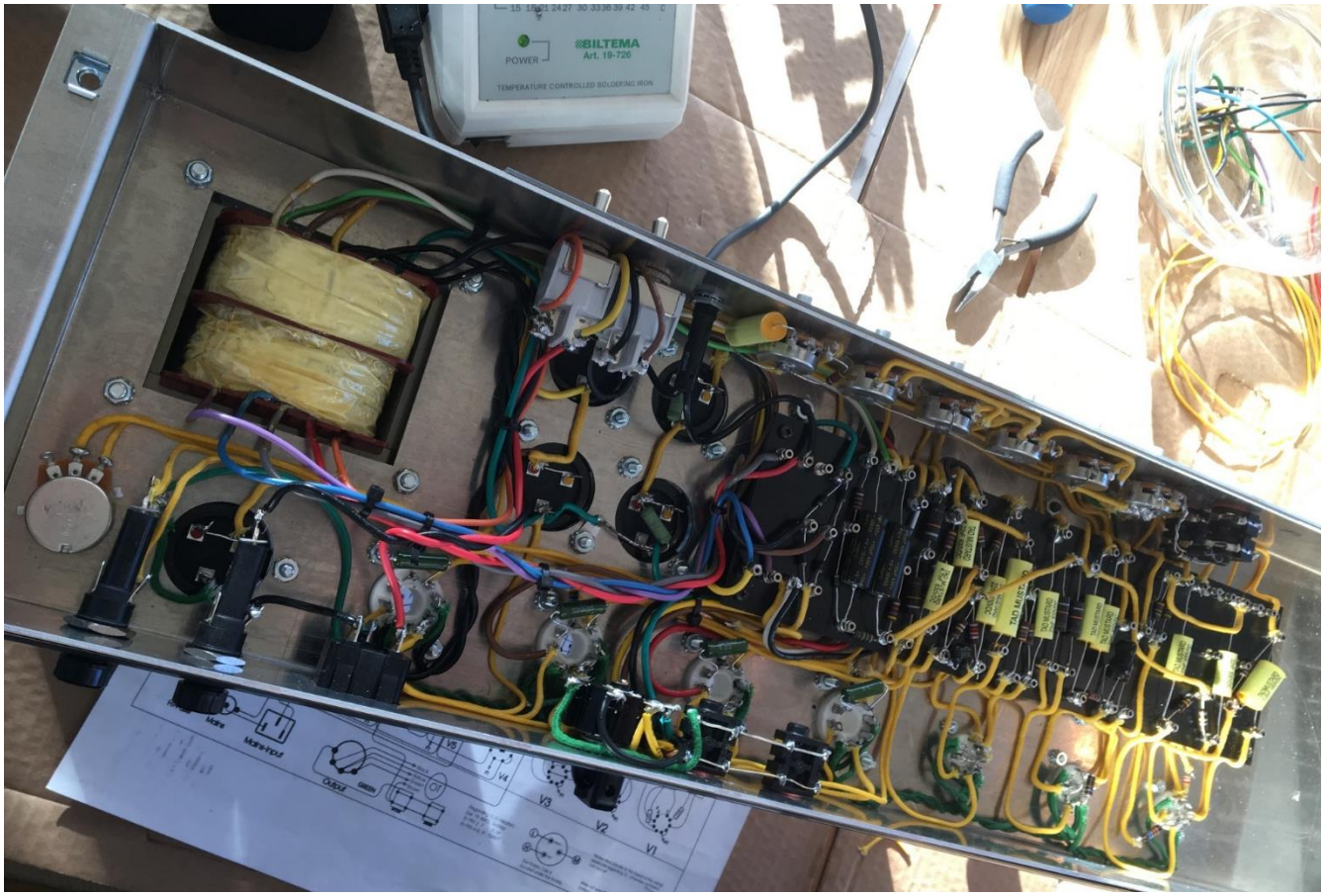
Selve gitarforsterkeren som er bygget i prosjektet er som nevnt bygd med utgangspunkt i et byggesett levert av Tube Amp Doctor. Byggesettet leveres i 257 deler med tilhørende kretsskjemaer som gir oversikt over hvert sitt område av forsterkeren.



Figur 2.6: Kretsskjema for hele forsterkeren. Hentet fra manualen som følger med byggesettet.

Figuren ovenfor viser kretsskjema for hele forsterkeren . Man kan gå løs på byggingen i hvilken rekkefølge man vil, men jeg valgte å holde gjøre det i en rekkefølge der chassiet var enkel å balansere opp ned samtidig som jeg sørget for at deler som kunne ødelegges ved et fall ble satt inn nært slutten.

Det som er av interesse for innplassering av utstyr, er at utgangsrørene, preamprør, potentiometre trafoer og analoge signalbehandlingskretser (på turret board) relativt separerte i layoutet. Kun rør og trafoer befinner seg på oversiden av chassiet. Det ble under bygging klart at ledninger med høyspenningsforsyning til rørene var spredt langs hele innsiden av chassiet. Dette medførte en viss engstelse for innsettelse av tilleggselektronikk (eksempelvis steppermotorer). I tillegg er det grunn til bekymring for at steppermotorene vil kunne introdusere støy i preamp-delen, ettersom denne ligger tett på potentiometrene.



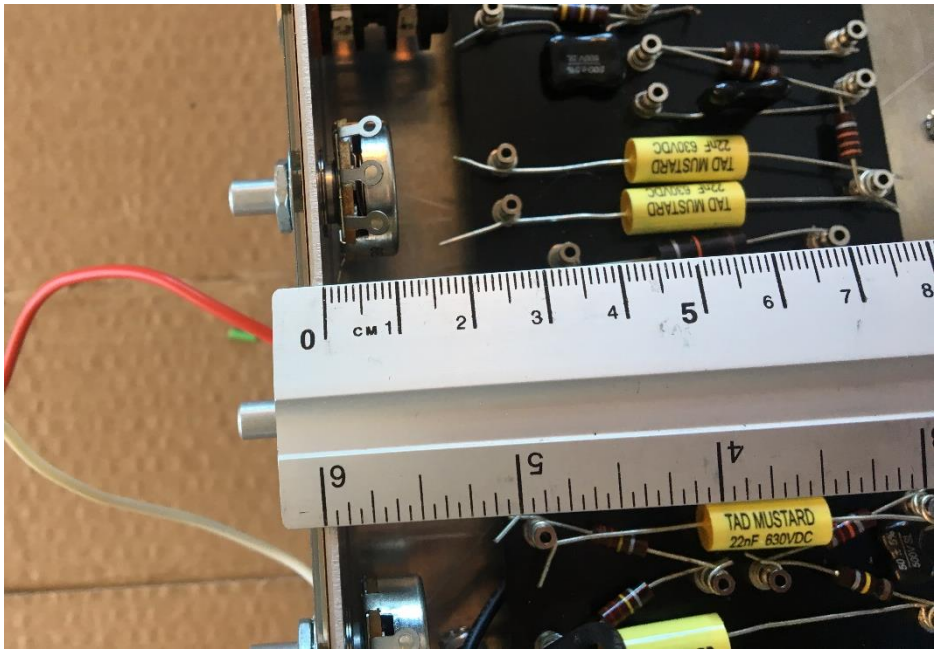
Figur 2.7: Inn- og undersiden av chassiet til gitarforsterkeren, ferdig loddet.

En løsning for integrering av steppermotorene som antakelig vil kunne unngå overnevnte problemer er:

1) Steppermotorene påmonteres tannhjul eller reimskive og settes på oversiden av chassiet, direkte over potentiometrene.
2) Nye potentiometre med litt lengre skaft monteres med klaring til aksling på innsiden med fastskrudd/pålimt tannhjul eller reimskive.
3) Det skjæres en slisse i chassiet slik at kjede eller reim får fri passasje.

Denne løsningen vil kreve potentiometre jeg for øyeblikket ikke har tilgang på, som også vil ta tid å fremskaffe. Jeg vurderer likevel løsningen som gjennomførbar. Det er også rikelig med plass til Belaen på innsiden av forsterkerkassen, men kan tenkes å fordre ekstra ventilasjon og maksimal avstand til utgangsrør på grunn av varmen.

Figur på neste side viser plassmessig mulighet for annen type potentiometer. Det må imidlertid en ekstra plate til for feste av potentiometer. Mellom denne platen og frontpanel vil tannhjul eller reimskive kunne få plass.



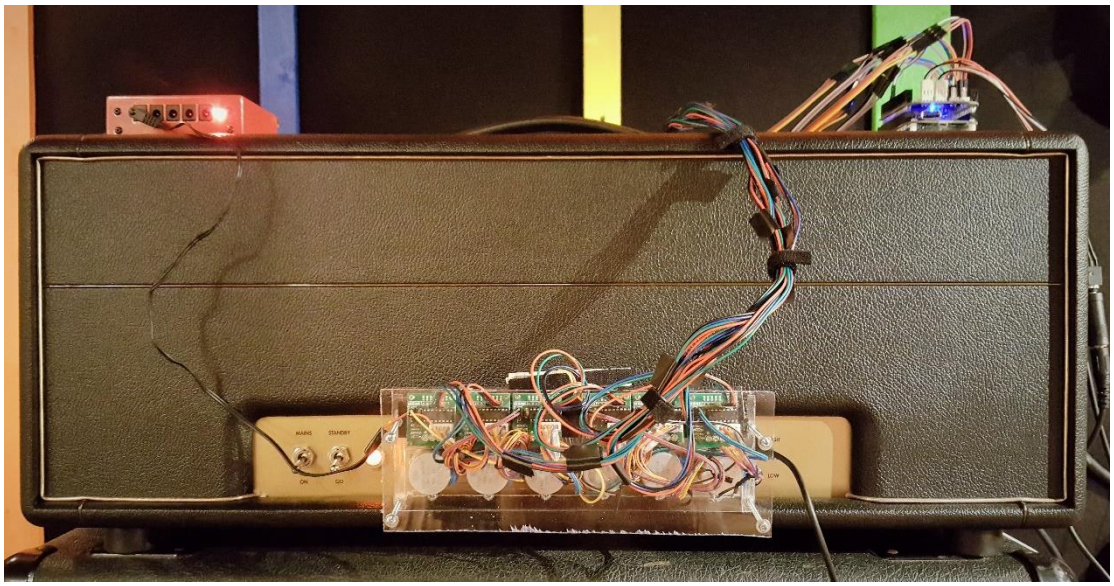
Figur 2.8: Tilgjengelig klaring mellom potentiometer og turret board er ca. 1 centimeter.

Figur 2.9: Ulike typer motorer, servoer og motoriserte potentiometre som er vurdert.

Figur 2.9 viser ulike motorer som var tilgjengelig via Ebay og innkjøpt for vurdering. De tre til høyre er ikke steppermotorer og vil være mer utfordrende å styre med binære signaler fra Belaen. Dette kunne dog vært oppnådd med

spesiallagde drivkretser. Valget falt på den lengst til venstre, som jeg har benyttet tidligere i bachelorprosjektet. Løsningen som innen tidsrammen kunne velges for oppgaven ble da en forbedret utgave av et oppheng med steppermotorer og drivkretser jeg bygde til mitt bachelorprosjekt, se figurer under.

Figur 2.10: Frontpanel på forsterkeren med fleksible mellomkoblinger for tilkobling av stepper motorer.



Figur 2.11: Prototypen skrudd fast på mellomkoblingene. Belaen på toppen til høyre, strømforsyning til drivkretser på toppen til venstre.

2.7 Hvordan gjøre en forsterker lydløs

Det å si at et Marshall-forsterker kan spille høyt er alt annet enn en overdrivelse. I et scenario der forsterkeren skal brukes uten at de som står i nærheten av den

er klar over det kan en hørselskade lett oppstå. Dette er ett av de sikkerhetsmessige kravene jeg har valgt å ta tak i under arbeidet med prosjektet.

Et signal som sendes gjennom en forsterker blir forsterket mange titalls desibel før det når utgangen, noe som medfører store mengder energi. Dersom denne energien ikke blir omsatt til noe annet (for eksempel til lyd gjennom et høyttalerkabinett) vil dette gjøres om til varme i forsterkerens siste ledd: utgangstransformatoren. Dette kan resultere i at viklingene på innsiden rett og slett brenner opp. Det er heller ikke slik at et vanlig lydkort kan motta dette signalet direkte – da vil samme skade skje ved lydkortets inngang i stedet. Løsningen vil kreve en innretning mellom gitarforsterkeren og lydkortet som kan senke dette signalets styrke. Jeg har valgt å gå for attenuering. I prosjektet benyttet jeg en Suhr Reactive Load. Dette er en reaktiv last, noe som kort forklart betyr at den tar inn signalet fra forsterkeren og omsetter deler av det til varme gjennom motstand. Motstanden i en reaktiv last er også, intuitivt nok, reaktiv, som betyr at motstanden og fase endrer seg noe med frekvensen den mates med. Den reaktive egenskapen skal noenlunde tilsvare den man finner i lasten som et høyttalerkabinett leverer og vil derfor gjøre at gitarforsterkeren i teorien oppfører seg likt som om den skulle vært brukt konvensjonelt.

En gitarforsterker brukes konvensjonelt sammen med et høyttalerkabinett. Høyttalerne som benyttes i slike kabinetter er mange og forskjellige, og kan drastisk endre den lydlige karakteren på det som sendes ut i rommet. Dette skyldes både de rent akustiske egenskapene til kabinettet og høyttalerne, men også frekvensresponsen til høyttalerne. En ofte brukt høyttaler til Marshall-forsterkere er Celestion Greenbacks, som har svært lav respons på frekvensinnhold over 6 kHz og under 80 Hz, men har en resonans som strekker seg mellom 2 og 4 kHz. For å ha en nøytral lytteopplevelse vil disse høyttalerne være uegnet, men til å farge gitarlyd kan den fjerne uønsket frekvensinnhold som oppstår av subharmonisk innhold og uønsket høyfrekvent innhold som følge av forvrengning. Videre ligger resonansen mellom 2 og 4 kHz i området der gitarplekter bryter gjennom og kan dermed gi økt opplevelse av anslag i det lydlige resultatet. Lyden av en gitarforsterker som vrenges uten denne filtreringen er ofte svært skarp og gir lav opplevelse av tonehøyde. Denne filtreringen er altså ønskelig, og jeg ønsker å oppnå dette uten å involvere et fysisk kabinett.

Dette kan gjøres på flere måter, deriblant analoge og digitale filtre, eller gjennom impulsresponser (Walker, 2005). Impulsresponser av gitarkabinetter har blitt et marked på nett, der aktører som Ownhammer, Redwirez og Celestion ofte omtales på diverse fora (eksempelvis https://forum.fractalaudio.com/forums). Slike impulsresponser lages gjennom å sende en rask puls gjennom et medium for så å måle reaksjonen i etterkant, her med en mikrofon. Dersom et separat signal konvolveres gjennom denne impulsresponsen vil man få en tilnærming til frekvensresponsen av utgangspunktet for impulsresponsen. Gjennom å konvolvere lydstrømmen fra den reaktive lasten vil brukeren derfor få en nær

tilnærming av lyden av et mikrofonopptak av en gitarforsterker gjennom et høyttalerkabinett.

Når Belaen har forsynt gitarforsterkeren med innsignal fra gitaristen i motsatt ende av kommunikasjonslinken mottar den dermed et høykvalitets attenuert utgangssignal fra Suhr Reactive Load som går inn i Belaen og sendes tilbake til gitaristen. Internt i programmet i Juce påføres konvolvering gjennom en impulsrespons av et Greenback-kabinett fanget opp med en Royer 121-mikrofon. Derfra går lydstrømmen ut fra Juce, inn på lydkortet og ut i monitorering. Dette var gjenstand for mye lek og interessant eksperimentering, se resultat.

Del 3: Resultat

3.1 Punktvis gjennomgang av problemstilling

Problemstillingene jeg har tatt for meg var som følger:

1. Hvilke tekniske utfordringer oppstår under utviklingen av en gitarforsterker som kan fjernstyres og brukes over nett?

Det er selvfølgelig mange praktiske og mekaniske utfordringer når så vidt ulike teknologier skal fungere sammen. Problemstillingen er forsøkt redegjort så godt som mulig i denne rapporten. Jeg vil her særlig peke på tilgang til egnede mekaniske komponenter som spesiell utfordring. Skal produktet bli perfekt og helintegrert kreves tilgang til verktøymaskiner og skreddersydde spesialbestilte komponenter (potentiometre, motorer, festemekanismer og innkapsling). Programvare som skal fungere godt i sanntid krever avanserte programmeringsteknikker (eksempelvis tråding), og feilsøking kan være vanskelig. Jeg har redegjort for to ytterligere utfordringer utenfor dette i kapittel 4.4.

2. Hvordan kan en Audio-over-IP-applikasjon bygges fra bunnen av ved hjelp av åpne digitale utviklingsverktøy?

Svaret er først og fremst at det er mulig med de verktøyene jeg har valgt. Det har vært en fordel å ta i bruk C++ som programmeringsspråk, da C++ ofte lar seg kompilere til rask kode og benyttes ofte i audio-sammenheng (VST, AAX skrives i C++). Koden jeg har skrevet kan relativt lett rekontekstualiseres i en annen audioapplikasjon, som for eksempel fjernstyrt analog ekkomaskin, gitarpedaler, kanalstriper til miksebord og klangrom etc.

3. Er rammeverket Juce og mikrokontrolleren Bela egnede verktøy for denne oppgaven?

Det korte svaret er ja. Jeg anser særlig Juce som et potent verktøy for slike anvendelser. Dette skyldes i stor grad kvaliteten på bibliotekene Juce tilbyr. Selve lyddelen av Belaen er av høy nok kvalitet til dette formålet. Litt svak prosessor på Belaen kan imidlertid føre til lang kompileringstid i større prosjekter. Total CPU-ressursforbruk, inkludert alle systemprosesser, ble hos programmet i Juce og Bela henholdsvis 1 og 45 %, da med forbehold om at programmene kjører på maskiner med vesentlig ulik prosesseringskraft.

3.2 Måloppnåelse, praktisk gjennomføring

Målsettingen for det praktiske arbeidet var som følger:

1. Det skal ikke være nødvendig for brukeren å ha annet utstyr enn hva man kan finne i et typisk hjemmestudio. Det vil si: datamaskin, lydkort og høyttalere (pluss gitar og jack-kabel).

Dette punktet er oppnådd til det fulle. Fordelen ved dette er potensiell kort tid brukt på oppsett fra brukerens side.

2. Kompetansegulvet for bruk skal ikke gå utover det å justere inn en gitarforsterker til ønsket lydlig resultat, bruk av enkle, digitale lydprosesseringsverktøy (av typen plugin for DAW-er) og å skrive inn korrekt IP-adresse og portnummer for enheten.

Jeg velger å anse dette punktet som oppnådd. Dog vil drøftingsdelen av oppgaven fremme noen forslag som vil kunne øke brukervennligheten.

3. Det grafiske brukergrensesnittet skal være oversiktlig og til en viss grad gjenspeile hvordan den faktiske gitarforsterkeren ser ut.

Dette faller på det subjektive.

4. En gitarforsterker brukes til vanlig i kombinasjon med et høyttalerkabinett, noe som kan være veldig høylytt. Av sikkerhetsmessige grunner bør forsterkeren kunne innrettes slik at den ikke er i stand til å forårsake en hørselskade der den er fysisk plassert under bruk.

Den reaktive lasten gjør selve forsterkeren praktisk talt lydløs. Lydnivået fra den reaktive lasten er svært lavt (interne vibrasjoner kan høres fra innsiden) – det går an å kommunisere med hvisking ved siden av den. Derfor velger jeg å anse dette målet som oppnådd.

5. Teknologien som muliggjør fjernstyringen og bruken skal helst bygges inn i forsterkeren for å minimere tid på oppsett.

Dette lot seg ikke gjennomføre innenfor tidsrammen med tilgjengelig mekanisk utstyr, verktøy og komponenter.

Som er generell egen vurdering vil jeg dra fram at forsterkeren og fjernstyringssystemet i et bruksscenario fungerer slik som målsettingen siktet etter på et funksjonelt nivå.

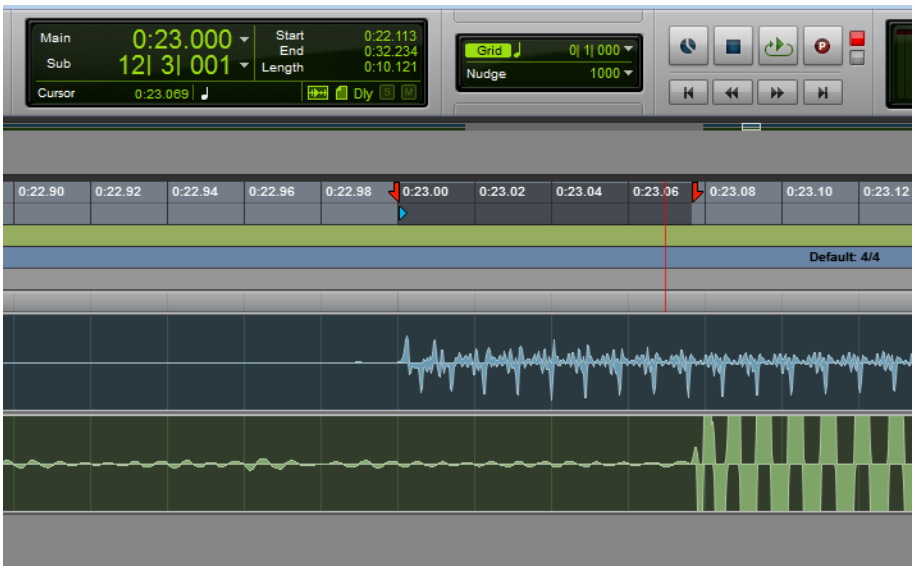
Del 4: Diskusjon og veien videre

4.1 Lydlig resultat

Jeg er særs fornøyd med det lydlige resultatet. Med mine ører, er det tydelig at det er en Marshall-stakk jeg hører i monitor. I forkant av prosjektet hadde jeg en, tydeligvis ubegrunnet, mistanke om at lydkonvertering hos Bela kanskje ville farge lyden merkbart. Dette var slett ikke tilfelle, og skaper motivasjon for videre eksperimentering med plattformen.

4.2 Latency

I digitale systemer er latency, som nevnt, uunngåelig. Den totale tidsforsinkelsen i systemet som foreligger har jeg målt til rundt 68 millisekund. Dette er for høyt til sanntidsbruk i en sanntids musikalsk setting. Reduksjon av tidsforsinkelse kan skje på flere måter. Den første er oppgradering til gigabit nettverksteknologi. Dernest systematisk reduksjon av buffer i alle ledd og eksperimentering med ulike pakkestørrelser og formattering. Denne typen optimalisering ble det ikke tid til i denne omgang. Jeg opplevde en stor nedgang i latency da jeg implementerte ASIO-kompatibilitet i Juce, fremfor Windows sine standarddrivere.



Figur 4.1: Enkel primitiv latencymåling. Utført med ekstern maskin fra fjernstyringssystemet, og målt mellom direktelyd fra gitar og utgang fra lydkortet i maskinen der Juce kjører.

4.3 Sikkerhet og konsekvenser

De positive sidene av AoIP har også en bakside. Et hvert datasystem bør antas å kunne ha sikkerhetshull, noe som kan ha konsekvenser. For AoIP-systemer er selve implementasjonen av lydoverføringen over nett fellesnevneren for hva som bør håndteres forsiktig. Cyberangrep mot dette kan ta form av direkte avlytting, sabotasje av tilgjengelighet og systemets integritet og åpning av kommunikasjonskanalen uten eierens viten eller samtykke. Sikkerhet mot slike angrep er en omfattende oppgave og ikke gjennomført, men vil være viktig i videreutviklingen mot et eventuelt ferdig produkt.

Videre er det også en generell samtykkeproblematikk tilstede ved bruk av systemer med nettbasert lydoverføring. En åpen mikrofon fanger opp lyd generelt – ikke bare de lydlige hendelsene den er tiltenkt å fange opp. Gjennom bruk av AoIP-systemer utsetter man alle rundt for de potensielle konsekvensene som ligger i AoIP-konseptets natur. Bruken av AoIP-teknologi burde derfor gjøres i en kontekst der omverden kan sies å være isolert. Videre bør det ikke være mulighet for at personer trer innenfor systemets lydfangerekkevidde uten å være klar over det eller uten å kjenne til de potensielle konsekvensene.

Angående sikkerhet er det også slik at AoIP-systemer kan misbrukes på lik linje med andre former for overvåkningsteknologi. Informasjonslekkasjer og avlytting ved hjelp av IoT- og AoIP-teknologi er dessverre å forvente fremover.

4.4 Absurd hardwarerelatert problematikk under gjennomføringen

I kapittel 2.7 gav jeg en oversikt over de generelle utfordringene man støter på i en AoIP-implementasjon. Når løsningene på alle disse utfordringene skal virke sammen kreves mye kode, noe som fort svekker oversiktligheten i arbeidet. Hvis det oppstår feil eller problemer eksternt fra selve programkoden kan det da være svært tidkrevende å utelukke syntaks- eller systemfeil. Det var særlig to slike tilfeller under mitt praktiske arbeid med prosjektet som krevde mye tid for å finne ut av. Den første var akutt blåskjermproblematikk. Under implementeringen av kommunikasjonen mellom Juce og Bela begynte jeg plutselig å få blåskjerm med omtrent 15-30 minutters intervaller. Symptomene rett før blåskjerm var at lyden ble hakkete. Min første mistanke var at dette var koderelatert, da kommunikasjonen er avhengig av teknikker for flytting av informasjon jeg ikke er erfaren med. Under feilsøkingen oppsto selvfølgelig en bekymring for hvilke konsekvenser blåskjermen kunne ha for både prosjektet mitt og datamaskinen (og dens programmeringsressurser) som prosjektet avhenger av. Feilen ble funnet etter en uke med febrilsk leting. Etter hvert som programkoden i Belaen vokste økte også strømbehovet – som trakk mer strøm fra datamaskinen enn den klarte å levere. Dermed kunne ikke datamaskinen håndtere kommunikasjon med lydkortet, og blåskjermen inntraff. En annen utfordring var at Juce ikke klarte å feste seg til en fast bufferstørrelse i bufferet opp mot lydkortet. Ved utskrift av

bufferstørrelse til konsoll var det nært tilfeldig hvilken verdi den ble satt til. Vanligvis settes bufferstørrelse til forskjellige toerpotenser (32, 64, 128, 256 osv.), men her altså helt vilkårlig. Grunnet mulighet for bufferstørrelser som ikke er delelige på heltall var enkel algoritmisk behandling av informasjon derfor ikke mulig. Feilen virket å være at M-Audio-lydkortet ikke lengre var støttet på Windows 10, da innkjøp av nytt lydkort endret oppførselen.

4.5 Brukervennlighet

Det er særlig et punkt som kan være frustrerende med å koble et slikt system opp mot ukjente og ulike lokalnett – IP-adressetildeling. I et videreutviklingsscenario ønsker jeg å implementere bonjour slik at Juce og Bela automatisk finner hverandre og sine respektive IP-adresser gitt at de er på samme lokalnett. I tillegg kan man tenke seg bruk av dynamisk DNS slik at enhetene har nås gjennom faste domenenavn. Dette kan også gjøres gjennom en egen skyløsning som både Juce og Bela kobler seg opp mot etter oppstart og registrerer seg.

Videre kan tilbakemeldinger fra musikere under et fremtidig utprøvingsscenario kunne gi innspill til forbedringer som man kanskje ikke umiddelbart ser selv.

Del 5: Konklusjon

Denne rapporten dokumenterer prosessen av å bygge en gitarforsterker som kan fjernstyres og brukes over nett. Resultatet og den praktiske utprøvingen viser at dette faktisk er mulig. Jeg tror tilnærmingen benyttet i dette prosjektet har et enda større potensial innen musikkteknologien enn hva som er dokumentert her.

At teknologien som er benyttet og muliggjør denne prototypen er såpass tilgjengelig tror jeg bærer bud om at vi vil se flere tilsvarende produkter i årene som kommer, både skreddersydde og kommersielle. Videre er teknologi som Bela og Juce så kostnadsmessig gunstig at det vil kunne åpne for ny og kreativ bruk. Større tilgjengelighet og bedre brukervennlighet i AoIP-teknologi vil styrke muligheter for ressursdeling, musikalsk samarbeid og samspill over avstand.

Latency er et generelt problem med alle digitale systemer for overføring og behandling av lyd. Likevel har systemer slik som Dante ved hjelp av spesialisert hardware oppnådd mer enn akseptable resultater, mens rent software-baserte systemer henger litt etter per nå. Prototypen jeg har skapt har forbedringsmuligheter på dette området, og konkrete tiltak er drøftet.

Avslutningsvis vil jeg si at dette har vært en krevende og spennende musikkteknologisk reise som har gitt et resultat jeg er veldig fornøyd med. Jeg er ytterligere inspirert til arbeid med denne type teknologi.

Referanser

Aalen, I. (2015). *Sosiale medier*: Fagbokforlaget.

Audinate. (2019). Audinate Web Page. Retrieved from https://www.audinate.com/

Bela. (2019). Bela Home Page. Retrieved from https://bela.io/

Cáceres, J.-P., & Chafe, C. (2010). JackTrip: Under the hood of an engine for network audio. *Journal of New Music Research, 39*(3), 183-187.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). *Introduction to algorithms*: MIT press.

Dictionary, C. (Ed.) (2019).

Dvecko, M. (2019). Introduction To Concurrent Programming: A Beginner's Guide. Retrieved from https://www.toptal.com/software/introduction-to-concurrent-programming

Emerick, G. (2013). *Here, there and everywhere*: Novo Século.

Foundation, A. (2019). Arduino Web Page. Retrieved from https://www.arduino.cc/

Foundation, P. (2019). Raspberry Pi Home Page. Retrieved from https://www.raspberrypi.org/

Hardin, M. J. (2019). *Innovative Music Production Approaches Empowered by Internet of Things Technologies*. Retrieved from https://marquesjhardin.wordpress.com/category/ph-d-research/page/1/

Jack, R. H., Stockman, T., & McPherson, A. (2016). *Effect of latency on performer interaction and subjective quality assessment of a digital musical instrument.* Paper presented at the Proceedings of the audio mostly 2016.

Jacktrip. (2019). Jacktrip Web Page. Retrieved from https://ccrma.stanford.edu/software/jacktrip/

Katz, M. (2010). *Capturing sound: how technology has changed music*: Univ of California Press.

Kurose, J. F., & Ross, K. (2017). *Computer networking: A top-down approach featuring the internet, 7/E*: Pearson Education India.

Lester, M., & Boley, J. (2007). *The effects of latency on live sound monitoring.* Paper presented at the Audio Engineering Society Convention 123.

McPherson, A. P., Jack, R. H., & Moro, G. (2016). Action-sound latency: Are our tools fast enough?

Norman, D. A. (2013). Design of everyday things: Revised and expanded. *Hachette, New York*.

ROLI. (2019). Juce Home Page. Retrieved from https://juce.com/

Stompenberg. (2019). Stompenberg FX. Retrieved from https://www.thomann.de/blog/en/stompenberg-effects/

Tanenbaum, A. S. (2016). *Structured computer organization*: Pearson Education India.

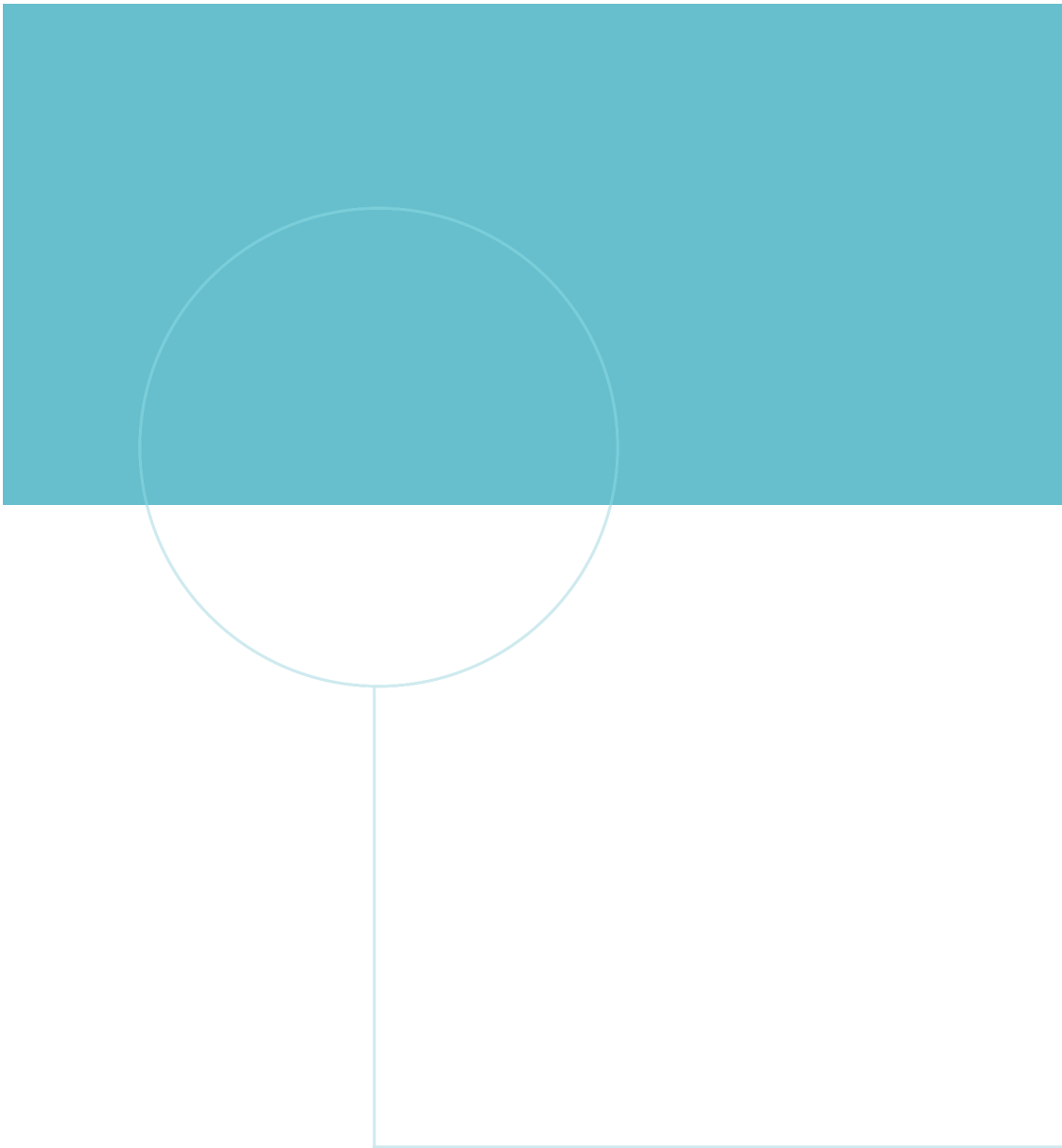
Valderhaug, A. K. (2016). Millionmannen. Retrieved from https://p3.no/musikk/millionmannen/

Walker, M. (2005). Convolution Processing With Impulse Responses. *Sound on Sound.* Retrieved from https://www.soundonsound.com/techniques/convolution-processing-impulse-responses

Del 6: Vedlegg

Vedlegg 1: Videoopptak av fjernstyringssystemet i bruk og layout for utprøvingsoppsett.

Vedlegg 2: Videoopptak av gitarforsterkerens ratt som blir manipulert av en oppheng med steppermotorer.



NTNU

Kunnskap for en bedre verden