VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

DIPLOMOVÁ PRÁCE

Brno, 2018 Bc. Robin Zoň

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ
DEPARTMENT OF TELECOMMUNICATIONS

SOFTWARE PRO DIGITÁLNÍ MIXÁŽNÍ PULT
SOFTWARE FOR DIGITAL MIXING CONSOLE

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. Robin Zoň

VEDOUCÍ PRÁCE
SUPERVISOR

doc. Ing. Jiří Schimmel, Ph.D.

BRNO 2018

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní obor **Audio inženýrství**
Ústav telekomunikací

Student: Bc. Robin Zoň ***ID:*** 165032
Ročník: 2 ***Akademický rok:*** 2017/18

NÁZEV TÉMATU:

Software pro digitální mixážní pult

POKYNY PRO VYPRACOVÁNÍ:

Navrhněte a realizujte software pro operační systém Windows, který bude umožňovat vícekanálové zpracování zvukových signálů v reálném čase minimálně 8 vstupními a 8 výstupními jednotkami, směrování signálu mezi těmito jednotkami a vkládání VST plug-in modulů. Jako zvukové rozhraní použijte technologii ASIO nebo ovladače operačního systému s nízkým dopravním zpožděním, např. WDM. Doplňte tento software o možnost dálkového řízení v sítích TCP/IP a grafickým uživatelským rozhraním pro webový prohlížeč.

DOPORUČENÁ LITERATURA:

[1] BOURLANGER, R., LAZZARINI, V. The Audio Programming Book. The MIT Press, 2011. ISBN: 978-0-2-2-01446-5

[2] PIRKLE, W. Designing Audio Effect Plug-ins in C++. Focal Press, 2013. ISBN: 978-0-240-82515-1

[3] KIENTZLE, T. A Programmer's Guide To Sound. Addison-Wesley, 1998. ISBN: 0-201-41972-6

Termín zadání: 5.2.2018 ***Termín odevzdání:*** 21.5.2018

Vedoucí práce: doc. Ing. Jiří Schimmel, Ph.D.
Konzultant:

prof. Ing. Jiří Mišurec, CSc.
předseda oborové rady

ABSTRAKT

Tato práce se zabývá návrhem a realizací softwaru pro digitální mixážní pult postaveného na platformě Windows. Tento software umožňuje vícekanálové zpracování zvukového signálu v reálném čase, připojení několika vstupních a výstupních jednotek, směřování signálu mezi těmito jednotkami a vkládání a správu VST plug-in modulů. Software využívá zvuková rozhraní připojená pomocí technologie ASIO. Práce je rozdělena na několik aplikací. Hlavní aplikace provádějící výpočty zvukových vzorků a vkládání a správu plug-in modulů je naprogramována v jazyce C++ s využitím technologie JUCE. Tu lze ovládat skrze své vlastní lokální rozhraní nebo webové ovládací rozhraní naprogramované v jazyce TypeScript, které využívá technologii React. Webové rozhraní umožňuje řízení VST plug-in modulů díky své vlastní implementaci řízení.

KLÍČOVÁ SLOVA

Software pro digitální mixážní pult, zpracování zvuku v reálném čase, Windows, ASIO, VST, JUCE, React, Node.js, C++, TypeScript, XML, JSX, OSC, TCP, WebSocket, WOAP, PAG

ABSTRACT

This thesis describes the design and implementation of a software for digital mixing console built on the Windows platform. This software is designed to offer real-time multi-channel audio processing using multiple input and output units, signal routing between these units and insertion and management of VST plug-in modules. The software uses an audio interface connected with ASIO technology. The thesis is divided into several applications. Main application which computes audio samples and allows insertion and management of plug-ins is programmed in C++ using JUCE technology. This application can be controlled with its own local graphical interface or with web control interface, which is programmed in TypeScript with the use of React technology. Web interface allows user to control VST plug-in modules with its own custom implementation of plug-in control.

KEYWORDS

Software for digital mixing console, real-time audio processing, Windows, ASIO, VST, JUCE, React, Node.js, C++, TypeScript, XML, JSX, OSC, TCP, WebSocket, WOAP, PAG

ZOŇ, Robin. *Software pro digitální mixážní pult*. Brno, 2018, 67 s. Diplomová práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedoucí práce: doc. Ing. Jiří Schimmel, Ph.D.

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „Software pro digitální mixážní pult“ jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu diplomové práce panu doc. Ing. Jiřímu Schimmelovi, Ph.D. za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Brno ....

podpis autora

Faculty of Electrical Engineering
and Communication
Brno University of Technology
Purkynova 118, CZ-61200 Brno
Czech Republic

http://www.six.feec.vutbr.cz

PODĚKOVÁNÍ

Výzkum popsaný v této diplomové práci byl realizován v laboratořích podpořených z projektu SIX; registrační číslo CZ.1.05/2.1.00/03.0072, operační program Výzkum a vývoj pro inovace.

Brno ....

podpis autora

EVROPSKÁ UNIE
EVROPSKÝ FOND PRO REGIONÁLNÍ ROZVOJ
INVESTICE DO VAŠÍ BUDOUCNOSTI

Obsah

Seznam obrázků

Seznam výpisů

Úvod

Vícekanálové zpracování zvuku je nedílnou součástí společenských událostí, koncertů nebo produkce hudebních děl. Analogové systémy pro zpracování zvuku jsou však mnohdy rozměrné, nekompaktní a cenově nedostupné, což společně s rostoucím požadavkem na nízkou cenu a jednoduchost transportu otevírá příležitost systémům digitálním.

Tyto systémy často poskytují vyšší modularitu díky zásuvným modulům zvukových efektů, které se zařazují do signálové cesty. Jednou z možností je použití technologie VST (Virtual Studio Technology), která je mezi uživateli velmi rozšířená. Digitální systém tak může nabízet volbu stovek nebo i tisíců zvukových efektů a zároveň zachovávat kompaktní rozměry s možností uložení a načtení uživatelských nastavení, transportu bez neustálého přepojování zvukových procesorů a přívětivou cenu.

Komerční řešení těchto systémů jsou však spjata s konkrétní hardwarovou platformou, na kterou jsou přizpůsobeny. To může přinášet omezení ve zvukové kvalitě, počtu vstupně-výstupních jednotek nebo dalších parametrech systému. Rozšíření či úprava takové platformy pak může být vlivem nekompatibility zařízení různých výrobců velmi nákladná nebo nemusí být vůbec možná.

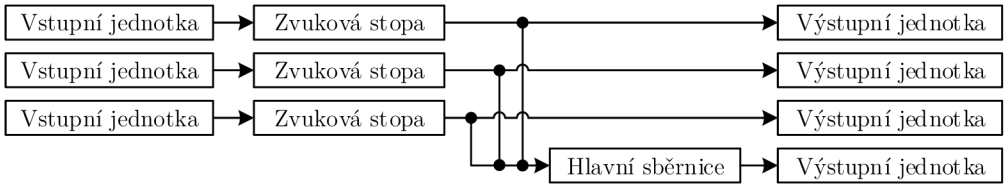
Cílem této práce je navrhnout a realizovat software umožňující vícekanálové zpracování zvukového signálu v reálném čase pro operační systém Windows. Tento systém má využívat technologii ASIO (Audio Stream Input/Output) a umožnit tak připojení k různým zvukovým zařízením, jichž je na trhu bohatý výběr. Software má dále nabízet vkládání a správu VST plug-in modulů a vzdálené řízení přes webový prohlížeč. Ve spojení s vhodnou hardwarovou konfigurací má pak tato práce sloužit jako software pro digitální mixážní pult.

1 Digitální mixážní pult

Digitální mixážní pult je zařízení, které umožňuje číslicové zpracování zvukového signálu několika kanálů a jejich číselnou sumaci. Zvukovými vstupy mixážního pultu bývají obvykle externí zdroje signálu (mikrofony, hudební nástroje, ostatní zvukové procesory) nebo méně často interní zdroje signálu, pokud mixážní pult umožňuje nahrávání a přehrávání zvukového signálu uloženého v interní paměti [1].

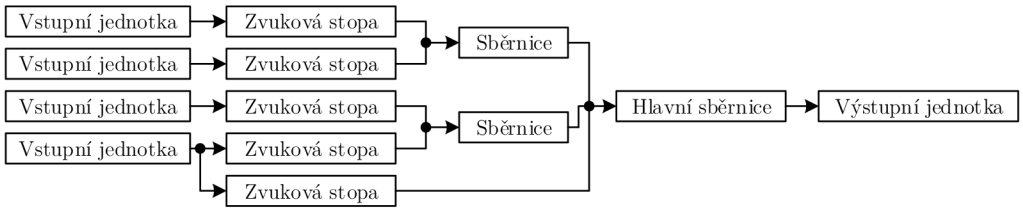
1.1 Struktura signálových cest

Jádrem digitálního mixážního pultu je jednotka umožňující práci se zvukovými kanály a jejich následnou sumaci. U jednoduchých pultů jsou vstupní jednotky staticky přiřazeny zvukovým stopám uvnitř zařízení a ty staticky přiřazeny jednotkám výstupním. Často jsou navíc všechny zvukové stopy přiřazeny do hlavní sběrnice (často zvané „Master"), která má své vlastní výstupní jednotky. Toto zapojení běžně mívají vícekanálové zvukové karty, které, ačkoli jsou primárně určeny pro nahrávání, je možné použít i jako mixážní pult (např. RME Fireface 800 [2]). Struktura zapojení jednoduchého mixážního pultu je ilustrována na obrázku 1.1.



Obr. 1.1: Vnitřní struktura jednoduchého digitálního mixážního pultu

U větších mixážních pultů je počet zvukových stop a sběrnic dynamický, což umožňuje vytvářet stromovou strukturu toku signálu. Zvukové stopy lze přiřadit do sběrnic a tyto sběrnice přiřadit hlavní sběrnici, viz obrázek 1.2. Tuto strukturu nabízí většina výrobců moderních digitálních mixážních pultů (např. Soundcraft Vi1 [3]).



Obr. 1.2: Vnitřní struktura digitálního mixážního pultu s dynamickými sběrnicemi

1.2 Zvukové stopy a sběrnice

Pro rozlišení mezi fyzickými kanály (vstupními a výstupními jednotkami) zařízení a interními kanály, kde probíhá číslicové zpracování, je pro interní kanály použit výraz „stopa" (v anglickém jazyce „track"). Někteří výrobci rozdělují interní kanály na stopy a sběrnice, kdy stopám lze přiřadit pouze jeden vstup a výstup, zatímco sběrnicím několik [4]. V této práci však toto rozdělení použito není, neboť i stopám lze přiřadit několik různých vstupů a výstupů. To můžou být buď fyzické jednotky mixážního pultu, nebo jiné stopy. Platí tedy, že sběrnice je zároveň stopou.

Stopy lze rozdělit dle spojení na tři typy: zvukovou stopu (*audio track*), skupinovou sběrnici (*bus track*, *mix bus* nebo *group*) a výstupní sběrnici (*output bus*). Zvukové stopě lze na vstup přiřadit vstupní jednotku zvukového zařízení a na výstup sběrnici. Skupinové sběrnici na vstup zvukové stopy nebo skupinové sběrnice a na výstup jiné skupinové sběrnice nebo výstupní sběrnice. Výstupní sběrnici lze přiřadit stejné stopy jako skupinové sběrnici, výstup je však možný pouze do výstupní jednotky mixážního pultu.

Speciálním případem skupinové sběrnice je efektová sběrnice (*effect bus* nebo *fx bus*). Je to taková sběrnice, která je přímo napojena na efektovou jednotku. Jejím vstupem je stejný signál jako u ostatních sběrnic, výstupem je však signál zpracovaný efektovou jednotkou. Takové sběrnice jsou užitečné z důvodu omezeného výpočetního výkonu, kdy by jinak nebylo možné aplikovat zvukové efekty na více stop (typicky efekty upravující dozvuk). To zároveň umožní jednodušší kontrolu nad efektovým procesorem, neboť není nutné nastavovat parametry zvukového efektu pro každou stopu zvlášť.

Stopa může mít několik zvukových kanálů, běžně jeden (monofonní signál) nebo dva (stereofonní signál). Větší konfigurace (např. prostorový zvuk) jsou dostupné pouze na specializovaných mixážních pultech (např. Yamaha DM2000VCM [5]) a pro účely živého ozvučení se používají zřídka.

Výše uvedená pojmenování jsou z akademického hlediska nepřesná a zavádějící, jsou však zavedena v praxi a jsou známa širokou veřejností.[1]

[1] Český jazyk nemá přesné překlady nebo ekvivalentní výrazy mnohých těchto termínů. Pojmenování proto není možné chápat doslovně. Z logiky věci je každá sběrnice sběrnicí skupinovou, stejně jako každá stopa zvukovou. Tato práce tedy používá obecné pojmenování „stopa" pro všechny výše popisované prvky vnitřní struktury mixážního pultu. Těmito prvky jsou: zvuková stopa, skupinová sběrnice, výstupní sběrnice a efektová jednotka.

1.3 Parametry stop

Zvukové stopy a sběrnice v digitálním mixážním pultu typicky obsahují několik parametrů, které jsou přebrány z analogových mixážních pultů [6]. Pro soubor těchto parametrů existuje ustálený název „Channel strip" [7]. Parametry lze rozdělit do několika sekcí:

Předzesilovač

Přístupný pouze pro zvukové stopy připojené na vstupní jednotku mixážního pultu. Umožňuje nastavení zesílení vstupního předzesilovače, otočení fáze, zapnutí 48V napájení vstupní linky předzesilovače (tzv. „fantomové napájení"), zapnutí útlumového článku či filtru typu horní propust, který lze ladit. Pokud není mixážní pult vybaven dálkově řiditelnými předzesilovači, není možné tyto parametry nastavit z řídící konzole a uživatel je musí nastavit manuálně u vstupní jednotky.

Šumová brána

Nastavení efektu typu šumová brána či častěji efektu expander [8]. Sekce může být doplněna o nastavení pásmové propusti řídící větve.

Ekvalizér

Nejčastěji parametrický, tří až pětipásmový ekvalizér s volitelnou frekvencí, zesílením a šířkou pásma. Krajní pásma lze přepínat mezi efekty typu shelf a peak. Jednodušší ekvalizéry nabízí pouze volitelnou frekvenci a zesílení.

Kompresor a limiter

Nastavení efektu typu kompresor nebo limiter [8] často obsahuje podobná nastavení jako šumová brána. Někdy je tato sekce zařazena před sekci ekvalizéru.

Zesílení a panorama

Nastavení zesílení bývá nejčastěji realizováno tahovým enkodérem stejně jako u analogových mixážních pultů. Nastavení panoramatu zajišťuje otočný potenciometr nebo enkodér s kruhovou indikací. Součástí sekce je také tlačítko funkce úplného ztlumení (*mute*), které při aktivaci nastaví zesílení na $-\infty\,\mathrm{dB}$.

Aby se ušetřil počet tahových potenciometrů a zmenšila se velikost celé konzole, jsou stopy organizovány do vrstev, mezi kterými je možné přepínat [9]. Mixážní pulty s touto funkcí mají zpravidla potenciometry motorizované, aby po přepnutí

do jiné vrstvy potenciometr nastavil svou polohu dle hodnoty zesílení příslušného kanálu. Někteří výrobci, zřejmě za účelem snížení výrobních nákladů, mixážní pulty neosazují motorizovanými potenciometry. Nastavení polohy potenciometru do odpovídající hodnoty musí uživatel provádět ručně.

1.4 Efektová jednotka

Digitální mixážní pult je vybaven efektovými jednotkami. Každá jednotka obsahuje jeden nebo více nezávislých zvukových efektových procesorů. Většina mixážních pultů nabízí omezený výběr těchto efektů a nedává žádnou možnost banku efektů rozšířit. Někteří výrobci však rozšíření umožňují, a to např. pomocí rozšiřující karty SoundGrid, která umožňuje připojení externího procesoru se systémem MultiRack dedikovaného pro výpočet zvukových efektů [10]. Výběr efektů je však omezen pouze na produkty od společnosti Waves, což možnosti rozšíření otevírá pouze z části.

Pokud není uživatel s výběrem zvukových efektů spokojen, může připojit externí efektový procesor. To ale nemusí být příliš praktické z ekonomických důvodů nebo zvláště tehdy, má-li být pult snadno přenositelný.

1.5 Vzdálené ovládání

Jedním z benefitů digitálních mixážních konzolí je také možnost vzdáleného řízení. Na dnešním trhu již většina konzolí podporuje ovládání pomocí technologie Ethernet nebo WiFi. Velmi častým typem vzdáleného řízení je řízení pomocí aplikace pro dotykové zařízení, nejčastěji tablet. Některé konzole jsou na tomto ovládání striktně založeny a nenabízejí žádnou jinou možnost ovládání (např. Mackie DL1608 [11]). Součástí této konzole je nabíjecí stanice pro tablet Apple iPad, který využívá aplikaci Mackie Master Fader pro řízení, případně přehrávání hudby. Nespornou výhodou tohoto řešení je možnost mít veškerou techniku potřebnou pro ozvučení společenské události na pódiu, čímž odpadá nutnost vést mikrofonní signál někdy i desítky metrů daleko k mixážnímu pultu. Zvukový technik pak není vázán na jedno pracovní stanoviště a může lépe reagovat na podmínky prostoru.

Některé mixážní pulty, např. Yamaha 01V96i [12], také nabízejí ovládání pomocí protokolu MIDI (Musical Instrument Digital Interface), čehož lze velmi dobře využít při studiové práci. Jednotlivé MIDI zprávy jsou přiřazeny funkcím jako nastavení zesílení jednotlivých kanálů, jejich ztlumení či nastavení parametrů efektů v efektových sběrnicích. MIDI, tak jak jej známe v současné době, bohužel neumožňuje ovládat více než šestnáct parametrů v dostatečně vysokém rozlišení,

které je nutné, mimo jiné, pro precizní nastavení hodnoty zesílení [13]. MIDI ovládání tedy často pokrývá pouze základní a omezenou funkcionalitu.

V neposlední řadě je nespornou výhodou digitálních mixážních pultů možnost uložit a načíst nastavení parametrů kanálů, sběrnic, efektů nebo celého pultu. Toto nastavení je možné přenášet mezi pulty stejného typu nebo výrobce. Zatímco načtení nastavení může trvat zlomek sekundy, ve světě analogových pultů tyto operace trvají i desítky minut, neboť je nutné je, až na výjimky, provádět ručně [14].

1.6 Propojení konzolí

Některé mixážní pulty nabízí možnost vzájemného propojení několika stejných konzolí, což přináší zajímavé možnosti rozšíření. Firma Presonus vyrábí celou sérii takovýchto mixážních pultů: Presonus StudioLive AI. Propojení těchto pultů je realizováno pomocí sběrnice FireWire a umožňuje spojení dvou libovolných konzolí o 16, 24 nebo 32 vstupních kanálech. Celkově je tedy možné získat až 64 kanálů [15].

Existují také konzole s ovládací konzolí fyzicky oddělenou od vstupně-výstupní jednotky (např. Allen & Heath iLive-T112 a iDR-64 [16]). Tato architektura rozšiřuje možnosti běžných pultů s pevným počtem vstupních a výstupních jednotek a zároveň elegantně řeší problematiku přenosu zvuku mezi pódiem a konzolí. Veškeré zvukové zpracování probíhá přímo u vstupních jednotek, kde je umístěn zvukový procesor. Mixážní pult je pouze ovládací konzolí, a pokud není potřeba nastavení měnit, může být dokonce vypnut.

2 Návrh softwaru

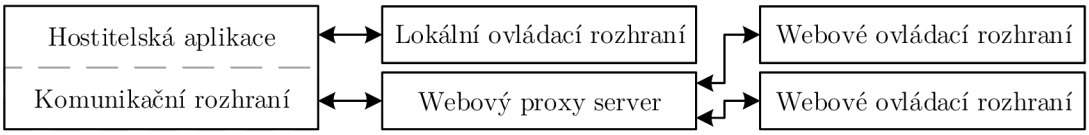
Cílem této práce je navrhnout software pro digitální mixážní pult. Tento software by měl odstranit některé nevýhody stávajících pultů, a to především:

- poskytnout modularitu efektových jednotek pomocí VST plug-in modulů,
- umožnit ovládání skrze webovou aplikaci,
- nabídnout neomezený počet kanálů a stop.

Tento software by měl být postaven na platformě Windows. Ačkoli komerční pulty (např. iLive-T112) preferují spíše varianty systému Unix, tato volba by omezila výběr VST modulů, neboť jich ani zdaleka není tolik k dispozici pro tuto platformu.[1]

Software je součástí projektu s názvem WOAP, jehož cílem je vytvořit digitální mixážní pult v rozměru 19“ skříně (*rack*) postavený na platformě Windows. Hardwarovou část tohoto projektu zajišťuje Bc. Libor Přibyl v diplomové práci s názvem Digitální mixážní pult.

Základem vývoje každého softwaru je separace částí aplikace tak, aby bylo možné jednotlivé části jednotlivě spravovat či měnit. Z tohoto důvodu je celý projekt rozdělen na několik samostatných aplikací dle obrázku 2.1.



Obr. 2.1: Návrh struktury softwaru pro digitální mixážní pult

Hostitelská aplikace

Aplikace WOAP Host zajišťuje práci se zvukovými vzorky, VST moduly, kanály a vším ostatním, co souvisí se zvukem a jeho zpracováním. Tato aplikace nabízí OSC (Open Sound Control) komunikační rozhraní protokolu TCP (Transmission Control Protocol) a vlastní, lokální grafické rozhraní. Této aplikaci se dále věnuje kapitola 3.

Webový proxy server

Proxy server WOAP WebProxy slouží pro převod OSC zpráv z protokolu TCP na protokol WebSocket. To umožní připojení k aplikaci pomocí webového prohlížeče, se kterým nelze, až na výjimky, komunikovat pomocí protokolu TCP. Proxy server je dále popsán v kapitole 4.

[1] Zjištěno porovnáním počtu dostupných VST plug-in modulů v internetové databázi KVR: <`http://kvraudio.com/plugins/vst-plugin`>.

Webové ovládací rozhraní

Webové ovládací rozhraní WOAP WebClient umožní plné ovládání hostitelské aplikace pomocí aktuálního webového prohlížeče na standardním zařízení více uživatelům zároveň. Připojuje se k hostitelské aplikaci skrze webový proxy server. Viz kapitola 5.

Tuto trojici aplikací tvořících jádro práce doplňuje sada pomocných aplikací popsaná v kapitole 6:

Nástroj pro tvorbu zobrazení plug-in modulů

Nástroj WOAP PAG Generator slouží pro jednoduché a uživatelsky přívětivé vytváření alternativních zobrazení plug-in modulů. Tato zobrazení jsou poté prezentována ve webovém rozhraní jako náhrada za originální zobrazení plug-in modulu dodaného výrobcem.

Webový server

Webový server WOAP WebServer je nutný pro zpřístupnění souborů webového rozhraní. Webové rozhraní je možné spouštět několika způsoby, díky této aplikaci však není nutné, aby instalaci a zprovoznění webového serveru řešil uživatel.

Spouštěč

Aplikace s názvem WOAP slouží pro jednoduché spuštění celého systému. Spouští hostitelskou aplikaci, webový proxy server a webový server.

3 Hostitelská aplikace

Hostitelská aplikace WOAP Host je srdcem celého projektu. Jedná se o samostatnou aplikaci zajišťující komunikaci se zvukovým rozhraním, správu, vytváření a propojení zvukových stop a sběrnic a správu a přidávání VST plug-in modulů. Aplikace je naprogramována v jazyce C++ s využitím frameworku JUCE, který do značné míry usnadňuje práci s mnoha jejími částmi, od práce se vzorky zvukového signálu po komunikaci s ostatními aplikacemi [17].[1]

Vstupním bodem aplikace je třída `Main`, která je vlastníkem všech ostatních tříd. Hlavní třídy aplikace implementují návrhový vzor Singleton [18]. Není tak nutné vzájemně vkládat reference hlavních tříd pro konstrukci tříd ostatních, a ačkoli je přenositelnost kódu snížena a odporuje to Deméteřinu principu [19], je toto řešení velmi jednoduché a spolehlivě splňuje svůj účel. Aplikace obsahuje desítky vlastních tříd, mezi ty největší a nejdůležitější, jež budou popsány níže, patří:

- `Bootstrapper`,
- `AccessControlCore`,
- `AudioCore`,
- `AudioCoreProcessor`,
- `MixerNode`,
- `APICore`,
- `ValidatableOSCMessage`,
- `OSC::RemoteClient`,
- `APIConsumer`,
- `ListenerBridges::Base`,
- `StateManager`,
- `ConfigurationManager`,
- `MainView`.

3.1 Třída pro inicializaci aplikace

Pro získání některých důležitých systémových dat ostatních tříd, např. seznamu přístupových práv (třída `UserPermission`), je využita třída `Bootstrapper`. Ostatní třídy se instanci třídy `Bootstrapper` před spuštěním aplikace registrují přidáním statické třídní proměnné `Bootstrapper::Initializer`. Při spuštění aplikace je všem těmto třídám instance třídy `Bootstrapper` předána, a tak může být provedeno zaregistrování všech systémových dat.

[1] Framework neboli aplikační rámec je softwarová struktura určená pro podporu vývoje aplikace. Slovo framework nemá žádný spisovný používaný ekvivalent, proto bude použito a skloňováno dle vzoru hrad.

3.2 Třída pro řízení přístupu

Pro řízení přístupu uživatelů, správu uživatelů a jejich přihlášení byla vytvořena třída `AccessControlCore`. Řízení přístupu je nutné pro zabezpečení aplikace, jinak by po připojení zařízení k veřejně dostupné síti mohl aplikaci ovládat kdokoli. V aplikaci lze vytvářet uživatele (třída `User`) a přiřazovat je do skupin (třída `UserGroup`). Skupiny obsahují různá oprávnění (třída `UserPermission`), a je tak možné jednoduše vytvořit administrátora se všemi oprávněními nebo uživatele, který bude mít přístup pouze k určité části aplikace. Tato oprávnění se vztahují pouze pro přístup skrz OSC komunikační rozhraní. Lokální ovládání není ověřováno, lokální uživatel má tedy vždy přístup ke všem částem aplikace. Konkrétní hodnoty oprávnění jsou indexovány, a to převážně kvůli rychlosti vyhledávání.

V aplikaci je také vytvořen tzv. anonymní uživatel (pokud není tato funkce deaktivována v nastavení aplikace). Jedná se o výchozího uživatele vytvořeného při spuštění aplikace, který je přiřazen všem kontrolním spojením. Autorizace uživatele není nutná, což může najít svůj účel v případě, že je zařízení připojeno pouze k lokální síti nebo zabezpečení přístupu probíhá na úrovni připojení k této síti. Nastavení oprávnění tohoto uživatele je možné omezit. Toto omezení je však možné udělat pouze úpravou zdrojového kódu.

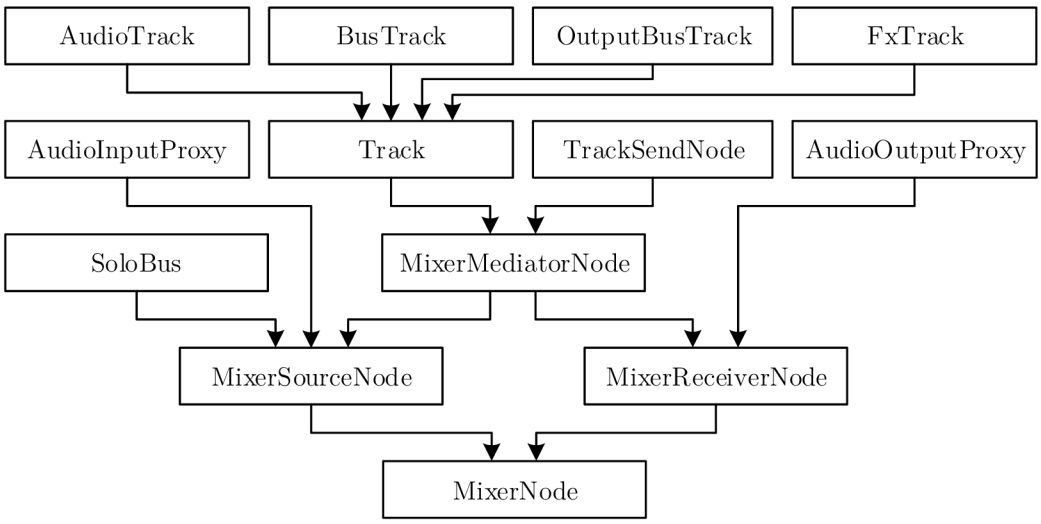
Třída `AccessControlCore` musí znát hodnoty všech oprávnění existujících v aplikaci. K tomu je využit `Bootstrapper` a jeho metoda `reportPermissions()`, do které je předáno pole instancí `UserPermission`. `Bootstrapper` si tyto instance uloží a instance třídy `AccessControlCore` si poté vytvoří jejich kopie pro vlastní potřebu. Není očekáváno, že by se seznam oprávnění měnil za běhu aplikace.

3.3 Třída zvukového jádra

Největší třídou projektu je třída `AudioCore`. Ta zajišťuje připojení ke zvukovému zařízení pomocí technologie ASIO, vytváření, správu a propojení zvukových stop a sběrnic. Třída je vlastníkem všech instancí tříd `MixerNode` a `MixerConnection`.

3.3.1 Zvukové stopy, sběrnice a zapojení

Hierarchie tříd zvukové části hostitelské aplikace, jejích stop, vstupů a výstupů je uvedena na obrázku 3.1.



Obr. 3.1: Hierarchie tříd struktury zpracování zvuku

`MixerNode` je základní třída reprezentující jeden prvek struktury zpracování zvuku. Uchovává identifikátor UUID (Universally Unique Identifier), název, barvu prvku a informaci o posledním přebuzení zvukového signálu (pokud jej stopa využívá). Pro všechny prvky platí pravidlo, že mohou obsahovat jednokanálový (monofonní) nebo dvoukanálový (stereofonní) signál. Jedná se o implementační omezení celé aplikace, ačkoli by rozšíření na vícekanálové stopy nebylo příliš složité.

`MixerSourceNode` je prvek reprezentující zdroj signálu, jehož součástí je zvuková vyrovnávací paměť obsahující vzorky výstupního signálu.[2] Tomuto prvku je možné přiřadit příjemce třídy `MixerReceiverNode`. Příjemce nemá svou vlastní vyrovnávací paměť, zvukové vzorky si musí získat ze zdrojových prvků, ke kterým je připojen.

[2]Vyrovnávací paměť si uchovává zvukové vzorky několika kanálů. Jedná se tedy o dvourozměrné pole hodnot, které je až na výjimky instancí šablonové třídy `AudioBufferExtended<>`, nejčastěji `AudioBufferExtended<double>`. Alias pro tuto třídu je `AudioBufferDouble`.

`MixerMediatorNode` kombinuje obě výše uvedené třídy. Jedná se tedy o prvek dědící jak `MixerSourceNode`, tak `MixerReceiverNode`, a je mu tedy možné přiřadit jak vstupy, tak výstupy.

`Track` je základní třída reprezentující stopu (dle kapitoly 1.2). Je abstraktní, dědí `MixerMediatorNode` a dále ji dědí a doplňují třídy `AudioTrack`, `MixBusTrack`, `OutputBusTrack` a `FxTrack`. `AudioTrack` odpovídá zvukové stopě, lze jí tedy na vstup přiřadit pouze vstupní jednotky zvukového rozhraní. `BusTrack` je skupinovou sběrnicí, `OutputBusTrack` hlavní skupinovou sběrnicí, která je jako poslední ve stromové struktuře a je jediná ze stop, které lze přiřadit výstupní jednotky. `FxTrack` je efektová sběrnice, jež je v systému pouze z důvodu uživatelské přívětivosti. Aplikace totiž umožňuje přidat zvukové efekty na jakoukoli stopu.

Každá stopa je nositelem několika parametrů, které jsou odvozeny od běžných mixážních pultů: parametr zesílení (*gain*), panorama (*pan*), ztlumení (*mute*), připojení k Solo sběrnici (*solo*) a označení pro nahrávání (*record arm*).[3] Průběh závislosti zesílení levého a pravého kanálu na hodnotě panoramatu (*pan law*) je lineární.

Do zvukového řetězce stopy lze zařadit VST plug-in modul (zabalen do třídy `AudioProcessorPlayerDouble`) anebo přídavný zvukový výstup (`TrackSendNode`, tzv. „send"), u kterého si uživatel může nastavit jeho pozici ve zvukovém řetězci stopy. Struktura nabízí ne až tak typickou možnost nastavit pozici tohoto zvukového výstupu za kterýkoli ze zásuvných modulů, což rozšiřuje možnosti běžných mixážních pultů, kde tato možnost často není nebo je omezena na konkrétní pozice. `TrackSendNode` je tedy možno zařadit před řetězec zásuvných modulů, za kterýkoli z modulů, na konec řetězce modulů (před zesílení a panorama) nebo konec řetězce stopy (za zesílení a panorama). Třída dědí `MixerMediatorNode` a rovněž obsahuje vlastní parametry zesílení a panoramatu.

Spojení všech prvků v této struktuře zajišťují instance třídy `MixerConnection` odkazující na zdroj (`MixerSourceNode`) a příjemce (`MixerReceiverNode`). Každý prvek může mít libovolný počet těchto spojení. Spojování prvků je ošetřeno tak, aby nenastala kruhová závislost, která by, v lepším případě, způsobila nefunkčnost zapojení. Příklad zapojení zvukové struktury je zobrazen na obrázku 3.2.

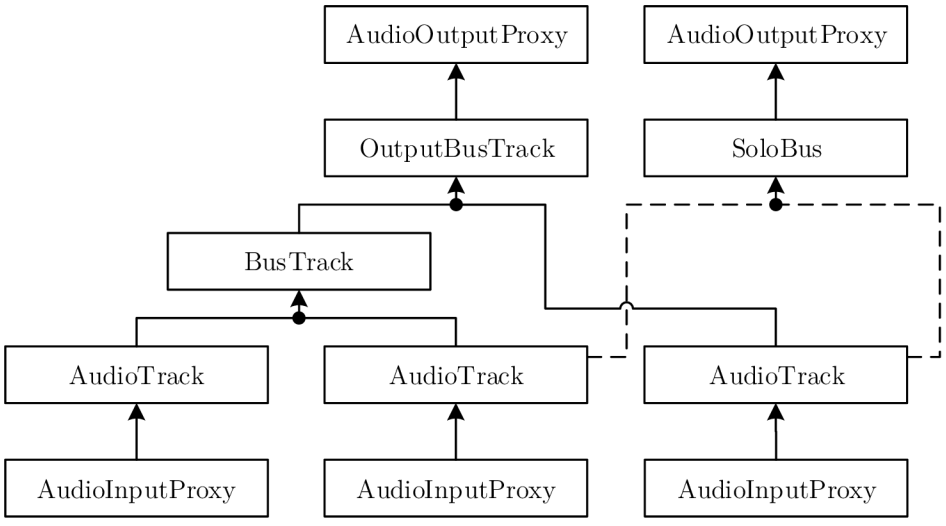
Speciálním typem sběrnice je `SoloBus`. Jak je již z názvu patrné, jedná se o sběrnici, do níž stopy s aktivovaným parametrem *solo* posílají svá výstupní data. Je to jediný prvek celého schématu, který nemá kompenzaci zpoždění (viz kap. 3.4.3). Ačkoli se jedná o sběrnici, tato stopa dědí pouze `MixerSourceNode` a je ze standardního schématu zpracování vyjmuta. Stopy, jež mají aktivovaný parametr

[3] Aplikace nemá implementováno nahrávání zvukového signálu. Pro účely rozšiřitelnosti má však každá stopa vlastní parametr pro označení stopy k nahrávání, který může být použit při vhodné kombinaci s ovládacím a zvukovým rozhraním.

`solo`, nejsou s touto sběrnicí propojeny pomocí třídy `MixerConnection`. Instance třídy `SoloBus` poslouchá změny parametru *solo* každé stopy a uchovává si seznam těchto stop. Zvukový signál odesílaný do této sběrnice je signálem stop z pozice před zesílením a panoramatem, tak jak je tomu běžně u ostatních mixážních pultů.

Třídy `AudioInputProxy` a `AudioOutputProxy` reprezentují hardwarové vstupní a výstupní jednotky. `AudioInputProxy` si na začátku zpracování zvukových vzorků získaných ze zvukového zařízení zkopíruje data kanálů, ke kterým je přiřazena, a tato data později předává zvukovým stopám k dalšímu zpracování. Opakem této třídy je třída `AudioOutputProxy`, jež získává vzorky z výstupních sběrnic, které má přiřazeny na svém vstupu, a jejich sumu poté ukládá do výstupního pole zvukového rozhraní.

Instance těchto dvou tříd jsou vytvářeny automaticky během inicializace nebo změny zvukového zařízení. Pro každý aktivní vstupní kanál zvukového rozhraní je vytvořena instance `AudioInputProxy` (jednokanálový vstup) a pro každou dvojici sousedních kanálů (vždy začínající lichým kanálem) další instance `AudioInputProxy` (dvoukanálový vstup). Díky automatickému vytváření těchto dvojic není nutné, na rozdíl od komerčních řešení, párování vstupních jednotek složitě nastavovat. Ačkoli toto řešení zdvojnásobí výkonové nároky na kopírování vstupních dat, je tento způsob velmi elegantní. Instance `AudioOutputProxy` jsou řešeny obdobným způsobem s tím rozdílem, že zde nedochází ke zvýšení výpočetní náročnosti, neboť pokud není do výstupu připojena žádná stopa, je pole výstupních vzorků ponecháno beze změny.



Obr. 3.2: Ukázka zapojení struktury zpracování zvuku

3.3.2 Podpora VST plug-in modulů

Pro účely práce s VST plug-in moduly byla v hostitelské aplikaci vytvořena třída `AudioProcessorPlayerDouble`, která z části kopíruje třídu `AudioProcessorPlayer` přiloženou v JUCE. Oproti svému vzoru poskytuje zpracování vzorků ve dvojí přesnosti, ovládání modulu pomocí OSC komunikačního rozhraní a několik dalších vylepšení pro specifické potřeby tohoto projektu. Nad rámec běžných ovládacích prvků dodaných výrobcem zásuvného modulu je v aplikaci navíc k dispozici přepínač přemostění (parametr *bypass*) a zapnutí (parametr *active*). Aplikace podporuje VST plug-in moduly zkompilovány pro 32bitovou architekturu, jejichž specifická kompatibilita je primárně určena frameworkem JUCE. Aplikace je mimo jiné závislá na správně udané hodnotě zpoždění zvukového signálu plug-in modulu. Nesprávné udání hodnoty má za následek rozfázování signálu.

3.4 Třída pro výpočet vzorků zvukového signálu

Pro výpočet vzorků zvukového signálu slouží třída `AudioCoreProcessor`, která nabízí zpracování signálu ve více vláknech a kompenzaci zpoždění vytvořeného zásuvnými moduly. Pro výpočet je implementována fronta úloh, kterou paralelně obsluhuje několik instancí třídy `AudioCoreProcessor::Worker` (dále jen `Worker`), každá spuštěná z vlastního vlákna. Počet instancí třídy `Worker` je roven počtu virtuálních jader procesoru, na kterém je aplikace spuštěna, a jejich spřažení je nastaveno tak, aby každé jádro obsluhovalo pouze jednoho z nich.

3.4.1 Průběh zpracování

Volnost propojení prvků zvukového schématu s sebou přináší obtížné řešení jejich výpočtu. Pořadí, v jakém je signál vypočítán, se totiž mění podle jejich propojení. Ve zpracování všech prvků platí pravidlo: prvek je zařazen ke zpracování, pokud jsou připraveny (zpracovány) všechny jeho vstupní prvky.

Zpracování signálu probíhá voláním metody `audioDeviceIOCallback()` a je voláno z tzv. „audio vlákna", které je externím vláknem obsluhujícím zpracování zvukového signálu připojeného zvukového zařízení. Na začátku volání této metody je fronta úloh prázdná. Nejprve je zjištěno, které z prvků schématu jsou připraveny a mohou být zpracovány okamžitě. To platí pro všechny stopy, které nemají přiřazeny žádné vstupní prvky a všechny instance třídy `AudioInputProxy`, neboť jejich vstupní data jsou vstupními argumenty metody `audioDeviceIOCallback()`. Úlohy připravených prvků jsou zařazeny do fronty a proběhne notifikování všech

instancí třídy `Worker` (metoda `notifyAllWorkers()`).[4] Audio vlákno nyní vyčkává na zpracování všech stop. Maximální doba zpracování je nastavena na 2 sekundy, poté dojde k předání dat zvukovému zařízení.

Při probuzení instance třídy `Worker` dojde k pokusu o vyčtení úlohy z fronty. `Worker` zkouší vyčíst úlohu celkem padesátkrát, a pokud žádnou úlohu nevyčte, opět se uspí. Pokud je úloha vyčtena, je ihned zpracována. Po zpracování úlohy jsou všechny výstupní prvky zpracovávaného prvku obeznámeny, že jejich vstupní prvek byl zpracován (metoda `notifyNodeJobDone()`). Pokud některý z výstupních prvků má již všechny vstupy připraveny, dojde k přidání jeho úlohy do fronty a opět proběhne notifikování všech vláken. Po zpracování posledního prvku je notifikována instance `AudioCoreProcessor` a zpracování je ukončeno.[5]

Pokud měla zpracovávaná stopa aktivován parametr *solo*, je dokončení zpracování oznámeno instanci třídy `SoloBus`. Ta svou úlohu obdobně předá ke zpracování, pokud jsou všechny stopy s aktivním parametrem *solo* zpracovány.

Všechny instance tříd `TrackSendNode` jsou zpracovány zároveň se stopou, ke které přísluší. Dokončení zpracování stopy je oznámeno i všem výstupním stopám všech instancí `TrackSendNode` příslušné stopy.

3.4.2 Úlohy a pomocná data

Pro redukci výpočetní náročnosti si třída `AudioCoreProcessor` do prvků zvukové struktury ukládá pomocná data (struktura `AudioCoreNodeProcessingState`), mezi něž patří: počet vstupních prvků, počet zpracovaných vstupních prvků, pole výstupních prvků, úloha a informace o tom, zda je stopa zpracována. Údaje o tom, zda byl prvek zpracován a kolik jeho vstupních prvků bylo zpracováno, jsou přenastaveny na začátku zpracování. Při změně zapojení prvků se tyto hodnoty přepočítají.

Každý prvek zvukové struktury je nositelem své vlastní úlohy, která dědí třídu `AudioCoreProcessor::ProcessJob`. Všechny úlohy nejprve získají data ze všech připojených zdrojových prvků a ty následně předají svému prvku ke zpracování. Při získávání dat se provádí konverze počtu kanálů dle nastavení prvku, jemuž úloha předává data. Třída `InputProxyProcessJob` navíc provádí převod vzorků z 32bitového čísla s plovoucí desetinnou čárkou (`float`) na číslo 64bitové (`double`). Instance třídy `OutputProxyProcessJob` provádí opačnou operaci. Celé zpracování je tedy 64bitové (vyjma plug-in modulů, které toto zpracování nepodporují).

[4] Notifikování vlákna (anglicky *notify thread*) znamená probuzení vlákna, pokud je uspáno.

[5] Probouzení vláken je na systémech Windows vlivem nízkého rozlišení plánovače úloh operace s časovou náročností v řádech stovek mikrosekund, což je jeden z důvodů nízkého výkonu aplikace při použití zvukových vyrovnávacích pamětí s časovou délkou menší než 2 milisekundy.

3.4.3 Kompenzace zpoždění

Zpoždění vzorků zvuku, které vzniká v jednotlivých plug-in modulech, je nutné kompenzovat tak, aby všechny výstupní signály byly časově vzájemně zarovnány. Neměla by tedy nastat situace, kdy je jeden z kanálů zpožděn a druhý ne.

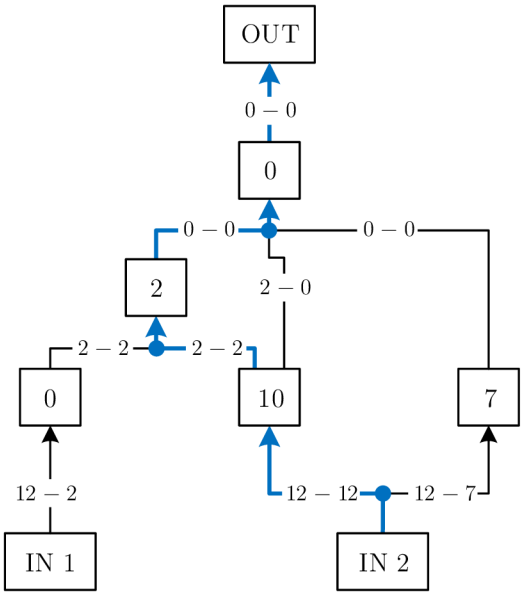
Kompenzace se provádí umělým zpožďováním nezpožděných (či méně zpožděných) větví stromové struktury, kdy každé propojení má vlastní hodnotu zpoždění. Kompenzace zpoždění začíná určením nejvyšší hodnoty zpoždění v celé stromové struktuře (metoda `AudioCore::recountGlobalTreeLatency()`). To probíhá rekurzivním procházením stromové struktury a hledáním nejvíce zpožděné cesty.

Následně je nutné určit hodnoty zpoždění každého propojení, k čemuž slouží metoda `MixerNode::getLatencyDiffForNode()` vracející hodnotu dle vztahu

$$\delta_t = \delta_{t1} - \delta_{t2}, \tag{3.1}$$

kde δ_{t1} je maximální hodnota zpoždění zdrojového prvku propojení ve stromové struktuře bez započteného vlastního zpoždění a δ_{t2} je maximální hodnota zpoždění příjemce ve stromové struktuře včetně započteného vlastního zpoždění. Tyto hodnoty jsou uloženy do pomocných dat (viz kap. 3.4.2) a jejich výpočet probíhá pouze při změně zapojení nebo při přidání, odebrání nebo přesunutí plug-in modulu.

Pro prvky `AudioInputProxy` platí, že jejich zpoždění ve stromové struktuře je rovno globálnímu maximu zpoždění stromové struktury.

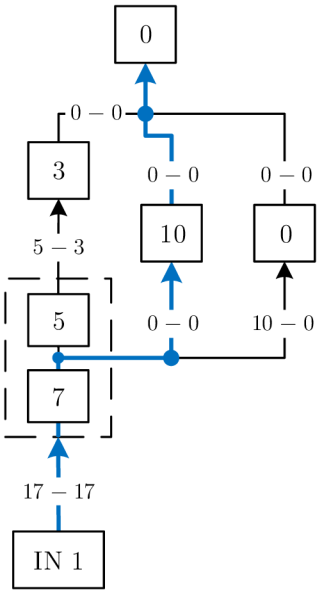


Obr. 3.3: Ukázka výpočtu zpoždění zvukové struktury

Na obrázku 3.3 lze názorně vidět jednotlivé hodnoty zpoždění prvků a propojení ve zvukové struktuře. Struktura začíná dvěma prvky `AudioInputProxy` a končí

jedním prvkem `AudioOutputProxy`. Dále je zde zařazeno několik stop s uvedenou hodnotou vlastního zpoždění (vzniklého např. zařazením odpovídajících zásuvných modulů). Hodnoty uvedené u propojení jsou hodnoty zpoždění δ_{t1} a δ_{t2} dle vztahu 3.1. Nejvíce zpožděná cesta, se zpožděním 12 vzorků, je zaznačena modře.

Pokud má některý z prvků přiřazen přídavný výstup `TrackSendNode`, je nutné kompenzovat i jej, což probíhá podobným způsobem s tím rozdílem, že instance třídy `TrackSendNode` není zpožděna o celé zpoždění prvku, který ji vlastní, ale pouze o jeho část.



Obr. 3.4: Ukázka výpočtu zpoždění stop s přídavným výstupem

Přerušovaná čára na obrázku 3.4 znázorňuje zvukovou stopu se dvěma vloženými plug-in moduly (s uvedeným zpožděním) a jedním přídavným výstupem mezi těmito moduly. Nejvíce zpožděná část, taktéž zaznačena modře, má 17 vzorků.

Pro účely kompenzace zpoždění vznikla třída `AudioBufferExtended`, která reprezentuje zvukovou vyrovnávací paměť a jejíž vzorky jsou rozděleny na dvě části: „minulé vzorky" a „aktuální vzorky". Uchování předešlých vzorků do značné míry zjednodušuje realizaci zpoždění, neboť není nutné implementovat zpožďovací linku, jak tomu běžně bývá v signálových procesorech. Instancím lze jednoduše nastavit délku obou částí a zavoláním metody `shiftSamplesToHistory()` posunout aktuální vzorky do části vzorků minulých. Přistoupit k těmto vzorkům je možné zápornou indexací pole, `buffer[0][-5]` tedy vrací hodnotu vzorku prvního kanálu se zpožděním 5 vzorků.

3.5 Třída komunikačního rozhraní

Komunikační rozhraní aplikace zajišťuje třída `APICore`. Ta obsahuje instance klientů třídy `OSC::RemoteClient` připojených k této aplikaci a umožňuje komunikaci s nimi. Při spuštění aplikace se spustí TCP server přidružený k portu 5020, který vyčkává na připojení klientů.

3.5.1 Protokol OSC

Open Sound Control je protokol určený pro komunikaci zvukových zařízení. Struktura každé zprávy je rozdělena na dvě části: adresu (řetězec znaků) a seznam argumentů. Protokol verze 1.0, jenž je podporován frameworkem JUCE, podporuje čtyři datové typy argumentů: `integer`, `float`, `string` a `blob` [20]. Pro přenos strukturovaných dat a polí je využit formát JSON (JavaScript Object Notation), který je díky své jednoduchosti běžně používán i mimo webová prostředí [21]. Data jsou tedy zakódována do tohoto formátu a odeslána jako argument datového typu `string`.[6]

Protokol OSC se v poslední době těší oblibě a vzhledem k tomu, že neexistuje žádná rozšířená alternativa, je protokol použit v této práci.

3.5.2 Zprávy

OSC zprávy posílané mezi hostitelskou a ovládací aplikaci jsou neformálně rozděleny na jednosměrné a obousměrné, příchozí a odchozí. Všechny typy zpráv jsou zdokumentovány v souboru `APIDoc.md` (`APIDoc.pdf`), který je součástí přílohy B.1.

Jednosměrné zprávy dědí třídu `ValidatableOSCMessage`, není možné na ně odpovědět a jsou (až na výjimky) pro hostitelskou aplikaci příchozí. Tato třída je obalem nad instancí třídy `juce::OSCMessage` obsahující adresu i argumenty zprávy. Přímá práce s třídou `OSCMessage` však není příliš přívětivá, a tak je dalším děděním třídy `ValidatableOSCMessage` prostá zpráva doplněna o tzv. „gettery", což jsou intuitivně pojmenované metody vracející hodnoty argumentů zprávy (např. pro zprávu `TrackSetName` metoda `getNewName()` vracející první argument). Práce se zprávami je tedy velmi jednoduchá a riziko programátorské chyby je podstatně sníženo. Zároveň dochází k automatické dokumentaci kódu.

Obousměrné zprávy dědí třídu `ValidatableOSCMessageWithId` (ta dále dědí `ValidatableOSCMessage`). Prvním z dvojice zpráv je požadavek (příchozí), na nějž je odeslána odpověď (odchozí). Příkladem je zpráva `AudioCoreGetTracksList`, kterou si klient požádá o seznam stop s odpovědí `AudioCoreTracksList` obsahující

[6] Strukturovaná data jsou částečně podporována v protokolu OSC verze 1.1.

seznam identifikátorů požadovaných prvků. Oboustranné zprávy jsou párovány pomocí čísla zprávy uloženého v prvním argumentu s datovým typem `integer` a je tedy možné jednoduše určit, ke kterému požadavku přísluší příchozí odpověď. Obecně platí, že hostitelská aplikace žádné požadavky neodesílá, proto si identifikátory zpráv spravuje klient.

Zprávy jsou při přijetí validovány. Je nutné dodržet počet argumentů a správný tvar adresy. Při zkompilování aplikace v režimu *debug* se při přijetí kontrolují i datové typy argumentů. V případě, že zpráva není validní nebo nastane při jejím dekódování chyba, je klientovi odeslána zpráva `ApiCoreMessageRejected`. Stejnou zprávu klient obdrží, pokud není autorizován.

3.5.3 Přihlášení

Připojení ke komunikačnímu rozhraní je zabezpečeno pomocí přihlášení a přístupových práv (viz kap. 3.2). Pokud je zapnuto anonymní přihlášení, je možné s hostitelskou aplikací komunikovat okamžitě po úspěšném spojení. V opačném případě je však nejprve nutné zaslat autentizační údaje a provést přihlášení. Klient tedy zašle zprávu `ApiCoreLogin` s jeho přístupovým heslem, na kterou získá odpověď `ApiCoreLoginResponse` obsahující návratový kód, zda bylo přihlášení úspěšné, či naopak, a v případě úspěšného přihlášení také autentizační identifikátor. Ten je pak možné použít pro opětovné přihlášení namísto uživatelova hesla. To je velice praktické, neboť uživatel není neustále vyzýván k zadání hesla, což snižuje efektivní dobu načítání aplikace. Platnost identifikátoru je 30 dní, přičemž při každé přijaté zprávě je jeho platnost prodloužena. Pro přihlášení uživatele pomocí autentizačního identifikátoru slouží zpráva `ApiCoreAuthenticate`.

3.5.4 Průběh komunikace

Hostitelská aplikace ve výchozím stavu neodesílá klientům žádné zprávy. Klienti si o zprávy musí požádat. Aplikace využívá návrhového vzoru Observer pattern [22], který definuje poslouchaný subjekt a posluchače (anglicky *listener*). Posluchač se u subjektu zaregistruje a subjekt následně při každé změně svých vlastních dat oznámí posluchači, že se hodnoty změnily. Hostitelská aplikace je subjektem a klient posluchačem. Klient se tedy musí u hostitelské aplikace registrovat, typicky zavoláním zpráv typu `AddListener`.[7]

[7]Pro účely ovládání skrz webové rozhraní WOAP WebClient není tento způsob tím nejefektivnějším, neboť je při spuštění aplikace nutné odeslat desítky zpráv pro zaregistrování klienta k poslechu všech změn. Komunikační rozhraní hostitelské aplikace je však vytvořeno pro univerzální použití. Řídící rozhraní totiž nemusí využívat všechny funkce.

Příkladem je zpráva `TrackAddControlsListener` pro získání informací o změnách hodnot ovládacích prvků stopy, jež zaregistruje klienta k poslouchání změn hodnot parametrů *gain*, *pan*, *mute*, *solo* a *recordArm*. Při změně hodnoty hostitelská aplikace odešle klientovi zprávu s novou hodnotou parametru, např. zprávu `TrackGainChanged`. Pro odhlášení posluchače od poslouchaného subjektu je možné využít zprávy typu `RemoveListener`, např. `TrackRemoveControlsListener`. Odhlášení klienta od všech subjektů je zároveň provedeno automaticky při odpojení klienta.

3.5.5 Vnitřní implementace komunikačního rozhraní

Pro implementaci komunikačního rozhraní v hostitelské aplikaci byla vytvořena třída `APIConsumer`, jež reprezentuje instanci používající komunikační rozhraní. V aplikaci je několik tříd, které tuto třídu dědí a toto rozhraní využívají (např. `AudioCore` nebo `Track`). Každá z instancí třídy `APIConsumer` musí implementovat metodu `getMessageTypesToListenTo()` určující typy OSC zpráv, která třída může přijímat, a metodu `messageReceived()` definující chování pro každou z těchto zpráv.

Vstupním parametrem metody je také zpráva `juce::OSCMessage`, kterou je nejprve nutné „zabalit" do některé z vyšších tříd (např. `TrackSetName`). Následně je možné provést validaci této zprávy zavoláním metody `testValidity()` nebo `testValidityAndPermissions()`, pokud je zpráva chráněna přístupovými právy, a v případě, že validace proběhne úspěšně, je následně možné se zprávou dále pracovat. Výpis 3.1 znázorňuje zpracování zprávy `TrackSetName`, která slouží pro nastavení názvu stopy. Pokud zpráva není validní, nebo je odeslána bez adekvátních oprávnění, systém automaticky odešle klientovi zprávu `ApiCoreMessageRejected`.

Výpis 3.1: Ukázka části zdrojového kódu zpracování OSC zprávy

```
case MessageType::SetName:
{
    TrackSetName msg(message, this);
    if (!testValidityAndPermissions(client, msg, Permission::ModifyTrack))
        return;

    setName(msg.getNewName());
    break;
}
```

Uložení klienta, který se zaregistroval k poslouchání určitého typu zpráv, zajišťují instance tříd dědící třídu `ListenerBridges::Base`. Tyto instance obsahují referenci na instanci, ke které jsou přidruženy a kterou poslouchají (např. instanci třídy `Track`), a klienta, kterého notifikují, pokud nastane změna hodnot. Zavoláním

zprávy typu `AddListener` (např. `TrackAddInfoListener`) se vytvoří nová instance třídy `ListenerBridges::Base` (např. `TrackInfoListenerBridge`), která se zaregistruje k poslouchání událostí (o změně jména nebo barvy stopy). V případě změny jména nebo barvy instance třídy `TrackInfoListenerBridge` odešle odpovídající zprávu svému klientovi.

3.6 Třída správce stavu aplikace

Aplikace je vybavena třídou `StateManager`, jež nabízí funkci uložení aktuálního stavu, což eliminuje potřebu pro znovunastavení při každém spuštění. Ukládání stavu probíhá při ukončení aplikace a v pravidelných intervalech, což je velmi užitečné v případech náhlého vypnutí aplikace nebo celého zařízení. Stav je uložen do souboru `state.xml` do kořenového adresáře hostitelské aplikace. Načtení stavu probíhá při spuštění aplikace. Pokud není soubor se stavem k dispozici, je vytvořeno výchozí rozhraní obsahující tři zvukové stopy, jednu skupinovou sběrnici a jednu výstupní sběrnici.

Součástí stavu jsou informace o všech zvukových stopách, sběrnicích, vstupních a výstupních jednotkách, plug-in modulech a propojeních. Do tohoto stavu se neukládá nastavení zvukového rozhraní. To je uloženo v souboru `audioConfig.xml`.

Práce se stavem aplikace je možná také skrz komunikační rozhraní: uložení stavu zprávou `StateManagerSaveState`, načtení stavu `StateManagerLoadState` apod.

3.7 Třída správce konfigurace aplikace

Hostitelská aplikace nabízí nastavení několika systémových parametrů pomocí třídy `ConfigurationManager`. Hodnotami, které lze nastavit (včetně výchozích hodnot uvedených v závorce), jsou:

- povolení načtení předchozího stavu aplikace (povoleno),
- časový interval ukládání stavu aplikace (10 sekund),
- povolení anonymního přihlášení (povoleno),
- administrátorské heslo vzdáleného ovládání (admin),
- povolení vytvoření bezdrátové sítě (zakázáno),
- název bezdrátové sítě (WOAP),
- heslo bezdrátové sítě (adminwoap).

Nastavení se ukládají do souboru `config.xml` v kořenovém adresáři hostitelské aplikace. Tento soubor je možné jednoduše upravit s použitím libovolného textového editoru a hodnoty tak nastavit ještě před spuštěním aplikace. Soubor je uložen pouze v případě, že uživatel výchozí konfiguraci změní, a pokud není nalezen, aplikace

provede nastavení výchozích hodnot. Při změně nastavení aplikace, vyjma změny administrátorského hesla vzdáleného ovládání, je nutné restartovat aplikaci.

Správa bezdrátové sítě v rámci hostitelské aplikace je experimentální funkce, jež nemusí fungovat na všech zařízeních. Síť by standardně měla fungovat na zařízení s bezdrátovou síťovou kartou s podporou tzv. „hostované sítě" [23].[8] Pokud je vytvoření bezdrátové sítě povoleno, je při spuštění aplikace vytvořena odpovídající bezdrátová síť. Výchozí branou této sítě je běžně adresa `192.168.137.1`, na níž je dostupný webový server.

3.8 Grafické rozhraní

Grafické rozhraní aplikace odpovídá běžným standardům a zvyklostem s ohledem na zachování maximální jednoduchosti. Rozhraní bylo navrženo pro ovládání jedním prstem na dotykovém zařízení tak, aby se eliminovala možnost nechtěného aktivování funkce, kterou uživatel neměl v úmyslu aktivovat. To je velmi užitečné mimo jiné v případě, kdy dotykové zařízení nereaguje dostatečně přesně, či zcela vynechává. Zobrazení celé aplikace využívá mírně zaoblených rohů, což vede ke zjednodušení orientace a vyšší přívětivosti rozhraní [24].

Rozhraní je optimalizováno pro zobrazení s rozlišením HD (High Definition, 1280×720 obrazových bodů) a vyšším. Pro optimální zobrazení je doporučeno rozlišení Full HD (1920×1080 obrazových bodů), které je vhodné pro práci s plug-in moduly, jejichž zobrazení jsou často rozměrná a u malých rozlišení by mohly přesáhnout až do oblasti ovládání zesílení stop, což je bezesporu nejdůležitější prvek zobrazení.[9] Aplikace je spuštěna v okně. Je možné ji maximalizovat tlačítkem maximalizace nebo přepnout do režimu na celou obrazovku klávesovou zkratkou `CTRL + L`.

Na obrázku 3.5 lze vidět zobrazení seznamu stop a v pravé části detail zvolené stopy. Každá stopa v seznamu stop obsahuje (shora): seznam plug-in modulů, název, tlačítka *Mute* a *Solo*, ovládání panoramatu, ovládání zesílení s indikací úrovně signálu a symbol pro manipulaci se stopou (ikona trojice horizontálních čar). Detail stopy obsahuje název, volbu přiřazení vstupních prvků (*Input*), volbu barvy (*Color*), volbu přiřazení výstupních prvků (*Output*), seznam plug-in modulů (*Plugins*), seznam přídavných zvukových výstupů (*Sends*), tlačítka *Mute* a *Solo*, ovládání panoramatu a zesílení.

Prázdná oblast nad seznamem stop je určena pro zobrazení oken grafických rozhraní plug-in modulů. Tato okna si pamatují svou pozici před zavřením, je

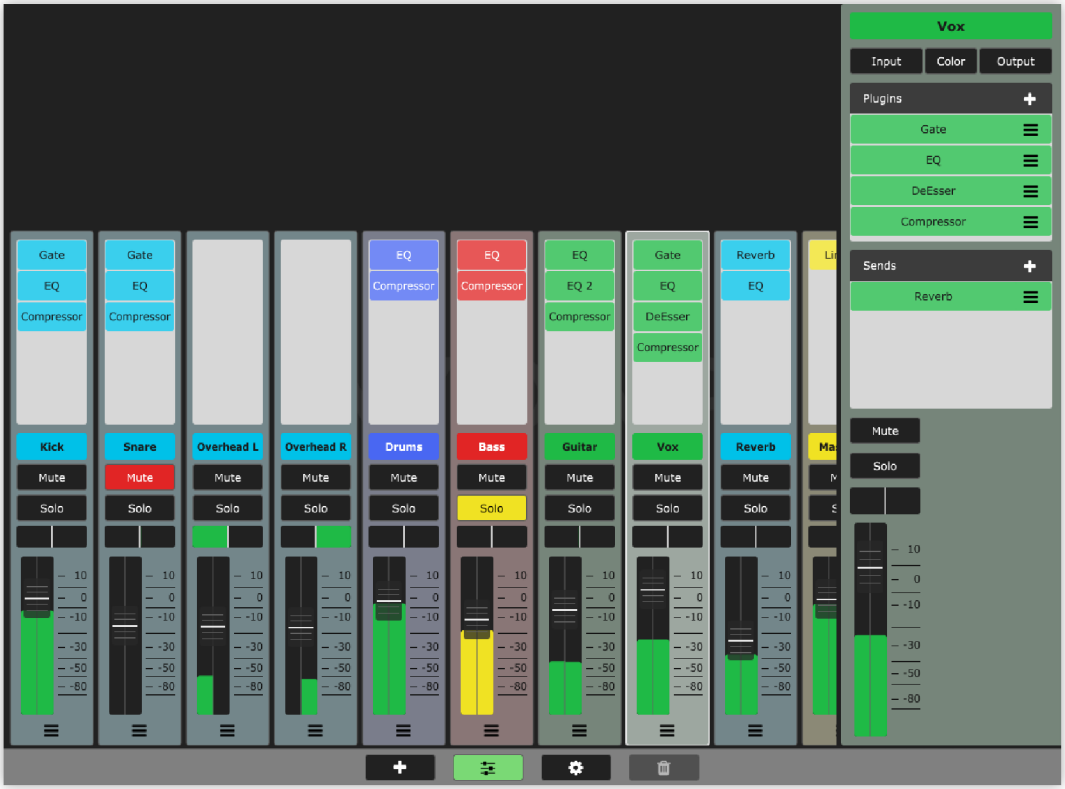
[8] Podporu hostované sítě lze zjistit ze systémové konzole příkazem: `netsh wlan show drivers`

[9] Platí pro zařízení s vypnutým zvětšením. Systémy Windows 10 mají u displejů s Full HD rozlišením ve výchozí instalaci zvětšení $1,25$.

tedy velmi jednoduché si tuto plochu rozvrhnout dle svých představ. Zobrazení zásuvných modulů obsahují kromě samotného modulu navíc svůj název a tlačítka *Active* a *Bypass* (viz obr. 5.1).

Ve spodní části aplikace se nachází lišta se čtveřicí tlačítek (zleva): přidání stopy, zobrazení mixu, zobrazení nastavení a koš.

Grafické komponenty dědí třídu `juce::Component` a mohou se skládat z jiných komponent. Hlavní třídou zobrazení nesoucí všechny grafické komponenty je třída `MainView` obsahující zobrazení zvukové části aplikace (třída `AudioCoreView`), zobrazení nastavení (třída `SettingsView`) a tlačítka dolní lišty.



Obr. 3.5: Zobrazení hostitelské aplikace

Indikace úrovně signálu

Indikace úrovně signálu využívá výpočet špičkové hodnoty s klouzavým průměrováním (*moving average* [25]). Indikovaný signál je signál na konci zvukového řetězce stopy (za zesílením a panoramatem). Pokud má stopa aktivován parametr *solo*, je indikace zobrazena žlutě a indikovaný signál je signál na konci řetězce zásuvných modulů před zesílením a panoramatem.

V případě, že dojde k přebuzení, je indikátor zobrazen červeně. K tomu může dojít pouze u zvukových stop, kdy dojde k přebuzení vstupní jednotky (určeno ze

vstupního signálu zvukové stopy), nebo výstupních sběrnic, kdy dojde k přebuzení výstupní jednotky (určeno z výstupního signálu skupinové sběrnice). Přebuzení je určeno z maximální absolutní hodnoty signálu a jeho práh je nastaven na −3 dBFS, uživatel je tak včas varován před skutečným zkreslením zvukového signálu. Indikace přebuzení trvá 300 milisekund.

3.8.1 Ovládání

Ovládání aplikace je jednoduché a intuitivní a podobá se tomu, jaké lze najít ve většině mobilních aplikací nebo aplikacích přizpůsobených pro ovládání dotekem. Aplikace využívá pouze tři gesta: kliknutí, dvojí kliknutí a posuv (tažení). Pro uživatele nezkušené s dotykovými zařízeními může být práce se složitějšími gesty velmi obtížná, z tohoto důvodu aplikace nevyužívá žádné další typy gest.

Gesta posuvu

Gesta posuvu se využívají pro nastavení zesílení a panoramatu, přesouvání oken a ve všech seznamech, které lze posouvat „chycením" za určenou část položky a tažením. Gesta posuvu jsou ošetřena před nechtěným kliknutím. Provede-li uživatel gesto posuvu do libovolného směru začínající v oblasti tlačítka a návrat do původní pozice, je gesto vyhodnoceno pouze jako posuv a k stlačení tlačítka nedojde. To platí pro tlačítka *Mute*, *Solo*, *Active*, *Bypass* a položky tlačítkových seznamů.

Seznam stop je možné posouvat v horizontální rovině gestem posuvu v přirozeném směru „chycením" v oblasti názvu stopy, tlačítek *Mute* a *Solo* nebo decibelové škály. Stopy je možné přesouvat (a měnit tak jejich pořadí) „chycením" za symbol pro manipulaci na jejím spodním okraji nebo smazat přesunutím na tlačítko koše.

Seznamy plug-in modulů nebo přídavných výstupů implementují totožný způsob ovládání, posuv je však v tomto případě vertikální a „chycení" je možné v celé šíři položky seznamu vyjma symbolu pro manipulaci.

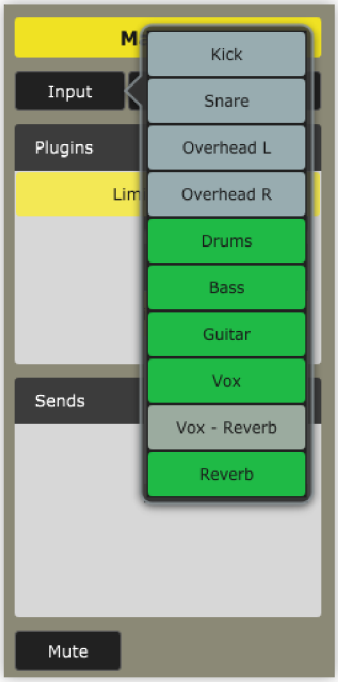
Funkční prvky

Přidání plug-in modulu je možné dvěma způsoby, a to buď dvojím kliknutím do prázdného prostoru seznamu modulů v zobrazení stopy nebo kliknutím na tlačítko se symbolem plus nad seznamem modulů v detailu stopy. Zobrazí se seznam typů VST plug-in modulů, kdy po zvolení jednoho z typů dojde k přidání nové instance na konec seznamu modulů a otevření zobrazení.

Kliknutím na položku zásuvného modulu v seznamu je otevřeno příslušné zobrazení nebo přesunutí zobrazení na popředí, pokud je již zobrazení otevřeno.

Dvojím kliknutím na název stopy v detailu stopy, název plug-in modulu v zobrazení plug-in modulu a název přídavného výstupu v jeho detailu je možné název editovat. Maximální délka názvu je nastavena na 12 znaků. Výchozí název instancí plug-in modulů je přebrán z názvu jejich typu. Ten může přesáhnout 12 znaků a nemusí tak být příliš čitelný. Je na uživateli, zda si instanci modulu přejmenuje ihned po vytvoření nebo ponechá výchozí. Názvy nastavené přes komunikační rozhraní nejsou nijak ošetřovány a je tedy možné nastavit i delší názvy, pokud to připojené rozhraní vyžaduje.

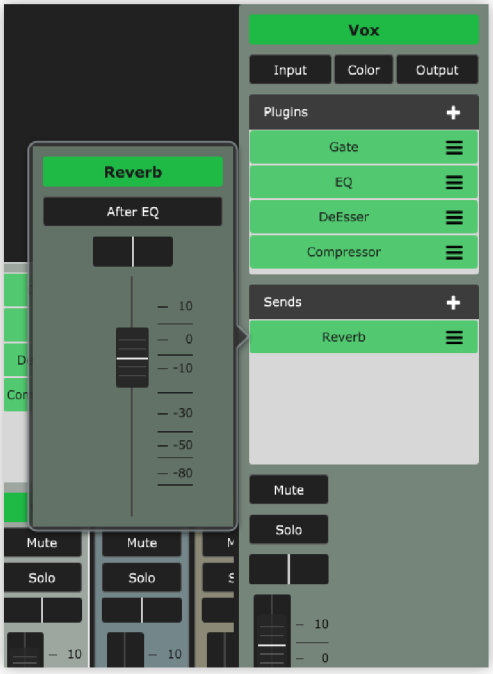
Seznam vstupních a výstupních prvků stopy (otevřen kliknutím na tlačítko *Input* nebo *Output*) zobrazuje všechny dostupné prvky, ke kterým se stopa může připojit. Prvky, ke kterým je již připojena, jsou zobrazeny zeleně (viz obr. 3.6). K připojení nebo odpojení prvku dojde po kliknutí na příslušný prvek.



Obr. 3.6: Zobrazení seznamu vstupních prvků stopy

Kliknutím na položku přídavného zvukového výstupu stopy se zobrazí jeho detail obsahující název, nastavení pozice ve zvukovém řetězci, panoramatu a zesílení (viz obr. 3.7). Pro přidání nového přídavného zvukového výstupu stopy slouží tlačítko se symbolem plus nad seznamem těchto položek, kdy si uživatel vybírá pozici ve zvukovém řetězci.

Kliknutím na symbol pro manipulaci stopy nebo její název se stopa zvolí a zobrazí se její detail. Pro opačnou akci může uživatel kliknout do prázdné plochy mimo seznam stop nebo prázdné plochy spodní lišty.



Obr. 3.7: Zobrazení detailu přídavného zvukového výstupu

Tlačítko se symbolem plus v dolní liště slouží pro přidávání stopy. Ta je vytvořena za aktuálně zvolenou stopu nebo na konec seznamu, pokud není žádná stopa zvolena. Výchozí barvy stop jsou: zelená pro zvukovou stopu, světle modrá pro efektovou stopu, tmavě modrá pro skupinovou sběrnici a žlutá pro výstupní sběrnici.

3.8.2 Zobrazení nastavení aplikace

Kliknutím na tlačítko nastavení v dolní liště (symbol ozubeného kola) dojde k přepnutí na zobrazení nastavení aplikace (viz obr. 3.8), jež má několik částí:

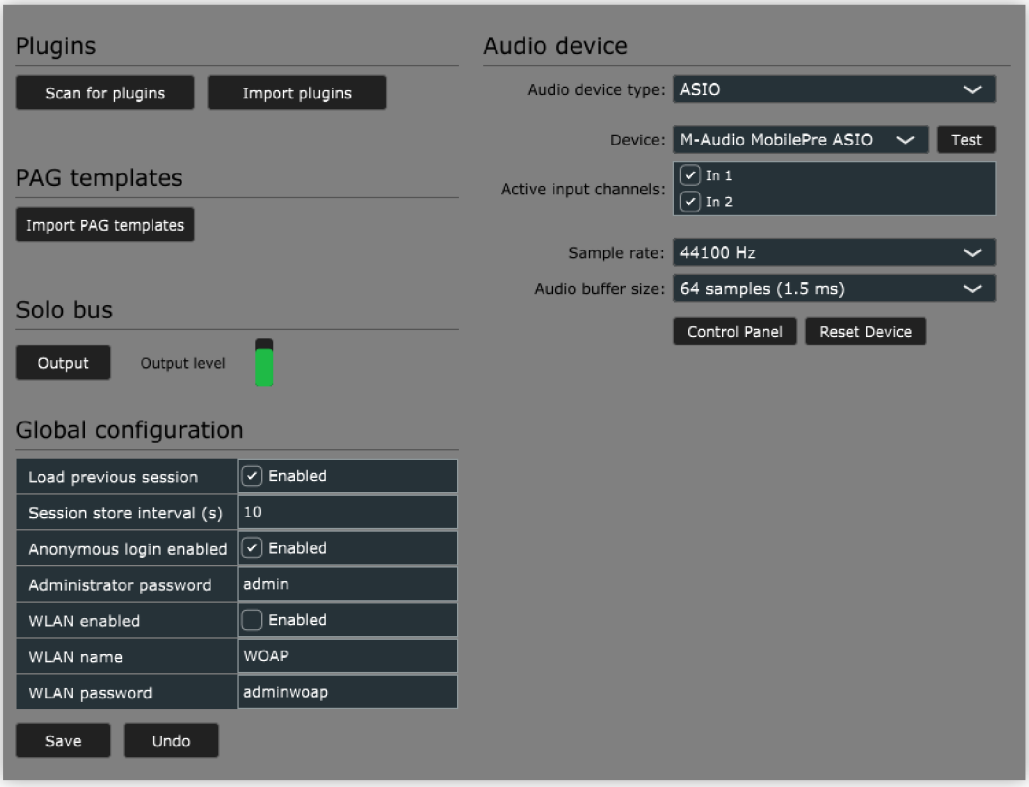
- správu plug-in modulů (*Plugins*),
- import alternativních zobrazení plug-in modulů (*PAG templates*),
- nastavení sběrnice Solo (*Solo bus*),
- nastavení globální konfigurace (*Global configuration*),
- nastavení zvukového zařízení (*Audio device*).

Správa plug-in modulů

Přidávání VST plug-in modulů je zjednodušeno použitím správce modulů a jeho funkce pro import. Kliknutím na tlačítko *Import plugins* je otevřen průzkumník souborů, kde uživatel může zvolit moduly, které chce importovat. Následně pro každý typ modulu proběhne vytvoření instance modulu a dotázání se uživatele, zda jej chce importovat. Modul nemusí být kompatibilní nebo může být chybný,

proto je před dotázáním uživatele krátká časová prodleva, která ponechá modulu čas na inicializaci. Při schválení importu modulu je soubor modulu zkopírován do složky *Plugins* v adresáři aplikace WOAP Host. Pokud je modul chybný, aplikace se může neočekávaně ukončit nebo zastavit, není tedy doporučeno provádět import ve chvíli, kdy probíhá používání aplikace pro účely zpracování zvuku. Seznam nekompatibilních modulů je uložen do souboru *pluginBlacklist.txt* adresáře aplikace WOAP Host.

Druhým způsobem importování VST modulů je zkopírování modulů do složky *Plugins* a následné stisknutí tlačítka *Scan for plugins*.[10] Tato metoda vytvoří instanci každého z modulů pro účely zjištění jeho informací, uživateli však nenabídne možnost import odmítnout.



Obr. 3.8: Zobrazení nastavení hostitelské aplikace

Import alternativních zobrazení plug-in modulů

Importovat alternativní zobrazení plug-in modulů lze kliknutím na tlačítko *Import PAG templates*. Tím je otevřen průzkumník souborů, podobně jako při importu zásuvných modulů, kdy si uživatel zvolí soubory, které chce importovat. Během importu je každá ze šablon částečně validována, přičemž nevalidní šablony nejsou

[10]Složka *Plugins* se vytvoří automaticky při prvním spuštění aplikace.

importovány. Název souboru šablony je při importu změněn na název identifikátoru typu plug-in modulu.

Importovat je taktéž možné zkopírováním šablon do složky *pag-templates* v kořenovém adresáři aplikace WOAP WebClient, přičemž název souboru musí odpovídat identifikátoru typu plug-in modulu.[11]

Pokud není v kořenovém adresáři hostitelské aplikace složka s aplikací webového rozhraní, je import šablon sice proveden, ale ve webovém rozhraní šablony nebudou nalezeny. V tomto případě je nutné importovat šablony ručním kopírováním. Tato situace může nastat např. v případě, že je hostitelská aplikace spouštěna z jiného zařízení než aplikace webového serveru.

Nastavení sběrnice Solo

Aplikace umožňuje přiřazení sběrnice *Solo* do kterékoli výstupní jednotky zařízení. To je možné provést stisknutím tlačítka *Output* a zvolením příslušné jednotky. Vedle tlačítka je indikátor výstupní úrovně sběrnice (nezobrazuje přebuzení sběrnice).

Nastavení globální konfigurace

Sekce globální konfigurace nabízí nastavení hodnot třídy `ConfigurationManager` a jejich uložení (tlačítkem *Save*). Kliknutím na tlačítko *Undo* se hodnoty vrátí do aktuálně uloženého stavu.

Nastavení zvukového zařízení

Sekce pro nastavení zvukového zařízení nabízí nastavení typu zařízení (*Audio device type*), vstupního zařízení (*Input*), výstupního zařízení (*Output*), aktivních vstupních kanálů (*Active input channels*), aktivních výstupních kanálů (*Active output channels*), vzorkovací frekvence (*Sample rate*) a délky zvukové vyrovnávací paměti (*Audio buffer size*). Některá nastavení nemusí být k dispozici. U některých zařízení bylo zjištěno, že nastavení délky vyrovnávací paměti nebo vzorkovací frekvence se nijak neprojeví. Je tedy nutné tuto hodnotu nastavit přímo v ovladači dodaném výrobcem a aplikaci restartovat.

[11] Identifikátor plug-in modulu lze vyčíst ze šablony z parametru `id` elementu `<plugin>`.

3.9 Rozšíření aplikace

Hostitelská aplikace obsahuje spoustu funkcí, které z ní dělají mocný nástroj pro práci se zvukovými signály v reálném čase. Existují však funkce, jimiž disponují konkurenční produkty a které tato aplikace nenabízí, a to např.:

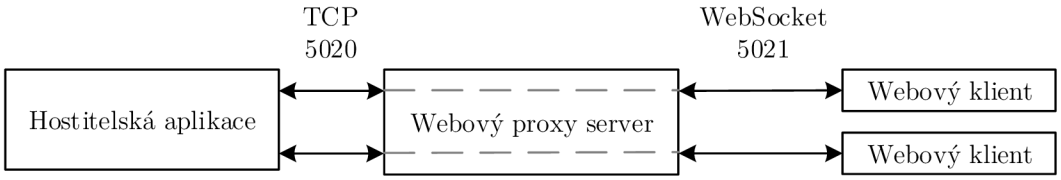
- správa uživatelských přednastavení,
- podpora MIDI zpráv a MIDI nástrojů,
- nastavení zpoždění signálu stop,
- nastavení parametrů vstupních jednotek,
- volba pozice výstupu stop pro sběrnici Solo,
- vícekanálový záznam zvuku.

4 Webový proxy server

Konzolová aplikace WOAP WebProxy slouží pro převod OSC zpráv z protokolu TCP na protokol WebSocket. Jedná se o velice jednoduchou aplikaci, jež na jedné straně komunikuje s hostitelskou aplikací a na straně druhé s klienty webového ovládacího rozhraní, viz obr. 4.1.

Jak je již uvedeno v kapitole 2, komunikační rozhraní aplikace WOAP Host využívá protokol TCP. Přímá komunikace pomocí protokolu TCP je však v prohlížečích možná pouze použitím technologie WebRTC (Web Real-Time Communication [26]), jejíž implementace je složitá a bez přístupu k internetu nemusí fungovat. Je proto vhodné použít některou z alternativ, kterými je protokol HTTP (Hypertext Transfer Protocol) nabízející spojení typu požadavek-odpověď, nebo protokol WebSocket, jenž nabízí spojení obousměrné. Řízení aplikace v reálném čase vyžaduje nízké dopravní zpoždění zpráv, vysokou datovou propustnost a spolehlivý obousměrný přenos dat, protokol WebSocket je tedy nejlepší, pokud ne jedinou, volbou [27].

Aplikace je naprogramována v jazyce TypeScript, je kompilována do jazyka JavaScript a spouštěna v prostředí Node.js. Vstupním bodem aplikace je skript `index.ts`, který definuje třídu `WebProxyServer` a `WebProxyClient`. Při spuštění aplikace dojde k vytvoření instance třídy `WebProxyServer`. Ta spustí WebSocket server přiřazený k portu 5021, který vyčkává na připojení klienta. Při připojení dojde k vytvoření nové instance třídy `WebProxyClient` a ta dále provede připojení k hostitelské aplikaci (protokol TCP, port 5020). Po úspěšném spojení je klientovi odeslána zpráva `ApiCoreReady` signalizující, že je hostitelská aplikace připravena a může být zahájena komunikace. V případě, že dojde k přerušení spojení mezi hostitelskou aplikací, je klientovi odeslána zpráva `ApiCoreError`. Přijaté zprávy jsou okamžitě předány protistraně a nejsou nijak specificky dekódovány ani validovány.



Obr. 4.1: Propojení aplikací pomocí webového proxy serveru

5 Webové ovládací rozhraní

Webové ovládací rozhraní WOAP WebClient slouží pro plnohodnotné řízení hostitelské aplikace. Jedná se o webovou aplikaci naprogramovanou v jazyce TypeScript s využitím technologie React [28], která je spustitelná na standardním zařízení s aktuálním webovým prohlížečem a je tedy možné ji využívat jak na mobilních telefonech a tabletech, tak laptopech nebo stolních počítačích.

Oproti hostitelské aplikaci je webové rozhraní sestaveno z menších funkčních bloků, neboť neprovádí žádnou práci se vzorky zvukového signálu a nemusí zajišťovat správu a načítání zásuvných modulů. Webové rozhraní nenabízí zobrazení a editaci systémových nastavení.

Aplikace je rozdělena na datovou a prezenční vrstvu a je zde ve velké míře využit návrhový vzor Observer pattern, podobně jak je tomu u hostitelské aplikace. Zprávy komunikačního rozhraní a způsob jejich zpracování jsou taktéž implementovány obdobným způsobem, rozdíly jsou pouze ve způsobu zápisu vlivem odlišností jazyků C++ a TypeScript, sémanticky se však vůbec neliší.

5.1 Datová vrstva

Datová vrstva aplikace obsahuje modely a úložiště. Model reprezentuje soubor dat vztahující se k jednomu logickému celku (např. instanci hostitelské aplikace), úložiště naopak obsahují data, jež není možné modelem abstrahovat.

5.1.1 Modely

Model je souborem dat vztahujícím se k jednomu logickému celku. Příkladem je model `TrackModel`, který je přímou reprezentací třídy `Track` hostitelské aplikace. Obsahuje tedy identifikátor, název, barvu, hodnotu zesílení, panoramatu, seznam plug-in modulů (modelů `TrackPluginModel`), seznam přídavných zvukových výstupů (modelů `TrackSendModel`) a několik dalších informací.

Modely mohou přímo využívat komunikační rozhraní. Při vytvoření své instance odešlou zprávy typu `AddListener` a tím se zaregistrují k poslechu změn hodnot dat hostitelské aplikace. Pokud se některá z hodnot změní, odešle hostitelská aplikace zprávu (`TrackNameChanged`) příslušnému modelu, ten si aktualizuje jeho vnitřní stav a oznámí prezenční vrstvě (nebo jiným posluchačům), že došlo ke změně.

V případě, že uživatel změní hodnotu v prezenční vrstvě, dojde k okamžitému nastavení hodnoty v modelu a oznámení této změny ostatním posluchačům modelu. Následně se odešle nová hodnota hostitelské aplikaci (např. zprávou `TrackSetName`), která vyvolá výše uvedený proces. Příchozí zpráva změny (`TrackNameChanged`)

následně u klienta, který změnu vyvolal, nevyvolá žádnou událost, neboť se hodnoty shodují. Okamžité nastavení hodnoty v modelu je velmi důležité pro zachování rychlé odezvy aplikace, jinak by při pomalém připojení mohla prezenční vrstva reagovat se zpožděním.

5.1.2 Úložiště

Úložiště (anglicky *store*) je souborem dat určitého typu, které nelze abstrahovat modelem. Příkladem je úložiště `SystemStore` obsahující, mimo jiné, informace o systému a příznak úspěšného spojení aplikace nebo přihlášení uživatele. Úložiště mohou využívat komunikační rozhraní obdobným způsobem jako modely. Zavoláním metody `Store::reloadStore()`, kterou implementují všechna úložiště, jsou všechny hodnoty úložiště smazány (resp. obnoveny do výchozího stavu) a případně načteny znovu (pokud využívají komunikační rozhraní). Tato metoda je využita při načtení aplikace a v případě, že dojde k přerušení spojení, neboť již není zaručena aktuálnost dat.

5.2 Prezenční vrstva

Prezenční vrstva aplikace využívá technologii React, která zavádí koncept grafických komponent, čímž velmi zjednodušuje práci s dynamickým grafickým rozhraním a podstatně snižuje výpočetní náročnost. Grafické komponenty (třídy dědící `React.Component`) přímo pracují s modely a úložišti a často jsou jejich posluchači. Rozhraní je téměř shodné s rozhraním hostitelské aplikace, proto zde jednotlivé funkční a grafické prvky nebudou nijak popsány.

5.3 Alternativní zobrazení VST plug-in modulů

Webová aplikace umožňuje ovládání VST plug-in modulů a využívá k tomu vlastní způsob vykreslení zobrazení nazvaný PAG (Plugin Alternative GUI). Základem tohoto zobrazení je XML (Extensible Markup Language) konfigurační soubor, který obsahuje šablonu příslušného modulu. Tato šablona je napsána ve vlastním jazyce PAG-JSX vycházejícího z jazyka JSX (JavaScript XML), což je jazyk používaný právě technologií React [29]. Jedná se o nástavbu jazyka HTML, obohaceného o elementy odpovídající komponentám (jejich názvy jsou s velkým počátečním písmenem). Šablony jsou tak velmi jednoduše modifikovatelné a jejich vytvoření a editace nevyžaduje příliš odborných znalostí.

Volitelnou součástí konfiguračních souborů jsou skripty v jazyce JavaScript, které jsou spuštěny při vykreslení těchto zobrazení. To umožní vývojářům přizpůsobit si grafické rozhraní dle jejich libosti.

Pokud chce uživatel zobrazit plug-in modul, který nemá přidružený žádný XML konfigurační soubor, nebo je soubor nevalidní, je vykreslena výchozí šablona obsahující základní ovládací prvky pro každý parametr modulu. Toto zobrazení však není uživatelsky příliš přívětivé.

Konfigurační soubory zpracovává instance třídy `PAGConfigParserXML`, jež převádí XML elementy na jejich třídní reprezentaci. Instance třídy `PAGViewCompiler` následně kompiluje šablonu zobrazení pro její vykreslení.

Struktura konfiguračního souboru je rozdělena na čtyři části (viz výpis 5.1): seznam parametrů (`<parameters>`), seznam šablon (`<views>`), skript pro obsluhu zobrazení (`<script>`) a seznam obrázků (`<images>`).

Výpis 5.1: Schéma PAG konfiguračního souboru

```
<PAGConfig version="1.0">
  <plugin ref="pluginName" version="1" id="123456">
        <parameters>
            ...
        </parameters>
        <views>
            ...
        </views>
        <images>
            ...
        </images>
        <script type="text/javascript">
          ...
        </script>
    </plugin>
</PAGConfig>
```

5.3.1 Seznam parametrů

První částí konfiguračního souboru PAG je seznam parametrů plug-in modulu. Ten je nutný pro párování parametrů s ovládacími prvky šablony uvedenými níže. Prvek seznamu, element `<patameter>`, obsahuje svůj název (atribut `ref`), typ (atribut `type`) a případně pole hodnot jako své potomky. Typ hodnoty parametru je `float` pro spojitou množinu hodnot, nebo `switch` pro diskrétní množinu hodnot. Pro typ

`switch` lze dále uvést konkrétní hodnoty jako potomky elementu, a to elementy `<step>` s atributem `value`. Ve výchozím stavu má parametr s typem `switch` dvě hodnoty, a to 0 a 1. Hodnoty parametrů jsou vždy v rozsahu 0 až 1 a nejsou nijak validovány. Typy parametrů jsou znázorněny ve výpisu 5.2.

Výpis 5.2: Elementy typu `<parameter>` PAG konfiguračního souboru

```
<parameters>
    <parameter ref="param1" type="float" />
    <parameter ref="param2" type="switch" />
    <parameter ref="param3" type="switch">
        <step value="0" />
        <step value="0.5" />
        <step value="1" />
    </parameter>
</parameters>
```

5.3.2 Seznam šablon

Druhou, neméně důležitou částí konfiguračního souboru je seznam šablon zobrazení plug-in modulu (viz výpis 5.3). Ačkoli současná verze aplikace podporuje pouze jednu šablonu v rámci tohoto seznamu, s ohledem na budoucí rozšíření aplikace byla tato část taktéž vytvořena jako seznam. Prvkem seznamu je element `<view>` nesoucí následující atributy: příznak označení výchozí šablony (`default`, není v této verzi podporován a může být vynechán), název šablony (`name`, není podporován a může být vynechán) a jazyk šablony (`lang` podporován pouze `pag-jsx`, musí být uveden).[1]

Výpis 5.3: Element typu `<view>` PAG konfiguračního souboru

```
<views>
    <view default="1" name="view1" lang="pag-jsx">
        ...
    </view>
</views>
```

[1] Je předpokládáno, že v budoucím rozšíření aplikace dojde k podpoře několika šablon zobrazení pro každý z plug-in modulů. V tom případě by parametr `default` označoval, které ze zobrazení bude načteno jako výchozí, a parametr `name` identifikoval jednotlivá zobrazení. Mezi těmito zobrazeními by mohlo být přepínáno pomocí vloženého skriptu, ať už na základě hodnoty některého z parametrů nebo uživatelovy akce.

Potomci elementu `<view>` jsou elementy zobrazení v jazyce PAG-JSX, který je hybridem mezi jazyky HTML a JSX se sadou předdefinovaných komponent. Elementy PAG-JSX mohou být všechny HTML elementy, jež jsou běžně podporovány, nebo elementy z předdefinované sady elementů PAG uvedených níže.

Element PAGPluginName

Element `<PAGPluginName>` reprezentuje název plug-in modulu, jehož zobrazení prezentuje. Element nemá potomky ani atributy.

Element PAGPluginId

Element `<PAGPluginId>` reprezentuje identifikátor plug-in modulu, jehož zobrazení prezentuje. Element nemá potomky ani atributy.

Element PAGImage

Element `<PAGImage>` reprezentuje obrázek načtený z konfiguračního souboru PAG. Obsahuje atribut `ref` odkazující na prvek v seznamu obrázků a nemá žádné potomky.

Element PAGParameter

Element `<PAGParameter>` je párovým elementem označujícím kontext parametru. Obsahuje atribut `ref`, který je použit pro párování parametru (viz kap. 5.3.1). Jeho potomky mohou být všechny PAG-JSX elementy.

Element PAGParameterName

Element `<PAGParameterName>` reprezentuje název parametru, v jehož kontextu je umístěn. Element nemá potomky ani atributy.

Element PAGParameterValue

Element `<PAGParameterValue>` reprezentuje hodnotu parametru, v jehož kontextu je umístěn. U modulů, jež podporují textové hodnoty, je tato hodnota textová.[2] Element nemá potomky ani atributy.

[2]Některé `plug-in` moduly nabízí textové hodnoty parametrů, což zvyšuje čitelnost hodnoty. Příkladem může být modul, který pro hodnotu 0 parametru dvoupolohového přepínače vrátí textovou hodnotu Off a pro hodnotu 1 hodnotu On.

Element PAGRotarySlider

Element `<PAGRotarySlider>` reprezentuje grafické zobrazení rotačního enkodéru parametru, v jehož kontextu je umístěn. Má atribut `size` pro zvolení velikosti v pixelech, `color-accent` pro barvu indikátorů, `color-fill` pro barvu výplně a přepínač `inverted` pro otočení směru rotace. Element nemá potomky.

Element PAGLinearSlider

Element `<PAGLinearSlider>` reprezentuje grafické zobrazení posuvného enkodéru parametru, v jehož kontextu je umístěn. Má atributy `width` a `height` pro zvolení šířky a výšky v pixelech, `color-accent` pro barvu indikátorů, `color-fill` pro barvu dráhy, přepínač `inverted` pro otočení směru tahu a přepínač `valueBar` pro skrytí barevné indikace dráhy (ve výchozím stavu zapnuto). Orientace elementu je nastavena dle jeho rozměrů: je-li jeho šířka menší než výška, je zobrazen vertikálně. Element nemá potomky.

Element PAGButton

Element `<PAGButton>` reprezentuje grafické zobrazení tlačítka, v jehož kontextu je umístěn. Má atributy `width` a `height` pro zvolení šířky a výšky v pixelech, `color-accent` a `color-fill` pro zvolení barevného schématu, přepínač `inverted` pro inverzi stavu a atribut `text` pro zadání textu tlačítka. Element nemá potomky.

Příznakové atributy elementů mohou nabývat hodnotu 0 nebo 1, atributy barev `color-accent` a `color-fill` hexadecimální zápis (např. `#AB12CD`) nebo dekadický zápis (např. `rgb(42, 42, 42)` či `rgba(42, 42, 42, 0.2)` pro barvu s alfa kanálem).

Předdefinované elementy obsahují minimum vlastních stylistických prvků a zobrazení je tak značně závislé na rozvržení pomocí běžných HTML elementů a kaskádových stylů. To dává vývojáři možnost přizpůsobit si rozhraní dle svých potřeb tak, aby co nejvíce odpovídalo nativnímu zobrazení zásuvného modulu dodaného výrobcem. Výpis 5.4 zobrazuje použití šablonových elementů všech typů.

Výpis 5.4: Ukázka zdrojového kódu šablony PAG

```
<div style="width: 200px; height: 200px;" class="demoView">
    <PAGPluginName />
    <PAGPluginId />
    <PAGImage ref="background" />
    <div style="left: 50px; position: absolute;">
        <PAGParameter ref="param1">
            <PAGParameterName />
            <PAGRotarySlider size="50"
                             color-fill="#181F22"
                             color-accent="#42A2C8"
                             inverted="1"
            />
            <PAGParameterValue />
        </PAGParameter>
    </div>
    <div style="left: 100px; position: absolute;">
        <PAGParameter ref="param2">
             <PAGButton width="50"
                        height="30"
                        text="On"
                        color-fill="#181F22"
                        color-accent="#42A2C8"
             />
        </PAGParameter>
    </div>
    <div style="left: 150px; position: absolute;">
        <PAGParameter ref="param3">
            <PAGLinearSlider width="20"
                             height="50"
                             color-fill="#181F22"
                             color-accent="#42A2C8"
            />
        </PAGParameter>
    </div>
</div>
```

5.3.3 Skript pro obsluhu zobrazení

Vývojáři šablony je umožněno vložení vlastního skriptu v jazyce JavaScript, který možnosti šablony posouvá ještě dále. Skript umožňuje definici několika metod a přístup k metodám a členským proměnným instance třídy `TrackPluginModel` příslušného modulu. Tato instance je uložena v proměnné `pluginModel` v rámci tohoto skriptu (viz výpis 5.5).

Pomocí skriptu je možné odposlouchávat událost otevření zobrazení (metoda `viewCreated()`), uzavření zobrazení (metoda `viewWillBeDeleted()`), změnu hodnoty parametru (metoda `parameterChanged()`), změnu příznaku zapnutí (metoda `activeChanged()`), změnu příznaku přemostění (metoda `bypassChanged()`) nebo změnu názvu modulu (metoda `nameChanged()`).

Přístup k proměnné `pluginModel` může představovat potenciální riziko kvůli neoprávněnému přístupu do části modulu, která by neměla být přímo měněna. Z tohoto důvodu je doporučeno, aby uživatel nepoužíval PAG konfigurační soubory získané z neověřených zdrojů.

Výpis 5.5: Rozhraní skriptu pro obsluhu zobrazení PAG

```
pluginModel.setExternalListener({
    viewCreated: function() {},
    viewWillBeDeleted: function() {},
    parameterChanged: function(parameterModel, newValue, newValueText) {},
    activeChanged: function(newState) {},
    bypassChanged: function(newState) {},
    nameChanged: function(newName) {}
});
```

5.3.4 Seznam obrázků

Obrázky, jež chce vývojář uložit společně se šablonou, je možné vkládat jako elementy typu `<image>`, potomky elementu `<images>`. Element `<image>` má atribut `ref` obsahující unikátní identifikátor obrázku, který je určen pro párování s elementem `<PAGImage>` popsaným výše. Potomkem elementu jsou obrazová data zakódována do formátu Base64 (viz výpis 5.6). Tato data je možné zalamovat, neboť všechny mezery a zalomení jsou při čtení odstraněny. Obrázek je následně zobrazen jako HTML element typu `<image>`, podpora formátu obrázku je tedy přímo určena použitým webovým prohlížečem. Doporučeným formátem je formát PNG (Portable Network Graphics), neboť zachovává dobrou kvalitu po kompresi, podporuje alfa kanál a je podporován na všech prohlížečích již několik let [30].

Výpis 5.6: Element typu `<image>` konfiguračního souboru PAG

```
<image ref="background">
    c3VjaCBoaWRkZW4gdmVyeSBiYXNlNjQ=
</image>
```

5.3.5 Ukázka zobrazení

Výsledné alternativní zobrazení plug-in modulu může relativně přesně odpovídat tomu, jež je dodáno výrobcem. Na obrázku 5.1 je porovnáno originální zobrazení plug-in modulu Boot EQ MKII (vlevo) s alternativním zobrazením (vpravo).

Obr. 5.1: Porovnání zobrazení plug-in modulu

6 Podpůrné aplikace

Trojici aplikací popsaných výše doplňují podpůrné aplikace, jež uživateli usnadní spuštění aplikace a vytváření alternativních zobrazení plug-in modulů.

6.1 Nástroj pro tvorbu zobrazení plug-in modulů

Aplikace WOAP PAG Generator slouží pro jednoduché a uživatelsky přívětivé vytváření alternativních zobrazení plug-in modulů. Tato zobrazení slouží pro prezentování ve webovém rozhraní jako náhrada za zobrazení dodaného výrobcem modulu.

Obr. 6.1: Aplikace pro tvorbu alternativních zobrazení plug-in modulů

Ovládání aplikace je jednoduché a intuitivní. Po spuštění aplikace může uživatel načíst zásuvný modul, jehož zobrazení chce vytvořit. To lze provést kliknutím na položku nabídky *File* a *Open plugin file* nebo přetažením souboru zásuvného modulu do levé části okna. Následně je modul načten a zobrazen v plovoucím okně (na obrázku 6.1 vpravo). Zobrazení v levé části aplikace není aktivní zobrazení plug-in modulu, jedná se pouze o jeho snímek (anglicky *screenshot*), který poslouží jako základ pro vložení alternativních ovládacích prvků.

Uživatel může snímek plug-in modulu vytvořit kdykoli znovu stisknutím tlačítka *Create plugin snapshot* v pravé horní části aplikace. To může být pro některé plug-in moduly užitečné, v případě modulu Boot EQ MKII (na obrázku) např. kvůli tomu, že aktivováním parametru *Tube on* dojde k rozsvícení elektronky, která v zobrazení působí lepším dojmem. Pokud uživatel okno s aktivním zobrazením zavře, je možné jej znovu otevřít tlačítkem *Open plugin editor*, které se objeví na místě tlačítka *Create plugin snapshot*.[1] Pokud je uživatel se snímkem spokojen, může začít vytvářet ovládací prvky.

Vkládání ovládacích prvků

Tažením za ovládání parametru v aktivním okně plug-in modulu dojde k zobrazení ovládacích prvků a typu parametru v pravé dolní části hlavního okna aplikace. Uživateli je doporučeno táhnout za parametr v celém jeho rozsahu, aby aplikace mohla správně určit, zda se jedná o přepínač, anebo parametr se spojitou množinou hodnot. Následně si uživatel zvolí vhodný ovládací prvek ze sady, jež je mu zobrazena, a tento prvek přetažením umístí na vhodnou pozici do prostoru snímku modulu.

Rozměry ovládacího prvku lze změnit tažením za jeho okraje. Změnit barvu ovládacího prvku kliknutím na ukazatel barvy *Fill* a *Accent* a následnou volbou barvy ze zobrazené palety či kliknutím na tlačítko *Pick* pod ukazatelem barvy a vybráním barvy ze snímku plug-in modulu (tzv. funkce kapátko). Posuvem kolečka myši nad ovládacím prvkem dojde ke změně hodnoty parametru, uživatel si tak může jednoduše vyzkoušet, jak bude zobrazení vypadat pro různé hodnoty parametru. Obrácení směru rotace nebo tažení ovládacího prvku je možné kliknutím na položku *Inverted*.

Každý parametr může mít přiřazen pouze jeden ovládací prvek. Odebrat ovládací prvek lze kliknutím pravým tlačítkem myši a následně volbou *Delete*. Změnu textu ovládacího prvku tlačítka (pokud je parametr dvoupolohovým přepínačem) lze provést dvojím kliknutím na tlačítko a zadáním nového řetězce. Uživatel může tento proces opakovat pro každý další parametr zásuvného modulu, dokud nejsou všechny ovládací prvky parametrů vhodně nahrazeny těmi alternativními.

[1] K vytvoření snímku modulu je nutné otevřít jeho zobrazení. U některých modulů je otevření velmi pomalé, a proto může při prvotním načtení zásuvného modulu dojít k pořízení vadného snímku, jenž je nejčastěji průsvitný, neboť je místo plug-in modulu sejmuto zobrazení pracovní plochy uživatele.

Vkládání grafických elementů

V pravé horní části aplikace se nachází dvojice geometrických tvarů – čtverec a kruh. Ty slouží pro překrytí nepotřebných částí zobrazení plug-in modulu, nejčastěji indikátorů nebo animovaných částí. Jejich vložení do zobrazení, nastavení barev a velikosti a odebrání je shodné s ovládacími prvky. Tyto tvary jsou vždy umístěny pod úrovní ovládacích prvků a není tedy možné, aby překrývaly ovládací prvky.

Uložení a načítání šablon

Po dokončení práce se zobrazením je nutné šablonu uložit, a to kliknutím na položku nabídky *File* a *Export XML config file*. Výslednou šablonu je možné nahrát prostřednictvím hostitelské aplikace funkcí *Import PAG templates*. Uživatel může výslednou šablonu znovu otevřít a upravit kliknutím na položku nabídky *File* a *Open XML config file*, kdy nejprve zvolí šablonu, již chce načíst, a následně plug-in modul, který k této šabloně přísluší. Veškeré vlastní úpravy šablony jsou během načtení zahozeny, neboť je aplikace nemusí umět interpretovat.

Průvodce vytvořením šablony

Součástí aplikace je taktéž experimentální průvodce spustitelný kliknutím na položku nabídky *Help* a *Start tutorial*. V pravém dolním rohu aplikace se zobrazí nápověda, jež uživatele vede krok za krokem od načtení plug-in modulu po uložení šablony. Je vhodné si okna aplikace rozvrhnout tak, aby aktivní editor zásuvného modulu byl viditelný zároveň s hlavním oknem aplikace. Průvodce automaticky mění hodnotu každého z parametrů, zatímco uživatel je vyzván k přiřazení ovládacích prvků a nastavení jejich velikostí a barev. Nelze však jednoduše rozpoznat specifické chování plug-in modulů a parametrů, a některé vlastnosti parametrů (např. skutečný počet kroků parametru) tak nemusí být správně určeny. Průvodce je možné ukončit z nabídky *Help* kliknutím na *Stop tutorial*.

Rozšíření alternativního zobrazení

S ohledem na různorodost zobrazení plug-in modulů by bylo vhodné aplikaci rozšířit tak, aby alternativní zobrazení působilo ještě více přirozeně a bylo funkčně použitelné na všech modulech. Mezi tyto rozšíření patří:

- volba začátečního a konečného úhlu dráhy rotačního enkodéru,
- volba zkosení průběhů,
- přidání výsuvných seznamů,
- přidání komponenty pro (sloupcovou) indikaci hodnoty parametru,
- podpora více zobrazení pro jeden modul.

6.2 Webový server

Aplikace WOAP WebServer je konzolová aplikace, která zpřístupňuje soubory webového rozhraní tak, aby je bylo možné spustit z webového prohlížeče. Aplikace využívá technologii Mongoose (Mongoose Embedded Web Server Library), jež je ověřená a jednoduchá pro použití.[2] Díky tomu má vlastní zdrojový kód této aplikace pouze několik desítek řádků. Webový server je přiřazen k portu 80, což je výchozí port pro komunikaci v protokolu HTTP pro přenos webových stránek. Tento server zpřístupňuje všechny soubory v jeho kořenovém adresáři kromě sama sebe a je možné jej tedy použít i pro jiné účely než pro tuto práci. Výchozím souborem (indexem) je soubor `index.html`.

6.3 Spouštěč

Pro snadné spouštění celé platformy je součástí práce aplikace s názvem WOAP, která spouští hostitelskou aplikaci, webový server a webový proxy server. Aplikace žádným způsobem neověřuje běh spuštěných aplikací. Pro účely nasazení aplikací v systémech pro koncového zákazníka je doporučeno všechny aplikace spouštět jako služby systému Windows, neboť je následně zaručen jejich běh a v případě, že se kterákoli z aplikací ukončí, je systémem Windows automaticky opět spuštěna.

[2]Mongoose Embedded Web Server Library: <`https://cesanta.com/proto-http.html`>.

7 Závěr

V práci jsem se zabýval návrhem a implementací softwaru pro digitální mixážní pult postaveného na platformě Windows, který umožňuje vícekanálové zpracování zvukových signálů v reálném čase v libovolném počtu vstupních a výstupních jednotek, směřování signálu mezi těmito jednotkami a vkládání a správu VST plug-in modulů. V práci taktéž zahrnuji aplikaci pro vzdálené řízení pomocí webového prohlížeče, která umožňuje ovládání softwaru pomocí laptopu, stolního počítače nebo mobilního zařízení více uživatelům zároveň. Grafická rozhraní aplikací odpovídají současným standardům a jsou kompatibilní jak se zařízeními využívajícími myš, tak se zařízeními s dotykovým displejem.

Zdrojový kód je napsán dle současných konvencí s ohledem na případné rozšíření, a ačkoli kód není příliš okomentován, jsou třídy, funkce a proměnné nazvány tak, aby z jejich názvu a kontextu bylo snadné určit jejich význam. Práci tvoří přibližně 40 000 řádků vlastního zdrojového kódu (53 000 včetně prázdných řádků a komentářů). Výsledné zpracování je rozděleno na tři hlavní a tři pomocné aplikace.

Jádro práce tvoří hostitelská aplikace umožňující komunikaci se zvukovým rozhraním, vytváření zvukových stop a sběrnic, jejich propojování, přidávání VST plug-in modulů a práci s nimi. Tato aplikace obsahuje vlastní uživatelské rozhraní a zároveň nabízí řízení pomocí OSC komunikačního rozhraní protokolu TCP.

Vzdálené ovládání je možné ve webovém prohlížeči pomocí webového ovládacího rozhraní. To nabízí ovládání hostitelské aplikace v plném rozsahu, a to včetně řízení VST plug-in modulů, jež je realizováno vlastní technologií nazvanou PAG, která dokáže nahradit ovládání modulu dodaného jeho výrobcem. Toho je docíleno pomocí konfiguračních souborů ve formátu XML, které definují podobu alternativního zobrazení plug-in modulů ve webové aplikaci.

Webové rozhraní komunikuje prostřednictvím protokolu WebSocket, zatímco hostitelská aplikace využívá pouze protokol TCP, proto je mezi těmito aplikacemi zařazen webový proxy server, který komunikaci překládá.

Pomocné aplikace zahrnují aplikaci pro jednoduché vytváření konfiguračních souborů alternativních zobrazení VST plug-in modulů, aplikaci webového serveru pro zjednodušení spouštění webového ovládacího rozhraní a spouštěcí aplikaci pro jednoduché spuštění všech hlavních aplikací zároveň.

Výsledné zpracování řeší nedostatky jiných komerčních řešení a přináší zcela nové možnosti ovládání aplikací svého druhu. Použitím vhodných technologií došlo k minimalizaci nákladů a požadavků na hardwarové rozhraní, a to vše se zachováním vysoké modularity. Aplikace může konkurovat řešením jiných výrobců a je připravena pro nasazení do reálného použití.

Literatura

[1] SCHIMMEL, Jiří. *Studiová a hudební elektronika*, 2. vydání, Brno: Vysoké učení technické v Brně, 2015, str. 72. ISBN: 978-80-214-4452-2.

[2] PSW. *RME Interfaces On Tour With Roger Hodgson And His Band*, ProSoundWeb [online], 14. 12. 2015 [cit. 1. 5. 2018]. Dostupné z: <http://prosoundweb.com/channels/live-sound/rme_interfaces_for_roger_hodgson_and_his_band/>.

[3] *Soundcraft Vi1 User Guide*. Soundcraft [online], srpen 2010 [cit. 1. 5. 2018]. Dostupné z: <http://soundcraft.com/en/products/vi1>.

[4] EARGLE, John a FOREMAN, Chris. *Audio engineering for sound reinforcement*, Milwaukee, Wis: Hal Leonard, 2002, str. 81–84. ISBN: 0-634-04355-2.

[5] *Multichannel Monitoring Tutorial Booklet*, 2. vydání. Yamaha Corporation [online], květen 2015 [cit. 1. 5. 2018]. Dostupné z: <http://www.yamahaproaudio.com/europe/en_gb/products/mixers/dm2000vcm/downloads.jsp>.

[6] SELF, Douglas. *Audio Engineering Explained*, Burlington: Taylor & Francis, 2009, str. 114–120. ISBN: 978-0-240-81273-1.

[7] FALCONER, Joel. *Understanding Your Mixer: The Channel Strip*, Envato Tuts+ [online], 3. 11. 2009 [cit. 1. 5. 2018]. Dostupné z: <http://music.tutsplus.com/tutorials/understanding-your-mixer-the-channel-strip--audio-2940>.

[8] ZÖLZER, Udo. *DAFX - Digital Audio Effects*, 1. vydání, New York: John Wiley & Sons, Ltd, 2002, str. 95–105. ISBN: 0-471-49078-4.

[9] IZHAKI, Roey. *Mixing Audio: Concepts, Practices and Tools*, Burlington: Taylor & Francis, 2013, str. 136–138. ISBN: 978-0-240-52222-7.

[10] CERRA, Steve La. *Masterclass: Using Waves MultiRack Live at FOH*, Electronic Musician [online], 23. 9. 2014 [cit. 1. 5. 2018]. Dostupné z: <http://emusician.com/how-to/masterclass-using-waves-multirack-live-at-foh>.

[11] WHITE, Paul. *Mackie DL1608: Digital Mixer With iPad Control*, Sound On Sound [online], říjen 2012 [cit. 1. 5. 2018]. Dostupné z: <http://soundonsound.com/reviews/mackie-dl1608>.

[12] *Yamaha 01V96i Reference Manual*. Yamaha Corporation [online], 2011 [cit. 1. 5. 2018], str. 100–108. Dostupné z: <http://www.yamahaproaudio.com/global/en/products/mixers/01v96i/downloads.jsp>.

[13] SCHIMMEL, Jiří. cit. d., str. 171–172.

[14] SWALLOW, Dave. *Live Audio: The Art of Mixing a Show*, Burlington: Taylor & Francis, 2010, str. 102–103. ISBN: 978-0-240-81604-3.

[15] *StudioLive^{TM} AI-Series Mixers: Owner's Manual*. PreSonus Audio Electronics, Inc. [online], 23. 2. 2017 [cit. 1. 5. 2018], str. 84–88. Dostupné z: <http://presonus.com/products/StudioLive-3242AI/downloads>.

[16] *iLive Reference Guide: Part 1 – Hardware*. Allen & Heath [online], 30. 10. 2010 [cit. 1. 5. 2018], str. 6. Dostupné z: <http://allen-heath.com/key-series/ilive-series/#tab4>.

[17] ROBINSON, Martin. *Getting Started with JUCE*, Birmingham: Packt Publishing Ltd, 2013, 158 str. ISBN: 978-1-78328-331-6.

[18] SINGER, Adam B. *Practical C++ Design: From Programming to Architecture*, The Woodlands: Apress, 2017, str. 34–35. ISBN: 978-1-4842-3056-5.

[19] LIEBERHERR, Karl. *Law of Demeter: Principle of Least Knowledge*, Boston: College of Computer and Information Science, Northeastern University [online], [cit. 1. 5. 2018]. Dostupné z: <http://ccs.neu.edu/home/lieber/LoD.html>.

[20] WRIGHT, Matt. *The Open Sound Control 1.0 Specification* [online], 26. 3. 2002 [cit. 1. 5. 2018]. Dostupné z: <http://opensoundcontrol.org/spec-1_0>.

[21] SRIPARASA, Sai Srinivas. *JavaScript and JSON Essentials*, Birmingham: Packt Publishing Ltd, 2013, 120 str. ISBN: 978-1-78328-603-4.

[22] SINGER, Adam B. cit. d., str. 41–51.

[23] HALSEY, Mike a BALLEW, Joli. *Windows Networking Troubleshooting*, Yorkshire: Apress, 2017, str. 36–38. ISBN: 978-1-4842-3222-4.

[24] Anthony. *Why Rounded Corners Are Easier on the Eyes*, UX Movement [online], 17. 8. 2011 [cit. 1. 5. 2018]. Dostupné z: <http://uxmovement.com/thinking/why-rounded-corners-are-easier-on-the-eyes>.

[25] SELF, Douglas. cit. d., str. 433–434.

[26] LORETO, Salvatore a ROMANO, Simon Pietro. *Real-Time Communication with WebRTC: Peer-to-Peer in the Browser*, Sebastopol: O'Reilly Media, Inc., 2014, 164 str. ISBN: 978-1-449-37187-6.

[27] Gamesparks. *Why the REST don't use WebSockets*, Game Sparks Technologies Ltd [online], [cit. 1. 5. 2018]. Dostupné z: <`http://gamesparks.com/blog/why-the-rest-dont-use-websockets`>.

[28] ROBBESTAD, Sven A. *ReactJS Blueprints*, Birmingham: Packt Publishing Ltd, 2016, 422 str. ISBN: 978-1-78588-654-6.

[29] VIPUL, A. M., SONPATKI, Prathamesh. *ReactJS by Example - Building Modern Web Applications with React*, Birmingham: Packt Publishing Ltd, 2016, str. 21–43. ISBN: 978-1-78528-964-4.

[30] BENDELL, Colin, KADLEC, Tim, WEISS, Yoav, PODJARNY, Guy, DOYLE, Nick a MCCALL, Mike. *High Performance Images: Shrink, Load, and Deliver Images for Speed*, Sebastopol: O'Reilly Media, Inc., 2016, str. 34–45. ISBN: 978-1-491-92580-5.

Seznam symbolů, veličin a zkratek

ASIO	Audio Stream Input/Output
dBFS	Decibels relative to full scale
GUI	Graphical User Interface
HD	High Definition
HTTP	Hypertext Transfer Protocol
JSON	JavaScript Object Notation
JSX	JavaScript XML
MIDI	Musical Instrument Digital Interface
OSC	Open Sound Control
PAG	Plugin Alternative GUI
PNG	Portable Network Graphics
TCP	Transmission Control Protocol
UUID	Universally Unique Identifier
VST	Virtual Studio Technology
WebRTC	Web Real-Time Communication
XML	Extensible Markup Language

Seznam příloh

A Spustitelné soubory

Všechny spustitelné soubory jsou přiloženy v adresáři `Executable`. Uživatel může spouštět jednotlivé aplikace zvlášť nebo využít aplikaci `WOAP.exe`, která spustí hostitelskou aplikaci, webový server a webový proxy server. Všechny aplikace jsou kompatibilní se systémy Windows 7 a novějšími. Pro jejich spuštění není vyžadována instalace žádných závislostí.

```
\
├── WebClient
│   ├── icons
│   ├── pag-templates
│   ├── static
│   ├── index.html
│   └── WOAP WebServer.exe ..... webový server
├── WebProxy
│   └── WOAP WebProxy.exe ..... webový proxy server
├── WOAP Host.exe ..... hostitelská aplikace
├── WOAP PAG Generator.exe ..... generátor zobrazení plug-in modulů
└── WOAP.exe ..... spouštěč
```

Struktura adresáře `Executable`

B Zdrojové kódy

Zdrojové kódy aplikací jsou přiloženy ve složce `Sources`. Jsou napsány v jazyce C++ a TypeScript a tvoří je desítky tisíc řádků vlastního kódu. Jejich kompilace vyžaduje vývojové prostředí Microsoft Visual Studio 2017 v konfiguraci pro kompilaci jazyka C++ a běhové prostředí Node.js (<`http://nodejs.org`>) nainstalované ve výchozí instalaci (tedy včetně nástroje npm).

Kompilace dále vyžaduje zdrojové soubory technologií ASIO, JUCE a VST, které jsou chráněny licenčními podmínkami zakazujícími jejich opětovnou distribuci. Jejich zdrojové soubory je nutné ručně stáhnout a vložit do odpovídajících adresářů dle následující struktury.

```
\
├── ASIO
│   ├── asio.h
│   ├── asiodrvr.h
│   └── ...
├── JUCE
│   ├── examples
│   ├── extras
│   ├── modules
│   └── ...
├── VST3 SDK
│   ├── base
│   ├── bin
│   └── ...
├── WOAP Host
├── WOAP PAG Generator
├── WOAP Runner
├── WOAP WebClient
├── WOAP WebProxy
└── WOAP WebServer
```

Struktura adresáře `Sources`

ASIO SDK: <`http://steinberg.net/en/company/developers.html`>.
Je nutné zkopírovat pouze hlavičkové soubory z adresáře `common`.

Framework JUCE verze 5.2.0.:
<`http://github.com/WeAreROLI/JUCE/tree/5.2.0`>.

VST3 SDK verze 3.6.7.:
<`http://www.steinberg.net/sdk_downloads/vstsdk367_03_03_2017_build_352.zip`>.

B.1 Hostitelská aplikace

Hostitelská aplikace je naprogramována v jazyce C++ a využívá technologie JUCE, ASIO a VST. Je uložena v adresáři `WOAP Host` kořenového adresáře zdrojových souborů.

Aplikace využívá externí modul *danlin_fontawesome* pro zobrazení ikon a piktogramů.[1] Jeho zdrojové soubory jsou přibaleny v adresáři `Modules` a není tak nutné je stahovat.

Součástí zdrojových souborů je taktéž dokumentace zpráv komunikačního rozhraní `Misc\APIDoc.md` ve značkovacím jazyce Markdown a `Misc\APIDoc.PDF` ve formátu PDF.

```
\
├── Builds
│   └── VisualStudio2017
├── JuceLibraryCode
├── Misc
│   ├── APIDoc.md ..... dokumentace komunikačního rozhraní
│   └── APIDoc.pdf ..... dokumentace komunikačního rozhraní
├── Modules
│   └── danlin_fontawesome
├── Source ..... vlastní zdrojové kódy
├── WOAP Host.jucer ..... soubor projektu pro aplikaci Projucer
└── icon.png
```

Struktura adresáře `Sources\WOAP Host`

Kompilace

Kompilace probíhá ve vývojovém prostředí Visual Studio 2017 a vyžaduje zdrojové soubory technologií ASIO, JUCE i VST. Pro kompilaci je nutné:

1. otevřít soubor `Builds\VisualStudio2017\WOAP Host.sln`,
2. zvolit režim kompilace *debug* nebo *release*,
3. spustit kompilaci.

Aplikace `WOAP Host.exe` je pro režim *debug* zkompilována v adresáři `Builds\VisualStudio2017\Win32\Debug`.

Aplikace `WOAP Host.exe` je pro režim *release* zkompilována v adresáři `Builds\VisualStudio2017\Win32\Release`.

[1] Modul *danlin_fontawesome*: <`http://github.com/danlin/danlin_modules`>.

B.2 Webový proxy server

Webový proxy server je naprogramován v jazyce TypeScript a spouštěn v prostředí Node.js. Je uložen v adresáři `WOAP WebProxy` kořenového adresáře zdrojových souborů. Konfigurační soubor `configuration.ts`, jenž je součástí zdrojových souborů, obsahuje nastavení příznaku, zda se jedná o režim vývoje aplikace (`development`) a zda se mají zobrazovat výpisy ladění aplikace (`debug`). Ve výchozím jsou tato nastavení vypnuta.

```
\
├── build
├── messages
├── osc
├── patches
├── scripts
├── sockets
├── configuration-manager.ts
├── configuration.ts ..... konfigurační soubor aplikace
├── icon.ico
├── index.ts
├── package-lock.json
├── package.json
└── tsconfig.json
```

Struktura adresáře `Sources\WOAP WebProxy`

Kompilace

Kompilace probíhá z příkazové řádky a vyžaduje běhové prostředí Node.js a připojení k internetu. Pro kompilaci je nutné:

1. spustit příkazový řádek z adresáře `Sources\WOAP WebProxy`,
2. stáhnout potřebné knihovny příkazem `npm install` (nutné pouze jednou),
3. spustit kompilaci jazyka TypeScript příkazem `npm run tscompile`,
4. spustit sestavení příkazem `npm run build`.

Aplikace `WOAP WebProxy.exe` je zkompilována v adresáři `build`. Spustit ji lze přímo nebo v prostředí Node.js příkazem `node build\index.js`.

B.3 Webové ovládací rozhraní

Webové ovládací rozhraní je naprogramováno v jazyce TypeScript a využívá technologii React. Je uloženo v adresáři `WOAP WebClient` kořenového adresáře zdrojových souborů. Konfigurační soubor `src\configuration.ts`, jenž je součástí zdrojových souborů, obsahuje, mimo jiné, příznak, zda se jedná o režim vývoje aplikace (`development`). Ve výchozím je toto nastavení vypnuto.

```
\
├── build
├── config
├── public
│   ├── icons
│   ├── pag-templates
│   └── index.html
├── runtime
│   └── WOAP WebServer.exe ..................................... webový server
├── scripts
├── src ..................................................... vlastní zdrojové soubory
│   └── configuration.ts ..................... konfigurační soubor aplikace
├── package-lock.json
├── package.json
└── tsconfig.json
```

Struktura adresáře `Sources\WOAP WebClient`

Kompilace

Kompilace probíhá z příkazové řádky a vyžaduje běhové prostředí Node.js a připojení k internetu. Pro kompilaci je nutné:

1. spustit příkazový řádek z adresáře `Sources\WOAP WebClient`,
2. stáhnout potřebné knihovny příkazem `npm install` (nutné pouze jednou),
3. spustit kompilaci jazyka TypeScript příkazem `npm run tscompile`,
4. spustit sestavení příkazem `npm run build`.

Soubory jsou zkompilovány v adresáři `build`. Součástí kompilovaných souborů je i aplikace `WOAP WebServer.exe`, jež je zkopírována z adresáře `runtime`. Pro účely vývoje lze aplikaci spustit příkazem `npm start`.

B.4 Generátor zobrazení plug-in modulů

Nástroj pro generování alternativních zobrazení plug-in modulů je naprogramován v jazyce C++ a využívá technologie JUCE a VST. Zdrojový kód je uložen v adresáři `WOAP PAG Generator` kořenového adresáře zdrojových souborů.

```
\
├── Builds
│   └── VisualStudio2017
├── JuceLibraryCode
├── Source ..................................... vlastní zdrojové kódy
├── WOAP PAG Generator.jucer ..... soubor projektu pro aplikaci Projucer
└── icon.png
```

Struktura adresáře `Sources\WOAP PAG Generator`

Kompilace

Kompilace probíhá ve vývojovém prostředí Visual Studio 2017 a vyžaduje zdrojové soubory technologií JUCE a VST. Pro kompilaci je nutné:

1. otevřít soubor `Builds\VisualStudio2017\WOAP PAG Generator.sln`,
2. zvolit režim kompilace *debug* nebo *release*,
3. spustit kompilaci.

Aplikace `WOAP PAG Generator.exe` je pro režim *debug* zkompilována v adresáři `Builds\VisualStudio2017\Win32\Debug`.

Aplikace `WOAP PAG Generator.exe` je pro režim *release* zkompilována v adresáři `Builds\VisualStudio2017\Win32\Release`.

B.5 Webový server

Aplikace webového serveru je naprogramována v jazyce C++ a využívá technologie JUCE a Mongoose (Mongoose Embedded Web Server Library). Zdrojové soubory Mongoose jsou přibaleny a není nutné je tedy stahovat. Tato aplikace je uložena v adresáři `WOAP WebServer` kořenového adresáře zdrojových souborů.

```
\
├── Builds
│   └── VisualStudio2017
├── JuceLibraryCode
├── Mongoose ..................................... zdrojové kódy technologie Mongoose
├── Source ....................................... vlastní zdrojové kódy
├── WOAP WebServer.jucer ......................... soubor projektu pro aplikaci Projucer
└── icon.png
```

Struktura adresáře `Sources\WOAP WebServer`

Kompilace

Kompilace probíhá ve vývojovém prostředí Visual Studio 2017 a vyžaduje zdrojové soubory technologie JUCE. Pro kompilaci je nutné:

1. otevřít soubor `Builds\VisualStudio2017\WOAP WebServer.sln`,
2. zvolit režim kompilace *debug* nebo *release*,
3. spustit kompilaci.

Aplikace `WOAP WebServer.exe` je pro režim *debug* zkompilována v adresáři `Builds\VisualStudio2017\Win32\Debug`.

Aplikace `WOAP WebServer.exe` je pro režim *release* zkompilována v adresáři `Builds\VisualStudio2017\Win32\Release`.

B.6 Spouštěč

Spouštěcí aplikace WOAP Runner (`WOAP.exe`) je naprogramována v jazyce C++ a využívá technologii JUCE. Její zdrojové soubory jsou uloženy v adresáři `WOAP Runner` kořenového adresáře zdrojových souborů.

```
\
├── Builds
│   └── VisualStudio2017
├── JuceLibraryCode
├── Source ..................................... vlastní zdrojové kódy
├── WOAP Runner.jucer ..... soubor projektu pro aplikaci Projucer
└── icon.png
```

Struktura adresáře `Sources\WOAP Runner`

Kompilace

Kompilace probíhá ve vývojovém prostředí Visual Studio 2017 a vyžaduje zdrojové soubory technologie JUCE. Pro kompilaci je nutné:

1. otevřít soubor `Builds\VisualStudio2017\WOAP Runner.sln`,
2. zvolit režim kompilace *debug* nebo *release*,
3. spustit kompilaci.

Aplikace `WOAP Runner.exe` je pro režim *debug* zkompilována v adresáři `Builds\VisualStudio2017\Win32\Debug`.

Aplikace `WOAP Runner.exe` je pro režim *release* zkompilována v adresáři `Builds\VisualStudio2017\Win32\Release`.