STŘEDOŠKOLSKÁ ODBORNÁ ČINNOST

Obor č. 18: Informatika

SEDAS: ATC simulátor s implementací AI pseudopilotů

Daniel Pojhan
Plzeňský kraj

Plzeň, 2025

STŘEDOŠKOLSKÁ ODBORNÁ ČINNOST

Obor č. 18: Informatika

SEDAS: ATC simulátor s implementací AI pseudopilotů

SEDAS: ATC simulator with the use of AI pseudopilots

Autor: Daniel Pojhan
Škola: Gymnázium, Plzeň, Mikulášské nám. 23,
Mikulášské náměstí 808/23, 326 00 Plzeň
Kraj: Plzeňský kraj
Konzultant: doc. Ing. Bc. Vladimír Socha, Ph.D.

Plzeň, 2025

Prohlášení

Prohlašuji, že jsem svou práci SOČ vypracoval samostatně a použil jsem pouze prameny a literaturu uvedené v seznamu bibliografických záznamů.

Prohlašuji, že tištěná verze a elektronická verze soutěžní práce SOČ jsou shodné.

Nemám závažný důvod proti zpřístupňování této práce v souladu se zákonem č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších předpisů.

V Plzni dne: _____ _____

Daniel Pojhan

Poděkování

Rád bych poděkoval panu doc. Ing. Bc. Vladimíru Sochovi, Ph.D. z Fakulty dopravní ČVUT za odborné konzultace týkající se teorie letectví a řízení letového provozu. Poděkování patří také Ing. Tomáši Malichovi za jeho cenné rady, které pomohly k formulaci postupu práce.

Anotace

Tato práce se zabývá implementací AI pseudopilotů do trénování řídících letového provozu. Systém implementuje 3 zásadní úlohy: rozpoznávání řeči, zpracovávání přirozeného jazyka a syntézu řeči. Práce se tak snaží optimalizovat využití lidských zdrojů v rolích pseudopilotů, kteří jsou k chodu ATC simulace nezbytní, pomocí metod strojového učení. Výstupem je pak volně konfigurovatelný a dostupný ATC simulátor ve formě desktopové aplikace.

Klíčová slova

řízení letového provozu; desktopová aplikace; automatické rozpoznávání hlasu; syntéza řeči; zpracovávání přirozeného jazyka; strojové učení; pseudopilot

Annotation

This work aims at the implementation of AI pseudopilots in the training of Air traffic control officers. The system implements 3 key tasks: automatic speech recognition, natural language processing and speech synthesis. The system is thus trying to optimize the usage of human resources in the roles of pseudpilots, who are essential for a flawless ATC simulation operation, with the use of machine learning methods. The output of this work is therefore a freely configurable and available ATC simulator in the form of an desktop application.

Keywords

air traffic control; desktop application; automatic speech recognition; speech synthesis; natural language processing; machine learning; pseudopilot

Obsah

Seznam zkratek

Zkratky v letectví

	Air Traffic Control
ATC	Služba poskytovaná pilotům pohybující se v řízeném vzdušném prostoru z důvodu optimalizace letecké dopravy
ATCo	**Air Traffic Controller** Člověk zajišťující službu ATC ve vzdušném prostoru
	Controller-Pilot Data Link Communications
CPDLC	Metoda komunikace mezi ATCo a pilotem pomocí textových zpráv přes datalink spojení
	International Civil Aviation Organisation
ICAO	Mezinárodní organizace pro civilní letectví přidružená k OSN, standardizuje leteckou legislativu
	European Union Aviation Safety Agency
EASA	Agentura Evropské unie, reguluje a dohlíží na civilní leteckou dopravu v Evropě
	International Virtual Aviation Organisation
IVAO	Nezisková organizace provozující bezplatnou online síť letecké komunikace pro nadšence ATC a letectví
	Virtual Air Traffic Simulation Network
VATSIM	Nezisková organizace podobná organizaci IVAO. Oproti IVAO má rozšířenější komunitu
	Federal Aviation Administration
FAA	Agentura ministerstava dopravy USA, reguluje a dohlíží na civilní leteckou dopravu nad Americkým leteckým prostorem
	Air traffic service
ATS	služby, které jsou letům v rámci vzdušného prostoru poskytovány (ATC, FIS, ALRS)
	Control tower controller
TWR	Služba ATC zajišťující bezpečný provoz na přistávací dráze, pojezdových drahách a v bezprostředním okolí letiště
APP	**Approach controller** Služba ATC zajištující bezpečný provoz v širším okolí letiště
	Area controller
ACC	Služba ATC zajišťující bezpečný provoz v příslušné řízené oblasti (v ČR se např.: jedná o celé území státu)
	Transition level
TL	nejnižší možná výška, ve které je letadlo povinno přepnout z jednotky stop na jednotku flight level (FL)

ROT	**Rate of turn** Doba, za kterou letadlo změní svůj kurz o nějaký úhel. Udáváno ve stupních za sekundu
KIAS	**knots-Indicated air speed** rychlost letadla která je indikována pilotovi na rychloměru (jednotka v uzlech)
MLW	**Maximum landing weight** Maximální hmotnost letadla, se kterou je možno bezpečně přistát
OEW	**Operating empty weight** Minimální hmotnost letadla, ve kterém se nachází jen posádka bez pasažérů
MTOW	**Maximum takeoff weight** Maximální hmotnost letadla, se kterou je možno bezpečně vzlétnout
ROC	**Rate of climb** Značí rychlost stoupání letadla, obecně je tato hodnota stanovena na stopy za minutu (Boeing 737-900 má ROC 2000ft/min)
ROD	**Rate of descent** Značí rychlost klesání letadla, také ve stopách za minutu (Boeing 737-900 má ROD 3500ft/minutu)
PSR	**Primary surveillance radar** Radarový systém určený k lokalizaci letadla pomocí emitování signálu, který se po dosažení odrazí zpátky k radaru, rotující anténou
SSR	**Secondary surveillance radar** Radarový systém na bázi PSR, kromě lokalizace zde ale probíhá i přenos dat mezi letadlem a radarem
ADS-B	**Automatic Dependent Surveillance - Broadcast** Nástupce SSR a PSR, letadlo díky transpondéru samo posílá informace o poloze, volací znak, atd.

Technické zkratky

VoIP	**Voice over IP** Protokol umožňující přenos hlasu ve vormě paketů protokolu TCP/UDP
GUI	**Graphical user interface** Uživatelské rozhoraní ovládané pomocí grafických prvků
ASR	**Automatic speech recognition** Obor strojového učení, který ze zabývá rozpoznáváním mluvené řeči
TTS	**Text to speech** Obor strojového učení, kteý řeší problematiku umělé tvorby lidské řeči
NLP	**Natural language processing** Soubor technik, které se analyzují a zpracovávají přirozený jazyk ve formě textu

PTT	**Push to talk** Způsob komunikace kde v případě, že chce ATCo mluvit, musí stisknout tlačítko po dobu mluvení. Tento typ komunikace je využíván v ATC
CUDA	**Compute Unified Device Architecture** Architektura vyvíjena společností NVIDIA pro akceleraci programů na GPU
MVP	**Minimum viable product** Produktová strategie která je založena na myšlence, kde se produkt nachází ve stavu prototypu, ale pokrývá většinu plánovaných funkcionalit, které pak vylepšuje
FOSS	**Free and open-source software** Označení pro software, který je otevřený a svobodný. Uživateli je tak umožněn přístup ke zdrojovému kódu.
IPC	**Inter process communication** Komunikační protokol, který umožňuje vyměňovat data mezi dvěma nezávisle na sobě běžícími procesy. Toho se dá docílit buď pomocí socketů, pipes nebo sdílené paměti
WAV	**Waveform audio file format** Zvukový formát vytvořený IBM a Microsoft, určený k ukládání zvuku na PC

1 ÚVOD

Řízení letového provozu (ATC) je důležitý prvek bezpečnosti a efektivity letecké dopravy, který umožňuje řídit a monitorovat leteckou dopravu ve vzdušném prostoru i na letištích. Předchází tak srážkám a optimalizuje tok letového provozu, což je z hlediska dnešní hustoty provozu v leteckém prostoru zásadní. Tuhle činnost zajišťují řídící letového provozu (ATCo), kteří většinou hlasově komunikují s piloty prostřednictvím radiového spojení.

Aby se ATCos mohli na svou práci důkladně připravit, musí projít náročným výcvikem ve specializovaných střediscích ATC. Výcvik probíhá na moderních simulátorech, které realisticky napodobují řízený vzdušný prostor nebo letištní prostředí. Vzhledem k tomu, že ATC je interaktivní činnost vykonávaná převážně hlasovou interakcí, je přítomnost tzv. pseudopilotů nezbytná. Ti v simulacích zastávají roli pilotů a odpovídají na hlasové pokyny řídících podobně jako v reálném provozu. Tím se ale bohužel zvýší počet personálu, který je nutný při obsluhování ATC simulátoru a samotný správný chod simulace je tak závislý na dostupnosti pseudopilotů. Řešením by ale mohlo být nahrazení pseudopilotů AI modulem, který by jejich činnost mohl zastávat.

Hlavní inspirací pro tento projekt byla stáž na Fakultě dopravní ČVUT, kde byla možnost se s takovým systémem seznámit a studovat metody ATC. V průběhu stáže mě napadla myšlenka, jakým by se proces vyučování ATC dal zefektivnit právě pomocí metod strojového učení. Následně jsem začal vyvíjet projekt vlastního ATC simulátoru jménem SEDAS (Scalable and Easily Deployable ATC Simulator). Projekt je zatím koncipován jako „Proof of concept“, jeho hlavním cílem je otestovat možnost nahrazení lidských pseudopilotů metodami strojového učení a používat tak jen tzv. AI pseudopiloty. Zároveň implementuje řadu dalších funkcionalit, které do boudoucna zjednoduší uživateli práci se softwarem. Vzhledem ke komplexitě projektu (zejména pak všech výzev implementace vlastního ATC simulátoru a legislativy s tím spojené) byla na vedlejší funkcionality aplikace využita strategie MVP (Minimum viable product). Některé funkce tak budou v budoucnu dopsány a optimalizovány.

2 TEORETICKÝ RÁMEC

V některých oblastech, zejména pak na dálkových tratích, se využívá jenom digitální datová komunikace (CPDLC), která přenáší instrukce ve formě textových zpráv. Ve většině situací se však řídící setkají s hlasovou komunikací, na kterou je nezbytné se připravit v rámci výcvikových simulací. V práci jsem se zaměřoval zejména na hlasovou komunikaci, která je mnohem náročnější na automatizaci než komunikace textová.

V ATC se jako primární jazyk využívá angličtina. Letecká komunikace má frazeologii standardizovanou Mezinárodní organizací pro civilní letectví (ICAO), ve které se používá mezinárodní hláskovací abeceda [1]. Vzdušný prostor nad stanoveným územním celkem se dělí do několik podprostorů z nichž každý má jiné pravidla a každý poskytuje jiné letové provozní služby (ATS) [2]. V rámci zachování jednoduchosti výstupního simulátoru bylo zaměření práce mířeno jen na jednu službu ATS, a tou je právě služba ATC. Samotná služba ATC se dělí ještě na letištní službu řízení (TWR), přibližovací službu řízení (APP) a oblastní službu řízení (ACC). Další zaměření simulátoru bylo na oblasti ACC, jelikož APP a TWR by přidávalo další část frazeologie, komplexitu a nové scénáře, které by se musely do simulátoru integrovat.

Hlavní problematikou bylo zvoleno vektorování letadel, které má ve své podstatě pár příkazů, které pouze mění směr a letovou hladinu letadla. Sama ATC úloha tedy spočívá v přiřazování vektorů (headingů) letadlům tak, aby se držely dané trasy a aby v prostoru nevznikaly kolize mezi letadly. Tato úloha je jakýmsi vstupním testem ATCos, protože prověří jejich znalost prostorového vidění a plánování tratí pro co nejefektivnější provoz, což je základ u ATC.

Pseudopiloti jsou v simulaci klíčovým prvkem. Dodávají totiž ATCos hlasový feedback jejich příkazů, od kterých se odvíjí celý chod simulace. Aby se však pseudopiloti a ATCos neslyšeli i bez sluchátek, jsou většinou oddělení v místnostech a komunikaci zprostředkovává ATC simulátor samotný pomocí VoIP (Voice over Internnet Protocol) protokolu. Samotnou interakci pseudopilotů s letadlem (změny výšky, rychlosti a směru) zajišťuje separátní software s GUI přímo pro pseudopiloty, který je odlišný od GUI ATCos.

Pro zajištění efektivního výcviku řídících letového provozu existuje řada specializovaných ATC simulátorů, které umožňují realistickou simulaci hlasové komunikace a interakce mezi ATCos a pseudopiloty. Tyto simulátory se liší svou úrovní sofistikovanosti, způsobem implementace komunikačních systémů a rozsahem podporovaných funkcionalit. Správné porozumění frazeologii, přesná interpretace pokynů a schopnost rychlé reakce na provozní situace jsou klíčové dovednosti řídících letového provozu, simulátory tudíž hrají zásadní roli v jejich výcviku.

Jedním z nejznámějších simulátorů v evropském kontextu je simulátor ESCAPE od společnosti Eurocontrol [3]. Jeho verze, ESCAPE-light [4], je distribuována do různých univerzit a výzkumných pracovišť, kterým je například katedra letecké dopravy na Fakultě dopravní ČVUT (FD ČVUT, viz obrázek 1).

Tento simulační software je zde využíván pro výuku řídích letového provozu. Na simulátoru ESCAPE-light, který se na FD ČVUT nachází, jsou ATCos a pseudopiloti rozděleni do dvou místností. V jedné místnosti se nacházejí ATCos, kteří interagují se simulátorem, a v druhé jsou pseudopiloti, kteří pomocí GUI mění parametry řízených letadel a hlasově komunikují s ATCos. Celý simulátor je konstruován z několika počítačů, každý pro jednoho ATCo a pseudopilota, na kterých běží Windows 10 a v nich virtualizované CentOS prostředí ve kterém je ESCAPE simulátor spuštěn. Komunikace mezi piloty je zprostředkovávána lokálním ČVUT FD serverem platformy TeamSpeak.

Mezi další ATC simulátory, které jsou spíše mířené pro nadšence ATC, je simulátor Aurora od IVAO [5]. IVAO (International Virtual Aviation Organization) je nezisková organizace, která sdružuje nadšence do řízení letového provozu a pilotování letadel. Po registraci se lze připojit na tzv. IVAO Network buď jako ATCo (pomocí Aurory), nebo jako virtuální pilot a v celosvětové ATC simulaci tak interagovat pomocí VoIP protokolu v rámci jednotlivých letišť a řízených okrsků. Simulace se snaží být co nejreálnější a používá standardní procedury ATC a frazeologii. Mezi alternativy IVAO pak existuje i VATSIM (Virtual Air Traffic Simulation network) [6]. Oba dva tyto simulátory jsou tedy založeny na komunikaci s ostatními uživateli na globálním serveru. Od ESCAPE a SEDAS se liší tím, že jejich software není určen pro lokální použití.

Související práce, od kterých se tento projekt odvíjel a stavěl na nich, byly většinou vědecké články a bakalářské práce. Jako například v práci [7], kde byl navrhnut pipeline AI modulu, který v sobě odbsahoval model ASR (automatic speech recognition), NLP (natural language processing) a následně výstupní TTS (text to speech) model, který dával syntetizovaným hlasem zpátky feedback. Architekturou jejich řešení se tento projekt částečně inspiroval, použily se

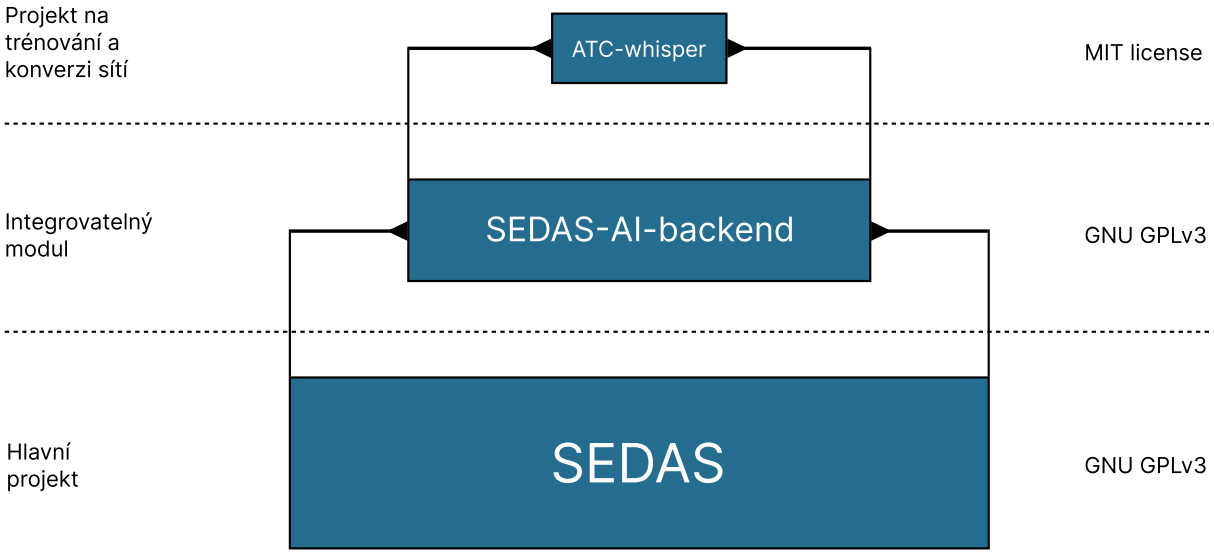
Obr. 1: ATC simulátor na ČVUT FD zachycený během simulace stanoviště APP pro letiště Václava Havla v Praze, kde si studenti procvičují standardizovanou frazeologii, koordinaci příletů a odletů a interakci mezi ATCos a pseudopiloty v realistickém prostředí.

však mnohem novější modely, zejména pak v ASR části, kde byl namísto skrytého markovského modelu použit state-of-the-art model Whisper [8] od společnosti OpenAI, který byl dotrénovaný na ATC komunikaci v bakalářské práci [9]. Část TTS pak využívala model z výzkumu modelu VITS [10], více však v kap. 4.

Problém je však ten, že většina těchto prací je stavěna na simulátoru ESCAPE, který ale není pro všechny dostupný, protože je využíván zejména k výzkumu a dodáván je jen do výzkumných institucí a univerzit. Pro ATC nadšence tedy špatně dosažitelná varianta. SEDAS je ale Open-source a umožňuje svým uživatelům spouštět lokální simulace ATC v prostředí, kde jsou pseu-dopiloti nahrazeni AI pseudopiloty. Tím je možné se vzdělávat v ATC bez potřeby více lidí a specializovaného hardwaru.

3 METODICKÝ RÁMEC PROJEKTU

Projekt je strukturován do 3 částí (viz obrázek 2), a to: software na konverzi sítí do formátu, který projekt podporuje (ATC-whisper), integrovatelný AI modul (SEDAS-AI-backend) a hlavní projekt SEDAS. ATC-whisper slouží pro konverzi dotrénovaného ASR modelu Whisper do formátu GGML [11]. SEDAS-AI-backend je pak modul, který je zaintegrován do hlavního projektu, avšak pro testovací účely dokáže fungovat jako samostatná jednotka. Přímo volá Whisper model a zároveň také zapojuje model pro zpracovávání příkazů (NLP) a také systém Piper [12] pro syntézu řeči (TTS), který je nadstavbou výše zmíněného modelu VITS.



Obr. 2: Schéma struktury projektu SEDAS (Scalable and Easily Deployable ATC Simulator)

Dokumentace projektů se na jednotlivých Github repozitářích (SEDAS, SEDAS-AI-backend, ATC-whisper, přílohy 1 2 3) nachází ve formě README markdown souborů. K lepší dokumentaci celé aplikace a jejího používání uživatelem je ještě napsaná dokumentace s využitím platformy Read the Docs. Ta je napsaná v Češtině a Angličtině a má vlastní url, pod kterou se dá k dokumentaci dostat (příloha 5).

Dokumentace projektů na Githubu je spíše určena pro replikaci postupů pro spuštění kódu v developement módu, aby se na vývoji v budoucnu mohli podílet i další uživatelé. Všechny projekty se tak na Githubu nachází pod otevřenou licencí GNU GPLv3 nebo MIT license. Vzhledem k narustajícímu počtu repozitářů, z nichž každý má jinou funkci, se na Githubu vytvořila organizace SEDAS-DevTeam, která vlastní všechny výše zmíněné repozitáře (příloha 4).

3.1 Použité technologie

Výběr technologií byl koncipován tak, aby bylo v nich jednoduché psát kód v rámci „Proof of Concept“ návrhu aplikace. V budoucnu se i po soutěži SOČ plánuje část prvků přepsat do jiných

jazyků/technologií v rámci zvýšení efektivity softwaru. Hlavní aplikační rámec byl napsán v Typescriptu [13] v běhovém prostředí NodeJS [14] s využitím frameworku Electron [15] jako hlavního GUI. Zejména byly vybírány technologie které jsou open-source, projekt samotný je totiž veřejně dostupný a tak je vhodné se FOSS iniciativy držet.

3.1.1 Electron

Electron je framework pro psaní aplikací v Javascriptu s využitím webových toolkitů (VueJS, React, atd.). Byl zvolen pro svou jednoduchost, multiplatformní podporu a velkou komunitní základnu. Spoustu společností používá Electron pro svoje aplikace, mezi nejznámější patří Slack, Skype, Visual Studio Code a Microsoft Teams. Sám Electron je vydávaný pod MIT licencí, spadá tedy do kategorie open-source projektů, což ještě víc podpořilo důvod ho použít. Dalším faktem je, že sám simulátor ESCAPE částečně využívá webových technologií pro tvorbu GUI pro ATCos.

V rámci vytváření GUI se ale žádné specifické webové toolkity nepoužívaly. Bylo by zbytečné implementovat komplexní nadstavbu, když se prozatím jedná o prototyp desktopové aplikace. Většina GUI kódu je proto psaná v HTML/CSS/JS. V rámci budoucího vývoje ale bude zapotřebí, kvůli neustálému zvyšování komplexity kódu, použít nějakou webovou nadstavbu, která usnadní vývoj grafických prvků.

3.1.2 Typescript

Pro vývoj aplikačního backendu byla použita nadstavba Javascriptu jménem Typescript. V rámci bezpečnosti tak bylo zajištěno statické typování, aby nedocházelo k chybám v rámci výpočtů, či popřípadě jiným nevyžádaným chybám, které by mohly nastat dynamickým typováním. Typescript nabízí mnohem lepší strukturování kódu než Javascript a také detekuje chyby během kompilace programu, což zjednodušuje vývoj.

3.1.3 C++, CMake

V rámci zefektivnění chodu aplikace a jeho optimalizace bylo využito knihovny node-addon-api. Ta umožňuje kompilovat C++ kód tak, aby mohl být přímo volán v běhovém prostředí NodeJS. Výhody tohoto postupu je rychlejší kód než v nativním Javascriptu. Mezi další výhody je pak větší bezpečnost a optimalizace. Volané metody v aplikačním backendu jsou rozděleny podle výpočetní náročnosti a ty, které představovaly pro systém jistý bottleneck, byly přepsány do C++. Výsledkem je, že výpočty prostředí a letadel (otáčení, zvyšování/snižování letové hladiny, zrychlování/zpomalování) jsou napsané jako addony, zatímco ostatní backendové funkce, které nejsou tolik náročné, jsou ponechány v Typescriptu.

V kap. 3, jde vidět schéma zapojení jednotlivých projektů do sebe (kód je členěn do několika podpojektů pro zachování srozumitelnosti kódu). SEDAS-AI-Backend je celý napsaný v C++,

avšak bez použití node-addon-api. Je to díky tomu, že tento AI backend běží jako samotný proces, aby nebyl blokován jinými funkcemi backendu, které by mohly zvýšit reakční dobu AI pseudopilotů. Pro zprostředkování komunikace mezi aplikací (SEDAS) a backendem pro AI (SEDAS-AI-backend) byl vytvořen wrapper který zajišťuje socket komunikaci na lokálním hostovi. Pro SEDAS-AI-backend byl použit konfigurační systém CMake pro automatizaci překladu kódu.

3.1.4 GGML, Whisper.cpp

V rámci AI backendu projektu se používal ASR model Whisper [8]. Pro jeho implementaci a akceleraci na GPU/CPU se využívá knihovny Whisper.cpp [16]. Ta je pak postavena na knihovně GGML [11] vydávané pod licencí MIT. Tato knihovna strojového učení umožňuje low-level implementace inference neuronových sítí. Knihovna má take širokou hardwarovou podporu a umožňuje akcelerovat modely pomocí architektury CUDA (Compute unified device architecture), což znatelně zrychluje inferenci modelu a pro nás tak snižuje reakční dobu AI pseudopilota. Samotná knihovna Whisper.cpp má také rozsáhlou cross-platform podporu (Vulkan, CoreML, OpenBLAS). Pro naše účely však stačí akcelerace pomocí CUDA, protože software byl prozatím testován na počítačích s Nvidia GPU.

3.1.5 Hugging Face, PyTorch

V rámci využití Whisper modelu pro transkripci hlasového výstupu ATCo bylo stavěno na předchozím výzkumu od FIT VUT, kde byl model dotrénovaný na ATC komunikaci [9]. Váhy tohoto modelu se nachází na portálu Hugging Face. Hugging Face slouží jako databáze různých modelů a datasetů řešící různé problematiky, včetně problematiky ASR. Důvod využití dotrénovaného modelu bylo, že ATC komunikace je lehce rozdílná od standardně strukturované řeči. Model je tedy ideální na použití pro naše účely. Vzhledem k tomu, že na Hugging Face jsou ale modely převážně nahrávány ve formátu safetensor, je nutná konverze do GGML formátu. Pro naše účely je totiž formát nepoužitelný, protože ho GGML nepodporuje. To bylo zajištěno skriptem pro konverzi modelu, kde bylo využito knihovny transformers (vyvíjené Hugging Face) a Pytorch (Python knihovna pro strojové učení, vyvíjena společností Meta).

3.1.6 Git

Pro správu verzování projektu a vytváření backupů se používal software Git společně s platformou Github, kde je projekt volně přístupný pod licencemi MIT (ATC-Whisper) a GNU GPLv3 (SEDAS-AI-backend, SEDAS). Pro integraci projektů do finální aplikace SEDAS byly použity Git submodules, které umožňují v repozitáři odkazovat na jiné, externí repozitáře. Ty jsou pak vyklonovány jedním příkazem a pak už se s nimi pracuje obdobně jako se standardními Git repozitáři. Tato integrace repozitářů do sebe umožňovala jednodušší integraci celého projektu,

aniž by bylo zapotřebí psát nějaké další pomocné funkce. Této vlastnosti Git submodules bylo zejména využito při integraci AI backendu do hlavního softwaru.

3.2 Funkční požadavky a návrh simulátoru

ATC simulátory fungují jako simulované verze radarů (PSR - Primary surveillance radar, SSR - secondary surveillance radar) či ADS-B (Automatic Dependent Surveillance-Broadcast). Simulátor musí obsahovat mapu/prostředí, ve kterém se letadla budou pohybovat a také musí mít stanovenou obnovovací frekvenci. Ta je do simulátorů implementována kvůli tomu, že k zpracování signálu a aktualizaci mapy musí radar provést jednu celou rotaci. Rychlost rotace ale není v letectví pevně stanovena. Někde se tak můžeme setkat s dobou rotace 10s, někde pak 5-6s [17]. Simulátor má zatím obnovovací frekvenci stanovenou na 1Hz (doba rotace 1s), aby nebyla simulace zbytečně náročná při prvotním testování. V budoucnu ale bude obnovovací frekvence volně nastavitelná.

Simulátor také musí mít zabudovaný systém, který bude definovat tzv. transition level (TL) [18]. Jedná se o správním orgánem ATC stanovenou vertikální vrstvu vzdušného prostoru, kde se přepíná z jednotky stop na jednotku flight level (FL) a naopak. Výška letadla se měří pomocí výškoměru, který měří atmosferický tlak. V různých regionech se však atmosferický tlak může měnit a piloti by tak museli konstantně upravovat nastavení výškoměru. Proto bylo od určité výšky zaveden systém TL, neboli nejnižší možná výška, ve které už se musí používat univerzální jednotka flight level. Každý stát si stanovuje svůj TL sám, například v Kanadě a USA je TL stanoven v 18 000ft. V rámci zachování jednoduchosti implementace simulátoru je zbytečné se snažit simulovat prostředí s atmosferickým tlakem. Každopádně je důležité z hlediska zachování realističnosti simulace, aby převod byl od jisté výšky aplikován. Nejnižší TL v ČR je na 1000ft [19], ten je také v simulátoru použit. V simulátoru je ale možné TL měnit.

Komunikace mezi ATCos a pseudopiloty se ve všech ATC simulátorech zprostředkovává pomocí VoIP protokolu. Každý ATCo a pseudopilot mají vlastního klienta, se kterým se připojí na jeden společný server (který v tomto případě imituje jednu frekvenci, na kterou by všichni piloti byli naladěni). V SEDASu se ale žádný VoIP klient a server nemusí existovat, protože komunikace od ATCo je nahrávána pomocí volání PTT (Push-to-talk) signálu, který začne registrovat zvuk z mikrofonu a následně ho pošle do ASR na zpracování. Samotný hlasový výstup pseudopilota se pak jen spustí přímo ATCo ve zvukovém výstupu.

Mezi další důležité věci, které je potřeba udělat ke správně napsanému ATC simulátoru je také systém volacích znaků. Mezinárodní standard [20] se řídí několika způsoby vytváření volacích znaků pro letadla, které jsou uvedeny v tabulce. 1. Oproti tabulce, která je definována v ICAO Annex 10 vol. II [1] byly do této tabulky přidány také požadavky Federální letecké správy (FAA) [21], které jsou od ICAO normy lehce rozdílné. Prozatím práce funguje na úplných volacích znacích Typu A. Do budoucna by se však toto mělo změnit a simulátor by měl podporovat všechny volací znaky ICAO i FAA.

Tab. 1: Možnosti vytváření volacího znaku letadla.

	Typ A		Typ B	Typ C
Úplný volací znak	ABCDEFG ABC1234 ABCD123	Airbus ABCDEFG	Rushair BCDE	Rushair 1234
Zkrácený volací znak	ADE ACDE	Airbus DE Airbus ABDE	Rushair DE Rushair BDE	*Žádná forma zkrácení*

3.3 Výpočty pohybu letadel

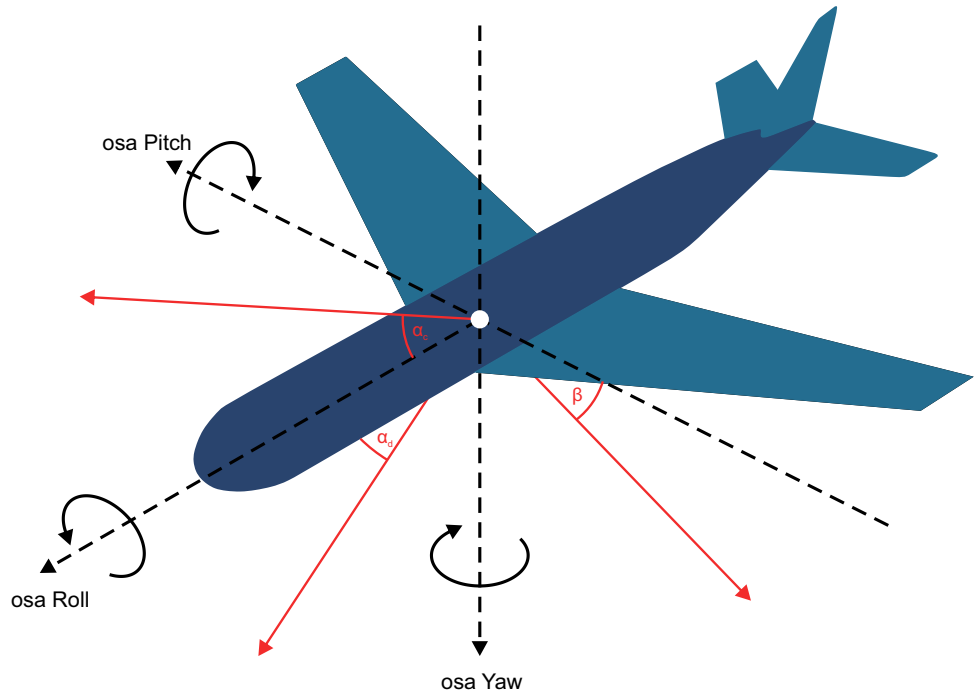
Pro správné fungování ATC simulačního softwaru je zásadní mít korektně definovaný matematický popis pohybu letadel. Tento model byl v rámci implementace záměrně zjednodušen, neboť cílem simulace je především věrohodná imitace letového chování, nikoli fyzikální přesnost do nejmenších detailů. Použité výpočty vycházejí převážně z publikace *Pilot's Handbook of Aeronautical Knowledge* [22], vydané FAA, a byly doplněny o relevantní poznatky z článků dostupných na platformě Skybrary (wiki zaměřená na ATC teorii, spravovaná organizacemi EUROCONTROL a ICAO).

Vzhledem k tomu, že letectví operuje primárně s imperiálními jednotkami, byly veškeré výpočty prováděny v tomto jednotkovém systému. Za účelem zachování jednoduchosti implementace se v aktuální fázi předpokládají pouze lineární změny rychlosti, výšky a směrového vektoru letu. Zároveň nebyly zohledněny individuální aerodynamické a výkonnostní charakteristiky různých typů letadel. Simulátor tedy prozatím pracuje s jedním typem „průměrného" dopravního letadla, pro které jsou všechny výpočty modelovány s využitím konstantních parametrů.

Na obrázku 3 je znázorněno prostorové rozložení úhlů, které definují orientaci letadla. Úhly α_c (climb - stoupání) a α_d (descent - klesání), představující hodnoty úhlu pitch, popisují podélný náklon letadla a slouží jako základ pro výpočet změny letové hladiny (viz kapitola 3.3.3). Úhel β (bank angle), určující příčný náklon (roll), je následně využit pro výpočet míry zatáčení – tzv. Rate of Turn (ROT) (viz kapitola 3.3.1).

Výpočty veličin používaných v simulaci pro model standardního chování letadla vycházejí z technických údajů dopravního letounu Boeing 737-800, který je nejpopulárnější variantou řady Boeing 737. Klíčové parametry byly čerpány z oficiálních charakteristik poskytnutých výrobcem [23], kde jsou mimo jiné uvedeny maximální přistávací hmotnost (MLW) a provozní prázdná hmotnost (OEW). Vzhledem k tomu, že simulace uvažuje průměrné provozní chování, byla výsledná hmotnost letadla m získána zprůměrováním těchto dvou hodnot.

Další potřebné veličiny, jako je například dostupný tah motoru nebo charakteristiky stoupání a klesání, byly převzaty z technických článků publikovaných na stránkách Skybrary [24]. Na základě těchto dat lze určit hodnotu tahové síly F, přičemž výsledné zrychlení letadla a se poté



Obr. 3: Osy letadla určující jeho klonění (Roll), klopení (Pitch) a zatáčení (Yaw) s indikací úhlů klonění (β), klesání (α_d) a stoupání (α_c).

vypočítá dle vztahu

$$a = \frac{F}{m}, \tag{1}$$

přičemž výsledek je nutné převést do jednotek ft/s^2 kvůli konzistenci s ostatními výpočty v imperiálním systému.

Hodnota β, která představuje úhel náklonu letadla během zatáčky, byla stanovena na základě údajů z dokumentu ICAO Doc 8168 [18]. Tento dokument specifikuje standardní podmínky pro tzv. *airplane holding*, tedy manévr, kterým řídící letového provozu dočasně oddaluje přiblížení letadla k letišti, například kvůli přetížení vzdušného prostoru nebo probíhajícím nouzovým postupům. Během holdingu se letadlo pohybuje po předem definovaném obrazci s konstantní rychlostí a úhlem náklonu. Právě z tohoto důvodu byla hodnota β zvolena jako standardizovaný úhel 25°, běžně používaný letadly během tohoto manévru.

Úhly stoupání α_c a klesání α_d byly odvozeny na základě údajů o indikované vzdušné rychlosti (KIAS) a míře stoupání (ROC, Rate of Climb) či klesání (ROD, Rate of Descent). Konstanta f reprezentuje obnovovací frekvenci simulace. Přehled všech konstant použitých při výpočtech je uveden v tabulce 2.

Tab. 2: Atributy standardně se chovajícího dopravního letadla, na které se vztahují všechny výpočty

Atribut	β	m	F	a	α_c	α_d	f
Hodnota	25°	54 365 kg	117 kN	4.18 ft/s^2	4°	7°	1Hz

3.3.1 Otáčení

Pro výpočet změny kurzu letadla na libovolnou cílovou hodnotu je nutné znát jeho rychlost otáčení, označovanou jako Rate of Turn (ROT). Tato veličina udává, o kolik stupňů se změní kurz za jednotku času, v tomto případě za jednu sekundu. ROT se určuje pomocí vztahu (2):

$$ROT = \frac{1091 \cdot \tan(\beta)}{v} \tag{2}$$

kde *ROT* představuje rychlost otáčení ve stupních za sekundu, β je úhel náklonu letadla (tzv. bank angle, neboli roll) a v označuje aktuální rychlost letu v uzlech. Konstanta 1091 je výsledkem převodu jednotek mezi uzly a stopami za sekundu v kontextu výpočtu úhlové rychlosti.

Jakmile je ROT spočítán a simulátor obdrží instrukci ke změně kurzu, stačí podle obnovovací frekvence simulace průběžně přičítat nebo odečítat hodnotu ROT, dokud aktuální kurz nedosáhne cílové hodnoty.

Vzhledem k tomu, že letadlo se může otáčet buď doprava nebo doleva, je třeba tuhle volbu směru otáčení přesně řídit. V rámci ATC frazeologie existují specifické pokyny jako *"turn right"* (otáčení výhradně vpravo), *"turn left"* (otáčení výhradně vlevo), nebo *"fly heading"* (pilot zvolí optimální směr otáčení). Výpočet tedy musí zohlednit požadovaný směr rotace dle zadání řídícího letového provozu (ATCo). Kompletní algoritmus řízení otáčení letadla je uveden v ukázce algoritmu 1.

Po výpočtu nového kurzu v jednotlivých krocích simulace je zároveň nutné stanovit podmínku, která určuje, kdy bylo požadované otáčení dokončeno a letadlo tak může přejít na ustálený let po cílovém kurzu. Tato logika je implementována prostřednictvím výpočtu proměnných h_n (nový aktuální kurz) a c_h (změna kurzu na jeden výpočetní cyklus). Porovnání těchto hodnot s cílovým kurzem h_d je vyjádřeno pomocí rovnice (3), která vrací booleovskou hodnotu indikující, jestli má letadlo i nadále pokračovat v zatáčce.

$$c = \begin{cases} \textbf{false} & \text{pokud } |h_d - h_n| < c_h \\ \textbf{true} & \text{jinak} \end{cases} \tag{3}$$

Tato logika je pak aplikována i při výpočtech změn výšky a rychlosti, kde dochází k analogickému problému konvergence k cílové hodnotě. V těchto případech jsou proměnné h_d a h_n nahrazeny aktuální a cílovou výškou, a c_h je nahrazeno hodnotou c_l, která představuje krok změny hladiny (tzv. level change) na jeden výpočetní cyklus.

Algoritmus 1: Algoritmus pro výpočet otáčení letadla

Vstupní hodnoty:

h_d - nový směr, do kterého se letadlo musí dostat

h - směr, ve kterém se letadlo právě nachází

ROT - Rate-of-turn (rychlost otáčení ve stupních za sekundu)

f - refresh rate simulátoru

Výstupní hodnoty:

c_h - změna směru, který letadlo vykoná

h_n - nový směr letadla po části rotace

d_{cw} = mod($h_d - h + 360$, 360) *// Vypočítá clockwise rotaci pomocí modulo operátoru*

d_{ccw} = mod($h - h_d + 360$, 360) *// Vypočítá counterclockwise rotaci*

turn_right = false

if command = `"turn-any"` **then**

    turn_right = ($d_{cw} \leq d_{ccw}$)

else if command = `"turn-right"` **then**

    turn_right = true

else if command = `"turn-left"` **then**

    turn_right = false

end if

$c_h = ROT \cdot (1/f)$

if turn_right **then**

    h_n = mod($h + c_h$, 360) *// V případě že h_n bude > 360*

else

    h_n = mod($h - c_h$, 360)

end if

3.3.2 Změna rychlosti

Aby simulace zahrnovala jak zrychlení, tak zpomalení, je zaveden výpočet směrového koeficientu k, který na základě rozdílu mezi cílovou a aktuální rychlostí určuje směr změny:

$$k = \frac{v_f - v}{|v_f - v|}, \quad k \in \{-1, 1\} \tag{4}$$

Vzhledem k tomu, že simulace uvažuje pouze lineární změny pohybu, je aktualizace rychlosti v každém kroku simulace triviální — aktuální rychlost se upravuje o velikost zrychlení vynásobenou obnovovací frekvencí a směrovým koeficientem:

$$v_n = v + k \cdot \frac{1}{f} \cdot a \tag{5}$$

Tento výpočet probíhá iterativně až do momentu, kdy rozdíl mezi aktuální a cílovou rychlostí klesne pod prahovou hodnotu (viz logika rozhodování rovnice (3)).

3.3.3 Změna výšky

Výpočet vertikální změny výšky probíhá analogicky ke změně rychlosti. I v tomto případě je nejprve stanoven směrový koeficient k na základě rozdílu mezi aktuální výškou l a cílovou výškou l_f, který udává směr vertikálního pohybu (stoupání nebo klesání).

Letadlo však během stoupání a klesání využívá rozdílné úhly náklonu trupu, které jsou definovány jako α_c (úhel stoupání) a α_d (úhel klesání) – viz tabulka 2. Proto je třeba podle hodnoty k zvolit odpovídající úhel α_{sel}, který bude použit pro výpočet vertikální složky pohybu. Tento výběr je definován následovně:

$$p_{sel} = \begin{cases} \alpha_c & \text{pokud } k = 1 \\ \alpha_d & \text{pokud } k = -1 \end{cases} \tag{6}$$

Výška v následujícím kroku simulace se následně vypočítá pomocí vztahu:

$$l_n = l + k \cdot \frac{1}{f} \cdot v \cdot \sin(p_{sel}) \tag{7}$$

kde l značí aktuální výšku, v indikovaná vzdušná rychlost, f je obnovovací frekvence simulace a α_{sel} zvolený úhel dle směru pohybu. Výpočet probíhá iterativně až do okamžiku, kdy aktuální výška dosáhne cílové hodnoty, přičemž rozhodovací logika je opět založena na rovnosti (3).

3.4 Backend aplikace

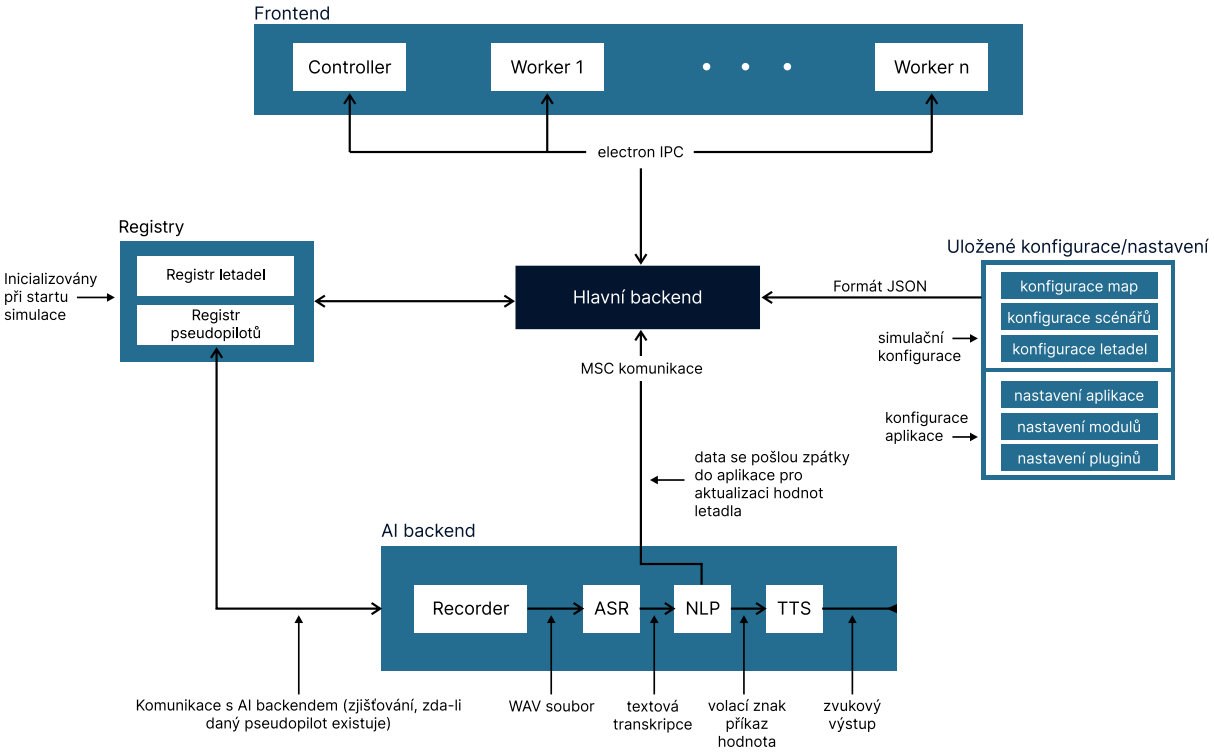
Aplikace nejlépe funguje v prostředí více monitorů, což je pro řídící letového provozu standardem. Tento design je z důvodu, že na jednom monitoru bude kontrolní panel pro řízení simulace (zastavování, spouštění, nastavování map/scénářů, inicializace pseudopilotů), na dalších monitorech pak už vizualizace map pro ATCos. Možnost konfigurace je buď „leftmost“ (tzv. kontrolní panel se inicializuje na monitoru který je nejvíce vlevo, ostatní okna se pak na monitorech inicializují zleva do prava) a „rightmost“ (to samé, akorát zprava). Aplikace ale podporuje i jednomonitorové prostředí, kde se inicializuje jen panel pro řízení simulace a jedna ATC mapa.

Jako další velice důležitý prvek ATC simulátoru je simulace virtuálního času. K tomu, aby se nám letadla nějak pohybovala, je nutno do aplikace dát kód, který nezávisle na okolí bude backendu posílat data o změně času. V hlavním backendu aplikace je proto vytvořeno vlákno, které běží jako separátní proces a samo si počítá hodiny. Zatím je toto implementované pomocí javascriptových timeout funkcí, které nejsou vždy přesné, do budoucna bude proto důležité toto

přepsat do stabilnějšího jazyku či node-api-addonu. Pro prototyp je ale prozatimní přesnost dostačující.

Na obrázku 4 lze vidět celé schéma backendu aplikace. Aplikační backend je strukturován co nejmodulárněji, aby se daly části jeho kódu volně nahrazovat a nezávisle vylepšovat. To budou v budoucnu umožňovat zejména pluginy a moduly. Kód na registraci pluginů je teprve ve fázi vývoje. Naopak moduly, které narozdíl od pluginů budou v softwaru už na začátku zaintegrovány do systému, mají už hotovou implementaci komunikace s aplikačním jádrem. Zatím je v SEDAS zabudován pouze jediný modul, a to je SEDAS-AI-backend. Moduly budou většinou psané v jazyku C++, zejména díky jeho rychlosti a také bezpečnosti. Narozdíl od ostatních částí aplikace fungují moduly nezávisle na hlavním backendu a komunikace je jim zajišťována pomocí localhost socket komunikace (od toho také název MSC komunikace - Module-Socket-Communication, což je standard, definovaný v rámci projektu, pro bezpečnější komunikaci mezi moduly a aplikací). Mezi frontendem a backendem funguje tzv. IPC (Inter-Process-Communication) komunikace. V projektu byla napsána vlastní nadstavba, která umožňuje bezpečné odesílání zpráv a registraci kanálů pro větší robustnost komunikace (kap. 3.6.1). Frontend (kap. 3.7) je pak rozdělen na různá okna podle multi-monitor konfigurace. Controller okno umožňuje nastavení celé simulace, zatímco Worker okna (ve schématu na obrázku 4 definována jako 1 až N podle počtu monitorů) jsou užívána jako jednotlivá GUI rozhraní pro ATCos.

Aplikace během chodu simulace inicializuje dva datové registry. První slouží k uchovávání aktivních letadel na mapě, druhý k evidenci aktivovaných pseudopilotů.



Obr. 4: Schéma backendu aplikace

A

```
{
    "FILENAME": "maze.json",
    "AIRPORT_NAME": "MAZE",
    "TYPE": "ACC",
    "CODE": "",
    "COUNTRY": "",
    "CITY": "",
    "DESC": "Training map for all ATC beginners, goal of this map is to teach ATC trainees plane coordination etc."
}
```

B

```
{
    "CONFIG": "maze_config.json",
    "ACC": {
        "SECTOR": [
            {"x": 150, "y": 150},
            {"x": 500, "y": 80},
            {"x": 900, "y": 250},
            {"x": 750, "y": 600},
            {"x": 200, "y": 500}
        ],
        "ARP": "none",
        "POINTS": [
            {"x": 300, "y": 300, "name": "VEPOT"},
            {"x": 550, "y": 450, "name": "AFIS"}
        ],
        "SID": "none",
        "STAR": "none"
    },
    "long": "none",
    "lat": "none",
    "zoom": "none",
    "scale": "0.005m",
    "scenarios": [
        {
            "name": "Empty scenario",
            "category": ["AI", "HE", "GL", "AE"],
            "wtc_category": ["UL", "L", "M", "H", "J"],
            "apc_category": ["TODO"],
            "flight_schedules": []
        },
        {
            "name": "Scenario 1 Basic",
            "category": ["AI", "HE", "GL", "AE"],
            "wtc_category": ["UL", "L", "M", "H", "J"],
            "apc_category": ["TODO"],
            "flight_schedules": [
                {
                    "category": "AI",
                    "wtc": "L",
                    "time": "00:05:00",
                    "special_event": "none"
                },
                {
                    "category": "AI",
                    "wtc": "M",
                    "time": "00:00:15",
                    "special_event": "none",
                    "departure": "VEPOT",
                    "arrival": "AFIS",
                    "transport_points": []
                },
                {
                    "category": "HE",
                    "wtc": "L",
                    "time": "00:12:00",
                    "special_event": "none"
                }
            ]
        },
        {
            "name": "Scenario 1 Advanced",
            "category": ["AI", "HE", "GL", "AE"],
            "wtc_category": ["UL", "L", "M", "H", "J"],
            "apc_category": ["TODO"],
            "flight_schedules": [
                {
                    "category": "AI",
                    "weight_category": "L",
                    "time": "5min",
                    "special_event": "none",
                    "departure": "VEPOT",
                    "arrival": "AFIS",
                    "transport_points": []
                }
            ]
        }
    ]
}
```

Obr. 5: Konfigurace mapy pro zajštění základních (A) a komplexních informací (B).

Registry mají pevně stanovenou strukturu a umožňují manipulaci se záznamy, jako je registrace a deregistrace pseudopilotů, vytváření a terminace letadel či aplikování příkazů ovlivňujících jejich pohyb, t.j. vektorování.

Vzhledem k tomu, že projekt je z části psaný v Javascriptu, jsou veškeré konfigurace stanoveny ve formátu JSON. Konfigurace jsou tak kompatibilní s programovacím jazykem projektu. Ve formátu JSON jsou tak popsány mapy, scénáře a letadla, ale i různé nastavení layoutů a konfigurace pluginů a modulů. Konfigurace jsou tak user-readable a systém zároveň umožňuje uživateli přidávat další vlastní konfigurace, popř. měnit existující.

To lze vidět na obrázcích 5-A a 5-B. Mapa má napsané dvě na sebe odkazující konfigurace. Konfigurace A slouží aplikaci pouze k zjištění základních informací mapy, načítá se tedy dřív. Samotná konfigurace B se načítá pouze v době, kdy např. uživatel konfiguraci vybral v panelu nastavení simulace (obrázek 10). V konfiguraci jsou zaznamenané všechny souřadnice bodů,

které tvoří vzdušný prostor, nachází se zde i konfigurace latitude/longitude (v případě, že chceme mapu odkazovat na reálné místo na zemi), měřítko a všechny scénáře, které může uživatel u mapy vybrat.

3.5 Vykreslování GUI pro ATCos

Veškeré vykreslování letadel (v rámci imitování SSR/PSR radaru v simulaci) bylo zprostředkovávané prostřednictvím HTML Canvas. Pro vytváření prototypu to byl nejjednodušší způsob pro vytváření měnící se 2D grafiky ve webovém prostředí. Vzhledem k tomu, že pro aktualizaci polohy letadla v Canvasu je nutné celý Canvas smazat a následně překreslit, bylo potřeba v rámci zachování textury mapy (která je také vykreslována na Canvasu) a bodů trajektorie letadla, nutno přidat více Canvas objektů, které se navzájem překrývají a mají nastavenou jistou transparentnost, aby byl vidět i canvas pod nimi. Zároveň vzhledem k tomu, že štítky letadle jsou možné posouvat myší, je nutné přidat ještě Canvas další. Celá konfigurace lze vidět v tabulce 3

Tab. 3: Canvas konfigurace

Canvas	Objekty pro vykreslování
Canvas 1	Letový prostor, body, letiště, pozadí (není nutno překreslovat)
Canvas 2	Letadla (jejich vektory), body trajektorií letadel (nutno překreslovat periodicky)
Canvas 3	Štítky letadel (překreslování pouze po interakci s uživatelem - mouse drag)

3.6 Bezpečnostní mechanismy a robustnost

Při vývoji vlastního ATC simulátoru je nutné brát potaz v to, aby byl software co nejvíce robustní proti jakýmkoli chybám, které v takto komplexním softwaru mohou nastat. Proto bylo v kódu zavedeno několik bezpečnostních mechanismů, které by měly posilovat rezistenci systému (jako např. posílená IPC komunikace a logování).

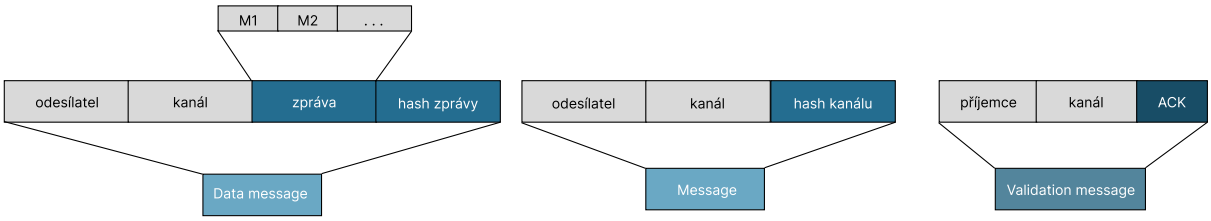
Pro backend byl vybrán Typescript, který oproti Javascriptu zajišťuje bezpečnost datových typů a zároveň také upozorňuje na chyby při kompilaci. Pro všechny C++ metody jsou napsané Typescript metody, které je volají a odchytávají jejich chyby.

3.6.1 IPC komunikace

Pro komunikaci mezi oknem aplikace (Renderer) a hlavním procesem (Main) využívá knihovna Electron mechanismus IPC (Inter-Process Communication). Renderer a Main spolu komunikují prostřednictvím specifických kanálů, na kterých si vyměňují textové zprávy. Vzhledem k rostoucí komplexitě aplikace bylo nutné zavést standardizovaný způsob přenosu zpráv. Za tímto účelem byl vytvořen tzv. IPCwrapper, který umožňuje správu kanálů (jejich inicializaci a deakti-

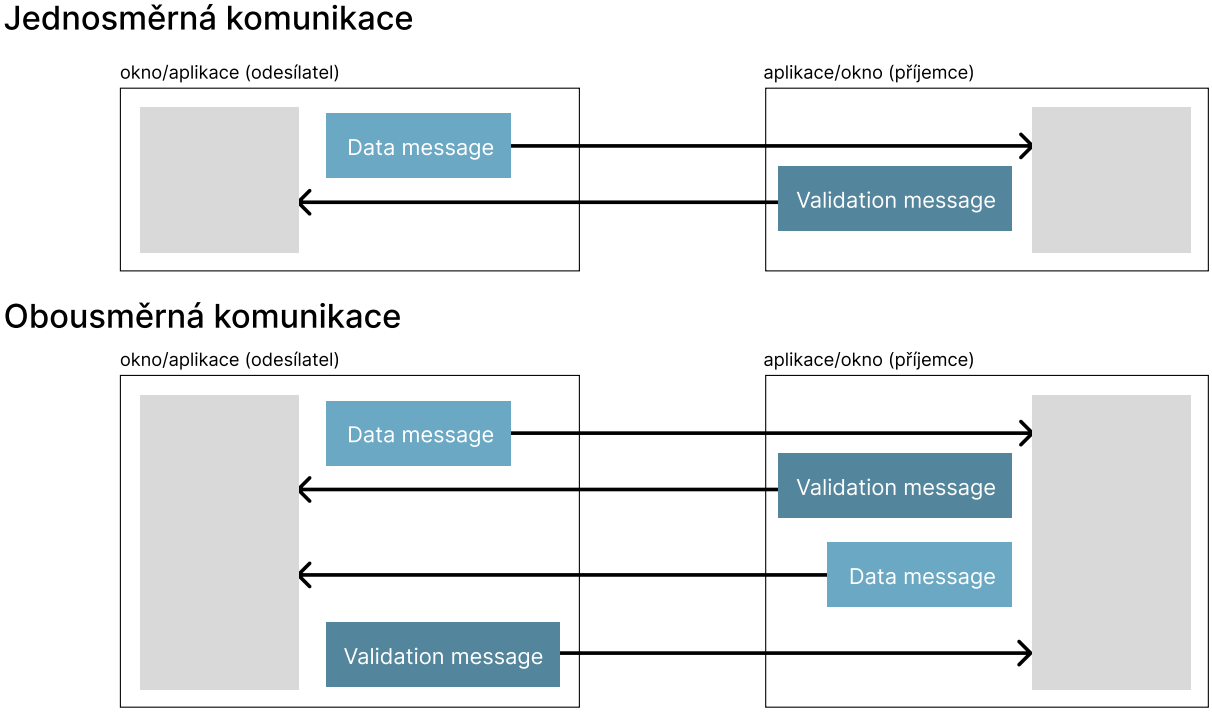
vaci) a zároveň ověřuje úspěšné doručení zpráv prostřednictvím validačních zpráv obsahujících ACK (Acknowledge) odpověď jako payload.

Společně s mechanismem IPCwrapper byl definován standard, podle kterého musí být v rámci aplikace SEDAS vytvářeny zprávy pro IPC komunikaci. Celkem existují tři typy zpráv, jak je znázorněno na schématu na obrázku 6.



Obr. 6: Definice zpráv pro IPC komunikaci

Prvním typem je Data message, která slouží k přenosu zpráv obsahujících data. Tento formát je navržen tak, aby umožňoval přenos více datových položek současně (M1, M2, ...). Druhým typem je Message, určený k emitování událostí, například v situacích jako „Uživatel stiskl tlačítko“. Tento typ zpráv neslouží k výměně dat, ale k notifikaci událostí v rámci aplikace. Posledním typem je Validation message, jejíž úkolem je potvrdit úspěšné přijetí zprávy. Tento validační mechanismus, spolu s abstrakcí komunikačního paradigmatu, je znázorněn na obrázku 7. Ke zjištění, zda nebyl obsah zprávy při komunikaci pozměněn, se z přijaté zprávy znovu vypočítá hash stejným algoritmem, jaký byl použit pro původní hash zprávy. Tyto hashe se pak následně porovnají.



Obr. 7: Schéma komunikace mezi Renderer a Main procesy

V IPCwrapperu je možné nastavit, jestli bude komunikace obousměrná, nebo jen jednosměrná. To pak zabraňuje chybám ve vývoji ve formě popletení kanálů. Validation message se však odesílají v obou módech.

3.6.2 Logování v aplikaci

Do simulátoru byl také přidán Logger, který shromažďuje informace o celém chodu programu a zaznamenává problémy a chyby do textových souborů. V současné době jsou implementovány dva nezávislé Logger mechanismy. Jeden ukládá záznamy z celého chodu aplikace (příklad záznamu lze vidět na obrázku 8), druhý jen záznamy z virtuálního prostředí ATC (v případě, kdyby totiž došlo k chybě v chodu simulace pak jde lépe nalézt původ chyby díky dedikovanému logování). Oba mají rozdílné soubory na zápis pro lepší přehlednost. SEDAS-AI-backend má také vlastního Loggera, ten však běží nezávisle na celém programu a sbírá data o transkripci a celém chodu konverzace mezi AI pseudopilotem a ATCo (příklad záznamu v kap. 4.4).

V případě jakéhokoli neočekávaného ukončení, SEDAS při inicializaci spouští kód na separátním vlákně, který pomocí nastaveného timeoutu (v uživatelském nastavení) periodicky ukládá stav aplikace a simulace. Zároveň také ukládá všechny logované zprávy, ale i data o monitorech, načtené mapě, typech letadel a zároveň všechny události simulace, které se staly. V budoucnu tak bude zajištěno, aby se v případě neočekávaného vypnutí dokázala aplikace znovu zapnout, načíst poslední backup a v simulaci pokračovat aniž by se tato změna nějak výrazně dotkla ATCo.

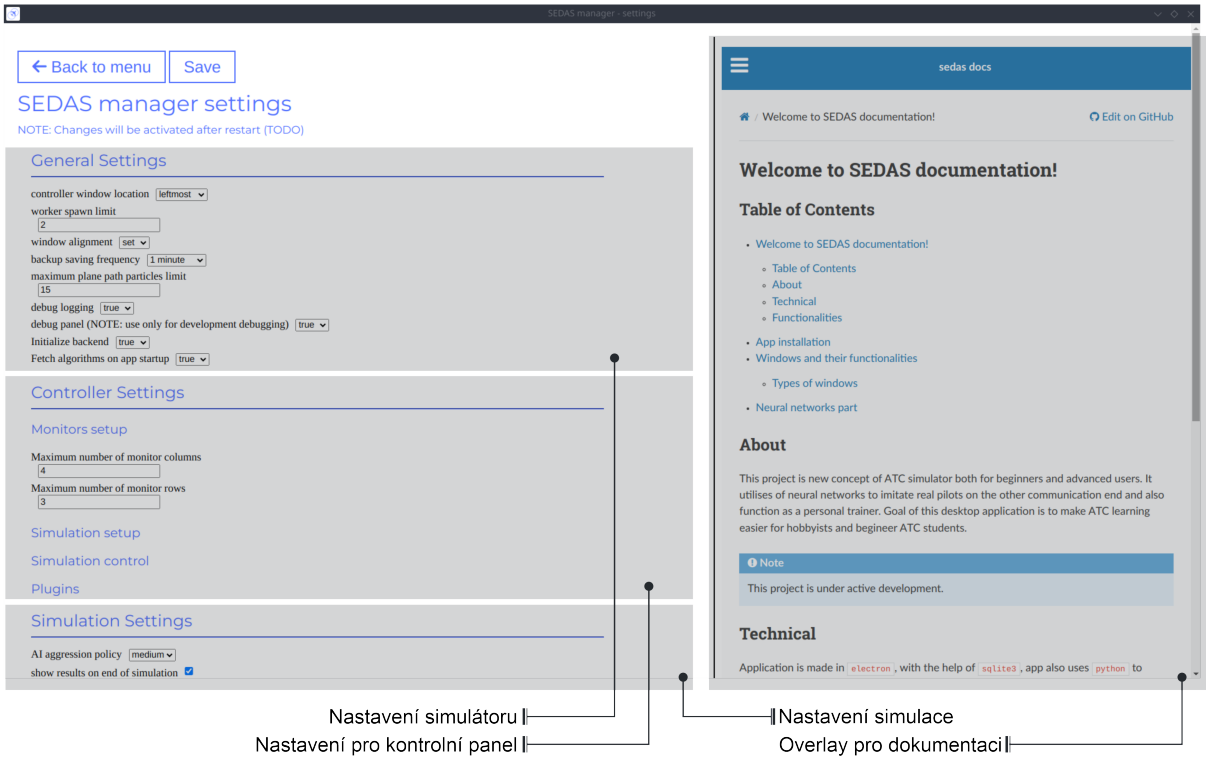
```
######################################
SEDAS manager v1.0.0 Linux linux 6.13.4-arch1-1
######################################
[10:03:59] (DEBUG) Addons loaded successfully
[10:03:59] (DEBUG) Created window object(win_type=none,path_load=/media/mywork/projects/sedas_dev/SEDAS-manager/src/res/html/other/load.html, coords=478,194)
[10:03:59] (DEBUG) Created window object(win_type=none,path_load=/media/mywork/projects/sedas_dev/SEDAS-manager/src/res/html/other/load.html, coords=2321,243.5)
[10:03:59] (DEBUG) Created window object(win_type=none,path_load=/media/mywork/projects/sedas_dev/SEDAS-manager/src/res/html/other/load.html, coords=4252,243.5)
[10:03:59] (DEBUG) APP-INIT
[10:03:59] (DEBUG) Performing HTTP GET on google servers for internet check
[10:03:59] (DEBUG) Lookup successful
[10:03:59] (DEBUG) Starting Backend because flag backend_init is true
[10:03:59] (DEBUG) BackupDB saving frequency is set to 60 seconds
[10:03:59] (DEBUG) Loading local plugins
[10:03:59] (DEBUG) Get display coords info for better window positioning
[10:03:59] (DEBUG) main-menu show
[10:03:59] (DEBUG) Created window object(win_type=none,path_load=/media/mywork/projects/sedas_dev/SEDAS-manager/src/res/html/other/main.html, coords=478,194)
[10:04:03] (DEBUG) worker show
[10:04:03] (DEBUG) Created window object(win_type=ACC,path_load=/media/mywork/projects/sedas_dev/SEDAS-manager/src/res/html/worker/worker.html, coords=1755,0)
[10:04:03] (DEBUG) worker show
[10:04:03] (DEBUG) Created window object(win_type=ACC,path_load=/media/mywork/projects/sedas_dev/SEDAS-manager/src/res/html/worker/worker.html, coords=3686,0)
[10:04:03] (DEBUG) controller show
[10:04:03] (DEBUG) Created window object(win_type=controller,path_load=/media/mywork/projects/sedas_dev/SEDAS-manager/src/res/html/controller/controller_set.html, coords=0,0)
[10:05:28] (DEBUG) Selected presets: MAŽE,command preset 1,Airbus preset
[10:05:28] (DEBUG) Created window object(win_type=none,path_load=/media/mywork/projects/sedas_dev/SEDAS-manager/src/res/html/other/load.html, coords=478,194)
[10:05:28] (DEBUG) Created window object(win_type=none,path_load=/media/mywork/projects/sedas_dev/SEDAS-manager/src/res/html/other/load.html, coords=2321,243.5)
[10:05:28] (DEBUG) Created window object(win_type=none,path_load=/media/mywork/projects/sedas_dev/SEDAS-manager/src/res/html/other/load.html, coords=4252,243.5)
[10:05:28] (DEBUG) Created worker widget window object(path_load=/media/mywork/projects/sedas_dev/SEDAS-manager/src/res/html/widget/worker_widget.html, coords=1805,50)
[10:05:28] (DEBUG) Created worker widget window object(path_load=/media/mywork/projects/sedas_dev/SEDAS-manager/src/res/html/widget/worker_widget.html, coords=3736,50)
[10:05:34] (DEBUG) user checked a map
[10:05:59] (DEBUG) Saving temporary backup...
[10:06:59] (DEBUG) Saving temporary backup...
[10:07:59] (DEBUG) Saving temporary backup...
[10:08:42] (DEBUG) Closing app... Bye Bye
[10:08:42] (DEBUG) Created window object(win_type=none,path_load=/media/mywork/projects/sedas_dev/SEDAS-manager/src/res/html/other/exit.html, coords=628,344)
[10:08:42] (DEBUG) terminating environment
[10:08:42] (DEBUG) stopping SEDAS modules
[10:08:44] (DEBUG) terminating backend worker
[10:08:44] (DEBUG) terminating database worker
[10:08:44] (DEBUG) exit

```

Obr. 8: Příklad výstupu globálního loggera aplikace

3.7 Uživatelské rozhraní

Samotné vytváření simulací v aplikaci se rozděluje na 2 části. Buď je možné si zapnout přímo předdefinovanou simulaci, neboli že v panelu nastavení simulace (obrázek 10) si uživatel navolí mapu, k ní příslušný scénář, typy letadel a typy povolených příkazů ATCo a software si sám sestaví příslušnou simulaci podle výběru uživatele (nutno podotknout, že tento systém není ještě kompletně doimplementován, více v kap. 5), a nebo si nastaví mapu s prázdným scénářem (Empty scenario) a dále si pak simulaci bude volně vytvářet za chodu podle sebe v panelu simulace (obrázek 12). Tato část je narozdíl od té první implementována. V následujících odstavcích je však i vysvětlení pro možnost navolení předdefinované simulace.

Uživatelské rozhraní je zejména děláno pro člověka, který je už v teorii ATC zkušený. V případě, že tomu ale tak není, je do částí aplikace zaintegrovaný panel pro wiki, který vysvětluje základy nastavování simulace, či aplikace samotné. V prvotním spuštění aplikace se otevře malé okno, ze kterého si uživatel může vybrat, zda-li chce zapnout už samotnou aplikaci a nebo před jejím zapnutím ještě jít do nastavení (Okno nastavení lze vidět na obrázku 9).



Obr. 9: Okno nastavení

Okno nastavení nabízí samo o sobě výše zmíněný panel dokumentace. Ten přímo replikuje samotnou dokumentaci, která je nahostována pomocí platformy Read the Docs (odkaz 5). Panel se tedy neobjeví, když nebude mít uživatel přístup k internetu, zbytek komponent aplikace ale touto změnou zůstane nedotčený. Samo nastavení je rozděleno do několika menších kategorií (General settings - nastavování chování aplikace, Controller settings - nastavování aplikace u

GUI ATCos, Simulation settings - nastavování chování simulace). Nutno podotknout, že ne všechny nastavené hodnoty dosud aplikace používá.

Když se uživatel nakonec rozhodne zapnout celý software, začnou se inicializovat všechny okna ATCos (obrázek 14) a také controller okno (ovládání všech prvků simulace). Controller okno je rozděleno do několika panelů: Setup (nastavení simulace), Monitors (nastavení zobrazení na monitorech), Simulation (simulace), Plugins (pluginy a další přípojný software) a Wiki (dokumentace), přičemž při prvotním zapnutí se okno zobrazí na panelu Setup (obrázek 10). Panel nastavení simulace, stejně tak jako panel simulace má kvůli nabytosti obsahem povolené scrollování. Panel nastavení pluginů není ještě kompletně hotov. I přesto, že částečné registrování pluginů je v aplikaci napsáno, není ještě natolik robustní, aby pro něj bylo zhotoveno GUI.

Obr. 10: Panel nastavení simulace

Tab. 4: Rozdělení kategorií letadel

Kategorie		WTC kategorie (podle MTOW)		
AI	Airplane	UL	Ultralight	454 kg či méně *
HE	Helicopter	L	Light	7 t či méně
GL	Glider	M	Medium	od 7 t do 136 t
AE	Aerostat	H	Heavy	136 t či více
		J	Super	Pouze 2 letadla: Airbus A380 (575 t) Antonov An-225 (640 t)

* ICAO definuje ještě další požadavky: Nemá více jak 2 sedadla, nepoužívá se pro hromadnou dopravu, KIAS nepřevyšující 35kt.

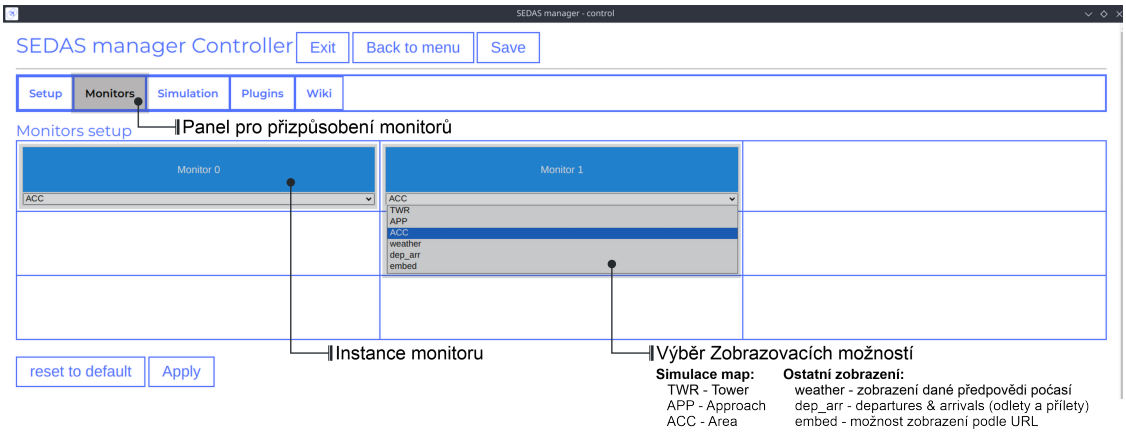
Panel nastavení simulace (obrázek 10) umožňuje uživateli vybrat mapu, na které se simulace bude odehrávat, a k ní příslušný scénář, který je v konfiguraci mapy napsán. Každá mapa ma tedy své vlastní scénáře, a postupným výběrem různých map se nám ukazují specifikace jejich scénářů. Každá mapa má také vlastní deskripci, ATC typ (zatím jsou podporovány pouze módy ACC), ICAO kódy letišť [25], Zemi a město. Mapy se však nemusí odkazovat na reálné místo, příkladem tomu je mapa MAZE, která je určena pro prvotní testování. Ta je koncipována jako testovací mapa ATCos pro trénink vektorování letadel.

Každý scénář má různé „tagy“ (category tags, weight category tags), které určují, jaké typy letadel se v simulaci mohou vyskytovat. Kategorie jsou rozepsané v tabulce 4 značí typy objektů, které se můžou ve vzdušném prostoru objevit. V definicích od EASA (Agentury Evropské unie pro bezpečnost letectví), která má v tomto ohledu stejné definice jako Úřad civilní letecké dopravy Spojeného království [26], je zaveden pojem balon, ne však Aerostat. Ten byl v rámci jednoduchosti značení typů v simulátoru zaveden, aby bral v potaz jak balony, tak vzducholodě. V rámci filtrování letadel v simulaci je ještě jedna kategorie jménem WTC (Wake turbulence category, také v tabulce 4). Ta zase klasifikuje letadla podle toho, jak moc turbulentí prostředí za sebou vytváří. To je důležité zejména vědět při přistávání nebo vzlétávání, kde mají obecně letadla větší náklon a tyto turbulentí prostředí by je mohly přivést do nestabilního chování, které by v nejhorším případě mohlo skončit havárií. Důvod výběru této klasifikace k simulátoru je, že se takto dá omezit jaké letadla budou v simulaci aktivní právě v závislosti na jejich maximální vzletové hmotnosti (MTOW).

Scénáře se tak dají ještě dále nastavovat, jako například deaktivací různých kategorií (software pak před zahájením simulace projde databázi letadel a eliminuje ty, které jsou součástí deaktivované kategorie). Následně také jde nastavit které datum a čas má simulace nastavit (zatím jen čistě estetické nastavení). Dále si pak uživatel vybere z jakých datových přednastavení ze kterého se letadla mají generovat (v případě výběru generování ze sady Airbus se pak budou generovat jen letadla společnosti Airbus). Dalším výběrem jsou pak datová přednastavení povolených příkazů (ty simulaci udávají, jaké příkazy budou povoleny, např. jenom vektorování leta-

del). Nutno podotknout, že selekce scénářů, letadel a příkazů zatím nemá hotovou implementaci ve softwaru. V budoucnu se totiž ještě musí dodělat implementace plánovaných simulací, které prozatím nejsou v provozu.

Poté, co uživatel má vybranou simulaci a její nastavení, a stiskne tlačítko „Confirm and setup“ se začne na oknech ATCo nastavovat prostředí. Mezitím má už uživatel možnost začít nastavovat jednotlivá okna podle jejich funkce (panel lze vidět na obrázku 11). Prozatím simulátor podporuje jen ACC simulace, připravená je však i struktura pro TWR a APP.



Obr. 11: Panel nastavení zobrazení na monitorech

V případě připojení monitorů, které budou mít pro ATCos využití jedině informativní (například informace o počasí na mapě, status odletů a příletů, či popř. replikace webového zdroje na okně - AisView, DronView), je možnost je takto nastavit v panelu nastavení zobrazení na monitorech.

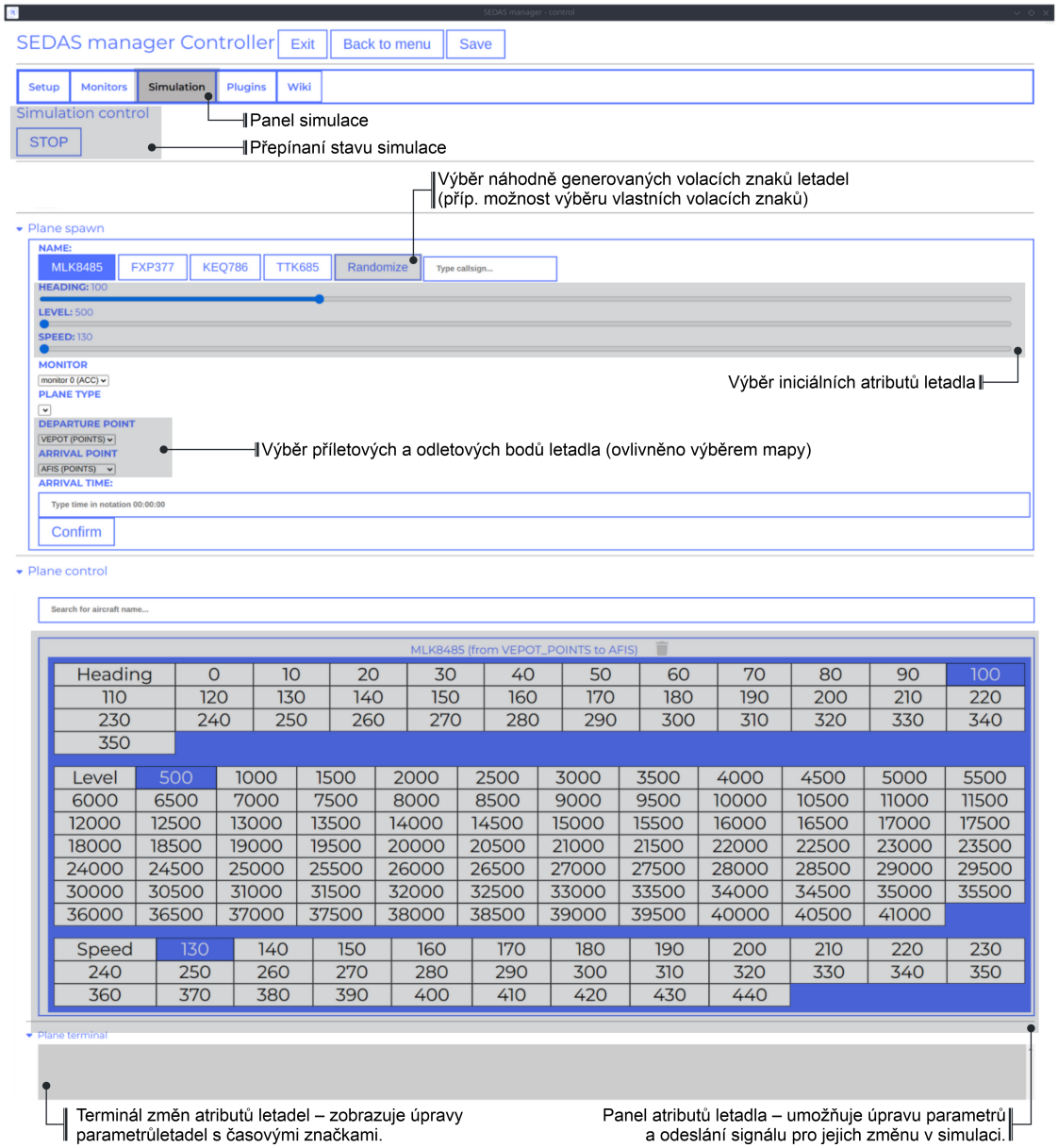
Mezi nejdůležitější panely na celé aplikaci je panel simulace (obrázek 12). V něm uživatel může ovládat chod simulace (generování nových letadel, vektorování letadel, zastavování simulace). Tento panel je zejména určen pro neplánované simulace, kde nejsou nastaveny generace různých letadel v různých časech.

Pro zajištění lepší orientace v aplikaci, zejména u méně zkušených uživatelů, je její součástí dokumentační panel (viz obrázek 13). Tento panel do značné míry replikuje strukturu a obsah oficiální dokumentace systému SEDAS, podobně jako tomu je v nastavení samotné aplikace. Jeho přínos však spočívá nejen v přehledném shrnutí ovládacích prvků a funkcionalit systému, ale také v možnosti seznámit se se základními principy ATC.

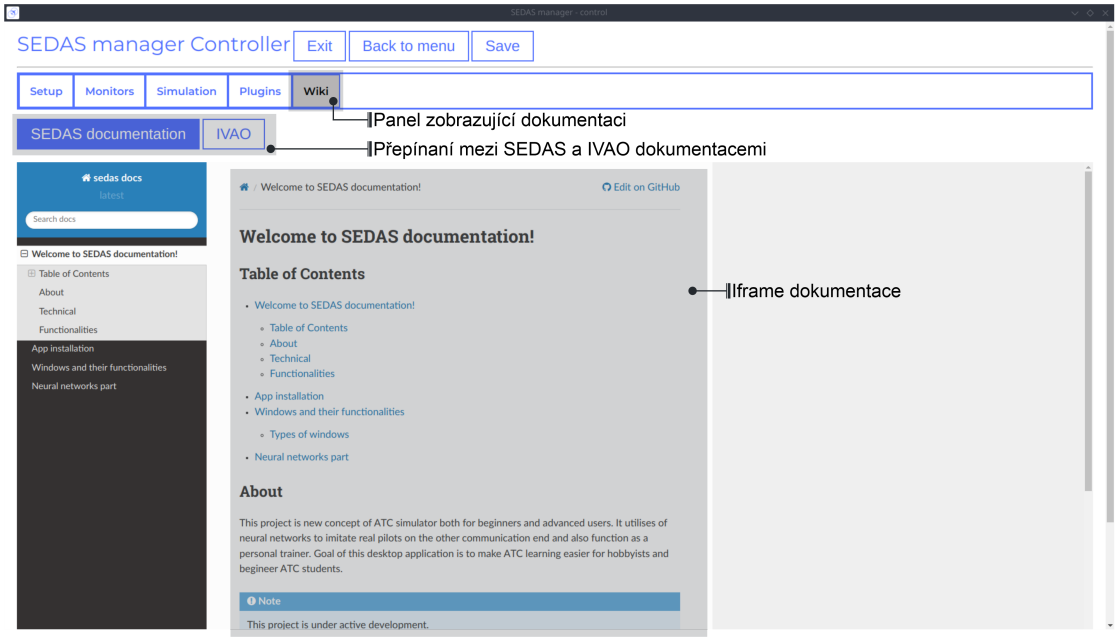
Uživatel zde nalezne tematicky tříděné teoretické informace, odkazy na odborné zdroje jako je Skybrary, a přepínač mezi dvěma základními režimy dokumentace – SEDAS a IVAO. Dokumentace IVAO obsahuje navíc i podrobné návody k praktickému výkonu role řídícího letového provozu, což může být obzvláště užitečné při výcviku nebo samostudiu.

Samotné rozhraní pro ATCo je znázorněno na obrázku 14. Grafický design tohoto uživatelského rozhraní byl částečně inspirován simulátorem ESCAPE-light, využívaným na Fakultě dopravní ČVUT (viz obrázek 1). Od něj se však odlišuje použitými barevnými schématy a odlišnou struk-

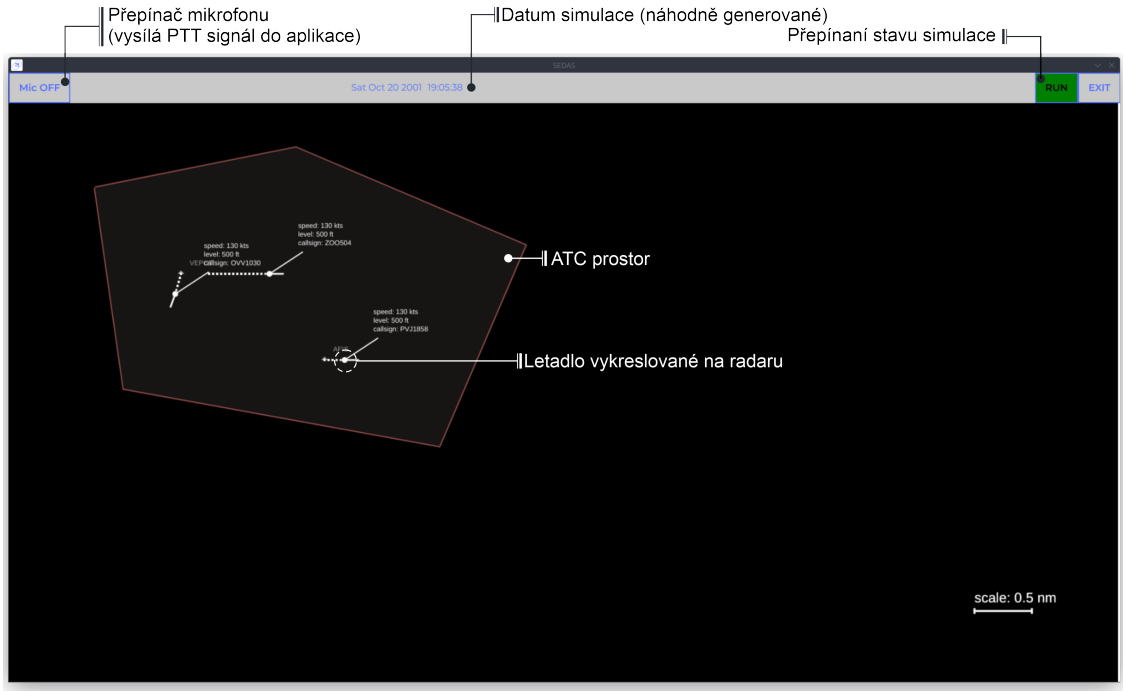
turou ovládacího panelu. V horní části rozhraní se nachází přepínač pro aktivaci a deaktivaci mikrofonu, který slouží k realizaci funkce Push-to-talk (PTT) – běžné součásti komunikačního systému v reálných i simulačních ATC prostředích. Kromě toho horní panel zobrazuje aktuální datum a čas v rámci simulace a umožňuje přepínání jejího stavu. ATCo má tedy možnost simulaci pozastavit nebo opětovně spustit podle potřeby bez nutnosti zásahu do řídicí konzole. Rovněž je zde integrována funkce pro úplné ukončení aplikace z pohledu ATCo, čímž odpadá nutnost přepínat se do aplikačního okna „Controller“ pro její manuální zavření.



Obr. 12: Panel simulace



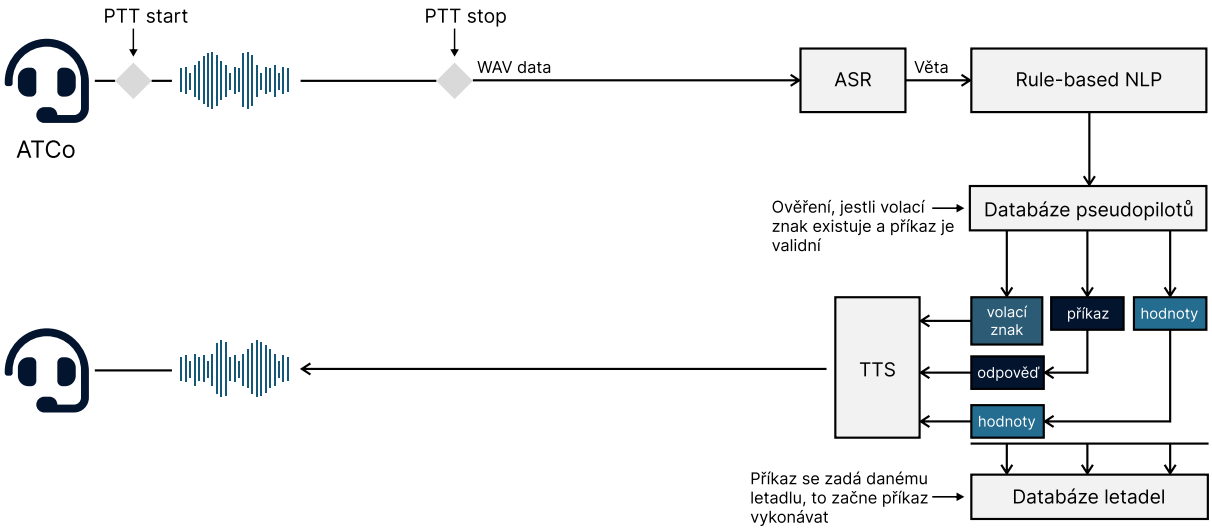
Obr. 13: Panel dokumentace



Obr. 14: Okno ATCo

4 IMPLEMENTACE AI PSEUDOPILOTŮ

AI modul je rozdělen na jednotlivé složky (ASR, NLP, TTS) které se spouští při zavolání PTT (Push to talk) signálu. Výstupem AI pseudopilota je syntetizovaná řeč, která se rovnou pošle do zvukového výstupu a signál do databáze letadel, který změní parametry letadla (natočení, rychlost, výšku) podle příkazu od ATCo. Struktura řešení nahrazení pseudopilota (viz obrázek 15) se částečně inspirovala pipeline z jiných výzkumů, které řešily podobnou problematiku [7].



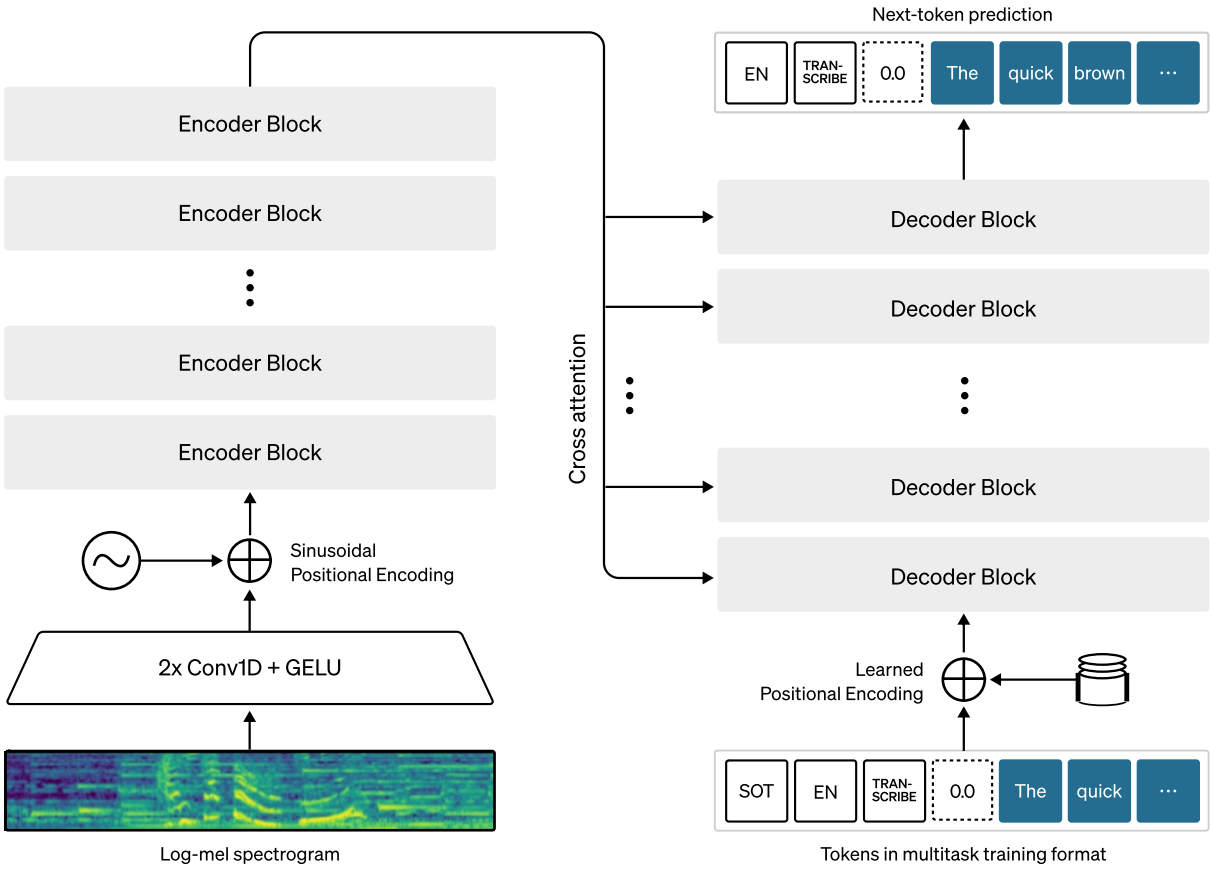
Obr. 15: Schéma struktury AI pseudopilota.

4.1 Rozpoznávání řeči

Rozpoznávání řeči bylo napsáno pomocí knihovny Whisper.cpp [16], což je inferenční nadstavba pro model Whisper [8], která umožňuje spouštět model na GPU pomocí Nvidia CUDA. To zrychluje celý proces odpovědi od AI pseudopilota. Jako model však nebyl použit výchozí Whisper model od OpenAI, ale jeho verze, která byla dotrénovaná na ATC komunikaci, Whisper-medium-ATC z práce [9], která se nachází na portálu Hugging Face.

Model z portálu byl ale ve formátu safetensor, který je podobný formátu pickle (formát sloužící k serializaci a deserializaci datových typů v jazyce Python), avšak je mnohem bezpečnější. Knihovna Whisper.cpp však podporuje jen formát GGML. Pro konverzi byl napsán vlastní skript (kap. 3.1.5), který se nachází v repozitáři ATC-whisper (příloha 3). Repozitář je rozdělen na dva git branche: *model-playground* a *main*. Branch model-playground obsahuje právě výše zmíněné skripty na konverzi modelů různých formátů, zatímco branch main obsahuje kód k dotrénování OpenAI whisper model na datech z projektu ATCOSIM [27]. Tato metoda však zatím nebyla využita, protože doteď model Whisper-medium-ATC na úkol vystačil.

ASR systém Whisper je založen na architektuře Transformer [8]. Vstupní audio si model rozdělí na 30 sekundové bloky které převádí do log-Mel spektrogramu (reprezentace zvukového signálu



Obr. 16: Architectura modelu Whisper dle [8]

pomocí Fourierovy transformace). Ten se následně pošle do enkodéru (obrázek 16), který ho zpracovává pomocí konvolučních vrstev a následně transformer bloků. Dekodér pak převádí reprezentace z enkodéru na textový výstup autoregresivně (tzv. token po tokenu) a tím je generován textový výstup.

4.2 Zpracování přirozeného jazyka

Zpracování textového výstupu je založené na Rule-based NLP metodách. Znamená to, že na základě jistých pravidel, kterými se text řídí, budeme dané slova zpracovávat. Tento přístup je efektivní a nevyžaduje tak přímo implementaci specifického algoritmu strojového učení. Vzhledem k tomu, že samotná ATC frazeologie je dostatečně strukturovaná, je toto použití prozatím dostačující. Vzhledem k zaměření projektu na vektorování, je implementace těchto metod jednoduchá a dá se napsat pomocí regulárních výrazů (regex) nebo if-else podmínek.

Jak už je výše v úvodu psáno, projekt je prozatím zaměřený na úlohu vektorování letadel, která je nejzákladnější úlohou ATC. Směr letadel se řídí podle tzv. headingů, které jsou určeny pomocí úhlů na kompasu. Letadlo tedy nikdy nemůže dostat více informací než jen jednu hodnotu a tou především bude nový směr a nebo letová hladina, kam se letadlo má dostat. Ve frazeologii jsou hodnoty tzv. „headingů“ dány vždy třímístnými hodnotami, tedy pokud by se letadlo navádělo na úhel 80°, ATCo by letadlu sdělil hodnotu „080“. Jak lze vidět v tabulce 5, pro číslo 9 jsou

zde 2 konverze. To je dáno tím, že ATC frazeologie nevyslovuje „nine“ ale pro srozumitelnější výslovnost ATCos říkají „niner“. Ne vždy toto ale ASR model přeloží správně, proto je třeba počítat s oběmi možnostmi.

Implementace celého Rule-based systému je v ukázce algoritmu 2. Zde je jen napsáno získávání volacího znaku, zbytek kódu (získání příkazu a hodnot příkazu) je totiž obdobný. Celé zpracování textu tedy produkuje 3 výstupy, těmi jsou volací znak, příkaz a hodnoty (obrázek 15). V NLP algoritmu je zmínka o 2 asociativních polích (V_{nato} a $V_{numeric}$). V_{nato} zajišťují konverzi na zkrácenou verzi z hláskovací abecedy, tedy aby volací znak letadla nebyl „Oscar Kilo Lima 4455“ ale „OKL4455“, protože v tomto formátu jsou všechny letadla v simulátoru zaznamenávána. Pole $V_{numeric}$ zajišťuje konverzi v případě, že ASR model jako výstup vrátí číslo ve slovním vyjádření namísto číselného formátu.

Tab. 5: Tabulka konverzí transkripce

V_{nato}				$V_{numeric}$	
alpha	A	november	N	zero	0
bravo	B	mike	M	one	1
charlie	C	oscar	O	two	2
delta	D	papa	P	three	3
echo	E	quebec	Q	four	4
foxtrot	F	romeo	R	five	5
golf	G	sierra	S	six	6
hotel	H	tango	T	seven	7
india	I	uniform	U	eight	8
juliet	J	victor	V	niner	9
kilo	K	whiskey	W	nine	9
lima	L	x-ray	X		
yankee	Y	zulu	Z		

Algorithm 2 Rule-based algoritmus pro získání volacího znaku z výstupu ASR
Vstupní hodnoty:
x - Vstupní transkripce z ASR
Výstupní hodnoty:
callsign - volací znak letadla
other_words - ostatní slova z transkripce po oddělení volacího znaku (prakticky tedy příkaz s hodnotami)

---

x = `"Oscar kilo lima 4455 fly heading 090"`

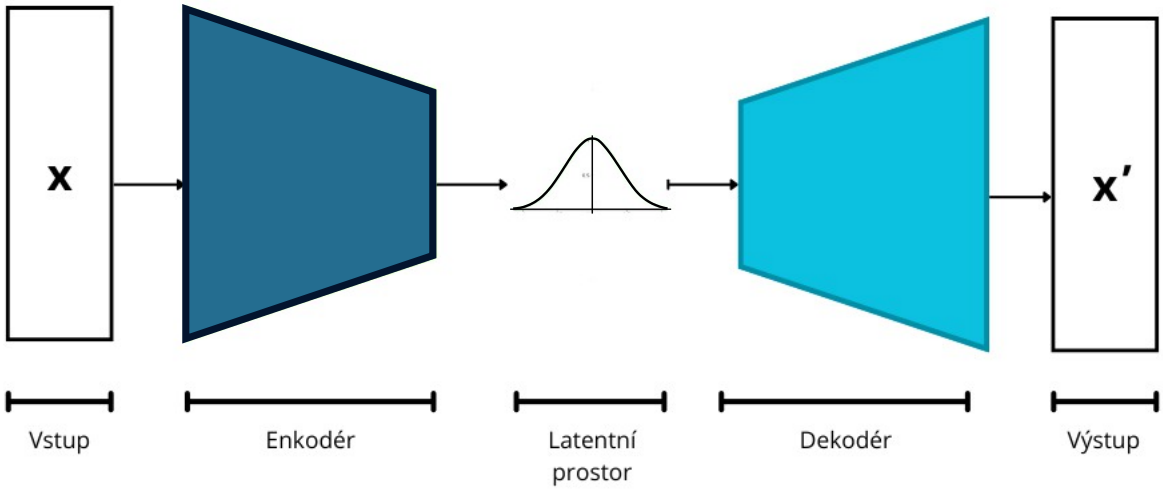
callsign = `" "`
other_words = x

```
X = split(x, "  ") // Rozdělí string na vektor substringů oddělených mezerami
for w in X do
    // Zjišťování, zda-li slovo není v NATO abecedě
    if w in V_nato then
        callsign += V_nato[w] // Přidá NATO znak do volacího znaku

        // Odstraní dané slovo z kopie proměnné x
        other_words = remove_string(other_words, w)

    // Zjišťování, zda-li vol. znak nemá v sobě čísla
    else
        if is_number(w) and length(w) ≥ 1 then
            callsign += w
            other_words = remove_string(other_words, w)

        // V případě, že ASR špatně přeloží hodnotu, např. místo "5"napíše "five"
        else if w in V_num then
            callsign += V_num[w]
            other_words = remove_string(other_words, w)
        else
            return callsign, other_words
        end if
    end if
end for
return callsign, other_words
```

4.3 Syntéza řeči

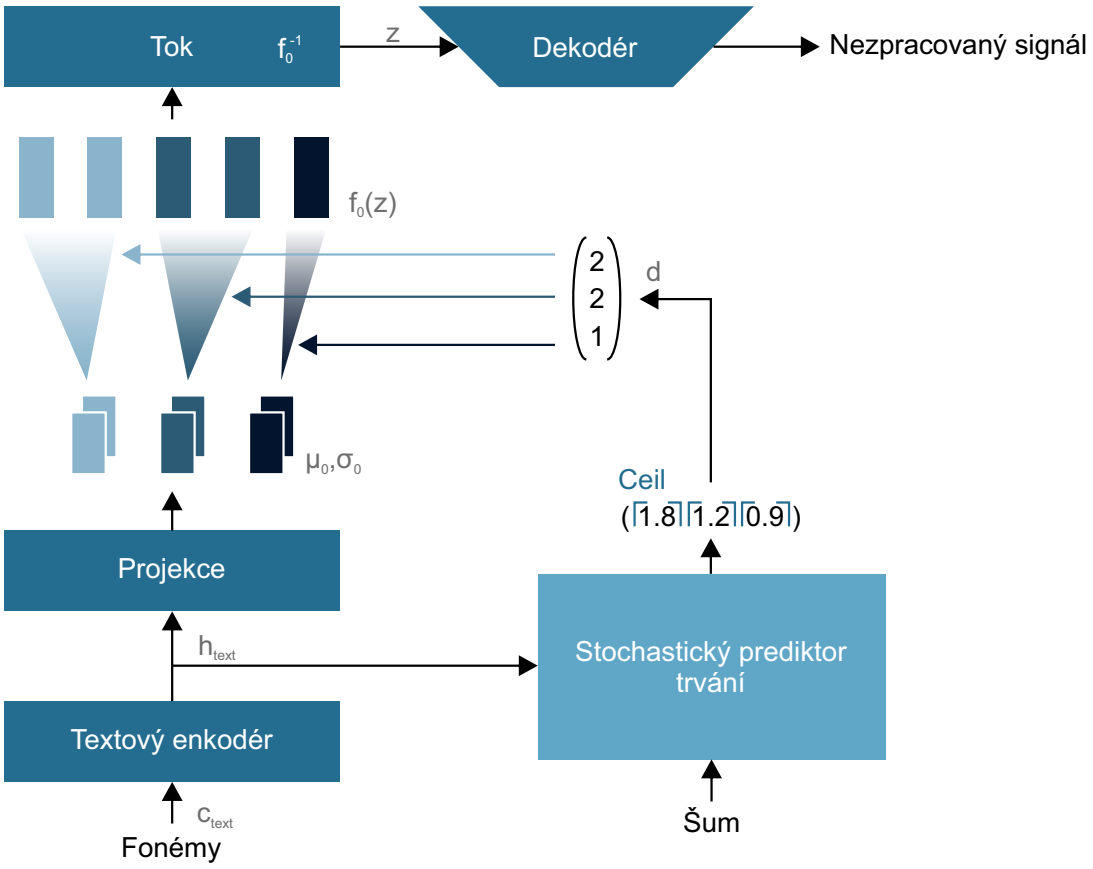
Zvolený systém Piper pro problematiku TTS byl zvolen kvůli efektivní inferenci modelu VITS v C++ a zároveň kvůli podpoře Nvidia CUDA. Samotný Piper systém je stavěn tak, aby se dal spustit i na méně výkonných zařízení, jako např. jednodeskových počítačích Raspberry Pi. Díky tomu není TTS systém tolik náročný na provoz. Piper využívá předtrénovaný model VITS [10] (Variational Inference for Text-to-Speech), který byl následně převeden do ONNX runtime, což je ekosystém na akceleraci inference existujících modelů.

VITS je založen na síti Conditional variational autoencoder (C-VAE) [28]. To jest rozšíření sítě VAE [29] (Variational autoencoder, viz obrázek 17), která funguje podobně jako autoenkodérové sítě. Vstup zpracovává enkodér, který kóduje data na vektor menší dimenze. Narozdíl od autoenkodérů se následně výstup z enkodéru převádí do Gaussovo distribuce v latentním prostředí, namísto toho, aby se jednalo jen o určitý bod v prostředí. Síť pak následně z distribuce náhodně vybere bod a ten následně zpracuje dekodér, který vygeneruje výstup. Oproti autoenkodérům tak VAE přidává jistou randomizaci a chová se tak více jako generativní model.



Obr. 17: Architektura VAE sítě

Oproti VAE má C-VAE zavedenou tzv. podmíněnost. Tedy k samotnému VAE je přidána ještě podmínka, která ovlivňuje proces kódování a dekódování. Na obrázku 18 je tak koncipován blok stochastického prediktoru chování (SDP), který predikuje délku fonémů, která je pro syntézu řeči zásadní.



Obr. 18: Architektura VITS dle [10]

Piper lze ovládat pomocí jediného binárního souboru, který spouštěním v terminálu generuje wavefile (WAV) soubor s hlasem pseudopilota. Zvuk je pak přímo spouštěn u ATCo přes systémový přehrávač audia. Projekt poskytuje syntézu mnoho hlasů a jazyků, pro nás je však nejdůležitější angličtina, která má v projektu Piper největší podporu. Každý hlas má vlastní JSON konfiguraci a ONNX runtime file, které binární soubor piper zavolá a podle toho sestaví hlas s příslušným textem. Pro co největší realističnost simulace je dobré mít co nejvíce hlasových konfigurací, aby nebyly v případě velkého množství AI pseudopilotů duplicitní. Do simulátoru je prozatím z projektu Piper zaintegrováno 19 hlasů z jazyku Angličtina či jeho variant.

V rámci dalšího zvýšení realističnosti simulace je do každé syntézy hlasu pomocí knihovny PortAudio (knihovna pro manipulaci s WAV soubory v C++) přidán ještě šum. Při reálné komunikaci mezi pilotem a ATCo šum vzniká špatným signálem či rušením. V simulátoru má šum u každého hlasu jinou intenzitu, aby si ATCos zvykli na komunikaci ve variabilním zašuměném prostředí.

```
Started recording
Stopped recording
Model out: [00:00:00.000 --> 00:00:04.820]   Charlie Bravo Alpha 1, 1-2-7, turn right heading 180.
Transcription: charlie bravo alpha 1 127 turn right heading 180
Pseudopilot response: turning right heading one, eight, zero,
Callsign: CBA1127
Plane callsign: CBA1127
Values: 180
Commands: turn-right

Started recording
Stopped recording
Model out: [00:00:00.000 --> 00:00:04.880]   November Foxtrot Tango 505, turn left heading 360.
Transcription: november foxtrot tango 505 turn left heading 360
Pseudopilot response: turning left heading three, six, zero,
Callsign: NFT505
Plane callsign: NFT505
Values: 360
Commands: turn-left

Started recording
Stopped recording
Model out: [00:00:00.000 --> 00:00:04.640]   Charlie bravo alpha one one two seven turn left heading three six zero
Transcription: charlie bravo alpha one one two seven turn left heading three six zero
Pseudopilot response: turning left heading three, six, zero,
Callsign: CBA1127
Plane callsign: CBA1127
Values: 360
Commands: turn-left

Started recording
Stopped recording
Model out: [00:00:00.000 --> 00:00:03.360]   November... uh...
Transcription: november uh
Pseudopilot response:
Callsign: N
Plane callsign: N
Values:
Commands:

Started recording
Stopped recording
Model out: [00:00:00.000 --> 00:00:05.840]   November Foxtrot Tango 505, turn right heading 090
Transcription: november foxtrot tango 505 turn right heading 090
Pseudopilot response: turning right heading zero, niner, zero,
Callsign: NFT505
Plane callsign: NFT505
Values: 090
Commands: turn-right
```

Obr. 19: Příklad komunikace ATCo a AI pseudopilota

4.4 Příklad komunikace

Na obrázku 19 lze vidět příklad komunikace ATCo a AI pseudopilota, přesněji řečeno jakým způsobem zpracovává kód pseudopilotů příkazy. Každá komunikace je oddělena prázdnými řádky. První dva řádky bloku komunikace označují volání PTT signálu (tzv. stisknutí tlačítka na ATCo GUI pro zapnutí/vypnutí mikrofonu). „Model out“ pak značí nezpracovaný výstup modelu Whisper (jde si povšimnout ještě nezpracovaných timestampů a také nekonzistentního přidávání mezer). „Transcription“ už je softwarově upravená transkripce ASR modelu. Dále si lze všimnout řádku „Pseudopliot response“, která značí text, který je odpovědí pseudopilota na ATCo příkaz (ten se pak dále syntetizuje). Zbylé řádky (Callsign/Plane callsign, Values, Commands) jsou výstupem Rule-based zpracování. Na řádku 33 lze vidět neúspěšnou komunikaci ze strany ATCo. Program už zde dále příkaz nezpracovával.

5 ZÁVĚR

Cílem projektu bylo vytvořit systém schopný nahradit lidské pseudopiloty pomocí metod strojového učení a tím optimalizovat tréninkový proces ATC. Projekt byl realizován ve třech samostatných modulech: ATC-whisper (zaměřeném na konverzi ASR modelu), SEDAS-AI-backend (integrující všechny metody ASR, NLP a TTS) a SEDAS (představujícím samotné simulační prostředí). Výsledkem je aplikace, která umožňuje definování mapových podkladů a scénářů včetně parametrizace frazeologie, tras a kategorie letadel. K realistickému vnímání simulace přispívá i přidaný šum do syntézy řeči, což zajišťuje věrnější zvukový výstup.

Simulátor je dostupný pod licencí GNU GPLv3 na platformě Github a je doplněn o podrobnou dokumentaci pomocí systému Read the Docs. I přes úspěšnou realizaci základního konceptu je aplikace zatím ve fázi prototypu a vyžaduje další vývoj. Zásadní výzvou do budoucna je optimalizace modelu AI, zejména snížení latence odpovědi ASR modelu a rozšíření NLP komponenty, která je dosud založena na rule-based metodách. Pro zajištění větší variability simulovaných scénářů a pokrytí širší škály řízených oblastí bude nutné tedy implementovat pokročilejší modely strojového učení.

Z hlediska simulačního prostředí bude zapotřebí přejít od lineárních výpočtů k modelům lépe aproximujícím reálnou dynamiku letů, tedy bude implementován vlastní aeromodel pro lepší popis pohybu letadel v závislosti na vnějších veličinách. Součástí plánovaného rozvoje je také restrukturalizace frontendové části s využitím webových frameworků a optimalizace aplikace pro lepší výkonnost. Další krokem je rozšíření kompatibility simulátoru na více operačních systémů, především Windows a macOS. Počítá se i rozšířením podpory na menší počítače, jako tomu je i u jednodeskových počítačů typu Raspberry Pi.

Kromě optimalizací samotné aplikace se počítá s rozšířením funkcionality, zejména zavedením jednotného audio systému s podporou různých vstupních a výstupních zařízení, což přispěje k větší multiplatformitě simulátoru. Důležitým aspektem dalšího vývoje je také implementace podpory pluginů, které umožní uživatelskou úpravu a rozšíření funkcionalit simulátoru. Již nyní jsou připraveny základní mechanismy pro jejich integraci, zbývá však dokončit uživatelské rozhraní pro jejich správu.

Plánovaný rozvoj pokrývá i vylepšení práce se scénáři a prostředím. Stávající systém konfigurace scénářů je funkční, ale v současné podobě limitovaný. Další vývoj povede k rozšíření podporovaných oblastí řízení, včetně přiblížení (APP) a letištní služby řízení (TWR). Rovněž se plánuje zlepšení simulace reálného času, což umožní přesnější modelování operačního prostředí ATC. Do budoucna se předpokládá implementace komplexnějších fyzikálních jevů, jako je simulace větru, která by ještě více přispěla k realistickému vnímání simulace a umožnila řídícím letového provozu lepší přípravu na reálné provozní podmínky.

Celkově lze konstatovat, že projekt dosáhl stanovených cílů a vytvořil základ pro budoucí rozvoj otevřeného a flexibilního ATC simulátoru. Díky modularitě a otevřenému přístupu k vývoji má

potenciál stát se platformou, která nejen usnadní přístup k výcviku řídících letového provozu, ale také poskytne prostor pro komunitní rozvoj a další inovace v této oblasti.

Použitá literatura

1. INTERNATIONAL CIVIL AVIATION ORGANIZATION. *Aeronautical Telecommunications Annex 10 Volume II*. Montreal, Quebec, Canada: International Civil Aviation Organization, 2001. kap. 5, [cit. 2025-02-24].
2. ŘÍZENÍ LETOVÉHO PROVOZU ČESKÉ REPUBLIKY. *VFR příručka ČR, VFR-GEN*. Jeneč, Česko: Letecká informační služba ŘLP ČR, 2024. GEN 6.1, [cit. 2025-02-24].
3. EUROCONTROL. *EUROCONTROL simulation capabilities and platform for experimentation*. 2018. Dostupné také z: `https://www.eurocontrol.int/simulator/escape`. [cit. 2025-02-24].
4. BOUCHAL, Albert; HAD, Petr; BOUCHAUDON, Philippe. The Design and Implementation of Upgraded ESCAPE Light ATC Simulator Platform at the CTU in Prague. In: *2022 New Trends in Civil Aviation (NTCA)*. 2022, s. 103–108. Dostupné z DOI: `10.23919/NTCA55899.2022.9934771`. [cit. 2025-02-24].
5. IVAO VZW. *Internation Virtual Aviation Organisation*. 2025. Dostupné také z: `https://www.ivao.aero/`. [cit. 2025-02-24].
6. VATSIM INC. *The International Online Aviation Network*. 2025. Dostupné také z: `https://vatsim.net/`. [cit. 2025-02-24].
7. PRASAD, Amrutha; ZULUAGA-GOMEZ, Juan; MOTLICEK, Petr; SARFJOO, Saeed; NIGMATULINA, Iuliia; VESELY, Karel. Speech and natural language processing technologies for pseudo-pilot simulator. *arXiv preprint arXiv:2212.07164*. 2022.
8. RADFORD, Alec; KIM, Jong Wook; XU, Tao; BROCKMAN, Greg; MCLEAVEY, Christine; SUTSKEVER, Ilya. Robust speech recognition via large-scale weak supervision. In: *International conference on machine learning*. PMLR, 2023, s. 28492–28518.
9. NEVAŘILOVÁ, Veronika. *Automatický přepis řeči letecké komunikace do textu*. 2024. Dostupné také z: `https://hdl.handle.net/11012/247442`. Bakalářská práce. Brno: Vysoké učení technické v Brně. Fakulta informačních technologií. Ústav počítačové grafiky a multimédií. [cit. 2025-02-24].
10. KIM, Jaehyeon; KONG, Jungil; SON, Juhee. Conditional Variational Autoencoder with Adversarial Learning for End-to-End Text-to-Speech. *ArXiv*. 2021, roč. abs/2106.06103. Dostupné také z: `https://arxiv.org/abs/2106.06103`. [cit. 2025-02-24].
11. GGML.AI. *GGML, tensor library for machine learning*. 2022. Dostupné také z: `https://github.com/ggml-org/ggml`. [cit. 2025-02-24].
12. HANSEN, Michael. *Piper: A fast, local neural text to speech system*. 2023. Dostupné také z: `https://github.com/rhasspy/piper`. [cit. 2025-02-24].
13. MICROSOFT. *Typescript, Javascript with syntax for types*. 2012. Dostupné také z: `https://www.typescriptlang.org/`. [cit. 2025-02-24].
14. OPENJS FOUNDATION. *NodeJS, run Javascript everywhere*. 2009. Dostupné také z: `https://nodejs.org/en`. [cit. 2025-02-24].
15. OPENJS FOUNDATION. *Electron, build cross-patform desktop apps with JS, HTML and CSS*. 2023. Dostupné také z: `https://www.electronjs.org/`. [cit. 2025-02-24].

16. GERGANOV, Georgi. *Whisper.cpp, Port of OpenAI's Whisper model in C/C++*. 2022. Dostupné také z: https://github.com/ggerganov/whisper.cpp. [cit. 2025-02-24].
17. SKYBRARY AVIATION SAFETY. *Surveillance*. 2021. Dostupné také z: https://skybrary.aero/articles/surveillance. [cit. 2025-02-24].
18. INTERNATIONAL CIVIL AVIATION ORGANIZATION. *Procedures for Air Navigation Services (PANS-OPS): Aircraft Operations, Volume I*. 6. vyd. Montreal, Quebec, Canada: International Civil Aviation Organization, 2018. Č. Doc 8168. ISBN 978-92-9258-670-6. [cit. 2025-02-24].
19. ŘÍZENÍ LETOVÉHO PROVOZU ČESKÉ REPUBLIKY. *VFR příručka ČR, VFR-ENR*. Jeneč, Česko: Letecká informační služba ŘLP ČR, 2025. ENR 3.1, [cit. 2025-02-24].
20. SKYBRARY AVIATION SAFETY. *Aircraft Call Sign*. 2021. Dostupné také z: https://skybrary.aero/articles/aircraft-call-sign. [cit. 2025-02-24].
21. FEDERAL AVIATION ADMINISTRATION. *FAA AC 120-26L: Assignment of Aircraft Call Signs and Associated Telephonies*. Washington D.C., USA: Federal Aviation Administration, 2016. kap. 5, [cit. 2025-02-24].
22. FEDERAL AVIATION ADMINISTRATION. *Pilot's Handbook of Aeronautical Knowledge (2025): FAA-H-8083-25C*. Washington D.C., USA: Skyhorse Publishing Inc., 2023. [cit. 2025-02-24].
23. BOEING COMMERCIAL AIRPLANES. *737 Airplane charasteristic for airport planning*. Arlington County, Virginia, USA, 2020. [cit. 2025-02-24].
24. SKYBRARY AVIATION SAFETY. *BOEING 737-900*. 2021. Dostupné také z: https://skybrary.aero/aircraft/b739. [cit. 2025-02-24].
25. INTERNATIONAL CIVIL AVIATION ORGANIZATION. *AFI planning and implementation regional group (APIRG)*. 2010. Dostupné také z: https://www.icao.int/wacaf/documents/apirg/sg/2010/afi_opmet_mtf2/docs/wp08.pdf. [cit. 2025-02-24].
26. UK CIVIL AVIATION AUTHORITY. *UK Air Operations Regulation*. 2024. Dostupné také z: https://regulatorylibrary.caa.co.uk/965-2012/Content/Regs/00040_art._2_Definitions.htm. [cit. 2025-02-24].
27. HOFBAUER, Konrad; PETRIK, Stefan; HERING, Horst. The ATCOSIM Corpus of Non-Prompted Clean Air Traffic Control Speech. In: *LREC*. Citeseer, 2008, sv. 3, s. 8. Č. 5.
28. SOHN, Kihyuk; LEE, Honglak; YAN, Xinchen. Learning Structured Output Representation using Deep Conditional Generative Models. In: CORTES, C.; LAWRENCE, N.; LEE, D.; SUGIYAMA, M.; GARNETT, R. (ed.). *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2015, sv. 28. Dostupné také z: https://proceedings.neurips.cc/paper_files/paper/2015/file/8d55a249e6baa5c06772297520da2051-Paper.pdf.
29. KINGMA, Diederik P.; WELLING, Max. Auto-Encoding Variational Bayes. *CoRR*. 2013, roč. abs/1312.6114. Dostupné také z: https://arxiv.org/pdf/1312.6114.

Seznam obrázků

Seznam tabulek

Seznam ukázek algoritmů

Přílohy

1. **Zdrojový kód SEDAS**
   `https://github.com/SEDAS-DevTeam/SEDAS-manager`
2. **Zdrojový kód SEDAS-AI-backend**
   `https://github.com/SEDAS-DevTeam/SEDAS-AI-backend`
3. **Zdrojový kód ATC-whisper**
   `https://github.com/SEDAS-DevTeam/ATC-whisper/tree/model-playground`
4. **SEDAS-DevTeam github**
   `https://github.com/orgs/SEDAS-DevTeam/repositories`
5. **SEDAS dokumentace**
   `https://sedas-docs.readthedocs.io/en/latest/`
6. **Ukázka funkcionalit SEDAS simulátoru**
   `https://www.youtube.com/watch?v=SKF-RxaYZEI`