VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

DIPLOMOVÁ PRÁCE

Brno, 2018 Bc. Michal Kolář

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY
DEPARTMENT OF CONTROL AND INSTRUMENTATION

KOMUNIKAČNÍ MODUL PRO SIMULÁTOR X-PLANE
COMMUNICATION MODULE FOR X-PLANE SIMULATOR

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Bc. Michal Kolář
AUTHOR

VEDOUCÍ PRÁCE — Ing. Miroslav Jirgl, Ph.D.
SUPERVISOR

BRNO 2018

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní obor **Kybernetika, automatizace a měření**
Ústav automatizace a měřicí techniky

Student: Bc. Michal Kolář ***ID:*** 146038
Ročník: 2 ***Akademický rok:*** 2017/18

NÁZEV TÉMATU:

Komunikační modul pro simulátor X-PLANE

POKYNY PRO VYPRACOVÁNÍ:

Cílem diplomové práce je návrh a realizace softwarového komunikačního modulu pro letecký simulátor X-PLANE 10, který bude schopen číst a zapisovat data ze/do simulátoru, pro možnost změny aktuální letové situace, resp. vybraných letových parametrů, a rovněž čtení těchto parametrů pro účely následného zpracování a analýzy. Výměna dat by měla probíhat s využitím některého ze standardních komunikačních protokolů.

1. Seznamte se se simulátorem X-PLANE 10 a stručně popište jeho základní vlastnosti.
2. Analyzujte možnosti čtení a zápisu parametrů simulace (simulátoru).
3. Navrhněte a definujte základní vlastnosti komunikačního modulu pro čtení a zápis dat.
4. Na základě předchozího návrhu realizujte a implementujte komunikační modul.
5. Ověřte a demonstrujte funkčnost řešení.
6. Sepište diplomovou práci.

DOPORUČENÁ LITERATURA:

LAMINAR RESEARCH. X-PLANE 10 Manual [online]. 10.40. 2013 [cit. 2018-01-25].

Termín zadání: 5.2.2018 ***Termín odevzdání:*** 14.5.2018

Vedoucí práce: Ing. Miroslav Jirgl, Ph.D.
Konzultant:

doc. Ing. Václav Jirsík, CSc.
předseda oborové rady

ABSTRAKT

V rámci diplomové práce byl řešen návrh komunikačního modulu pro letecký simulátor X-Plane, pomocí kterého je možné číst data ze simulátoru nebo změnit aktuální letovou situaci.

KLÍČOVÁ SLOVA

API, SDK, server, TCP, X-Plane, zásuvný modul

ABSTRACT

Within Mater's Thesis a design of the communication module for X-Plane simulator was worked out. It will enable to read data from simulation model or change current flight situation.

KEYWORDS

API, plugin, SDK, server, TCP, X-Plane

KOLÁŘ, Michal. *Komunikační modul pro simulátor X-PLANE*. Brno, 2018, 65 s. Diplomová práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace a měřicí techniky. Vedoucí práce: Ing. Miroslav Jirgl, Ph.D.

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „Komunikační modul pro simulátor X-PLANE“ 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 Ing. Miroslavu Jirglovi, Ph.D. za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Brno ....

podpis autora

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM VÝPISŮ

ÚVOD

Simulátor X-Plane je přední letecký simulační nástroj pro osobní počítače, který nabízí velmi realistický model. Díky věrné modelaci je simulátor X-Plane vhodný nástroj k výuce a udržování letových dovedností pilotů. Pro potřeby výuky umožňuje simulátor X-Plane využít instruktorskou stanici umístěnou v počítačové síti, která na základě pokynů letového instruktora ovlivňuje letovou situaci zkoušeného pilota.

Nicméně instruktorská stanice X-Plane má pro potřeby zadávající strany nevyhovující uživatelské prostředí, které neumožňuje modifikovat letovou situaci zkoušených pilotů, dle specifických potřeb. Zadavatel navíc potřebuje data simulačního modelu vyhodnocovat externími nástroji online během zkušebního letu, což instruktorská stanice neumožňuje. Na druhou stranu simulátor X-Plane nabízí sadu vstupně-výstupních metod pro komunikaci s externími nástroji.

Tato práce tedy vznikla na základě potřeby zadávající strany číst a zapisovat data do simulačního modelu X-Plane online po lokální síti během zkušebního letu. Práce se věnuje návrhu a následné realizaci komunikačního modulu pro letecký simulátor X-Plane, který by umožňoval číst a zapisovat data do simulačního modelu z instruktorského počítače umístěného v lokální síti dle specifických potřeb zadavatele.

1 SIMULÁTOR X-PLANE 10

X-Plane je komplexní a výkonný letecký simulátor pro osobní počítače, který nabízí velmi realistický model. X-Plane lze využít k simulaci letových vlastností s vysokou přesností. Díky věrné modelaci výkonu a ovládání letadla je X-Plane vhodný nástroj k výuce a udržování letových dovedností pilotů.

Pokud není explicitně uvedeno jinak, je obsah této kapitoly čerpán z oficiální dokumentace, viz [1].

X-Plane obsahuje jak podzvukovou tak nadzvukovou dynamiku, která uživateli umožňuje simulovat letové charakteristiky od nejpomalejších letounů po ty nejrychlejší. X-Plane zahrnuje více než 30 modelů letadel ve výchozí instalaci. Navíc lze simulátor doplnit o dalších 2000 modelů, které jsou dostupné online (mnohé z nic jsou k dispozici zcela zdarma). Dále si náročnější uživatelé mohou navrhovat své vlastní letouny.

Kompletní balíček scenérií pokrývá Zemi ve vysokém rozlišení od 74° severní šířky do 60° jižní šířky. Uživatelé mohou přistát na libovolném z více než 33000 letišť nebo na letadlových lodích, ropných plošinách, fregatách nebo helipadech budov. Mohou také realisticky modelovat let dálkově řízeného letounu, provést vypuštění raket *X-15* nebo pilotovat *Space Ship One* z mateřské lodi, provést opětovný vstup do atmosféry Země v raketoplánu, létat s dalšími uživateli přes LAN popřípadě internet nebo vypustit vodu na lesní požáry.

Počasí v X-Plane je variabilní od jasné oblohy s vysokou viditelností až po bouřky se silným větrem, turbulencemi a microburstry, popřípadě termální proudy pro kluzáky. Aktuální povětrnostní podmínky lze stáhnou z internetu a umožnit tak pilotovi létat za reálného počasí.

X-Plane obsahuje podrobné modelování selhání systémů, které může selhat ručně na příkaz instruktora, nebo nahodile v neočekávaných situacích. V každém okamžiku mohou selhat přístroje, motory, ovládače letu, řídicí kabely, antény, podvozek nebo cokoliv z desítek dalších systémů. Uživatelé mohou mít letového instruktora lokálně nebo prostřednictvím internetu pracujícího ze systému *Instructor's Operating Station*. Instruktor může měnit denní dobu, povětrnostní podmínky a stav selhání stovek systémů a komponent letadla. Dále může lektor kdykoli libovolně měnit polohu letounu.

Modely letadel jsou také velmi flexibilní, což uživateli umožňuje snadno modifikovat barvy, zvuky a přístrojové desky libovolného letounu. Vlastní návrhy letadel nebo helikoptér mohou být vytvořeny prostřednictvím přiloženého softwaru *Plane Maker*.

X-Plane využívají přední světové obranné složky, vzdušné síly, výrobci letadel a vesmírné agentury pro aplikace od letového výcviku až po koncepční návrh a tes-

tování letů.

X-Plane byl například užit při vyšetřování havárií. *Scaled Composites* využívali X-Plane k vizualizaci letů *Space Ship One* na okraj atmosféry v jejich tréninkovém simulátoru. *Kalitta* použila X-Plane k výcviku letu pilotů uprostřed noci na letadlech *Boeing 747*. *Northwest* a *Japan Airlines* využívá X-Plane k přezkoumání výcvikových letů. *Cessna* užívá X-Plane pro výcvik nových zákazníků v komplikovaném přístroji *Garmin G1000*. *Dave Rose* použila X-Plane k optimalizaci letounů pro dosažení vítězství na *Reno Air*. *NASA* užila X-Plane, aby otestovala vstup kluzáků do atmosféry Marsu.

1.1 Verze X-Plane simulátoru

X-Plane může být používán v široké škále situací od domácího až po komerční letecký výcvik. Standardní instalace X-Plane 10 je verze **Global** a je ideální pro téměř všechny domácí uživatele. Situace, které přesahují standardní domácí použití, vyžadují nákup *USB klíče*, který slouží k odemknutí funkcí verze **Professional**.

Simulátor X-Plane je dostupný pro tři nejběžnější softwarové platformy: *Microsoft Windows*, *Apple Mac OS* a operační systémy založené na jádru *Linux*.

X-Plane splňuje požadavky na certifikaci od *FAA* pro použití při zaznamenávání hodin do zkušebních letů a hodnocení. Tento certifikát umožňuje získat kredity na licenci soukromého pilota, opakovací trénink, hodiny školení přístrojů a hodiny na osvědčení *Airline Transport Certificate*. Nicméně certifikace *FAA* simulátoru vyžaduje verzi **X-Plane 10 Professional** a příslušný hardware (kokpit a letové ovladače).

1.1.1 X-Plane 10 Global

X-Plane 10 Global je standardní maloobchodní verze X-Plane simulátoru, kterou uživatel získá nákupem na oficiálních stránkách, viz [3]. Tato verze nevyžaduje připojení *USB klíče*. Více kopií X-Plane může běžet na více počítačích propojených do sítě, čehož lze využít pro externí vizualizaci, dohled z instruktorské stanice a podobně. Každý počítač v této síti vyžaduje vlastní DVD disk nebo digitální licenci simulátoru X-Plane. Systém založený na této verzi nelze certifikovat leteckým úřadem *FAA* ani jiným orgánem pro letecký výcvik, protože netestuje přítomnost letových ovladačů nebo použitelnou frekvenci snímkování.

1.1.2 X-Plane 10 Regional

Na rozdíl od verze **X-Plane 10 Global**, instalační disk **X-Plane 10 Regional** obsahuje pouze světové scenérie (např. Evropa nebo Severní Amerika). Tyto verze simulátoru X-Plane jsou k dispozici za nižší cenu než distribuce **X-Plane 10 Global**. Kromě scenérií je distribuce **X-Plane 10 Regional** shodná s distribucí **X-Plane 10 Global**.

1.1.3 X-Plane 10 Professional

Tato verze X-Plane je určená pro komerční použití a simulátory schválené *FAA*. Vyžaduje jeden *X-Plane Professional USB klíč* pro každou kopii X-Plane simulátoru v síti.

USB klíč je nutný pro *FAA* schválené simulátory, protože při spuštění X-Plane simulátoru kontroluje letové ovladače a snímkovací frekvenci podle požadavků *FAA*. Dále tento klíč umožňuje použití cylindrické a sférické projekce letové scény.

2 KOMUNIKAČNÍ MODUL

Tato práce vznikla na základě potřeby zadávající strany číst a zapisovat data ze simulačního modelu X-Plane online po lokální síti během letu. Cílem této práce je tedy návrh komunikačního modulu, který bude v nejširším možném záběru vyhovovat požadavkům zadávající strany. Dále by v rámci této práce měl být tento navržený modul implementován a otestován na X-Plane simulátoru v testovací laboratoři zadavatele.

2.1 Požadované vlastnosti

Požadované vlastnosti komunikačního modulu byly zadavatelem stanoveny takto:

1. Komunikační modul bude umožňovat čtení a zápis požadovaných dat do simulátoru X-Plane 10.
2. Modul bude komunikovat online po lokální počítačové síti během letu na simulátoru.
3. Modul bude ke komunikaci využívat standardní síťový komunikační protokol.
4. Samotná komunikace bude probíhat čistě textovou formou (data nebudou posílána ani přijímána v binární formě), aby bylo možné pracovat s komunikačním modulem z jednoduchého textového terminálu.
5. Komunikační modul bude požadovaná data ze simulátoru odesílat v přesně definovaném formátu dle přílohy A.
6. Komunikační modul bude číst a zapisovat data uvedená v tabulce 2.1.

Tab. 2.1: Soubor požadovaných dat pro čtení a zápis do simulátoru

n	Parametr	Popis parametru
0	`__alt,ftmsl`	The aircraft's altitude, in feet above mean sea level.
1	`AMprs,_inHG`	AMprs, in inches mercury.
2	`AMtmp,_degC`	AMtmp, in degrees Celsius.
3	`_beta,__deg`	The aircraft's angle of sideslip, in degrees.
4	`_elev,yoke1`	The user's elevator input, as a ratio to full upward deflection. For instance, a value of -1.000 indicated the user is commanding the aircraft's nose down as fast as possible.
5	`__lat,__deg`	The aircraft's latitudinal location, in degrees.
6	`__lon,__deg`	The aircraft's longitudinal location, in degrees.
7	`hding,__mag`	The aircraft's magnetic heading, in degrees.
8	`hding,_true`	The aircraft's true heading, measured in body-axis Euler angles.
9	`_Mach,ratio`	The aircraft's current speed, as a ratio to Mach 1.
10	`____P,rad/s`	Roll rate, measured in body-axes (when all is working as it should).
11	`pitch,__deg`	The aircraft's pitch, measured in body-axis Euler angles.
12	`____Q,rad/s`	Pitch rate, measured in body-axes (when all is working as it should).
13	`____R,rad/s`	Yaw rate, measured in body-axes (when all is working as it should).
14	`_real,_time`	The number of seconds, in the real world, elapsed since the simulator was launched.
15	`_roll,__deg`	The aircraft's roll, measured in body-axis Euler angles.
16	`ruddr,yoke1`	The user's rudder input, as a ratio to full rightward deflection. For instance, a value of -1.000 indicated the user is commanding the aircraft to yaw left as fast as possible.
17	`_Vind,_keas`	The craft's indicated airspeed, in knots equivalent airspeed (the calibrated airspeed corrected for adiabatic compressible flow at the craft's current altitude).
18	`_Vind,_kias`	The craft's indicated airspeed, in knots indicated airspeed.
19	`Vtrue,_ktas`	"The craft's true airspeed (the speed of the craft relative to undisturbed air), in knots true airspeed."
20	`__VVI,__fpm`	The aircraft's current vertical velocity, in feet per minute. If everything is working as it should, this is normal to the local horizon under the aircraft.

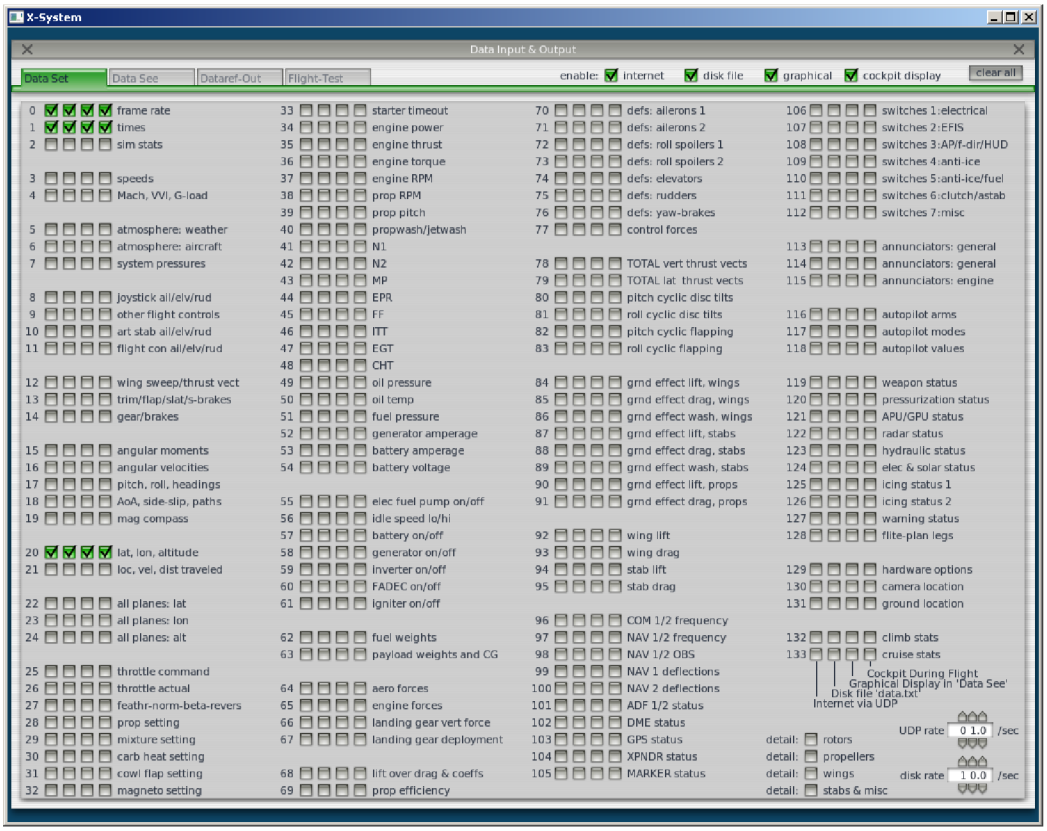
2.2 Analýza požadavků

Na základě požadovaných vlastností komunikačního modulu z kapitoly 2.1 byla provedena analýza realizovatelnosti jednotlivých požadavků zadavatele na komunikační modul.

2.2.1 Čtení a zápis dat do simulátoru

V referenci na požadavek číslo 1 z kapitoly 2.1 byly analyzovány možnosti čtení a zápisu dat do simulátoru X-Plane.

Simulátor X-Plane 10 umožňuje konfiguraci odesílání dat pomocí dialogového okna **Data Input & Output**, které je přístupné z vysouvacího menu **Settings** na horní liště hlavního okna simulátoru. Pod záložkou **Data Set** lze konfigurovat jednotlivé výstupy dat ze simulátoru, viz obr. 2.1. [1] U každé datové položky lze



Obr. 2.1: Konfigurace výstupních dat ze simulátoru

pomocí čtyř zaškrtávacích polí volit odpovídající typ výstupu:

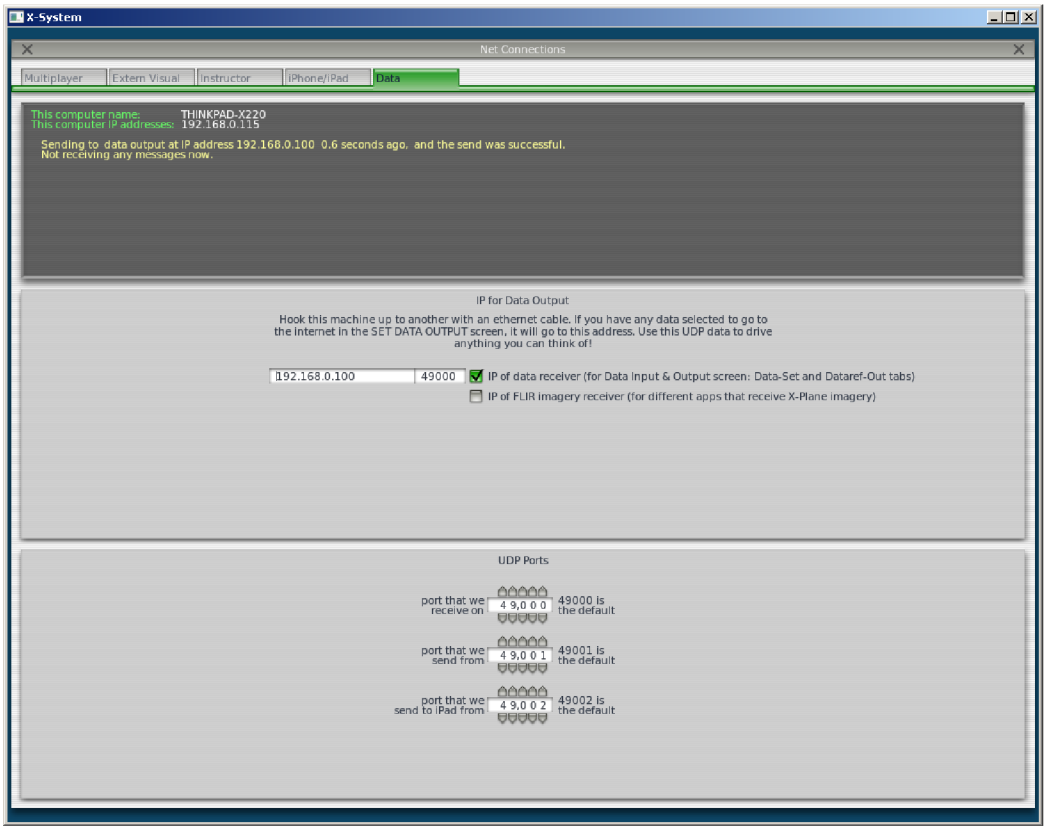
1. První pole odpovídá výstupu přes počítačovou síť pomocí UDP protokolu.
2. Druhé pole odpovídá výstupu do logovacího souboru `Data.txt` umístěného na lokálním disku v adresáři s nainstalovaným X-Planem.

3. Třetí pole odpovídá vizuálnímu výstupu ve formě vykreslovaného grafu, který je přístupný z dialogového okna **Data Input & Output** pod záložkou **Data See**.
4. Čtvrté pole odpovídá výstupu do kokpitu na obrazovku pilota. [1]

2.2.2 Online komunikace po síti

V referenci na požadavek číslo 2 byly analyzovány možnosti síťové komunikace online během letu na simulátoru.

X-Plane umožňuje posílat data po počítačové síti pomocí UDP protokolu. Tato data lze adresovat konkrétní IP adrese nebo plošně více účastníkům jako **broadcast**. Síťové rozhraní lze nakonfigurovat pomocí dialogového okna **Net Connections**, přístupného z vysouvacího menu **Settings**. V dialogovém okně **Net Connections**, viz



Obr. 2.2: Konfigurace síťového rozhraní

obr. 2.2, se v sekci **IP for Data Output** zadá požadovaná IPv4 adresa a komunikační port účastníka počítačové sítě, na kterém budou data ze simulátoru přijímána.

IP adresa slouží pro identifikaci konkrétního počítače v síti, nebo skupiny počítačů v případě *broadcastingu* a komunikační port identifikuje konkrétní proces běžící na cílovém počítači, pro který jsou odesílaná data určena.

Dále lze v sekci **UDP Ports** konfigurovat jednotlivé porty pro odesílání a příjem dat do simulátoru X-Plane. Pole **port that we receive on** identifikuje port, na kterém simulátor X-Plane naslouchá a přijímá UDP pakety. Na tento port by měly být adresovány UDP pakety určené ke zpracování simulátorem. Pole **port that we send from** identifikuje port, ze kterého X-Plane data odesílá do sítě. Tento údaj je možné zpracovat na přijímací straně. Lze ho například využít v případě, kdy jeden počítač přijímá data z více simulací současně. Pak by všechny simulátory X-Plane adresovaly data na stejný port, ale každý by měl nastaven unikátní port, ze kterého data odesílá. Podle tohoto portu by adresát rozeznal data od jednotlivých účastníků.

2.2.3 Standardní protokol

V referenci na požadavek číslo 3 byly analyzovány možnosti síťové komunikace založené na standardním komunikačním protokolu.

Z požadavku číslo 2 vyplývá, že analyzovat připadá v úvahu pouze síťovou komunikaci. Simulátor X-Plane využívá ke komunikaci po síti *User Datagram Protocol* (UDP). Tento komunikační protokol je standardizován a popsán v RCF768, viz [18]. Detailnější popis jakým způsobem X-Plane komunikuje přes UDP pakety je popsán v kapitole 2.3.

2.2.4 Textový formát dat

V referenci na požadavek číslo 4 byla analyzována možnost komunikovat v čistě textové formě bez nutnosti přenášet data v binárním tvaru. Tomuto omezení vyhovuje pouze výstup do logovacího souboru `Data.txt`, který generuje pouze znakový výstup:

- **Hlavičku** dat na začátku souboru ve formátu ERE[1]:
  `'^(  [_/a-ZA-Z-]{5},[_/a-ZA-Z-]{5} \| )+'`.
- **Data** ze simulátoru zapisovaná jako jeden řádek v pořadí odpovídajícímu hlavičce souboru s přednastavenou periodou ve formátu ERE:
  `'^(  \-?[ 0-9]+\.[0-9]{5} \| )+'`.

Vygenerovaný soubor `Data.txt` může vypadat například takto:

```
f-act,_/sec |   f-sim,_/sec |   frame,_time |   __cpu,_time |
   46.42475 |      19.90000 |       0.02154 |       0.00000 |
    7.35337 |      19.90000 |       0.13599 |       0.00000 |
    5.45272 |      19.90000 |       0.18339 |       0.14392 |
    6.63212 |      33.08620 |       0.15078 |       0.14392 |
    8.94932 |      32.69469 |       0.11174 |       0.14392 |
```

[1]IEEE POSIX Extended Regular Expressions [19]

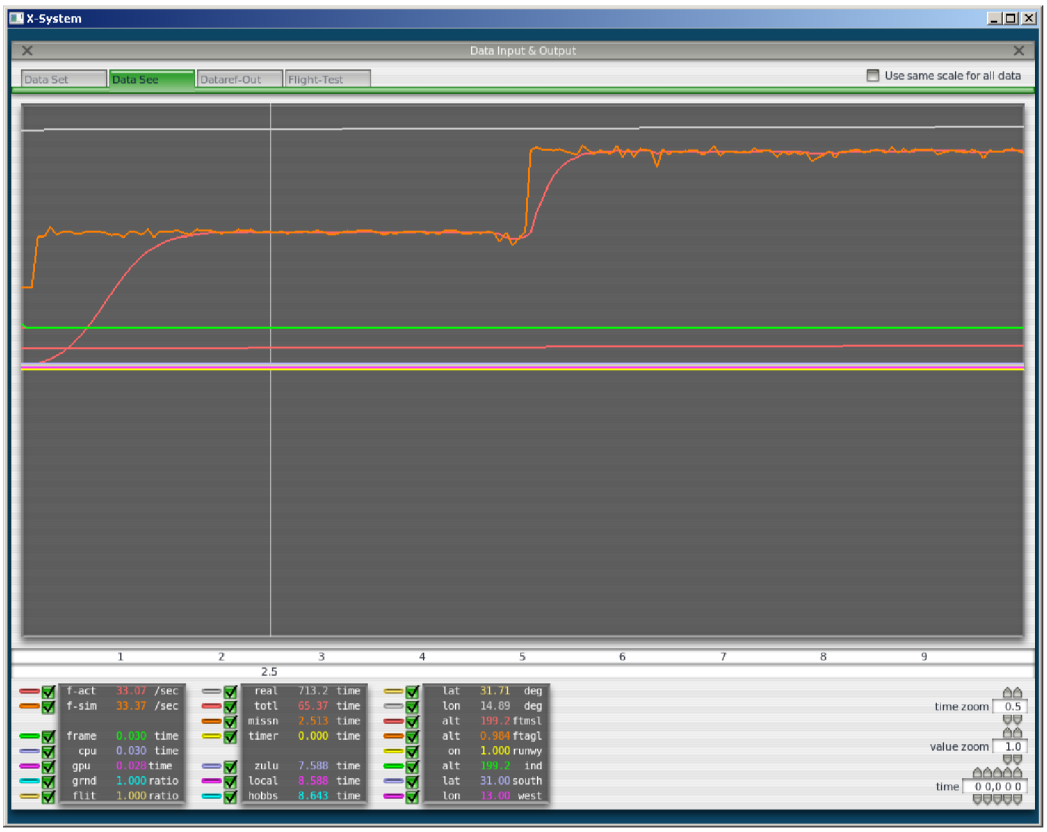
```
11.78324 |      33.02732 |       0.08487 |       0.14392 |
14.96273 |      32.88721 |       0.06683 |       0.07136 |
18.25886 |      32.87072 |       0.05477 |       0.07136 |
21.37691 |      33.12801 |       0.04678 |       0.07136 |
```

Vzorový soubor byl z důvodu úspory místa zkrácen v obou směrech. Tato úprava však nijak nedegraduje vypovídající schopnost vzoru.

2.2.5 Formát dat

V referenci na požadavek číslo 5 byla analyzována schopnost simulátoru X-Plane odesílat data v přesně definovaném formátu dle vzoru z přílohy A.

Vzhledem k tomu, že požadovaný formát může splňovat pouze textový výstup, redukuje se analýza pouze na výstup do logovacího souboru `Data.txt`. Analýzou rozdílů referenčního vzoru z přílohy číslo 5 a vlastního souboru vygenerovaného simulátorem X-Plane během zkušebního letu byla zjištěna shoda těchto formátů. Příklad výstupu dat do grafů je na obr. 2.3 a grafický výstup dat do kokpitu pilota na obr. 2.4.



Obr. 2.3: Datový výstup do grafů

Obr. 2.4: Grafický výstup dat do kokpitu (vlevo nahoře)

2.2.6 Požadovaná data

V referenci na požadavek číslo 6 byla analyzována schopnost simulátoru X-Plane číst a zapisovat data uvedená v referenční tabulce 2.1. Data uvedená v referenční tabulkách lze pro všechny uvedené metody datového výstupu nastavit pomocí dialogového okna **Data Input & Output**, které je přístupné z vysouvacího menu **Settings** na horní liště hlavního okna simulátoru.

2.3 UDP komunikace

V oficiálním manuálu simulátoru X-Plane 10 je zmínka o možnosti odesílat a přijímat data po síti pomocí UDP komunikace. Pro získání dalších informací o této metodě komunikace odkazuje manuál na dokument pod názvem '**Sending Data to X-Plane**' umístěný v adresáři s instalací simulátoru X-Plane 10, viz [2].

Tento dokument obsahuje nedostatečně obsáhlý a nestrukturovaný popis jakým způsobem je možné komunikovat se simulátorem pomocí UDP paketů. Vzhledem k nedostatečnosti popisu bylo studium této metody značně obtížné a opíralo se více o empirické znalosti získané při testování této komunikace, než o samotnou

dokumentaci. V následujících podkapitolách jsou uvedeny informace, které z této dokumentace bylo možné vytěžit.

2.3.1 Formát zprávy

Všechny UDP zprávy mají stejný formát a jsou tvořené:

- **Hlavičkou zprávy** o délce 5 znaků.
- **Datovou strukturou** obsahující informace, které se odesílají či přijímají. [2]

První 4 znaky hlavičky obsahují typ zprávy a poslední (pátý) znak je nulový '\0' (null terminated character). V dokumentu není uveden ucelený seznam všech typů zpráv a jejich specifikace, nicméně jsou zde uvedeny příklady definice datových struktur pro některé typy zpráv. Není známo zda-li tyto příklady pokrývají kompletní sadu zpráv, ale v tabulce 2.2 je vypsán seznam nalezených typů společně s jejich účelem.

Ze všech uvedených typů zpráv se nejzajímavěji jeví zpráva typu **DREF**, která umožňuje nastavit libovolnou hodnotu do libovolné datové reference s povolením zápisu. Příklad definice datové struktury zprávy typu **DREF** je uveden ve výpisu 2.1.

Výpis 2.1: Definice struktury pro DREF zprávy

```
struct dref_struct {
  xflt var;
  xchr dref_path[strDIM];
};
```

Kde datový typ `xflt` odpovídá 4-bajtovému datovému typu `float` s nativním pořadí bajtů, `xchr` odpovídá datovému typu `char` s nativním pořadím bajtů a `strDIM` odpovídá konstantě o hodnotě 500. [2]

UDP zpráva tohoto typu by mohla obsahovat následující data: `DREF` + `\0` + <4-bajtová hodnota 1> + `sim/cockpit/switches/anti_ice_surf_heat_left` + `\0`. Tato zpráva by měla zapnout vytápění povrchu křídel letounu a v binární formě by vypadala následovně:

```
00000000  4452 4546 0000 0000  0173 696d 2f63 6f63 |DREF.....sim/coc|
00000010  6b70 6974 2f73 7769  7463 6865 732f 616e |kpit/switches/an|
00000020  7469 5f69 6365 5f73  7572 665f 6865 6174 |ti_ice_surf_heat|
00000030  5f6c 6566 7400                         |_left.          |
```

2.3.2 Zarovnání

Struktura dat musí být bajtově zarovnaná podle aktuální implementace 64-bitového *X-Code kompilátoru*. To by mělo odpovídat zarovnání celých čísel (`int`) a čísel s plovoucí řádovou čárkou (`float`) na nejbližší 4 bajty a čísel s plovoucí řádovou čárkou

Tab. 2.2: Typy UDP zpráv simulátoru XPlane [2]

Typ	**Účel**
ACFN	Načíst letoun
BECN	Speciální zprávy při zapnutém broadcastingu
CHAR	Stisk klíče klávesnice
CMND	Spustit příkaz
DATA	Nastavit hodnotu datového výstupu
DCOC	Vybrat data pro výpis na obrazovku pilota
DREF	Nastavit datovou referenci na hodnotu
DSEL	Vybrat data pro odesílání přes UDP
DVIS	Nastavit pozici letounu
FAIL	Selhat systém
ISET	Nastavit možnosti internetu
LSND	Přehrát zvuk ve smyčce
MENU	Aktivovat položku menu
MOUS	Někde kliknout myší
NACF	Nastavit počet letadel na obloze
OBJL	Umístit objekt
OBJN	Načíst objekt
PAPT	Umístit na letiště
RECO	Zotavit systém
RPOS	Nastavit pozici letounu
RREF	Vyžádat odeslání požadovaných datových referencí
SOUN	Přehrát zvuk
SSND	Vypnout přehrávání zvuku ve smyčce
UCOC	Zrušit výběr dat pro výpis na obrazovku pilota
USEL	Zrušit výběr dat pro odesílání přes UDP
VEH1	Nastavit polohu jednoho letounu
VEHA	Nastavit polohu všech letounů

s dvojnásobnou přesností (`double`) na nejbližších 8 bajtů. [2]

To znamená, že pokud by před proměnou typu `double` byla uložena pouze jedna hodnota typu `int`, tato proměnná by zabrala také 8 B, aby zůstalo zachované zarovnání proměnných typu `double` na 8 B. Pokud struktura obsahující datový typ `double` obsahuje také pole proměnných typu `int`, toto pole je zarovnáno na 8 B pouze podle prvního prvku v poli a ostatní prvky pole jsou zarovnány na 4 B a každý prvek pole

vyjma prvního zabírá pouze 4 B. [2]

2.3.3 Pořadí bajtů

Veškerá komunikace před UDP pakety musí respektovat pořadí bajtů, které užívá počítač na kterém běží simulátor X-Plane, se kterým se komunikuje.

2.3.4 Ladění

Pro potřeby ladění komunikace založené na posílání UDP paketů je možné v dialogovém okně **Operations & Warnings** přístupném z vysouvacího menu **Settings** na horní liště simulátoru X-Plane zatrhnout volbu **dump net data to log.txt**. Tato volba by v případě chyby měla zajistit zaznamenání očekávané a obdržené velikosti přijaté struktury.

2.4 Vyhodnocení požadavků

Na základě provedené analýzy požadavků kladených na komunikační modul z kapitoly 2.1 byly vyvozeny následující závěry:

- Požadavek číslo 1 je realizovatelný pouze pomocí UDP zpráv, protože jako jediná z uvedených metod umožňuje zápis dat do simulátoru X-Plane.
- Požadavek číslo 2 přímo splňuje opět pouze komunikace pomocí UDP zpráv, ale nepřímo by se dala využít i metoda zápisu dat do souboru na disk. Za předpokladu, že by tento soubor byl čitelný během simulace (nebylo ověřeno), bylo by možné tato data číst a pomocí externího procesu komunikovat s požadovanými klienty v počítačové síti. Tato metoda by však nedokázala kompenzovat neschopnost zapisovat data do simulátoru.
- Požadavku číslo 3 na užití standardního síťového komunikačního protokolu metoda UDP zpráv vyhovuje. Standard je popsán v RCF768, viz [18].
- Požadavku číslo 4 na textovou komunikaci vyhovují pouze metody zápisu dat do souboru a výpis parametrů na obrazovku pilota. Výpis dat na obrazovku pilota je teoreticky možné pomocí metod zpracování obrazu dále vyhodnocovat, ale vzhledem k ekvivalentním datům z ostatních metod, které umožňují podstatně jednodušší zpracování, by toto řešení bylo nesmyslné.
- Požadavku číslo 5 na konkrétní formát textových dat přímo vyhovuje pouze metoda výpisu dat do souboru. Nepřímo by bylo možné do požadovaného formátu převést i data odesílaná před UDP zprávy a data vypsaná na obrazovku pilota.

- Požadavku číslo 6 na možnost čtení a zápisu dat uvedených v tabulce 2.1 vyhovuje pouze metoda UDP komunikace. Pro pouhé čtení dat ze simulátoru vyhovují všechny ostatní uvedené metody.

Pro přehlednější vyhodnocení jsou výsledky analýzy uvedeny v tabulce 2.3, kde **P** znamená přímo realizovatelné, **N** nepřímo realizovatelné a '–' nerealizovatelné. Z uvedené tabulky je zřejmé, že pro návrh komunikačního modulu by bylo nej-

Tab. 2.3: Realizovatelnost požadavků na komunikační modul

	Čtení a zápis	**Online síťová kom.**	**Stand. síť. proto.**	**Text. komunikace**	**Formát dat**	**Požadovaná data**
UDP zprávy	P	P	P	N	N	P
`Data.txt`	–	N	N	P	P	P
Obrazovka	–	N	N	P	N	P
Grafy	–	–	–	–	–	P

vhodnější zvolit metodu zasílání UDP zpráv, která přímo splňuje všechny požadavky kladené na komunikační modul až na požadavek komunikace textovou formou a komunikaci v definovaném tvaru. Komunikaci textovou formou by bylo možné realizovat nepřímo pomocí retranslačního uzlu, který by přijímal požadavky od klienta v počítačové síti v preferované textové formě. Tyto požadavky by přeložil do binárního formátu nativně podporovaného simulátorem X-Plane a odeslat simulátoru. Obdobně by retranslační uzel pracoval v případě příjmu dat ze simulátoru. Přijatá binární data by byla přeložena do textové formy dle preferencí zadávající strany a odeslána klientovi v síti.

V případě, že by nebyl kladen požadavek na zápis dat do simulátoru, dala by se pro komunikaci velmi podobně využít metoda zápisu dat do souboru. Retranslační uzel by četl přímo textová data ze souboru `Data.txt` a tato data beze změny přeposlal klientovi v síti. Vzhledem k požadavku na online komunikaci během letu by však tato metoda vyžadovala schopnost číst data ze souboru současně během zápisu dat v průběhu simulace.

Na základě vyhodnocení realizovatelnosti požadavků kladených na komunikační modul byl zahájen vývoj komunikačního modulu pomocí metody zasílání UDP

zpráv. Tento vývoj byl vzhledem ke zmíněným nedostatkům příslušné dokumentace neúspěšný. Velmi dobře fungovalo přijímání UDP zpráv ze simulátoru a jejich parsování na požadovaná data, ale z neznámých důvodů simulátor X-Plane nepřijímal žádné požadavky od klienta. Nebyla zaznamenána ani žádná chyba příjmu zprávy, která by měla být zalogována v souboru `log.txt`. Kvůli dlouhodobému neúspěchu rozchodit příjem zpráv na straně simulátoru bylo zahájeno pátrání na internetu po příčině. Příčina nebyla nalezena, ale z neoficiálních zdrojů bylo zjištěno, že komunikace simulátoru pomocí UDP zpráv je experimentální funkce, která nemusí být plně funkční a může se kdykoliv změnit bez zachování zpětné kompatibility.

Během hledání řešení problému byla objevena další metoda komunikace se simulátorem X-Plane. Jedná se o možnost doplnit funkcionalitu simulátoru pomocí externích zásuvných modulů. O této možnosti není v oficiálním manuálu žádná zmínka a proto tato možnost nebyla analyzována. Po krátkém studiu možností zásuvných modulů ovlivnit simulátor X-Plane bylo usouzeno, že tato metoda umožňuje splnit veškeré požadavky kladené na komunikační modul, navíc zajistit vyšší výkon a zpětnou kompatibilitu v případě přechodu na vyšší verzi simulátoru X-Plane. Bylo tedy upuštěno od snahy využít nativní metodu komunikace pomocí UDP zpráv a bylo zahájeno studium a návrh komunikačního modulu pomocí zásuvných modulů. O možnostech doplnění funkcionality simulátoru pomocí zásuvných modulů pojednává samostatná kapitola číslo 3.

3 X-PLANE PLUGIN SDK

X-Plane Plugin SDK je vývojové prostředí, které umožňuje uživateli nebo třetím stranám vytvářet zásuvné moduly (pluginy) pro X-Plane v *ANSI C* nebo v jiném binárně kompatibilním jazyce.

Pokud není explicitně uvedeno jinak je obsah této kapitoly čerpán z oficiální dokumentace, viz [4].

3.1 Přehled

3.1.1 Zásuvný modul

Zásuvný modul je spustitelný kód, který běží pod X-Plane simulátorem a rozšiřuje činnost X-Plane simulátoru. Zásuvné moduly jsou modulární a umožňují vývojářům modifikovat simulátor bez zdrojového kódu simulátoru.

3.1.2 Možnosti zásuvného modulu

Zásuvné moduly umožňují rozšířit možnosti letového simulátoru nebo získat přístup k datům simulátoru. Zásuvné moduly jsou odlišné od běžných programů v tom, že sami o sobě nejsou kompletní programy, ale pouze část kódu, která je vykonávána v rámci X-Plane za běhu simulátoru. Díky tomu, že zásuvné moduly běží přímo pod simulátorem, jsou schopny dosáhnout věcí, které běžné programy neumožňují, ale zároveň mají limitace, kterými běžné programy netrpí.

Zásuvné moduly jsou schopny:

- Vykonávat kód pod X-Plane simulátorem, buď kontinuálně (např. jednou za letový cyklus) nebo v reakci na událost simulátoru.
- Číst data ze simulátoru (např. modul může kontinuálně číst hodnoty letových přístrojů a zapisovat je do souboru).
- Zapisovat data do simulátoru a vykonávat tím akce nebo měnit chování (např. modul může měnit počasí na základě nějakého modelu).
- Vytvářet uživatelské prostředí v rámci simulátoru (např. modul může vytvořit vyskakovací okno s přístroji nebo dialogovým oknem).
- Řídit různé subsystémy simulátoru.

Některé části simulátoru je možné řídit jednoduše zápisem dat do simulátoru (např. nastavení výšky mraků). Ostatní subsystémy vyžadují více komplexní přístup a mají specifické API. Např. API pro programování Flight Management System.

3.1.3 DLL

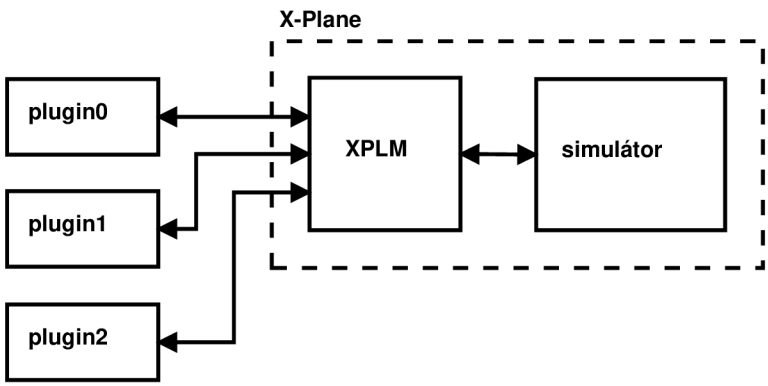
X-Plane zásuvný modul je Dynamic-Link Library (DLL). DLL je modul obsahující funkce a data, která mohou být použita jiným modulem (aplikací nebo DLL). [4][17]

DLL může definovat dva druhy funkcí: *exported* a *internal*. *Exported* funkce jsou určeny pro volání jinými moduly a také v rámci DLL, ve kterém jsou definovány. *Internal* funkce jsou obvykle určeny pro volání pouze v rámci DLL, ve kterém byly nadefinovány. Přesto, že DLL může data exportovat, obvykle jsou tato data používána pouze jejich funkcemi. Nicméně neexistuje ochrana proti čtení nebo zápisu jiným modulem na adresu těchto dat. [17]

DLL poskytují možnost modulárních aplikací tak, že jejich funkce může být aktualizována a znovu užita jednodušeji. DLL také pomáhá redukovat paměťovou režii pokud několik aplikací používá stejnou funkcionalitu ve stejný čas. Přesto, že každá aplikace obdrží vlastní kopii dat DLL, aplikace sdílí kód DLL. [17]

3.1.4 Zásuvné moduly a X-Plane

Systém zásuvných modulů simulátoru X-Plane je postaven na základě DLL. Centrální komponentu tohoto systému tvoří X-Plane Plugin Manager (XPLM). XPLM je knihovna řídící zásuvné moduly. XPLM je samo o sobě také DLL a X-Plane jej linkuje stejně jako ostatní zásuvné moduly. XPLM slouží jako centrální uzel v systému zásuvných modulů, viz obr. 3.1.



Obr. 3.1: XPLM komunikuje se simulátorem a zásuvnými moduly

Zásuvný modul obsahuje kód, který bude vykonáván v rámci simulátoru. Funkce, které jsou vyžadovány aby byly v rámci XPLM volány jsou exportovány a XPLM je nalezne v adresáři se zásuvnými moduly.

XPLM obsahuje funkce, které zásuvný modul potřebuje k modifikaci simulace, čtení dat, vytvoření uživatelského prostředí, atd. XPLM exportuje tyto funkce do svého adresáře, aby je zásuvné moduly mohly najít a zavolat.

Zásuvné moduly nikdy nekomunikují přímo s X-Plane simulátorem. Komunikace probíhá vždy skrz XPLM (z praktických důvodů lze prohlásit, že pro zásuvné moduly je XPLM přímo simulátor a není nutné znát detaily komunikace mezi XPLM a X-Plane simulátorem). Zásuvné moduly také nikdy nekomunikují přímo s jiným modulem, namísto toho posílají zprávy prostřednictvím XPLM.

Callback je funkce v zásuvném modulu, kterou volá XPLM, aby informovalo modul o dějích v simulátoru a umožnilo vykonat kód modulu. Callbacky jsou v zásuvných modulech dvojího druhu:

- **Vyžadované callbacky** jsou funkce, které zásuvný modul exportuje z DLL. XPLM tyto funkce vyhledá a ve vhodné době je zavolá. Modul musí exportovat všech pět vyžadovaných callbacků, v opačném případě nebude zásuvný modul správně zaveden. Nicméně funkce mohou být prázdné a nevykonávat žádné operace.
- **Registrované callbacky** jsou funkce, které umožňují přístup k dalším možnostem simulátoru. Například je lze využít k obsloužení události vyvolané klikem myši na vytvořené okno. Na rozdíl od **vyžadovaných callbacků**, **registrované callbacky** nemusí být exportovány. Namísto toho se přímo předá ukazatel na tuto funkci do XPLM.

Callbacky lze registrovat jak z vyžadovaných tak registrovaných callbacků. Například lze registrovat callback, který se vykoná s pětiminutovým zpožděním po inicializaci zásuvného modulu. Po těchto pěti minutách se při vykonávání callbacku vytvoří okno a registruje callback pro obsluhu kliku myší na vytvořené okno.

3.1.5 Signatura zásuvného modulu

Každý zásuvný modul má unikátní signaturu, která je předána do XPLM během první inicializace zásuvného modulu, viz výpis 3.1.

Výpis 3.1: Předání signatury do XPLM

```
PLUGIN_API int XPluginStart(char *outName,
                            char *outSig,
                            char *outDesc)
{
  strcpy(outSig, "VUT_FEKT_Brno.comm_mod_plgn");
  ...
```

Pomocí této signatury jsou jednotlivé zásuvné moduly jednoznačně identifikovány od ostatních modulů. Signatura je tvořena libovolným řetězcem znaků, ale pro minimalizaci možnosti volby identické signatury s jiným existujícím zásuvným modulem je doporučeno, aby signatura začínala názvem organizace zodpovědné za vytvoření

modulu. Příkladem takového řetězce může být signatura modulu vytvořeného pro měření výkonu motoru jako součást balíčku diagnostiky letu:
'`VUT_FEKT_Brno.diagnostics.engine_performance_meter`'.

Unikátnost a forma jednotlivých signatur vytvořených zásuvných modulů v rámci jedné organizace je již na zodpovědnosti a volbě každého vývojáře, popřípadě konvenci organizace.

3.1.6 Povolování zásuvných modulů

Zásuvný modul může být povolen či zakázán. Callbacky jsou volány pouze pro povolené zásuvné moduly. Zakázání umožňuje uživateli během probíhající simulace vypnout ty zásuvné moduly, u kterých si uživatel nepřeje, aby běžely nebo které se jeví jako problematické.

Všechny zásuvné moduly jsou během načítání zakázané a v okamžiku, kdy je simulace připravená ke spuštění jsou zásuvné moduly jeden po druhém povolovány. V okamžiku kdy uživatel ukončí simulaci jsou všechny zásuvné moduly zakázány a následně uvolněny. Pokud uživatel při spuštění simulátoru X-Plane podrží klávesu *Shift*, spustí se všechny zásuvné moduly v zakázaném stavu a tomto stavu setrvají, dokud je uživatel manuálně nepovolí.

V případě povolení či zakázání přijme zásuvný modul callback. Zásuvný modul nutně nemusí na tento callback reagovat. Pokud však alokuje další zdroje, je vhodné, aby tyto zdroje byly v případě zakázání modulu uvolněny. Jako příklad lze uvést odpojení od serveru v počítačové sítí během zakázání zásuvného modulu a následná obnova spojení se serverem při povolení.

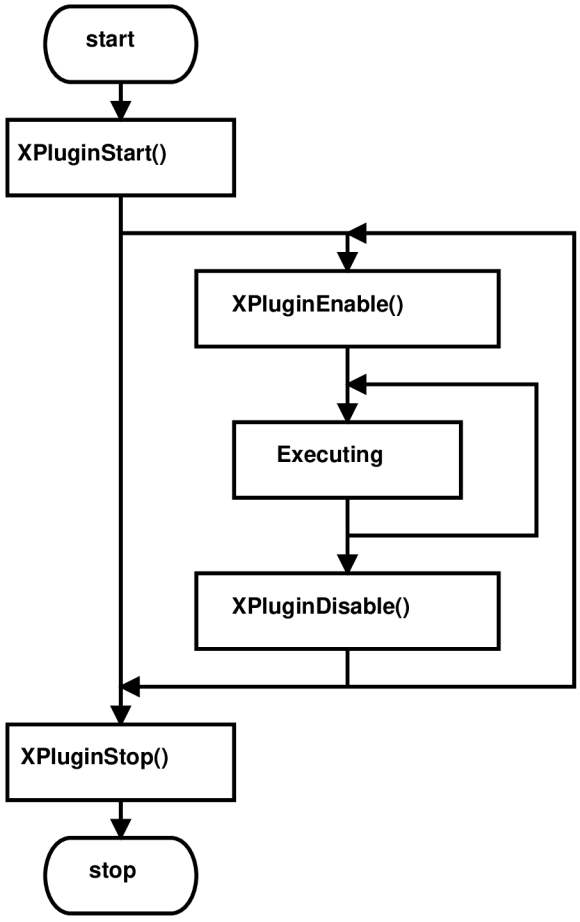
3.1.7 Provoz zásuvných modulů

Typický provoz zásuvných modulů je znázorněn na obrázku 3.2:

- Po startu simulátoru X-Plane 10 jsou postupně zavolány obslužné procedury `XPluginStart()` pro všechny zásuvné moduly umístěné v adresáři instalace simulátoru X-Plane 10 pod relativní adresou: `X-Plane 10\Resources\plugins`. Během této procedury je provedena potřebná inicializace samotného zásuvného modulu, registrace požadovaných callbacků a alokace trvalých zdrojů.
- Ihned po načtení všech zásuvných modulů jsou postupně všechny moduly povoleny. Těsně před povolení zásuvného modulu je vykonána jeho procedura `XPluginEnable()`. V této obslužné proceduře je vhodné alokovat dočasné zdroje.
- Po úspěšném povolení zásuvných modulů jsou tyto moduly ve stavu běžného pracovního chodu, viz blok **Executing** na obrázku 3.2. V každém simulačním cyklu jsou postupně obslouženy všechny povolené moduly a vykonány

relevantní registrované callbacky.

- Při ukončení simulátoru X-Plane nebo manuálně kdykoli během letu je zásuvný modul zakázán. Těsně před zakázáním modulu je obsloužena procedura `XPluginDisable()`, ve které by měly být uvolněny dočasné zdroje.
- Poté co je zásuvný modul v zakázaném stavu, může být opět povolen nebo v tomto stavu setrvat do ukončení simulátoru X-Plane. Do tohoto stavu se zásuvný modul může dostat ihned po spuštění simulátoru za předpokladu, že jsou tyto moduly explicitně zakázány.
- Před ukončením simulátoru X-Plane jsou vykonány obslužné procedury `XPluginStop()`, ve kterých by měly být uvolněny veškeré alokované zdroje a registrované callbacky.



Obr. 3.2: Provoz zásuvného modulu

3.2 Vyžadované callbacky

Po zásuvných modulech simulátoru X-Plane je vyžadováno 5 callbacků, které musí být implementovány: `XPluginStart()`, `XPluginStop()`, `XPluginEnable()`,

`XPluginDisable()` a `XPluginReceiveMessage()`, které umožní základní provoz zásuvných modulů v rámci simulátoru X-Plane. Podrobnější popis jednotlivých callbacků je uveden v následujících podkapitolách.

3.2.1 XPluginStart

Funkce `XPluginStart()` je zavolána ihned po načtení zásuvného modulu. Funkce provádí nutné inicializace zásuvného modulu. To zahrnuje vytváření uživatelského prostředí, registrace dalších callbacků, alokace zdrojů atp. Dále by tato funkce měla kopírovat řetězce znaků ukončené znakem '`0h`' do tří výstupních bufferů. Deklarace této funkce je následující:

```
PLUGIN_API int XPluginStart(char *outName,
                            char *outSignature,
                            char *outDescription);
```

kde jednotlivé argumenty mají následující význam:

- `outName` – Ukazatel na vyrovnávací paměť (buffer). Do tohoto bufferu je zapsán název zásuvného modulu ve formátu čitelném pro člověka.
- `outSignature` – Ukazatel na vyrovnávací paměť. Do tohoto bufferu je zapsána **Signatura** zásuvného modulu, viz kapitola 3.1.5.
- `outDescription` – Ukazatel na vyrovnávací paměť. Do tohoto bufferu je zapsán popis zásuvného modulu ve formátu čitelném pro člověka. V případě neúspěšného načtení modulu je do tohoto bufferu zapsána chybová zpráva.

Návratová hodnota je **1** pokud byl zásuvný modul úspěšně načten nebo **0** v případě neúspěchu. Pokud je návratová hodnota 0, zásuvný modul bude okamžitě uvolněn.

3.2.2 XPluginStop

Funkce `XPluginStop()` je zavolána těsně před uvolněním zásuvného modulu. Modul bude před voláním této funkce zakázán pomocí funkce `XPluginDisable()`, viz kapitola 3.2.4. Funkce by měla uvolnit registrované callbacky, vytvořené objekty a alokované zdroje. Deklarace této funkce je následující:

```
PLUGIN_API void XPluginStop(void);
```

3.2.3 XPluginEnable

Funkce `XPluginEnable()` je zavolána těsně před povolením zásuvného modulu. Zásuvný modul je povolen poté co byly všechny moduly načteny za předpokladu, že není příslušný zásuvný modul zakázán v nastavení X-Plane simulátoru. Tato

funkce je také volána v případě manuálního povolení zásuvného modulu uživatelem. Funkce `XPluginEnable()` nebude opět zavolána bez předchozího volání funkce `XPluginDisable()`, viz kapitola 3.2.4. V rámci této funkce je vhodné alokovat některé dočasné zdroje jako vytvoření a spuštění vláken nebo navázání síťové komunikace. Deklarace této funkce je následující:

```
PLUGIN_API void XPluginEnable(void);
```

Funkce nemá žádné argumenty ani návratovou hodnotu.

3.2.4 XPluginDisable

Funkce `XPluginDisable()` je zavolána těsně před zakázáním zásuvného modulu. Zakázaný modul nemůže přijmout žádná volání, uživatelské rozhraní modulu bude skryto. Tato funkce je také volána před uvolněním zásuvného modulu ještě před voláním funkce `XPluginStop()`, viz kapitola 3.2.2. V rámci této funkce by měla uvolnit významné zdroje a uvést zásuvný modul do stavu, ve kterém je očekáváno delší setrvání bez interakce se simulátorem X-Plane. Deklarace této funkce je následující:

```
PLUGIN_API void XPluginDisable(void);
```

Funkce nemá žádné argumenty ani návratovou hodnotu.

3.2.5 XPluginReceiveMessage

Funkce `XPluginReceiveMessage()` je zavolána z XPLM pokud je příslušnému zásuvnému modulu odeslána zpráva. Pomocí této funkce jsou přijímány jak soukromé zprávy adresované konkrétnímu modulu, tak plošné (broadcast) zprávy určené všem modulům. Deklarace této funkce je následující:

```
PLUGIN_API void XPluginReceiveMessage(XPLMPluginID inFrom,
                                      int inMessage,
                                      void *inParam);
```

kde jednotlivé argumenty mají následující význam:

- `inFrom` – Identifikátor zásuvného modulu, který odeslal zprávu.
- `inMessage` – Cele číslo identifikující odeslanou zprávu.
- `inParam` – Ukazatel na konkrétní data zprávy.

Funkce nemá žádnou návratovou hodnotu.

3.3 Registrované callbacky

Pro mnohé účely je nutné registrovat další callbacky. Typické užití těchto callbacků je například:

- Registrovaný callback pro obsluhu výběru položky menu zásuvného modulu pomocí knihovny `XPLMMenus.h`, viz [5].
- Registrovaný callback pro obsluhu kliku myší na grafické okno zásuvného modulu nebo callback pro samotné vykreslení grafického okna pomocí knihovny `XPLMDisplay.h`, viz [6].
- Registrovaný callback pro obsluhu přerušení od časovače pomocí knihovny `XPLMProcessing.h`, viz [7].

Registrovat je možné velké množství typů callbacků a jejich rozbor by byl nad rámec této práce. Podrobnější informace o možnostech registrace callbacků jsou k nalezení v oficiální dokumentaci, viz [4]. V této práci je uveden pouze jeden příklad registrační funkce `XPLMCreateWindow()`, která byla užita při implementaci komunikačního modulu, viz kapitola 3.3.1.

3.3.1 XPLMCreateWindow

Funkce `XPLMCreateWindow()` vytváří nové grafické okno, u kterého lze volit jeho umístění na obrazovce simulátoru X-Plane, jeho viditelnost a příslušné obsloužení vyvolaných událostí. Deklarace této funkce je následující:

```
XPLM_API XPLMWindowID XPLMCreateWindow(int inLeft,
                                       int inTop,
                                       int inRight,
                                       int inBottom,
                                       int inIsVisible,
                            XPLMDrawWindow_f inDrawCallback,
                             XPLMHandleKey_f inKeyCallback,
                    XPLMHandleMouseClick_f inMouseCallback,
                                      void *inRefcon);
```

kde jednotlivé argumenty mají následující význam:

- `int inLeft` – Souřadnice levé hrany grafického okna v pixelech.
- `int inTop` – Souřadnice horní hrany grafického okna v pixelech.
- `int inRight` – Souřadnice pravé hrany grafického okna v pixelech.
- `int inBottom` – Souřadnice spodní hrany grafického okna v pixelech.
- `int inIsVisible` – Viditelnost grafického okna: '`1`' – okno je viditelné, '`0`' – okno je neviditelné.

- `XPLMDrawWindow_f inDrawCallback` – Funkce pro obsluhu události vykreslení grafického okna, která umožňuje další manipulaci s oknem jako vypisování textu do okna a jiné. Definice datového typu ukazatele na funkci `XPLMDrawWindow_f` je uvedena v kapitole 3.3.3.
- `XPLMHandleKey_f inKeyCallback` – Funkce pro obsluhu události na stisk klávesy v grafickém okně. Definice datového typu ukazatele na funkci `XPLMHandleKey_f` je uvedena v kapitole 3.3.4.
- `XPLMHandleMouseClick_f inMouseCallback` – Funkce pro obsluhu události na klik myší do grafického okna. Definice datového typu ukazatele na funkci `XPLMHandleMouseClick_f` je uvedena v kapitole 3.3.5.
- `void *inRefcon` – Referenční konstanta pro průchod dat do callbacku. Podrobnější informace jsou uvedeny v [8].

Návratová hodnota je identifikátor vytvořeného grafického okna. Definice viz kapitola 3.3.2.

Argumenty `inLeft`, `inTop`, `inRight` a `inBottom` udávající souřadnice vykreslovaného okna v pixelech mají počátek souřadnic umístěný v levém spodním rohu okna simulátoru X-Plane.

3.3.2 XPLMWindowID

Datový typ `XPLMWindowID` je užit pro identifikátory grafických oken a je implementován jako obyčejný ukazatel bez datového typu. Definice datového typu [9]:

```
typedef void *XPLMWindowID;
```

3.3.3 XPLMDrawWindow f

Ukazatel na funkci datového typu `XPLMDrawWindow_f` je užit jako reference obsluhy události vyvolané vykreslením grafického okna. V rámci této funkce je možné používat API pro interakci se simulátorem X-Plane. Definice datového typu ukazatele na funkci [10]:

```
typedef void (*XPLMDrawWindow_f) (
                                    XPLMWindowID inWindowID,
                                            void *inRefcon);
```

Z definice je patrné, že ukazatel tohoto datového typu je zamýšlen pro použití na funkce s následující deklarací:

```
void functionName(XPLMWindowID inWindowID,
                            void *inRefcon);
```

kde jednotlivé argumenty mají následující význam:

- `XPLMWindowID inWindowID` – Identifikátor grafického okna.
- `void *inRefcon` – Referenční konstanta pro průchod dat do callbacku. Podrobnější informace jsou uvedeny v [8].

Funkce nemá žádnou návratovou hodnotu.

3.3.4 XPLMHandleKey f

Ukazatel na funkci datového typu `XPLMHandleKey_f` je užit jako reference obsluhy události vyvolané stiskem klávesy v grafickém okně nebo ztrátou fokusu klávesnice. Pokud je ztracen fokus klávesnice, argument `losingFocus` je nastaven na hodnotu '1'. V opačné případě byla stisknuta klávesa a argument `inKey` je nastaven na hodnotu stisknutého znaku. Definice datového typu ukazatele na funkci [11]:

```
typedef void (*XPLMHandleKey_f)(
                                XPLMWindowID inWindowID,
                                        char inKey,
                                XPLMKeyFlags inFlags,
                                        char inVirtualKey,
                                        void *inRefcon,
                                         int losingFocus);
```

Z definice je patrné, že ukazatel tohoto datového typu je zamýšlen pro použití na funkce s následující deklarací:

```
void functionName(XPLMWindowID inWindowID,
                          char inKey,
                  XPLMKeyFlags inFlags,
                          char inVirtualKey,
                          void *inRefcon,
                           int losingFocus);
```

kde jednotlivé argumenty mají následující význam:

- `XPLMWindowID inWindowID` – Identifikátor grafického okna.
- `char inKey` – Hodnota znaku stisknuté klávesy.
- `XPLMKeyFlags inFlags` – Příznak stisknuté klávesy. Podrobnější informace jsou uvedeny v [13].
- `char inVirtualKey` – Hodnota virtuálně stisknuté klávesy, pomocí API simulátoru.
- `void *inRefcon` – Referenční konstanta pro průchod dat do callbacku. Podrobnější informace jsou uvedeny v [8].

- `int losingFocus` – Příznak ztráty fokusu klávesnice. Pokud je ztracen fokus klávesnice argument je nastaven na hodnotu '1'.

Funkce nemá žádnou návratovou hodnotu.

3.3.5 XPLMHandleMouseClick f

Ukazatel na funkci datového typu `XPLMHandleMouseClick_f` je užit jako reference obsluhy události vyvolané klikem myší do grafického okna. Tato událost je vyvolána jak při stisku, tak při uvolnění tlačítka myši. Definice datového typu ukazatele na funkci [12]:

```
typedef int (*XPLMHandleMouseClick_f)(
                                      XPLMWindowID inWindowID,
                                               int x,
                                               int y,
                                    XPLMMouseStatus inMouse,
                                              void *inRefcon);
```

Z definice je patrné, že ukazatel tohoto datového typu je zamýšlen pro použití na funkce s následující deklarací:

```
int functionName(XPLMWindowID inWindowID,
                             int x,
                             int y,
                  XPLMMouseStatus inMouse,
                            void *inRefcon);
```

kde jednotlivé argumenty mají následující význam:

- `XPLMWindowID inWindowID` – Identifikátor grafického okna.
- `int x` – Souřadnice **x** kliku myši.
- `int y` – Souřadnice **y** kliku myši.
- `XPLMMouseStatus inMouse` – Stav stisku tlačítka myši. Podrobnější informace jsou uvedeny v [14].
- `void *inRefcon` – Referenční konstanta pro průchod dat do callbacku. Podrobnější informace jsou uvedeny v [8].

Návratová hodnota by měla být nastavena na hodnotu '0' pokud klik myší byl zpracován nebo na hodnotu '1' v případě, že obsluha nebyla vykonána.

3.4 Čtení a zápis dat do simulátoru

API datového přístupu umožňuje sdílet data zásuvného modulu se simulátorem X-Plane nebo ostatními moduly. *Data Access API*, viz [15], je hlavní nástroj pro přístup s datům nebo subsystémům simulátoru.

3.4.1 Datová reference

Datová reference je interní identifikátor proměnné simulátoru X-Plane nebo jiného zásuvného modulu. Veškerá komunikace se simulátorem probíhá čtením nebo zápisem dat do datových referencí.

Datové reference je zavedená kvůli izolaci zásuvných modulů od interních dat simulátoru X-Plane. Pokud by v budoucnu došlo ke změně interních proměnných simulátoru, datové reference můžou odkazovat na nové proměnné. Tyto datové jsou unikátní pouze v rámci jednoho běhu simulátoru a pří příštím spuštění mohou nabývat jiných hodnot. Z tohoto důvodu není vhodné přímo zakódovat datové reference do zásuvného modulu. Namísto toho se používají názvy datových referencí, pomocí kterých je skutečná reference nalezena až během inicializace zásuvného modulu při spuštění simulátoru X-Plane.

Název datové reference je globálně unikátní řetězec znaků, který je neměnný. Názvy datových referencí jsou hierarchicky uspořádány podle kategorií. Příkladem názvu datové reference může bý zeměpisná šířka polohy letounu:
`sim/flightmodel/position/latitude`
Tento název datové reference jednoznačně identifikuje příslušný parametr simulátoru a měl by zůstat nezměněn u nových verzí simulátoru X-Plane nebo *X-Plane Plugin SDK*.

Data Access API zprostředkovává přístup k objemnému množství datových referencí jejichž úplný seznam seřazený podle názvů datových referencí je uvede ve zdroji [16].

3.4.2 XPLMFindDataRef

Funkce `XPLMFindDataRef()` slouží pro získání datové reference pro požadovaný název datové reference. Deklarace této funkce je následující:

```
XPLM_API XPLMDataRef XPLMFindDataRef(const char *inDataRefName);
```

kde argument `const char *inDataRefName` je řetězec obsahující název datové reference. Funkce vrací datovou referenci pro příslušný název (ukazatel typu '`void *`'). Pokud nebyla datová reference nalezena, funkce vrací '`NULL`'.

3.4.3 XPLMGetData

Funkce `XPLMGetDatai()`, `XPLMGetDataf()` a `XPLMGetDatad()` slouží pro čtení požadovaných parametrů simulátoru X-Plane na základě datové reference a datového typu. Deklarace těchto funkcí je následující:

```
XPLM_API int XPLMGetDatai(XPLMDataRef inDataRef);
XPLM_API float XPLMGetDataf(XPLMDataRef inDataRef);
XPLM_API double XPLMGetDatad(XPLMDataRef inDataRef);
```

kde argument všech funkcí je datová reference na požadovaný parametr simulátoru a návratová hodnota je hodnota požadovaného parametru příslušného datového typu.

3.4.4 XPLMSetData

Funkce `XPLMSetDatai()`, `XPLMSetDataf()` a `XPLMSetDatad()` slouží pro zápis požadovaných parametrů simulátoru X-Plane na základě datové reference a datového typu. Deklarace těchto funkcí je následující:

```
XPLM_API void XPLMSetDatai(XPLMDataRef inDataRef, int inValue);
XPLM_API void XPLMSetDataf(XPLMDataRef inDataRef, float inValue);
XPLM_API void XPLMSetDatad(XPLMDataRef inDataRef, double inValue);
```

kde argument `inDataRef` všech funkcí je datová reference na požadovaný parametr simulátoru a argument `inValue` je požadovaná hodnota příslušného datového typu. Funkce nemá návratovou hodnotu a v případě, že požadovaný parametr není zapisovatelný nebo datová reference je nevalidní funkce nevykonává žádnou operaci.

4 NÁVRH A IMPLEMENTACE MODULU

Na základě kladených požadavků zadavatelem a analýzy realizovatelnosti těchto požadavků byl proveden návrh a implementace komunikačního modulu pro simulátor X-Plane pomocí *X-Plane Plugin SDK*.

4.1 Definice základních vlastností

Pro potřeby návrhu komunikačního modulu byly na základě požadavků kladených zadavatelem, viz kapitola 2.1, definovány tyto základní vlastnosti zásuvného modulu:

- Komunikační modul bude umožňovat čtení a zápis dat do simulátoru X-Plane.
- Komunikační modul bude implementovat síťový server se standardním protokolem pro komunikaci s klienty.
- Síťová komunikace s klienty bude probíhat textovou formou.
- Data odeslaná komunikačním modulem budou splňovat požadovaný formát dle přílohy A.
- Komunikační modul bude číst a zapisovat parametry uvedené v tabulce 2.1.

4.2 Identifikace komunikačního modulu

Pro implementaci zásuvného modulu, který je výsledným produktem této práce byl zvolen název **Communication module plugin** a dle požadavků kapitoly 3.1.5 byl jako identifikační signatura zásuvného modulu X-Plane zvolen řetězec
'`VUT_FEKT_Brno.comm_mod_plgn`'.
Umístění implementovaného zásuvného modulu bylo dle požadavků kapitoly 3.1.7 zvoleno do adresáře:
'`.\Resources\plugins\comm_mod_plgn\`'.
V tomto adresáři nebo v jeho podadresářích budou uloženy všechny součásti implementovaného komunikačního modulu.

4.3 Síťový protokol

Aby byly co nejlépe splněny požadavky zadavatele kladené na komunikační modul, zásuvný modul implementuje síťový server, který bude přijímat požadavky od klientů v počítačové síti a odesílat příslušná data. Síťový server je implementován pomocí knihoven *Winsock*, které se snaží návrh síťových aplikací přiblížit intuitivní metodě pomocí *socketů*, viz [20].

Po konzultaci se zadavatelem bylo rozhodnuto, že komunikace se serverem bude probíhat formou **dotaz-odpověď**. Při návrhu byly uvažovány pouze nejpoužívanější síťové komunikační protokoly *TCP* a *UDP*, které jsou vhodné pro tento typ aplikace. Navíc oba tyto protokoly je možné jednoduše implementovat v prostředí *MATLAB*, které bude použito k následnému zpracování a analýze dat získaných ze simulace. Protože komunikačnímu modelu **dotaz-odpověď** lépe vyhovuje *TCP*, byl tento protokol užit při implementaci komunikačního modulu. Tato volba přináší výhody v jednodušší práci s daty přenášenými po síti, neboť tato aplikační vrstva zajistí, že data dorazí v pořádku a ve správném pořadí (pokud nedojde k výpadku spojení). Nevýhoda této volby je vyšší režie spojená právě s touto bezpečnější potvrzovanou komunikací.

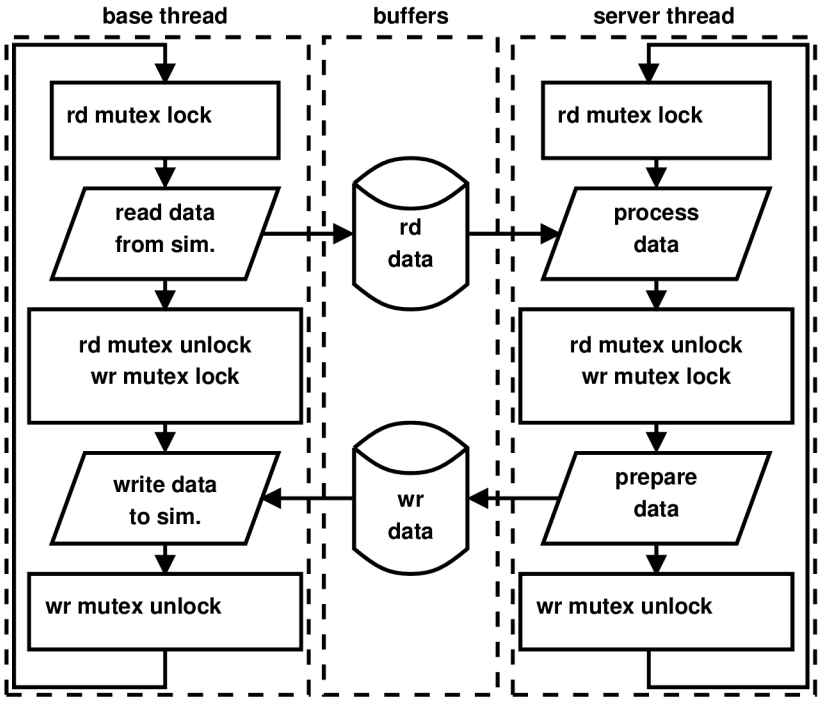
4.4 Čtení a zápis dat do simulátoru

Vzhledem ke skutečnosti, že během vykonávání kódu zásuvných modulů je samotná simulace pozastavena, viz kapitola 3, není vhodné, aby vykonávání implementovaného komunikačního modulu trvalo příliš dlouho z důvodu negativního ovlivnění plynulosti chodu samotné simulace letu.

Z výše uvedeného důvodu bude vytvořeno nové vlákno, které bude vykonáváno nezávisle na letovém cyklu simulátoru X-Plane. V původním vlákně implementovaného zásuvného modulu budou data ze simulátoru pouze načtena a uložena do mezipaměti, aby vykonávání zásuvného modulu trvalo nejkratší možnou dobu. Implementace síťového serveru a zpracování dat bude probíhat v nově vytvořeném vlákně nezávisle na letovém cyklu simulátoru X-Plane. Obdobně budou i data určená pro zápis do simulátoru přijatá serverem zpracována ve vytvořeném vlákně a výsledná data uložena do mezipaměti, která bude v původním vlákně zásuvného modulu pouze přepsána do simulátoru X-Plane. Tímto způsobem bude minimalizován vliv implementovaného zásuvného modulu na plynulost běhu simulace.

Aby nedošlo k nekonzistentnosti dat mezi oběma vlákny, je nutné použít některou z metod, která zajistí výlučný přístup do sdíleného paměťového prostoru. Zároveň je nutné zachovat co nejrychlejší chod zásuvného modulu. Tomuto požadavku vyhovuje použití globálních proměnných jako mezipaměť, do které mají přímý přístup obě vykonávaná vlákna. Tyto globální proměnné budou chráněny **mutexy**, viz [21], které zajistí výlučný přístup do paměti.

Princip přenosu dat mezi vlákny je znázorněn na obrázku 4.1, kde **base thread** je vlákno simulátoru X-Plane, ve kterém se cyklicky vykonává kód implementovaného zásuvného modulu. Toto vlákno pomocí mutexu uzamkne buffer pro čtení dat a následně přepíše buffer aktuálními daty ze simulátoru. Po ukončení zápisu dat je buffer opět pomocí mutexu uvolněn, aby k němu mělo přístup serverové vlákno.



Obr. 4.1: Diagram výměny dat mezi vlákny

Stejný postup se opakuje u dat určených k zápisu do simulátoru s tím rozdílem, že po uzamčení mutexem jsou data z bufferu čtena. Tento proces se vykoná s každým průchodem letového cyklu.

Obdobným způsobem pracuje se sdílenou pamětí serverové vlákno, viz blok **server thread** na obrázku 4.1. Serverové vlákno musí použít příslušné mutexy (stejné jako používá vlákno simulátoru) pro zajištění výlučného přístupu do sdílené paměti.

4.5 Konfigurace komunikačního modulu

Vzhledem k tomu, že komunikační modul implementuje vlastní síťový server, je nutné tento server také náležitým způsobem nakonfigurovat. Nastavení zásuvného modulu je možné realizovat mnoha způsoby. Pokud však budou uvažovány pouze metody, které si po vypnutí simulátoru X-Plane a jeho opětovném zapnutí zachovají nastavené parametry komunikačního modulu, redukuje se výběr pouze na metody, které při startu zásuvného modulu načtou konfiguraci z externího zdroje.

Jako nejjednodušší metoda se nabízí konfigurační soubor umístěný v adresáři se samotným zásuvným modulem. Dále, aby nebylo nutné nastavení provádět v rámci implementovaného komunikačního modulu pomocí nějakého konfiguračního okna, byl zvolen obyčejný textový soubor jednoduše čitelný uživatelem pomocí libovolného textového editoru.

Po konzultaci vlastností požadovaného komunikačního modulu se zadávající stranou byly zvoleny následující parametry konfigurace:

- *TCP* port, na kterém bude komunikační modul naslouchat.
- Maximální počet současně obsluhovaných klientů komunikačním modulem.
- Příznak viditelnosti okna komunikačního modulu.

4.5.1 Konfigurační soubor

Konfigurační soubor komunikačního modulu je umístěný na relativní adrese: '`.\Resources\plugins\comm_mod_plgn\comm_mod_plgn.conf`'.
Tento konfigurační soubor bude zpracováván zásuvným modulem implementovaným v programovacím jazyce *ANSI C* a proto bylo pro co nejjednodušší zpracování zvoleno *ASCII* kódování konfiguračního souboru.

Implementovaný zásuvný modul akceptuje parametry konfiguračního souboru v následujícím tvaru: '`<parameterName>=<value>`', kde každý parametr je uveden na novém řádu. Parametry konfiguračního souboru nejsou nijak verifikovány. Pokud se komunikačnímu modulu nepodaří parametr načíst správně z důvodu chybného zápisu, bude nastavena výchozí hodnota parametru. Výchozí hodnota parametru bude nastavena také v případě, že příslušný parametr není v konfiguračním souboru uveden.

Implementovaný zásuvný modul akceptuje následující parametry konfiguračního souboru:

- `SOCKET_BACKLOG` – Parametr udávající maximální počet současně obsluhovaných klientů komunikačním modulem. Výchozí a doporučená maximální hodnota: '`5`'. Tento parametr je určen pro budoucí verze komunikačního modulu. Aktuálně komunikační modul naváže spojení pouze s jedním klientem.
- `TCP_PORT` – Port na kterém bude *TCP* server naslouchat. Výchozí hodnota zvolená zadavatelem: '`50000`'.
- `WINDOW_VISIBLE` – Příznak viditelnosti okna komunikačního modulu: hodnota '`1`' pro viditelné okno a hodnota '`0`' pro skryté okno. Výchozí hodnota: '`1`' – viditelné okno.

Konfigurační soubor bude načten během vykonávání obslužné procedury `XPluginStart()` při inicializaci komunikačního modulu po spuštění simulátoru X-Plane. Příklad konfigurace, která nastaví *TCP* port 50032, jednoho síťového klienta a skrytí grafického okna komunikačního modulu je uveden ve výpisu 4.1.

Výpis 4.1: Příklad konfiguračního souboru

```
TCP_PORT=50032
SOCKET_BACKLOG=1
WINDOW_VISIBLE=0
```

4.6 Data pro čtení a zápis

Na základě souboru požadovaných parametrů pro čtení a zápis uvedených v tabulce 2.1 byly nalezeny příslušné parametry simulátoru X-Plane a jejich názvy referencí dostupné přes *Data Access API*. Tyto názvy datových referencí s příslušnými parametry (data požadovaná zadavatelem) jsou vedeny v tabulce 4.1, kde položka **w** udává zda-li je parametr zapisovatelný.

Pro požadovaný parametr '`_Mach,ratio`' se nepodařilo najít žádnou odpovídající datovou referenci a proto tento parametr nebude v komunikačním modulu implementován. Parametr '`_real,_time`' byla nalezena alternativa v podobě parametru '`_totl,_time`'.

Dále je některým parametrům z důvodu různé implementace nutné převádět jednotky. Jedná se o tyto parametry:

- `__alt,ftmsl` – výška letounu udaná ve stopách nad střední hladinou moře je interně implementována v metrech. Pro převod je použit následující vztah [22]:

$$alt_{\mathrm{f}} = \frac{3937 \cdot alt_{\mathrm{m}}}{1200}. \tag{4.1}$$

- `Vtrue,_ktas` – rychlost letounu udaná v uzlech je interně implementována v metrech za sekundu. Pro převod je použit následující vztah [23]:

$$v_{\mathrm{kn}} = \frac{900 \cdot v_{\mathrm{m/s}}}{463}. \tag{4.2}$$

4.6.1 Poloha letounu

Při konzultaci se zadavatel bylo zjištěno, že by bylo třeba pomocí komunikačního modulu měnit polohu letounu zadáním požadované zeměpisné šířky, délky a nadmořské výšky, což podle tabulky 4.1 není přímo možné.

Pro potřeby změny polohy letounu byly nalezeny tyto parametry simulátoru X-Plane:

- `sim/flightmodel/position/local_x`
- `sim/flightmodel/position/local_y`
- `sim/flightmodel/position/local_z`

Tab. 4.1: Datové reference přiřazené požadovaným parametrům

n	Parametr	Datová reference X-Pane API	w
0	__alt,ftmsl	sim/flightmodel/position/elevation	
1	AMprs,_inHG	sim/weather/barometer_current_inhg	
2	AMtmp,_degC	sim/cockpit2/temperature/outside_air_temp_degc	
3	_beta,__deg	sim/flightmodel/position/beta	
4	_elev,yoke1	sim/cockpit2/controls/yoke_pitch_ratio	✓
5	__lat,__deg	sim/flightmodel/position/latitude	
6	__lon,__deg	sim/flightmodel/position/longitude	
7	hding,__mag	sim/flightmodel/position/mag_psi	
8	hding,_true	sim/flightmodel/position/true_psi	
9	_Mach,ratio	-	
10	____P,rad/s	sim/flightmodel/position/Prad	✓
11	pitch,__deg	sim/flightmodel/position/true_theta	
12	____Q,rad/s	sim/flightmodel/position/Qrad	✓
13	____R,rad/s	sim/flightmodel/position/Rrad	✓
14	_totl,_time	sim/time/total_running_time_sec	✓
15	_roll,__deg	sim/flightmodel/position/true_phi	
16	ruddr,yoke1	sim/cockpit2/controls/yoke_heading_ratio	✓
17	_Vind,_keas	sim/flightmodel/position/indicated_airspeed2	✓
18	_Vind,_kias	sim/flightmodel/position/indicated_airspeed	✓
19	Vtrue,_ktas	sim/flightmodel/position/true_airspeed	
20	__VVI,__fpm	sim/flightmodel/position/vh_ind_fpm	✓

které reprezentují nativní souřadnice simulačního modelu. Tyto parametry jsou zapisovatelné a lze je tedy ke změně polohy využít, ale je nutné nejdříve konvertovat požadované souřadnice ve formě zeměpisné šířky, délky a nadmořské výšky do nativních souřadnic x, y a z. K tomuto účelu nabízí *API* simulátoru funkci `XPLMWorldToLocal()` [24]. Deklarace této funkce je následující:

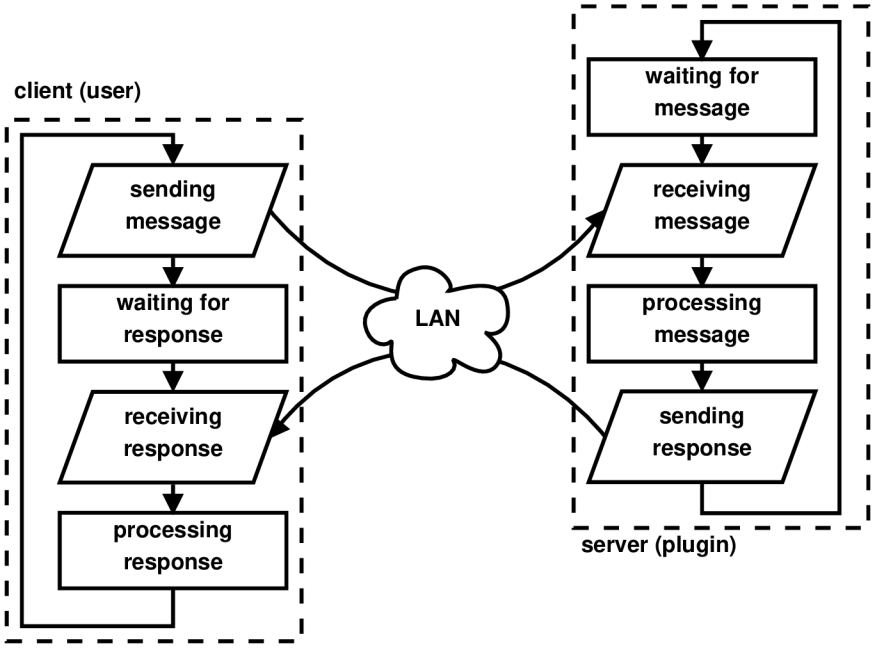
```
XPLM_XPLM_API void XPLMWorldToLocal(double inLatitude,
                                    double inLongitude,
                                    double inAltitude,
                                    double *outX,
                                    double *outY,
                                    double *outZ);
```

kde argumenty `inLatitude`, `inLongitude` a `inAltitude` jsou souřadnice zeměpisné šířky, délky ve stupních a nadmořské výšky v metrech typu `double`. Argumenty `*outX`, `*outY` a `*outZ` jsou ukazatele na proměnné typu `double` v nativních souřadnicích simulátoru X-Plane. Funkce nemá žádnou návratovou hodnotu.

Při požadavku uživatele na zápis libovolné souřadnice jsou všechny pomocí funkce `XPLMWorldToLocal()` zkonvertovány do nativních a ty následně zapsány.

4.7 Komunikace klienta se serverem

Dle požadavků zadavatele, bude zásuvný modul po síti komunikovat textovou formou. Princip komunikace mezi uživatelem a zásuvným modulem je znázorněn na obrázku 4.2. Pro potřeby čtení a zápisu dat do simulátoru X-Plane byly implemen-

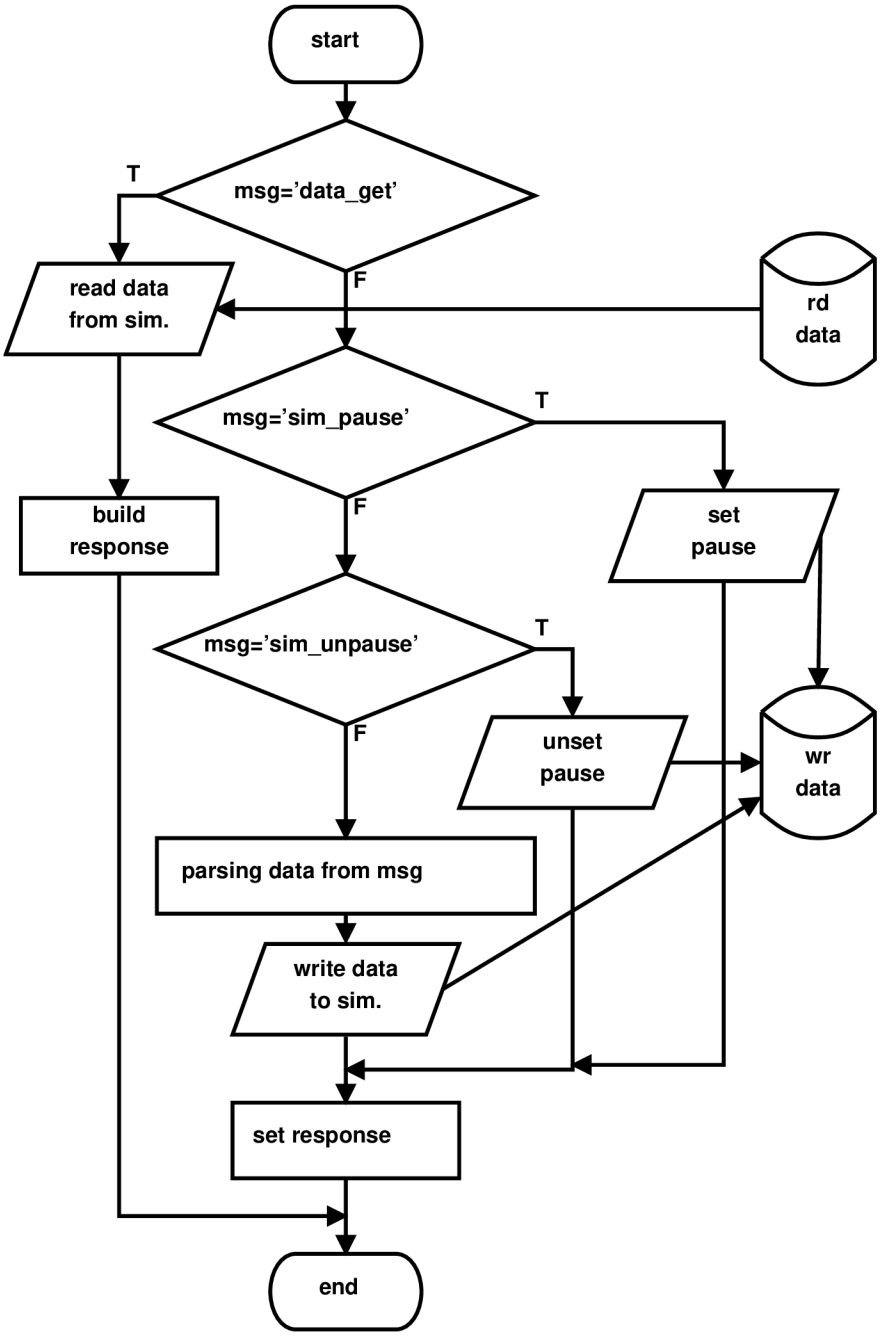


Obr. 4.2: Diagram komunikace uživatele se zásuvným modulem

továny následující zprávy:

- `data_get` – Požadavek na aktuální data simulátoru.
- `sim_pause` – Příkaz pro pozastavení simulace.
- `sim_unpause` – Příkaz pro opětovné spuštění simulace.
- **Cokoli mimo předchozí** – Příkaz pro zápis parametrů simulace. Obsah této zprávy závisí na konkrétní situaci, viz kapitola 4.7.2.

Tyto zprávy jsou po přijetí komunikačním modulem zpracovány dle diagramu znázorněném na obrázku 4.3.



Obr. 4.3: Diagram zpracování přijaté zprávy zásuvným modulem

4.7.1 Čtení dat

Pro získání aktuálních parametrů simulátoru X-Plane slouží zpráva '`data_get`'. Pokud implementovaný server obdrží tuto zprávu, odešle všechna požadovaná data uvedená v tabulce 4.1 (kromě parametru '`_Mach,ratio`') klientovi. Odeslaná data splňují přesně definovaný tvar, viz kapitola 2.1. Zkrácený příklad komunikace pro zprávu '`data_get`' je uveden ve výpisu 4.2.

Výpis 4.2: Příklad zprávy `data_get`

```
data_get
_totl,_time |   pause,_bool |    __lat,__deg |    __lon,__d
  487.13950 |              1 |       48.81821 |       16.076
```

Odeslaná data obsahují navíc parametr '`pause,_bool`', oproti souboru požadovaných dat, který indikuje pozastavení simulace, viz kapitola 4.7.3.

4.7.2 Zápis dat

Pro zápis dat do simulátoru X-Plane slouží zpráva začínající libovolným řetězcem znaků nekorespondujícím s ostatními zprávami uvedenými v kapitole 4.7. Zpráva pro zápis dat do simulátoru má následující formát:

```
<parameterName0>:<value0>;<parameterName1>:<value1>;
```

kde '`parameterName`' je název parametru dle tabulky požadovaných dat a '`value`' je požadovaná hodnota parametru. Počet parametrů je variabilní a závisí na konkrétním požadavku k zápisu. Po úspěšném přijetí požadavku k zápisu dat je odeslána potvrzovací zpráva '`data_received`'. Příklad zápisu nové polohy letounu je uveden ve výpisu 4.3. Pokud nebude název parametru správně rozpoznán, nedojde k žádné

Výpis 4.3: Příklad zápisu nové polohy letounu

```
__lat,__deg:49.0;__lon,__deg:16.0;
data_received
```

změně v simulátoru. Totéž nastane pokud se uživatel pokusí zapsat hodnotu do parametru, který je určen pouze pro čtení.

4.7.3 Pozastavení simulace

Během testovacích fází implementovaného komunikačního modulu zadávající stranou byl vznesen nový požadavek, a to implementovat příkazy pro pozastavení a opětovné spuštění simulace.

K tomuto účelu byla použita funkce `XPLMCommandKeyStroke()`, která slouží k virtuálnímu stisku klávesy a jejíž definice je následující [26]:

```
XPLM_API void XPLMCommandKeyStroke(XPLMCommandKeyID inKey);
```

kde argument typu '`inKey`' je požadovaná klávesa nastavená na hodnotu '`xplm_key_pause`' výčtového typu '`XPLMCommandKeyID`', viz [27].

Dále byla implementována datová reference pro čtení příznaku pozastavení: '`sim/time/paused`'.

Pro pozastavení chodu simulace slouží zpráva '`sim_pause`'. Pokud implementovaný server obdrží tuto zprávu, simulaci pozastaví a odešle klientovi potvrzovací zprávu '`data_received`'.

Pro opětovné spuštění simulace slouží zpráva '`sim_unpause`'. Pokud implementovaný server obdrží tuto zprávu, simulaci opět spustí a stejně jako v předchozím případě odešle potvrzovací zprávu '`data_received`'. Příklad pozastavení a opětovného spuštění simulace je uveden ve výpisu 4.4.

Výpis 4.4: Příklad pozastavení simulace

```
sim_pause
data_received
sim_unpause
data_received
```

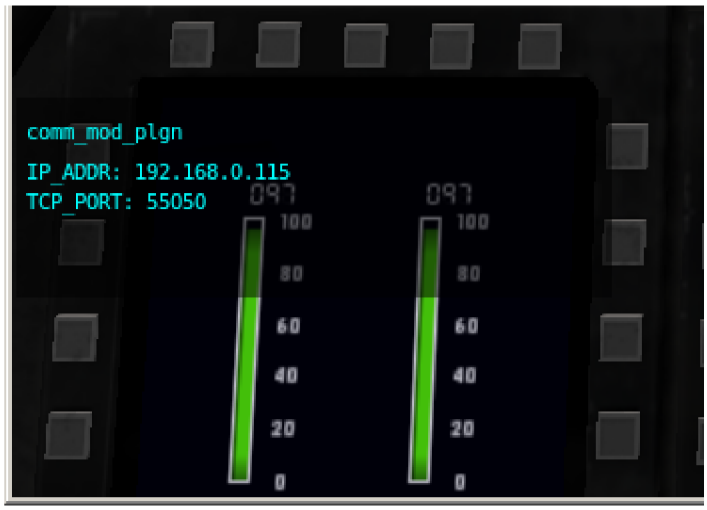
4.8 Vzhled modulu

Komunikační modul jako síťový server nepotřebuje žádné grafické prvky, nicméně pro rychlé nalezení aktuálních parametrů *TCP* portu a *IP* adresy komunikačního modulu bylo vytvořeno jednoduché průsvitné okno v levém dolní rohu okna simulátoru, které tyto parametry vypisuje. Vzhled okna byl odvozen od původních oken simulátoru pro výpis dat na obrazovku a v případě potřeby je možné toto okno skrýt pomocí příslušné konfigurace, viz kapitola 4.5.1.

4.9 Princip komunikačního modulu

Výsledný komunikační modul implementovaný pomocí *X-Plane Plugin SDK* bude pracovat následujícím způsobem:

1. Během spouštění simulátoru X-Plane bude komunikační modul načten jako *DLL* společně s ostatními zásuvnými moduly. Při načítání dojde během vykonávání obslužné rutiny `XPluginStart()` k inicializaci komunikačního modulu. Do této inicializace je zahrnuto načtení konfiguračního souboru a nastavení příslušných parametrů, načtení všech datových referencí pro požadované parametry simulátoru, vytvoření mutexů pro ochranu přístupu do mezipaměti a nastavení vlastní identifikace.



Obr. 4.4: Grafická podoba komunikačního modulu

2. Při povolení komunikačního modulu bude během vykonávání obslužné rutiny `XPluginEnable()` vytvořeno nové vlákno.
   (a) Původní vlákno je nyní v pracovním režimu a každý letový cyklus zapíše požadovaná data do bufferu pro čtení dat. Dále pokud je vystaven příznak zápisu, načte data z bufferu pro zápis a přepíše příslušné parametry simulátoru X-Plane.
   (b) Nové vlákno inicializuje síťový *TCP* server a začne naslouchat na požadovaném portu.
3. Poté co se k serveru připojí některý účastník sítě, začne server přijímat požadavky od klienta a odesílat reakce. V případě že od klienta přijde požadavek na odeslání, server tato data načte z bufferu, odešle klientovi a opět čeká na příchozí zprávu. Pokud přijme požadavek na zápis dat do simulátoru, zapíše tato data do bufferu pro zápis a opět čeká na novou zprávu.
4. Při ukončení simulátoru nebo zakázání komunikačního modulu uživatelem dojde během obslužné procedury `XPluginDisable()` k ukončení serverového vlákna.
5. Pokud dojde k opětovnému povolení komunikačního modulu, algoritmus se opakuje od bodu č. 2. V opačném případě zůstane komunikační modul nečinný až do ukončení simulátoru X-Plane, kdy během obslužné rutiny `XPluginStop()` dojde k uvolnění alokovaných zdrojů.

4.10 Vývojové prostředí

Zadávající strana provozuje simulátor X-Plane 10 na 64-bitové platformě operačního systému *Microsoft Windows 7*. Výsledný komunikační modul tedy bylo nutné imple-

mentovat na tuto platformu. Proto bylo ke kompilaci zdrojových souborů zvoleno nativní vývojové prostředí *Microsoft Visual Studio 2012*, které poběží na totožné platformě. Tím jsou minimalizována rizika spojená s křížovým překladem pro jinou platformu.

Další výhodou použití tohoto prostředí byla možnost použít vzorový projekt vytvořený vývojáři *X-Plane Plugin SDK* jako šablonu pro implementovaný komunikační modul, viz [25]. Tento projekt obsahoval veškerá nastavení pro úspěšný překlad projektu do výsledného *DLL*. Kromě naplnění projektu vlastními zdrojovými soubory zásuvného modulu bylo nutné pouze upravit název projektu, což samo o sobě byl netriviální úkon o desítkách operací, vzhledem k nepochopitelné složitosti konfigurace projektů vývojového prostředí *Microsoft Visual Studio* pomocí grafických oken a desítek nabídek.

Samotné zdrojové soubory jsou psané v programovacím jazyce *ANSI C* a vzhledem k nepochopitelným problémům vývojového prostředí *Microsoft Visual Studio 2012*, byl všechen zdrojový kód umístěn do jednoho jediného souboru. Všechny zdrojové soubory včetně zkompilovaného *DLL* jsou umístěny na přiloženém *CD*.

5 OVĚŘENÍ FUNKČNOSTI

Výsledná implementace komunikačního modulu **Communication module plugin** byla aplikována na letecký simulátor umístěný v laboratoři zadavatele. Vzhledem k tomu, že již při implementaci komunikačního modulu probíhalo průběžné testování správné funkce, nenastaly žádné komplikace při závěrečné implementaci na cílový letecký simulátor.

Na aplikovaném komunikačním modulu proběhla série testů, při kterých nebyla objevena žádná chyba implementace. Toto testování však nebylo zaměřeno na výjimky a okrajové podmínky a proto by bylo vhodné komunikační modul otestovat hlouběji, aby byla minimalizována pravděpodobnost výskytu chyb implementace.

Pro demonstraci funkčnosti komunikačního modulu byl vytvořen jednoduchý automatizovaný test v jazyce *Tcl/Expect*, viz [28]. Celý test je koncipován ze strany klienta, tedy uživatele komunikačního modulu. Vytvořený test názorně demonstruje základní síťovou komunikaci klienta s modulem simulátoru X-Plane, viz výpis 5.1.

Výpis 5.1: Automatizovaný test `testcase.exp`

```
#!/usr/bin/expect
set timeout 2
set ipaddr  "192.168.0.115"
set port    50000

proc abort {msg} {
  puts "\n\nABORTING␣$msg␣..."
  exit 1;
}

#  start client
spawn bash
send "/usr/bin/nc␣-v␣$ipaddr␣$port\n"
expect {
  timeout {abort "connecting"}
  -re "Ncat:␣Connected␣to␣$ipaddr:$port."
}

# send data request
send "data_get\n"
expect {
  timeout {abort "getting␣data"}
```

```
  -re "␣␣_totl,_time␣|␣␣␣pause,_bool␣|␣␣␣__lat,__deg␣|"
}

# pause simulation, change aircraft position
# and after 60s send data request
send "sim_pause\n"
expect {
  timeout {abort "pausing␣simulation"}
  -re "data_received"
}
send "__lat,__deg:49;__lon,__deg:16;__alt,ftmsl:3000;\n"
expect {
  timeout {abort "changing␣aircraft␣position"}
  -re "data_received"
}
after 60000
send "data_get\n"
expect {
  timeout {abort "getting␣data"}
  -re "␣␣_totl,_time␣|␣␣␣pause,_bool␣|␣␣␣__lat,__deg␣|"
}

# unpause after 4s
after 4000
send "sim_unpause\n"
expect {
  timeout {abort "unpausing␣simulation"}
  -re "data_received"
}
```

Výstup tohoto testu je uveden ve výpisu 5.2.

Výpis 5.2: Výstupní log testu

```
Ncat: Version 7.60 ( https://nmap.org/ncat )
Ncat: Connected to 192.168.0.115:50000.
data_get
sim_pause
  _totl,_time |   pause,_bool |   __lat,__deg |   __lon,_
_deg |   __alt,ftmsl |   ruddr,yoke1 |   _elev,yoke1 |
ailrn,yoke1 |   AMtmp,_degC |   alpha,__deg |   _beta,__d
```

```
eg |   _Vind,_kias |   _Vind,_keas |   hding,__mag |   __
__P,rad/s |   ____Q,rad/s |   ____R,rad/s |   Vtrue,_ktas
 |   _roll,__deg |   hding,_true |   pitch,__deg |   __VV
 I,__fpm |   __alt,ftagl |   AMprs,_inHG |
    373.50378 |              0 |      44.99919 |      17.0
0896 |    3912.17147 |       0.00000 |       0.89771 |
    0.12954 |       7.25228 |       9.73157 |      -0.412
74 |     445.08319 |     445.08319 |     109.80502 |
  0.38761 |       0.48478 |       0.06982 |     474.77414
 |      61.45321 |     114.14458 |       7.16050 |   -150
5.31128 |    2021.25452 |      25.92198 |
data_received
__lat,__deg:49;__lon,__deg:16;__alt,ftmsl:3000;
data_received
data_get
  _totl,_time |   pause,_bool |   __lat,__deg |   __lon,_
_deg |   __alt,ftmsl |   ruddr,yoke1 |   _elev,yoke1 |
ailrn,yoke1 |   AMtmp,_degC |   alpha,__deg |   _beta,__d
eg |   _Vind,_kias |   _Vind,_keas |   hding,__mag |   __
__P,rad/s |   ____Q,rad/s |   ____R,rad/s |   Vtrue,_ktas
|   _roll,__deg |   hding,_true |   pitch,__deg |   __VV
I,__fpm |   __alt,ftagl |   AMprs,_inHG |
    373.56238 |              1 |      49.00000 |      16.0
0000 |    3000.00000 |       0.00000 |       0.89771 |
    0.12954 |       7.24880 |       9.87040 |      -0.499
57 |     444.24405 |     444.24405 |     111.31425 |
  0.37753 |       0.45862 |       0.07108 |     474.12097
 |      62.92753 |     115.65384 |       7.68774 |   -130
7.78101 |    2008.99463 |      25.92029 |
sim_unpause
data_received
```

5.1 Měření zadavatele

Zadavatel uskutečnil vlastní testování implementovaného komunikačního modulu. Pří tomto testování bylo pro komunikaci se zásuvným modulem využito prostředí *MATLAB* jako klientské strany a pro potřeby diplomové práce byly zadavatelem dodány grafické výstupy provedeného měření, viz obrázky 5.1 a 5.2.

Obr. 5.1: Naměřená data 1

Obr. 5.2: Naměřená data 2

6 ZÁVĚR

Cílem této práce byl návrh a realizace softwarového komunikačního modulu pro letecký simulátor X-Plane 10, který by byl schopen číst a zapisovat data do simulátoru pro možnost změny aktuální letové situace, respektive vybraných letových parametrů, a rovněž čtení těchto parametrů pro účely následného zpracování a analýzy.

V rámci diplomové práce proběhlo seznámení se simulátorem X-Plane a v kapitole č. 1 byly stručně popsány jeho základní vlastnosti.

Dále byly v rámci kapitoly č. 2 analyzovány obecné možnosti čtení a zápisu parametrů modelu simulátoru. Na základě této analýzy byly nalezeny 4 metody poskytované simulátorem X-Plane, které lze využít pro čtení nebo zápis. Pro všechny tyto metody byla zkoumána splnitelnost požadavků kladených zadavatelem uvedených v kapitole č. 2.1 a výsledné shrnutí dosažitelnosti jednotlivých požadavků pro všechny metody bylo uvedeno v tabulce 2.3.

Na základě provedené analýzy možnosti čtení a zápisu dat do simulátoru X-Plane byla pro návrh komunikačního modulu vybrána metoda *UDP* zpráv. Testování této metody komunikace bylo neúspěšné. Úspěšně byly přijímány a zpracovávány všechny *UDP* zprávy odesílané simulátorem X-Plane, ale nepodařilo se přijmout jedinou zprávu simulátorem X-Plane od uživatele. Během zkoumání příčiny neúspěchu bylo z neoficiálních zdrojů zjištěno, že testovaná metoda je pouze experimentální funkcí, u které není zaručena funkčnost. Důvod neúspěchu zůstal neobjasněný, protože při hledání příčiny byla objevena nová stabilní metoda *zásuvných modulu*, která vyhovuje požadavkům zadavatele kladeným na komunikační modul a zároveň je garantována zpětná kompatibilita při přechodu na novou verzi simulátoru X-Plane.

V rámci kapitoly č. 3 bylo provedeno studium vývojového prostředí pro návrh zásuvných modulů a analýza komponent nutných pro úspěšný návrh komunikačního modulu.

Testování použitelnosti *zásuvných modulu* bylo úspěšné a proto byl v rámci kapitoly č. 4 zahájen návrh komunikačního modulu pomocí této metody. Během návrhu byl pro síťovou komunikaci na základě konzultace se zadavatelem zvolen *TCP*, který vyhovuje požadovanému modelu komunikace **dotaz-odpověď** a velmi usnadňuje implementaci (není nutné kontrolovat validitu a pořadí příchozích dat). Pro minimalizaci zpomalení chodu simulátoru v důsledku komunikačního modulu byla navržena implementace *TCP* serveru v odděleném vlákně, které bude vykonáváno nezávisle na běhu simulátoru a pro sdílení dat mezi vlákny byly použity globální proměnné chráněné pomocí mutexů.

Na základě předchozího návrhu byl komunikační modul implementován v programovacím jazyce *ANSI C*. Zdrojový kód komunikačního modulu byl kompilován v nativním vývojové prostředí *Microsoft Visual Studio 2012* pro 64-bitovou platformu

operačního systému *Microsoft Windows 7*, na které zadavatel provozuje simulátor X-Plane.

Výsledný komunikační modul byl úspěšně aplikován na simulátoru v laboratoři zadavatele a v rámci kapitoly č. 5 byl modul podroben sérii testů funkčnosti, při kterých nebyla objevena žádná chyba implementace. Funkčnost komunikačního modulu byla demonstrována na jednoduchém automatizovaném testu, viz výpis 5.1.

Další rozvoj komunikačního modulu by mohl směřovat k rozšíření souboru podporovaných parametrů pro čtení a zápis, doplnění implementace komunikačního modulu o metody umožňující nepřímo zapisovat data i do parametrů určených pouze pro čtení a optimalizaci zdrojového kódu pro dosažení vyššího výkonu, snadnější udržovatelnosti a lepší čitelnosti.

LITERATURA

[1] *X-Plane 10 Manual* [DVD]. 2016 [cit. 2018-04-15]

[2] *Sending Data to X-Plane* [DVD]. [cit. 2018-04-20]

[3] *X-Plane Flight Simulator* [online]. [cit. 2018-04-15]. Dostupné z: `http://www.x-plane.com/`

[4] *X-Plane Plugin SDK* [online]. [cit. 2018-03-07]. Dostupné z: `http://www.xsquawkbox.net/xpsdk/mediawiki/Main_Page`

[5] *XPLMMenus* [online]. [cit. 2018-05-05]. Dostupné z: `http://www.xsquawkbox.net/xpsdk/mediawiki/XPLMMenus`

[6] *XPLMDisplay* [online]. [cit. 2018-05-05]. Dostupné z: `http://www.xsquawkbox.net/xpsdk/mediawiki/XPLMDisplay`

[7] *XPLMProcessing* [online]. [cit. 2018-05-05]. Dostupné z: `http://www.xsquawkbox.net/xpsdk/mediawiki/XPLMProcessing`

[8] *WhyRefcons* [online]. [cit. 2018-05-06]. Dostupné z: `http://www.xsquawkbox.net/xpsdk/mediawiki/WhyRefcons`

[9] *WhyRefcons* [online]. [cit. 2018-05-06]. Dostupné z: `http://www.xsquawkbox.net/xpsdk/mediawiki/XPLMWindowID`

[10] *XPLMDrawWindow f* [online]. [cit. 2018-05-06]. Dostupné z: `http://www.xsquawkbox.net/xpsdk/mediawiki/XPLMDrawWindow_f`

[11] *XPLMHandleKey f* [online]. [cit. 2018-05-06]. Dostupné z: `http://www.xsquawkbox.net/xpsdk/mediawiki/XPLMHandleKey_f`

[12] *XPLMHandleMouseClick f* [online]. [cit. 2018-05-06]. Dostupné z: `http://www.xsquawkbox.net/xpsdk/mediawiki/XPLMHandleMouseClick_f`

[13] *XPLMHandleMouseClick f* [online]. [cit. 2018-05-06]. Dostupné z: `http://www.xsquawkbox.net/xpsdk/mediawiki/XPLMKeyFlags`

[14] *XPLMMouseStatus* [online]. [cit. 2018-05-06]. Dostupné z: `http://www.xsquawkbox.net/xpsdk/mediawiki/XPLMMouseStatus`

[15] *DataAccess* [online]. [cit. 2018-05-09]. Dostupné z: `http://www.xsquawkbox.net/xpsdk/mediawiki/DataAccess`

[16] *Datarefs for X-Plane* [online]. [cit. 2018-05-09]. Dostupné z: http://www.xsquawkbox.net/xpsdk/docs/DataRefs.html

[17] *Dynamic-Link Libraries* [online]. [cit. 2018-03-09]. Dostupné z: https://msdn.microsoft.com/en-us/library/windows/desktop/ms682589.aspx

[18] POSTEL, J. RCF 768: User Datagram Protocol. *IETF Tools* [online]. 28 August 1980 [cit. 2018-04-19]. Dostupné z: https://tools.ietf.org/html/rfc768

[19] Regular Expressions. *The Open Group Base Specifications* [online]. [cit. 2018-04-20]. Dostupné z: http://pubs.opengroup.org/onlinepubs/009695399/basedefs/xbd_chap09.html

[20] *About Winsock* [online]. [cit. 2018-05-10]. Dostupné z: https://msdn.microsoft.com/en-us/library/windows/desktop/ms737523.aspx

[21] *Mutex Objects* [online]. [cit. 2018-05-10]. Dostupné z: https://msdn.microsoft.com/en-us/library/windows/desktop/ms684266.aspx

[22] *Guide for the Use of the International System of Units* [online]. [cit. 2018-05-09]. Dostupné z: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/\nistspecialpublication811e2008.pdf

[23] *Units outside the SI* [online]. [cit. 2018-05-09]. Dostupné z: https://www.bipm.org/en/publications/si-brochure/chapter4.html

[24] *XPLMWorldToLocal* [online]. [cit. 2018-05-10]. Dostupné z: http://www.xsquawkbox.net/xpsdk/mediawiki/XPLMWorldToLocal

[25] *HelloWorld* [online]. [cit. 2018-05-10]. Dostupné z: http://www.xsquawkbox.net/xpsdk/mediawiki/HelloWorld

[26] *XPLMCommandKeyStroke* [online]. [cit. 2018-05-10]. Dostupné z: http://www.xsquawkbox.net/xpsdk/mediawiki/XPLMCommandKeyStroke

[27] *XPLMCommandKeyID* [online]. [cit. 2018-05-10]. Dostupné z: http://www.xsquawkbox.net/xpsdk/mediawiki/XPLMCommandKeyID

[28] *Expect* [online]. [cit. 2018-05-10]. Dostupné z: https://core.tcl.tk/expect/index

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

ANSI	American National Standards Institute
API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
CD	Compact Disc
DLL	Dynamic-Link Library
DVD	Digital Video Disc
FAA	Federal Aviation Administration
FEKT	Fakulta elektrotechniky a komunikačních technologií
IP	Internet Protocol
LAN	Local Area Network
MATLAB	Matrix Laboratory
NASA	National Aeronautics and Space Administration
OS	Operationg System
SDK	Software Development Kit
Tcl	Tool command language
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
USB	Universal Serial Bus
VUT	Vyskoké učení technické
XPLM	X-Plane Plugin Manager

alt_{f}	Nadmořská výška ve stopách
alt_{m}	Nadmořská výška v metrech
v_{kn}	Rychlost v uzlech
$v_{\mathrm{m/s}}$	Rychlost v metrech za sekundu

SEZNAM PŘÍLOH

A PŘÍKLAD FORMÁTU DAT

Tento soubor by získán od zadávajícího jako reference formátu dat simulátoru odesílaných z komunikačního modulu klientovi. Pro názornost bylo doplněno číslování řádků a znak konce řádku '$' které nejsou součástí vzoru, zalomení řádků je doplněné z důvodu nedostatku místa a nelze se ním řídit. Dále pro lepší čitelnost byl znak '20h' (mezera) nahrazen znakem '␣':

```
1 ␣␣_real,_time␣|␣␣␣_totl,_time␣|␣␣␣missn,_time␣|␣␣␣timer,_time␣|
  ␣␣␣_zulu,_time␣|␣␣␣local,_time␣|␣␣␣hobbs,_time␣|␣␣␣_Vind,_kias␣|
  ␣␣␣_Vind,_keas␣|␣␣␣Vtrue,_ktas␣|␣␣␣Vtrue,_ktgs␣|␣␣␣_Vind,__mph␣|
  ␣␣␣Vtrue,mphas␣|␣␣␣Vtrue,mphgs␣|␣␣␣_Mach,ratio␣|␣␣␣__VVI,__fpm␣|
  ␣␣␣Gload,norml␣|␣␣␣Gload,axial␣|␣␣␣Gload,_side␣|␣␣␣_elev,yoke1␣|
  ␣␣␣ailrn,yoke1␣|␣␣␣ruddr,yoke1␣|␣␣␣____Q,rad/s␣|␣␣␣____P,rad/s␣|
  ␣␣␣____R,rad/s␣|␣␣␣pitch,__deg␣|␣␣␣_roll,__deg␣|␣␣␣hding,_true␣|
  ␣␣␣hding,__mag␣|␣␣␣alpha,__deg␣|␣␣␣_beta,__deg␣|␣␣␣hpath,__deg␣|
  ␣␣␣vpath,__deg␣|␣␣␣_slip,__deg␣|␣␣␣__lat,__deg␣|␣␣␣__lon,__deg␣|
  ␣␣␣__alt,ftmsl␣|␣␣␣__alt,ftagl␣|␣␣␣___on,runwy␣|␣␣␣__alt,__ind␣|
  ␣␣␣__lat,south␣|␣␣␣__lon,_west␣|␣␣␣____X,____m␣|␣␣␣____Y,____m␣|
  ␣␣␣____Z,____m␣|␣␣␣___vX,__m/s␣|␣␣␣___vY,__m/s␣|␣␣␣___vZ,__m/s␣|
  ␣␣␣_dist,___ft␣|␣␣␣_dist,___nm␣|␣␣␣power,_1,hp␣|␣␣␣power,_2,hp␣|
  ␣␣␣thrst,_1,lb␣|␣␣␣thrst,_2,lb␣|␣␣␣trq_1,_ftlb␣|␣␣␣trq_2,_ftlb␣|
  ␣␣␣rpm_1,engin␣|␣␣␣rpm_2,engin␣|␣$
2 ␣␣␣␣␣␣0.66120␣|␣␣␣␣␣␣0.05025␣|␣␣␣␣␣␣0.05025␣|␣␣␣␣␣␣0.00000␣|
  ␣␣␣␣␣␣␣9.95984␣|␣␣␣␣␣␣10.95984␣|␣␣␣␣␣␣20.08635␣|␣␣␣␣␣␣0.00000␣|
  ␣␣␣␣␣␣␣4.98012␣|␣␣␣␣␣␣5.04642␣|␣␣␣␣␣␣0.16528␣|␣␣␣␣␣␣0.00000␣|
  ␣␣␣␣␣␣␣5.80732␣|␣␣␣␣␣␣1.22922␣|␣␣␣␣␣␣0.00765␣|␣␣␣␣␣␣1.49211␣|
  ␣␣␣␣␣␣␣1.97657␣|␣␣␣␣␣␣0.35880␣|␣␣␣␣␣␣-0.02533␣|␣␣␣␣␣␣0.00000␣|
  ␣␣␣␣␣␣␣0.00000␣|␣␣␣␣␣␣0.00000␣|␣␣␣␣␣␣-0.03485␣|␣␣␣␣␣␣-0.01143␣|
  ␣␣␣␣␣␣␣0.00005␣|␣␣␣␣␣␣0.26351␣|␣␣␣␣␣␣0.11602␣|␣␣␣␣␣152.64061␣|
  ␣␣␣␣␣␣147.97728␣|␣␣␣␣-153.64151␣|␣␣␣␣␣␣61.33526␣|␣␣␣␣␣148.75500␣|
  ␣␣␣␣␣␣␣81.09892␣|␣␣␣␣␣␣0.00000␣|␣␣␣␣␣␣49.24037␣|␣␣␣␣␣␣16.55242␣|
  ␣␣␣␣␣901.40643␣|␣␣␣␣␣␣0.06980␣|␣␣␣␣␣␣1.00000␣|␣␣␣␣1111.78638␣|
  ␣␣␣␣␣␣␣48.00000␣|␣␣␣␣␣␣15.00000␣|␣␣␣␣3809.86328␣|␣␣␣␣␣217.47549␣|
  ␣␣-26760.16797␣|␣␣␣␣␣␣␣0.04410␣|␣␣␣␣␣␣␣0.54289␣|␣␣␣␣␣␣␣0.07269␣|
  ␣␣␣␣␣␣␣0.07017␣|␣␣␣␣␣␣␣0.00001␣|␣␣␣␣␣␣14.41388␣|␣␣␣␣␣␣14.41387␣|
  ␣␣␣␣␣1191.78259␣|␣␣␣␣1196.82935␣|␣␣␣␣␣␣58.40470␣|␣␣␣␣␣␣58.40458␣|
  ␣␣␣␣␣1295.70923␣|␣␣␣␣1295.71594␣|␣$
3 ␣␣␣␣␣␣␣3.14991␣|␣␣␣␣␣␣␣0.10050␣|␣␣␣␣␣␣␣0.10050␣|␣␣␣␣␣␣␣0.00000␣|
  ␣␣␣␣␣␣␣9.95985␣|␣␣␣␣␣␣10.95985␣|␣␣␣␣␣␣20.08636␣|␣␣␣␣␣␣0.00000␣|
```

␣␣␣␣␣␣␣5.04190␣|␣␣␣␣␣␣5.10903␣|␣␣␣␣␣␣0.38872␣|␣␣␣␣␣␣0.00000␣|
␣␣␣␣␣␣␣5.87936␣|␣␣␣␣␣␣1.87965␣|␣␣␣␣␣␣0.00775␣|␣␣␣␣␣␣7.56803␣|
␣␣␣␣␣␣␣1.40957␣|␣␣␣␣␣␣0.21622␣|␣␣␣␣␣-0.01761␣|␣␣␣␣␣␣0.00000␣|
␣␣␣␣␣␣␣0.00000␣|␣␣␣␣␣␣0.00000␣|␣␣␣␣␣-0.07070␣|␣␣␣␣␣-0.02131␣|
␣␣␣␣␣␣␣0.00078␣|␣␣␣␣␣␣0.08659␣|␣␣␣␣␣␣0.06147␣|␣␣␣␣152.64192␣|
␣␣␣␣147.97859␣|␣␣␣-140.54683␣|␣␣␣␣␣60.42535␣|␣␣␣␣148.41907␣|
␣␣␣␣␣76.23223␣|␣␣␣␣␣-0.00760␣|␣␣␣␣␣49.24037␣|␣␣␣␣␣16.55242␣|
␣␣␣␣901.53259␣|␣␣␣␣␣␣0.19671␣|␣␣␣␣␣␣1.00000␣|␣␣␣1111.85596␣|
␣␣␣␣␣48.00000␣|␣␣␣␣␣15.00000␣|␣␣␣3809.86792␣|␣␣␣␣217.51396␣|
␣␣-26760.16016␣|␣␣␣␣␣␣0.10473␣|␣␣␣␣␣␣0.81614␣|␣␣␣␣␣␣0.17036␣|
␣␣␣␣␣␣␣0.19962␣|␣␣␣␣␣␣0.00003␣|␣␣␣␣␣14.86557␣|␣␣␣␣␣14.86552␣|
␣␣␣␣1145.98804␣|␣␣␣␣1149.19873␣|␣␣␣␣␣60.38425␣|␣␣␣␣␣60.38358␣|
␣␣␣␣1292.40747␣|␣␣␣␣1292.41919␣|␣$
4 ␣␣␣␣␣␣6.41252␣|␣␣␣␣␣␣0.15075␣|␣␣␣␣␣␣0.15075␣|␣␣␣␣␣␣0.00000␣|
␣␣␣␣␣␣␣9.95986␣|␣␣␣␣␣10.95986␣|␣␣␣␣␣20.08638␣|␣␣␣␣␣␣0.00000␣|
␣␣␣␣␣␣␣4.99223␣|␣␣␣␣␣␣5.05871␣|␣␣␣␣␣␣0.56739␣|␣␣␣␣␣␣0.00000␣|
␣␣␣␣␣␣␣5.82146␣|␣␣␣␣␣␣1.95820␣|␣␣␣␣␣␣0.00767␣|␣␣␣␣␣15.32258␣|
␣␣␣␣␣␣␣0.90075␣|␣␣␣␣␣␣0.18229␣|␣␣␣␣␣-0.01046␣|␣␣␣␣␣␣0.00000␣|
␣␣␣␣␣␣␣0.00000␣|␣␣␣␣␣␣0.00000␣|␣␣␣␣␣-0.10661␣|␣␣␣␣␣-0.02918␣|
␣␣␣␣␣␣␣0.00189␣|␣␣␣␣␣-0.19547␣|␣␣␣␣␣-0.01727␣|␣␣␣␣152.64632␣|
␣␣␣␣147.98299␣|␣␣␣-137.70317␣|␣␣␣␣␣61.64796␣|␣␣␣␣148.63539␣|
␣␣␣␣␣70.52229␣|␣␣␣␣␣-0.02054␣|␣␣␣␣␣49.24037␣|␣␣␣␣␣16.55242␣|
␣␣␣␣901.67047␣|␣␣␣␣␣␣0.33583␣|␣␣␣␣␣␣1.00000␣|␣␣␣1111.98218␣|
␣␣␣␣␣48.00000␣|␣␣␣␣␣15.00000␣|␣␣␣3809.87500␣|␣␣␣␣217.55605␣|
␣␣-26760.14844␣|␣␣␣␣␣␣0.15192␣|␣␣␣␣␣␣0.82530␣|␣␣␣␣␣␣0.24924␣|
␣␣␣␣␣␣␣0.34471␣|␣␣␣␣␣␣0.00006␣|␣␣␣␣␣15.32760␣|␣␣␣␣␣15.32755␣|
␣␣␣␣1088.35535␣|␣␣␣␣1093.97766␣|␣␣␣␣␣62.42525␣|␣␣␣␣␣62.42441␣|
␣␣␣␣1288.90198␣|␣␣␣␣1288.91699␣|␣$