České vysoké učení technické v Praze
Fakulta elektrotechnická
Katedra řídicí techniky

Diplomová práce

Integrace leteckého simulátoru do HIL simulátoru

Vypracoval: Jan Černý
Vedoucí práce: Ing. Libor Waszniowski, Ph.D.
Praha 2010

České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra řídicí techniky

ZADÁNÍ DIPLOMOVÉ PRÁCE

Student: **Bc. Jan Černý**

Studijní program: Elektrotechnika a informatika (magisterský), strukturovaný
Obor: Kybernetika a měření, blok KM1 - Řídicí technika

Název tématu: **Integrace leteckého simulátoru do HIL simulátoru**

Pokyny pro vypracování:

Seznamte se s rozhraním a komunikačním protokolem leteckého simulátoru a principy simulace a generování kódu v Simulinku.
Implementujte simulinkové bloky (s-fce a .tlc) podporující komunikaci Simulinkového modelu s leteckým simulátorem při simulaci i ve vygenerovaném kódu.
Použití demonstrujte na vhodné úloze.

Seznam odborné literatury:

Dodá vedoucí práce

Vedoucí: Ing. Libor Waszniowski, Ph.D.

Platnost zadání: do konce letního semestru 2009/10

prof. Ing. Michael Šebek, DrSc.
vedoucí katedry

L.S.

doc. Ing. Boris Šimák, CSc.
děkan

V Praze dne 27. 2. 2009

Prohlášení

Prohlašuji, že jsem svou diplomovou práci vypracoval samostatně a použil jsem pouze podklady (literaturu, projekty, SW atd.) uvedené v přiloženém seznamu.

V Praze, dne Jan Černý

Poděkování

Zde bych rád poděkoval všem, kteří mi poskytli cenné rady a pomoc při řešení technických problémů a obtíží při realizaci diplomové práce, především Ing. Liboru Waszniowskému Ph.D., Lukáši Hamáčkovi a Zdeňku Zechovskému.

Abstrakt

Tato diplomová práce popisuje vývoj uživatelského rozhraní integrujícího letecký simulátor FlightGear do simulátoru typu Hardware In the Loop (HIL) vyvinutého Katedrou řídící techniky a využívaného ve firmě Aero Vodochody a.s. Toto rozhraní umožňuje řídit a vizualizovat průběh simulací vykonávaných na HIL simulátoru. Výsledkem této práce je komplexní řešení pro vývoj, měření a testování řídicího systému, založeného na konceptu Flight By Wire, pro malé proudové letadlo.

Abstract

This master thesis describes the development of an interface integrating flight simulator FlightGear into Hardware In the Loop (HIL) based simulator developed by Department Of Control Engineering, used by Aero Vodochody a.s. This interface allows for visualization of simulations performed by the HIL simulator. This thesis results in a complex solution for development, measurement and testing of a small jet plane control system based on the Flight By Wire concept.

Obsah

1. Úvod 8
1.1. Úvod do problematiky 8
1.2. Cíl práce 8
1.3. HIL simulátor 9
1.4. Cíle této práce v rámci HIL simulátoru 11
1.5. Použitá terminologie 12
1.5.1. Osy letounu a řídící prvky 12
2. Výběr leteckého simulátoru 15
2.1. FlightGear 15
2.1.1. Stavové informace 16
2.1.2. Rozhraní 16
2.2. X-Plane 18
2.2.1. Stavové informace 18
2.2.2. Rozhraní 18
2.3. Microsoft Flight Simulator X 19
2.3.1. Stavové informace 19
2.3.2. Rozhraní SimConnect 19
2.4. Srovnání, výběr simulátoru 19
2.4.1. Srovnání 19
2.4.2. Výběr 20
3. Příslušenství k leteckému simulátoru 21
4. Rozhraní modelu pro HIL simulaci 22
4.1. Základní principy 22
4.2. Proces implementace bloku s podporou generování kódu 23
4.2.1. Implementace S-funkce pro simulaci 23
4.2.2. Vytvoření masky pro nastavení parametrů 24
4.2.3. Implementace TLC skriptu 25
5. FlightGear Blockset 27
5.1. Instalace knihovny 28
5.2. Společné vlastnosti pro všechny bloky 28
5.2.1. Komunikační rozhraní 28
5.2.2. Generování kódu – předávání parametrů 28
5.2.3. Generování kódu – společné funkce 28
5.3. Blok FlightGear Interface 28
5.3.1. Vstupy bloku 29
5.3.2. Výstupy bloku 29
5.3.3. Vstupní parametry S-funkce 30
5.3.4. Princip funkce 30
5.3.5. Implementace bloku 31
5.3.6. Komunikační rozhraní 32
5.3.7. Generování kódu 33
5.4. Blok IFDP UDP Interface 34
5.4.1. Vstupy bloku 35
5.4.2. Výstupy bloku 35
5.4.3. Vstupní parametry S-funkce 35
5.4.4. Princip funkce 36
5.4.5. Implementace bloku 36

5.4.6. Komunikační rozhraní 36
5.4.7. Generování kódu 37
5.5. Blok FG Data Sender 37
5.5.1. Vstupy bloku 38
5.5.2. Výstupy bloku 38
5.5.3. Vstupní parametry S-funkce 38
5.5.4. Princip funkce 39
5.5.5. Implementace bloku 39
5.5.6. Komunikační rozhraní 39
5.5.7. Generování kódu 39
5.6. Blok FG Data Receiver 39
5.6.1. Vstupy bloku 40
5.6.2. Výstupy bloku 40
5.6.3. Vstupní parametry S-funkce 41
5.6.4. Princip funkce 41
5.6.5. Implementace bloku 41
5.6.6. Generování kódu 41
5.7. Rozdíly v aplikaci bloků vyplývající z použitých protokolů 41
5.7.1. Bloky využívající TCP protokol 42
5.7.2. Bloky využívající UDP protokol 43
6. Závěr 45
7. Bibliografie 46
Příloha A 47
Příloha B 52
Příloha C 60
Příloha D 61

1. Úvod

1.1. Úvod do problematiky

Tato práce je součástí rozsáhlejší spolupráce Katedry řídicí techniky s firmou AERO Vodochody a.s. Součástí této spolupráce je vývoj konceptu řídicího systému typu *Flight By Wire (FBW)* pro malé proudové letadlo. Výsledky vývoje tohoto řídicího systému se experimentálně ověřují na hardwarovém simulátoru typu *Hardware In the Loop (HIL)*.

Neméně důležitou součástí vývoje jsou podpůrné nástroje poskytující rozhraní člověk-stroj (HMI), možnost vývoje a parametrizace modelu, a nástroje pro analýzu získaných výsledků.

Práce se zabývá vývojem zmíněného HMI. Účelem HMI je v tomto případě zprostředkování vizualizace průběhu simulace a umožnit tuto simulaci v reálném čase řídit. Koncepce celého projektu počítá s rozhraním tvořeným leteckým simulátorem v kombinaci s *Integrovaným systémem zpracování letových dat (IFDP)*.

1.2. Cíl práce

Cílem této práce je navrhnout a vytvořit vizualizační rozhraní platformy pro testování řídicího systému letounu v uzavřené smyčce (HIL simulátor). Tento simulátor je navržen jako univerzální simulační či měřící platforma, která by měla být využitelná i pro jiné projekty.

Systém pro AERO Vodochody a.s. se skládá z platformy pro simulaci modelu fyzikální soustavy v reálném čase, pro simulaci vstupních signálů a měření výstupních signálů řídicího systému. Simulaci a měření signálů provádějí distribuovaná zařízení navržená speciálně pro signály testovaného řídicího systému. Systém má být připraven na budoucí rozšíření o nové typy modulů.

Hlavním úkolem této práce je vytvořit uživatelské rozhraní tvořené leteckým simulátorem na PC platformě, k němuž jsou připojeny řídící prvky letounu (řídicí páka, páka přípusti a pedály). Simulátor je prostřednictvím blocksetu integrován v prostředí Matlab Simulink, které je využíváno pro tvorbu modelů HIL simulátoru.

Souvisejícím úkolem při tvorbě tohoto blocksetu je vytvoření bloku pro komunikaci s leteckým přístrojem *IFDP* (*Integrovaným systémem zpracování letových dat*) zobrazujícím letová data na displej MFD.

Dalším úkolem této práce byla vlastní integrace implementovaného rozhraní do HIL simulátoru. Tento úkol zahrnoval testování a zprovoznění jednotlivých částí rozhraní přímo v Aero Vodochody. Výsledkem integrace do HIL simulátoru má být komplexní řešení, které umožňuje ovládání řídicích veličin simulace letu, stejnými řídícími prvky jako v reálném letadle. Dále umožňuje vizualizaci průběhu simulace letu na PC a skutečném leteckém přístroji zobrazujícím letové údaje (například umělý horizont apod.). Díky propojení s prostředím Matlab Simulink disponuje toto řešení širokou škálou nástrojů pro diagnostiku vstupních i výstupních dat simulačního procesu.

Práce obsahuje popis řešení zadaného problému a dále popisuje implementaci tohoto řešení ve formě knihovny pro prostředí Simulink. Dokumentace ke knihovně, leteckému simulátoru a hardwaru je přílohou tohoto dokumentu.

Osnova práce

HIL Simulátor – kapitola 1.3

V této kapitole jsou v úvodu vysvětleny pojmy dále použité v této práci. Poté je zde popsán hardwarový simulátor využívaný firmou Aero Vodochody a.s. a cíle této práce v kontextu hardwarového simulátoru.

Výběr leteckého simulátoru – kapitola 2

Tato kapitola je věnována výběru vhodného leteckého simulátoru pro účely této práce. Obsahuje popis a zhodnocení dostupných produktů a zdůvodnění výběru zvoleného produktu.

Příslušenství k leteckému simulátoru – kapitola 3

V této kapitole je stručně popsáno příslušenství pro letecký simulátor FlightGear.

Rozhraní modelu pro HIL simulaci – kapitola 4

Tato část obsahuje stručný úvod do problematiky generování kódu pro cílovou platformu s využitím Real-Time Workshop. Dále tato kapitola popisuje, co vše obnáší implementace bloku podporujícího generování kódu.

FlightGear blockset – kapitola 5

Kompletní popis bloků vytvořených v rámci této práce seskupených do knihovny nazvané FlightGear blockset.

1.3. HIL simulátor

Tato část obsahuje úvod do problematiky této práce. Je zde vysvětlena koncepce řídicího systému Flight By Wire a koncepce simulátoru Hardware in The Loop vytvořeného pro výzkumné a měřící účely ve firmě Aero Vodochody a.s.

Flight By Wire

Cílem celého projektu pro Aero Vodochody je vývoj řídicího systém typu Flight By Wire pro malé proudové letadlo, jehož koncepce je pojata následujícím způsobem. FBW je primárním řídicím systémem a stávající mechanický řídicí systém je redundantním systémem tvořícím zálohu. Komplexní řešení řídicího systému umožňuje experimentálně ověřovat zákony řízení a interakci mezi oběma řídicími systémy.

Pro testování FBW systému jsou použity stojany s reálnými akčními členy, senzory a řídícími plochami. Manipulací s jednotlivými řídícími plochami či jejich zatěžováním se realizují vstupy či výstupy různých simulací v rámci testů řídicího systému FBW.

Hardware In the Loop

Základním principem simulace typu Hardware In the Loop (HIL) je propojení reálných hardwarových zařízení s počítačovým simulátorem, který v reálném čase provádí simuluje zbytek procesu v uzavřené smyčce. Hlavní důvody pro realizaci HIL simulace jsou následující. Díky využití modelu v uzavřené smyčce je simulace levnější, méně časově náročná a jednodušší než experimenty na skutečném zařízení. Dále je eliminováno riziko poškození reálného zařízení, které může být velmi drahé. Tento přístup k simulaci také umožňuje simulovat chování systému v extrémních pracovních bodech, které jsou s reálným zařízením těžko dosažitelné nebo existuje velké riziko jeho poškození. HIL simulace poskytuje na rozdíl od pouhé numerické simulace komplexnější výsledky díky zapojení reálných prvků.

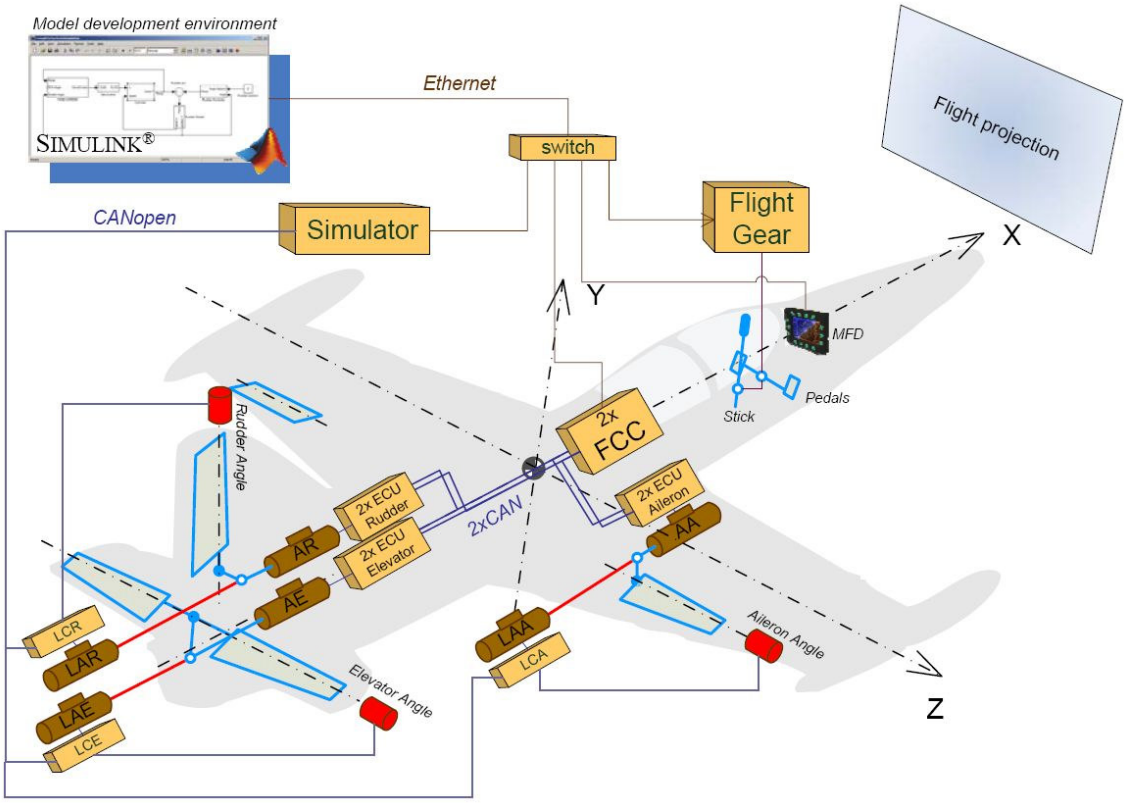
HIL simulátor sestrojený v Aero Vodochody a.s.

V našem případě je HIL simulátor sestrojen pro účely simulace letu letounu L-159. Hardwarová část simulátoru se skládá ze dvou hydraulických stojanů, stojanu s celým hydraulickým systémem letounu (letadlový stojan) a stojan s hydraulikou akčních prvků (aktuátorový stojan).

Letadlový stojan obsahuje kompletní mechanický a hydraulický řídicí systém pro tři osy letounu. Akční členy tohoto systému jsou rozšířeny o servomotory ovládané řídicím systémem navrženým podle konceptu FBW. Tato část simulátoru dále obsahuje kokpit s řídicí pákou a jednotku využívající letecký simulátor FlightGear pro vizualizaci letu. V kokpitu bude dále zapojen integrovaný systém zpracování letových dat (IFDP). Na obrázku 1 je zobrazen jeho výstup, tvořený multifunkčním displejem (MFD). Tento systém je blíže popsán v kapitole 5.4.

Naproti tomu aktuátorový stojan je podmnožina mechanického a hydraulického systému tvořená pouze dvěma akčními členy. Tento menší systém slouží především pro různé experimenty zahrnující měření frekvenčních a statických charakteristik akčních členů, odolnosti vůči různým typům rušících veličin, testování odezvy na poruchové stavy a další.

Oba tyto stojany jsou propojeny s vestavěným počítačovým simulátorem (1) s procesorem typu PowerPC a operačním systémem Linux (blok Simulator na obrázku 1). Ten v reálném čase provádí simulaci chování letounu respektive simulaci navrženou ve formě modelu v prostředí Matlab Simulink. Přístrojové vybavení simulátoru jako celku je tvořeno distribuovanými jednotkami (periferiemi), které komunikují s řídícím počítačem pomocí protokolu CANopen přes sběrnici CAN. Schéma realizace celého HIL simulátoru pro Aero Vodochody je zachyceno na obrázku 1.



Obrázek 1: Schéma zapojení HIL simulátoru, převzato z (2)

1.4. Cíle této práce v rámci HIL simulátoru

Cílem této práce je implementace knihovny (blocksetu) pro prostředí Matlab Simulink, která by rozšířila HIL simulátor o možnost vizualizace letových proměnných na různých platformách. Nedílnou součástí implementace knihovny bylo vytvoření podpůrných funkcí (bloků) umožňujících výměnu vybraných letových dat v rámci HIL simulátoru.

Součástí zadání byl požadavek, aby všechny implementované bloky podporovaly generování kódu přizpůsobeného pro spuštění na platformě vestavěného počítačového simulátoru (1). Tento požadavek představuje vytvoření TLC skriptů generujících kód v jazyce C vykazující stejnou funkčnost jako Simulinkové bloky. Obě tyto implementace musí splňovat požadavek zpracování dat v reálném čase. Tato problematika je blíže popsána v kapitole 4.2.3.

Výběr leteckého simulátoru a vhodného příslušenství

Prvním úkolem práce je výběr vhodného leteckého simulátoru pro PC platformu s vhodným komunikačním rozhraním, který by umožňoval vizualizaci letových proměnných dodaných externí aplikací. Dalším požadavkem je, aby letecký simulátor byl schopen zasílat externí aplikaci řídicí data získaná z řídicí páky, plynové páky a pedálů, která by dále ovlivňovala průběh simulace.

Dalším úkolem, souvisejícím s výběrem leteckého simulátoru, je specifikace požadavků na hardware a výběr vhodného příslušenství (řídicí páky, pedály).

Blok pro vizualizaci letových proměnných a zadávání řídicích dat

Následujícím krokem je implementace bloků pro Simulink, které by tvořily komunikační rozhraní mezi vybraným leteckým simulátorem a modelem v bloku Simulator (viz obrázek 1). Tyto bloky jsou součástí modelů vytvářených pro účely simulací prováděných na HIL simulátoru.

Vizualizace letových proměnných – IFDP

Součástí zadání je požadavek na navržení blok zajišťující plynulý datový tok mezi *integrovaným systémem zpracování letových dat (IFDP)* a vestavěným počítačovým simulátorem.

IFDP je dalším vizualizačním nástrojem, který je podrobněji popsán v kapitole 5.4. Tento systém umožňuje vizualizaci vybraných letových dat představujících orientaci letounu v prostoru, jeho polohu vzhledem k zemskému povrchu a některé další parametry jako rychlost, spotřebu paliva a další. V rámci HIL simulátoru představuje systém IFDP přístrojový panel simulovaného letadla.

Implementace podpůrných bloků

Nedílnou součástí integrace leteckého simulátoru FlightGear do HIL simulátoru je implementace podpůrných bloků různého určení. Tyto bloky umožňují vzájemné propojení dílčích částí HIL simulátoru, a implementují mezi těmito částmi cyklickou výměnu dat.

Demonstrace výsledků práce v Aero Vodochody a.s.

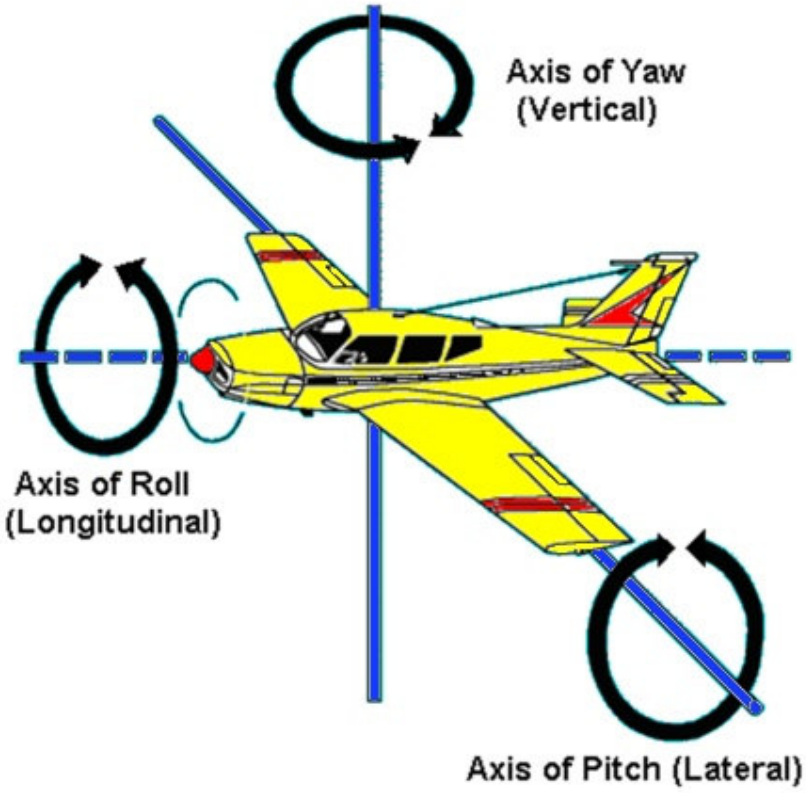
Posledním úkolem po dokončení implementace předchozích bodů je provedení a otestování vlastní integrace simulátoru FlightGear do HIL simulátoru a následná demonstrace funkčnosti výsledného řešení přímo ve firmě Aero Vodochody a.s.

1.5. Použitá terminologie

Poslední část této kapitoly je koncipována jako stručný vhled do letecké problematiky. Zde jsou vysvětleny základní termíny použité dále v rámci tohoto dokumentu.

1.5.1. Osy letounu a řídící prvky

Po vzletu letadla mění pilot pomocí jednotlivých řídících prvků pozici řídících ploch, a tím uvádí letadlo do rotačního pohybu. Tento rotační pohyb probíhá v trojrozměrném prostoru podle tří imaginárních os. Osy se protínají v těžišti letadla a jsou na sebe navzájem kolmé. Uspořádání os letadla názorně demonstruje obrázek 2.



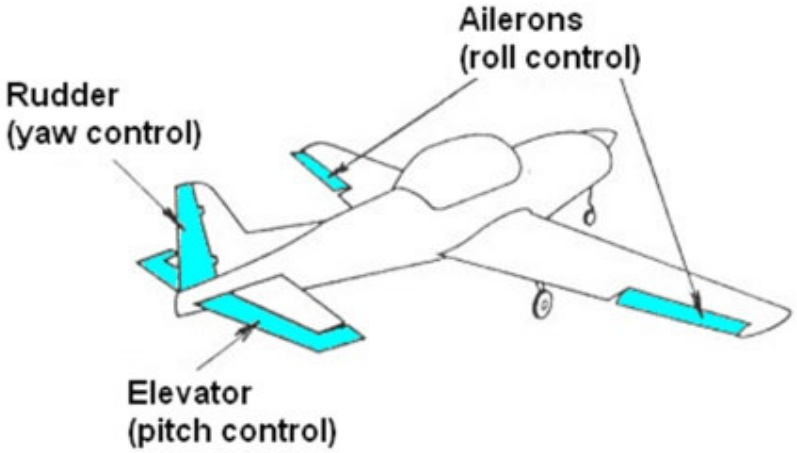
Obrázek 2: Osy letadla, převzato z (3)

Jednotlivé osy mají následující význam:

Osa procházející trupem letadla skrz ocas přímým směrem ven z kokpitu je osa *podélného náklonu* (***longitudinal/roll axis***). Pohyb v této ose je vyvolán změnou polohy (pohybem) *křidélek* (***ailerons***) umístěných na okraji křídel. Umístění jednotlivých řídících ploch je zobrazeno na obrázku 3.

Osa procházející oběma křídly je osa *příčného náklonu* (***lateral/pitch axis***). Pohyb v této ose je vyvolán pohybem respektive změnou polohy *výškového kormidla* (***elevator***) umístěného na zadním okraji zadních křídel.

Osa procházející těžištěm, kolmá na obě výše zmíněné osy, určuje *kurz* letadla (***vertical/yaw axis***). Pohyb v této ose je vyvolán změnou polohy *kormidla* (***rudder***) umístěného na zadní straně ocasní plochy letadla.



Obrázek 3:Řídící plochy letadla, převzato z (3)

2. Výběr leteckého simulátoru

Letecký simulátor slouží v simulačním řetězci jako vizualizační nástroj a také jako jednotka vstupních řídících dat. Je potřeba tedy vybrat takový simulátor, který je vybaven adekvátním vstupním a výstupním rozhraním.

V tomto případě je potřeba přenášet z leteckého simulátoru do HIL simulátoru řídící údaje (akční zásahy pilota) o výchylce *výškového kormidla* (***elevator***), *křidélek* (***ailerons***), kormidla (***rudder***) a údaje o aplikovaném tahu (***throttle***). Opačným směrem, tedy z HIL simulátoru do leteckého simulátoru se přenáší pozice letadla vzhledem k zemskému povrchu (*zeměpisná šířka*, *zeměpisná délka* a *nadmořská výška* – ***latitude***, ***longitude***, ***altitude***) a *Eulerovy úhly* (*kurz*, *příčný a podélný náklon* – ***yaw***, ***pitch***, ***roll***). Tyto veličiny stačí ke kompletnímu popisu polohy letadla v prostoru.

Zároveň je potřeba, aby komunikační protokol obsahoval prostor pro možné budoucí rozšíření komunikace mezi simulátory (hlavně směrem do leteckého simulátoru) o další stavové informace, jež budou vyčíslovány HIL simulátorem. Mezi tyto stavové informace patří například informace o rychlosti letounu, vertikální rychlost apod. Rozšíření o další stavové informace by v leteckém simulátoru umožnilo vizualizovat kompletní informace o stavu letounu na jednotlivých přístrojích.

Při výběru byly simulátory posuzovány hlavně podle vlastností jejich vstupně výstupních rozhraní. Princip simulace, simulační engine a další technické detaily týkající se implementace leteckého simulátoru nejsou z hlediska tohoto projektu podstatné.

Všechny posuzované programy obsahují v zásadě totožné nebo podobné funkce. Kromě simulace vlastního letu simulují věrně povětrnostní podmínky. Uživatel si může z Internetu stáhnout různé modely letadel, scenerii zahrnující většinu zemského povrchu a letišť, či si vytvářet své vlastní modely letadel. Tyto a další rozšiřující funkce či schopnosti daných programů nejsou z pohledu této práce rozhodující.

Hlavním kritériem pro výběr simulátoru byly vlastnosti rozhraní pro přístup ke stavovým veličinám simulovaného letounu. Z nich je potřeba načíst řídící veličiny (řídicí páka, kormidlo apod.) a poté je potřeba mít možnost stavové veličiny upravovat. Druhým hlavním kritériem bylo, aby vybraný simulátor byl stabilní projekt s dostupnou dokumentací a podporou, kterou by bylo možné při realizaci této práce využít.

Těmto kritériím odpovídají tři simulátory. Ty lze rozdělit do dvou skupin. První skupinou jsou komerční letecké simulátory, do kterých spadá X-Plane a Microsoft Flight Simulator X. Druhou skupinu tvoří open source projekty, a z ní pro tuto práci připadá v úvahu pouze simulátor FlightGear.

2.1. FlightGear

FlightGear je open source letecký simulátor. Vývoj simulátoru byl započat v roce 1996 a stále probíhá. Simulátor podporuje všechny rozšířené PC platformy od Windows, Linuxu až po MacOS a Solaris.

Cílem tohoto projektu je nabídnout alternativu ke stávajícím komerčním simulátorům, kterou je možné využít pro výzkumné a vědecké účely. Problémem komerčních simulátorů byla, hlavně dříve, jejich malá rozšiřitelnost, a tím pádem i malá využitelnost pro výzkumné

aplikace. FlightGear je vyvíjen jako alternativa, kterou lze volně modifikovat a rozšiřovat. Celý program i kompletní zdrojový kód je volně dostupný na Internetu a podléhá GNU-GPL licenci.

2.1.1. Stavové informace

Stavové informace jsou ve FlightGearu koncentrovány v takzvaném *Property Tree*. Property Tree zajišťuje přístup k velkému množství stavových informací, které jsou řazeny do různých kategorií podle toho, k čemu se vztahují. Změnou hodnot stavových proměnných lze jednoduše měnit parametry simulace za běhu programu. K Property Tree se dá přistupovat následujícími způsoby:

- Programově z kódu napsaném v C/C++, který je zkompilován v rámci aplikace FlightGear.
- Pomocí Nasal skriptů. Nasal je skriptovací jazyk implementovaný v rámci aplikace FlightGear.
- Pomocí socketů, které umožňují příjem a vysílání dat z FlightGearu.
- Přes rozhraní TelNet
- Přes HTML rozhraní

2.1.2. Rozhraní

FlightGear disponuje pestrou paletou komunikačních protokolů. Mezi těmito protokoly lze nalézt protokoly s pevným datovým rámcem, protokoly u nichž lze volit strukturu přenášených dat a protokoly, které jsou určeny pro komunikaci se specifickým zařízením nebo aplikací.

Dále jsou zmíněny pouze protokoly významné pro tuto práci. Zbylé protokoly, jimiž FlightGear disponuje, nebyly pro tento projekt relevantní nebo byly méně vhodné, než ty zmíněné níže. Všechny uvedené protokoly lze buď přenášet po síti Ethernet, po sériové lince nebo jejich pakety ukládat do souborů či načítat ze souborů.

Generic

Komunikační protokol *generic* umožňuje vytvořit vstupní/výstupní protokol podle definice v konfiguračním XML souboru. Tento konfigurační soubor se poté uloží do adresáře $FG_ROOT/Protocol/ a od té chvíle ho lze využívat.

Pomocí tohoto protokolu lze číst hodnoty vybraných stavových veličin z Property Tree nebo je tam zapisovat. To, jaké stavové veličiny bude paket obsahovat, definuje konfigurační XML soubor. Data zasílaná protokolem mohou být v ASCII, binárním nebo XML formátu.

Příklad definice jednoduchého protokolu

Takto vypadá XML konfigurace jednoduchého protokolu, kterým jsme schopni z FlightGearu číst stavové proměnné rychlost v uzlech a kurz letadla v radiánech. Data jsou v ASCII formátu a oddělení jednotlivých paketů je znak nové řádky ‚\n‘.

```
<?xml version="1.0"?>
<PropertyList>
<generic>
```

```
  <output>
    <line_separator>newline</line_separator>
    <var_separator>newline</var_separator>
    <binary_mode>false</binary_mode>

    <chunk>
      <name>speed</name>
      <format>V=%d</format>
      <node>/velocities/airspeed-kt</node>
    </chunk>

    <chunk>
      <name>heading (rad)</name>
      <format>H=%.6f</format>
      <type>float</type>
      <node>/orientation/heading-deg</node>
      <factor>0.0174532925199433</factor>
    </chunk>
  </output>

</generic>
</PropertyList>
```

Native-Ctrls

Native-Ctrls je protokol s pevnou strukturou paketu. Paket je tvořen strukturou v jazyce C respektive C++ a obsahuje následující významné stavové informace:

- hodnoty řídících vstupů (křidélka, kormidlo, klapky apod.)
- stavové informace motoru(ů)
- poruchy motoru(ů)
- informace o stavu palivových nádrží
- informace o větru a turbulencích
- informace o tlaku a teplotě

Native-FDM

Native-FDM je protokol s pevnou strukturou paketu. Paket je tvořen stejně jako u Native-Ctrls strukturou v jazyce C/C++. Paket obsahuje následující významné stavové informace:

- zeměpisnou pozici a nadmořskou výšku
- orientace letadla v prostoru (Eulerovy úhly)
- rychlosti v různých osách (rychlost letounu, vertikální rychlost apod.)
- stav motoru(ů)
- pozice řídících ploch letounu (kormidlo, klapky apod.)

V případě této úlohy by komunikace pomocí zvoleného protokolu, definovaného simulátorem FlightGear, probíhala prostřednictvím sítě Ethernet, protože to je jediné rozhraní podporované jak počítačem BOA5200 tak simulátorem FlightGear. Jako přenosové protokoly lze využít jak TCP tak UDP.

2.2. X-Plane

X-Plane je komerční letecký simulátor vyvíjený firmou Laminar Research. Simulátor podporuje platformy PC, s operačním systémem Windows, Linux, MacOS, a platformu iPhone/iPod Touch. X-Plane je nabízen i v profesionální verzi, která nabízí rozšířené a velmi kvalitní možnosti vizualizace a simulace. Díky tomu je tento simulátor využíván i některými společnostmi jako pomůcka při procesu návrhu komponent pro letecký průmysl či jako nástroj pro tréninkové či výzkumné účely.

Uživatelům je umožněno vytvářet pro simulátor vlastní plugin moduly, kterými lze rozšířit jeho rozhraní, letecký model nebo vytvořit zcela nové funkce.

2.2.1. Stavové informace

X-Plane umožňuje přístup k velkému spektru stavových veličin. Ty jsou opět rozděleny do kategorií, podle toho, k čemu přísluší (rychlost, pozice apod.). Tyto informace lze za běhu programu číst i do nich zapisovat. Ke stavovým proměnným lze přistupovat vytvořením pluginu nebo přes UDP rozhraní.

2.2.2. Rozhraní

Ke stavovým informacím (proměnným) lze přistupovat dvěma způsoby. Oba způsoby jsou nastíněny v HTML souboru `Sending Data to X-Plane.html` popisujícím výměnu dat mezi X-plane a externími aplikacemi, který je přílohou této práce. Tento dokument odkazuje i na dokumentaci k vývoji pluginu pro X-Plane (4). K simulátoru X-Plane je pro tyto účely k dispozici SDK, jehož součástí je API umožňující čtení a zápis stavových dat z/do X-Plane.

Tím se také dostáváme k prvnímu způsobu získání stavových dat z X-Plane, kterým je vytvoření vlastního pluginu. Nově vytvořený plugin může číst stavové informace, a ty lze potom zpracovat libovolným způsobem v rámci pluginu. Tedy například zasílat dál prostřednictvím nějaké sítě či rozhraní.

Druhý způsob přístupu ke stavovým informacím je možný pomocí UDP protokolu. Odhlédneme-li od tvorby pluginů X-Plane disponuje pouze jedním, a to UDP rozhraním, jehož prostřednictvím lze stavové proměnné číst nebo do nich zapisovat. To, jaké proměnné se zasílají ze simulátoru do externí aplikace, lze definovat přímo v konfiguraci simulátoru po jeho spuštění.

Níže je přiblížen způsob výměny dat mezi simulátorem X-Plane a externí aplikací přes rozhraní s UDP protokolem. Všechny zprávy vyměněné mezi těmito dvěma stranami mají následující formát:

- Message Prologue - Identifikátor zprávy – 5 znaků
- Data Input Structure – Struktura obsahující data zprávy, kterou chceme přijmout nebo odeslat.

Pomocí těchto zpráv můžeme provádět následující úkony: přijímat nebo odesílat zpět hodnoty vybraných stavových informací, měnit konfiguraci zasílání stavových dat (tedy aktivovat zasílání určitých stavových informací nebo je deaktivovat), vyvolat poruchu určitého zařízení či je opět „opravit“ a řadu dalších. Kompletní seznam zpráv pro komunikaci s X-Plane lze nalézt v souboru `Sending Data to X-Plane.html` na přiloženém CD.

2.3. Microsoft Flight Simulator X

Microsoft Flight Simulator je druhým zástupcem z kategorie komerčních leteckých simulátorů. Flight Simulator je produkt vyvíjený již od roku 1982 a Flight Simulator X je jeho desátá verze uvolněná pro americký trh v roce 2006. Tento simulátor podporuje pouze platformu PC s operačním systémem Windows XP nebo vyšší.

Microsoft také poskytuje vývojářům třetích stran SDK (5) pro vývoj nástaveb pro Flight Simulator. K němu poskytuje kompletní volně dostupnou dokumentaci prostřednictvím MSDN (Microsoft Developer Network) (5). Flight Simulator X je postaven na komerční simulační vývojové platformě Microsoft ESP (6), která je využívána i dalšími společnostmi při vývoji leteckých aplikací.

2.3.1. Stavové informace

Microsoft Flight Simulator X umožňuje stejně jako předešlé simulátory přístup k širokému spektru stavových informací rozdělených do kategorií. Kompletní přístup k těmto stavovým proměnným je možný přes rozhraní *SimConnect*.

2.3.2. Rozhraní SimConnect

Součástí vývojových nástrojů, které jsou distribuovány s produktem Flight Simulator X je komunikační rozhraní SimConnect (5). Rozhraní SimConnect umožňuje asynchronní přístup k stavovým proměnným a událostem uvnitř simulátoru.

Rozhraní SimConnect je implementováno na základě architektury klient-server. Komunikace mezi klientem (aplikace třetích stran) a serverem (Flight Simulator X) probíhá pomocí protokolu TCP/IP. SimConnect dále poskytuje programátorské API pro tvorbu klientských aplikací ve formě DLL knihovny.

2.4. Srovnání, výběr simulátoru

2.4.1. Srovnání

Při srovnání získaných poznatků o jednotlivých simulátorech začneme u skupiny komerčních simulátorů. Ty disponují solidním komfortem pro vývojáře nástavbových aplikací. K simulátoru poskytují SDK a kompletní dokumentaci (k SDK i vlastnímu simulátoru). Pokud ale chceme využívat tyto SDK je potřeba je nejdříve nainstalovat a nastavit správně vývojové prostředí, ve kterém programujeme.

Využijeme-li simulátor **X-plane,** můžeme se vydat dvěma cestami řešení. Tou první je implementace vlastního pluginu pro X-Plane, který by načítal hodnoty pro nás zajímavých stavových proměnných, a ty dále zpracoval. To by vyžadovalo určité předzpracování (převod na jiné jednotky apod.) a následné zaslání po síti pomocí protokolu TCP nebo UDP. Pro tyto účely by bylo potřeba navrhnout vhodný protokol.

Druhou cestou je využití UDP rozhraní simulátoru X-Plane a protokolu na něm implementovaném. Po zvolení této varianty je nutné naprogramovat dekodér a kodér UDP zpráv používaných při výměně dat se simulátorem.

Simulátor **Flight Simulator X** představuje oproti X-Plane o něco komplexnější řešení. Poskytované SDK, a tedy i programátorské API nabízí širší paletu funkcí a nástrojů. Je zde tedy o něco lepší podpora pro vývoj aplikací třetích stran než u X-Plane, ale ta se vzhledem k rozsahu této práce nijak markantně neprojeví. Při využití Flight Simulatoru je jediná možnost, jak realizovat výměnu stavových informací mezi simulátorem a externí aplikací, a tou je vytvořit klientskou aplikaci využívající rozhraní SimConnect.

FlightGear, jediný zástupce kategorie open source simulátorů, na rozdíl od komerčních simulátorů neposkytuje pro vývojáře externích aplikací žádné SDK. Bohužel neposkytuje ani kompletní dokumentaci, která by byla dostupná na Internetu, kde by byly popsány na jednom místě všechny věci technického rázu (nastavení různých konfiguračních XML, programátorská dokumentace, apod.). Naštěstí lidé zabývající se jeho vývojem jsou velmi ochotní a není problém od nich získat potřebné informace.

Na druhou stranu tento simulátor disponuje dobře využitelným rozhraním. Pro tuto práci lze využít datovou výměnu pomocí protokolů *NativeCtrls* a *NativeFDM*. Tyto protokoly jsou reprezentovány neměnnými datovými strukturami v jazyce C respektive C++, a tudíž není třeba tyto pakety dekódovat, ale lze přistupovat přímo k jednotlivým proměnným. Tyto protokoly také obsahují pestrou paletu stavových proměnných, pro která přenášejí data, takže při případném rozšiřování komunikace o další stavy není potřeba měnit protokol přenosu.

Tím, že tento simulátor neposkytuje SDK, odpadá problém s jeho instalací a nastavováním nějakého vývojového prostředí, což lze brát v rámci této práce jako plus, protože záběr této práce není tak veliký, aby plně využil možnosti API poskytovaných komerčními simulátory.

2.4.2. Výběr

Jak již bylo dříve řečeno, všechny tři simulátory poskytují z hlediska tohoto projektu dostatečné vizualizační nástroje a komfort pro obsluhu simulátoru. Specifické rozdíly ve vlastnostech simulátorů nejsou v tomto projektu důležité.

Pro účely většího projektu by byl nejvhodnější Microsoft Flight Simulator X, který poskytuje nejlepší programátorskou podporu. Náš projekt, ale není natolik složitý, aby tyto silné ale zároveň komplikované nástroje naplno využil. Tento simulátor byl tedy z výběru vyřazen.

Při rozhodování, zda použít X-Plane nebo FlightGear, bylo rozhodnuto podle následujících kritérií. X-Plane podobně jako Flight Simulator X poskytuje dobré SDK. U FlightGearu byla sice absence SDK, ale možnosti poskytované simulátorem pro tuto úlohu byly v podstatě totožné se simulátorem X-Plane. U simulátoru FlightGear byla také výhoda, že s ním již měli zkušenosti lidé z Katedry řídicí techniky, tudíž s nimi bylo možno osobně konzultovat případné problémy.

Po zvážení výhod poskytovaného SDK a výhod plynoucích z možnosti využít zkušeností kolegů, jsem se rozhodl, že druhá varianta mi přinese větší užitek. Výběr tedy padl na simulátor **FlightGear**. To, že FlightGear neposkytuje žádné SDK ani API realizující spolupráci se simulátorem, nepředstavovalo nakonec při implementaci Simulinkového blocksetu žádný zásadní problém.

3. Příslušenství k leteckému simulátoru

Příslušenství k leteckému simulátoru tvoří ovládací páka s plynovou pákou a letecké pedály. Toto příslušenství bylo vybráno na základě požadavků firmy Aero Vodochody, které zahrnovaly hlavně kvalitní a realistické zpracování věrně simulující ovládací prvky reálného letounu, na které byl kladen veliký důraz.

Po zvážení dostupných produktů byly nakonec vybrány produkty *Hotas Cougar* od firmy Thrustmaster a *Rudder pedals* od firmy PFC.

Thrustmaster Hotas Cougar

Jedná se o věrnou repliku ovládacích prvků z letounu F-16. Tato sestava obsahuje ovládací a plynovou páku. Na obou ovládacích prvcích je dohromady 28 plně programovatelných tlačítek, jež se programují pomocí speciálního přiloženého softwaru. Vše je zpracováno v masivním kovovém provedení. Hotas Cougar se připojuje k PC pomocí jednoho USB konektoru a vyžaduje instalaci ovladačů přiložených na CD.

PFC Rudder Pedals

Tyto masivní ocelové pedály přesně odpovídají požadavkům na věrnou simulaci reálných leteckých pedálů díky přiměřené tuhosti ve všech osách a adekvátnímu mechanickému zpracování. Pedály se připojují k PC pomocí USB, a není pro ně třeba instalovat žádný speciální ovladač.

4. Rozhraní modelu pro HIL simulaci

Rozhraní modelu v Simulinku pro HIL simulaci vytvoříme tak, že implementujeme blok, který toto rozhraní v modelu reprezentuje. Pro tento blok bude spolu s modelem vygenerován kód, který bude spustitelný na vestavěném počítači v HIL simulátoru. V této kapitole je nastíněn postup vytvoření takového bloku.

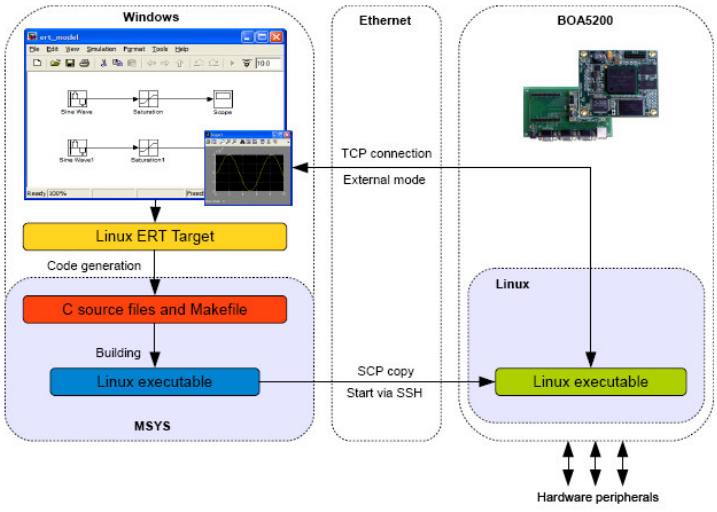
4.1. Základní principy

Generování kódu libovolného Simulinkového modelu umožňuje nástroj Real-Time Workshop, který je součástí instalace Matlabu. Tento nástroj poskytuje uživateli možnost přenosu návrhu řešení specifického problému ve formě Simulinkového modelu na cílovou platformu (v dokumentaci nazvanou *Target*).

Přenos řešení spočívá ve vygenerování kódu v jazyce C, který obsahuje stejnou funkčnost jako původní model. Síla tohoto nástroje spočívá v tom, že je možné vygenerovat kód, který lze přizpůsobit v podstatě libovolné cílové platformě. To, jakým způsobem se má vygenerovaný kód přizpůsobit, aby byl kompatibilní s cílovou platformou (s její hardwarovou konfigurací, operačním systémem apod.) určuje takzvaný *target,* definovaný v rámci Matlabu. Pozor nezaměňujte tento target s Targetem představujícím cílovou platformu. Bohužel tímto způsobem je nastavena terminologie v rámci dokumentace k targetu využitému v této práci.

V HIL simulátoru je použit jako cílová platforma vestavěný počítač BOA5200 (1) s operačním systémem Linux. Pro přizpůsobení generovaného kódu této platformě je využit target *Linux ERT Target*. Tento target byl navržen Pavlem Jelínkem v rámci jeho diplomové práce (7) na Katedře řídicí techniky a dále vyvíjen Lukášem Hamáčkem v rámci diplomové práce (8) pro tutéž katedru. Bližší informace o tomto targetu lze nalézt ve zmíněných pracích.

Princip generace kódu pomocí Real-Time Workshop pro platformu BOA5200 a jeho spuštění na této platformě znázorňuje obrázek 4, který následuje.



Obrázek 4: Princip generování a spuštění kódu, převzato z (8)

Výstupem procesu generace kódu jsou soubory se zdrojovým kódem a makefile. Zdrojový kód je s využitím makefilu zkompilován, poté nahrán na cílovou platformu a spuštěn.

4.2. Proces implementace bloku s podporou generování kódu

V dalších bodech této kapitoly je stručně nastíněno, co obnáší implementace Simulinkového bloku včetně podpory generování kódu pro platformu BOA5200. Podrobnosti o problematice jednotlivých níže uvedených bodů lze nalézt v dokumentaci (nápovědě) k Matlabu (9) v příslušných sekcích.

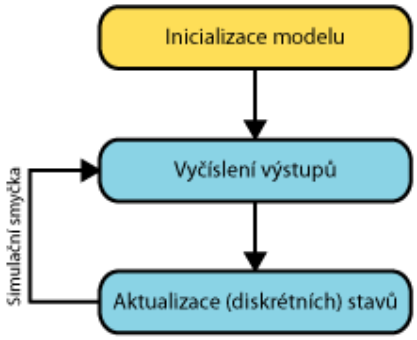
4.2.1. Implementace S-funkce pro simulaci

Funkcionalita každého bloku je implementována v S-funkci v jazyce C/C++. S-funkci je potřeba před použitím zkompilovat do MEX souboru. S-funkce jsou tvořené takzvanými *callback metodami*. Tyto metody obsahují kód vykonávaný v různých fázích zpracování Simulinkového modelu.

Zpracování modelu probíhá postupně po jednotlivých fázích. První fází je inicializace modelu. V této fázi se určí pořadí zpracování jednotlivých bloků, vyhodnotí se jejich vstupní parametry, přiřadí se jim paměť atd.

Po té Simulinkový engine přejde do *simulační smyčky,* ve které jsou jednotlivé fáze nazývány *simulačními kroky.* V každém simulačním kroku jsou bloky zpracovávány v pořadí určeném během inicializační fáze. Zpracováním se míní zavolání příslušné callback metody odpovídající simulačnímu kroku.

Následující obrázek ilustruje simulační kroky využité pro implementaci bloků knihovny FlightGear Blockset. Simulační smyčka je v tomto případě zjednodušená oproti smyčce obecných S-funkcí. Důvodem je to, že ne všechny simulační kroky obecné S-funkce jsou implementovanými bloky využity.



Obrázek 5: Zpracování bloků FlightGear blocksetu Simulinkovým enginem.

Většina bloků z knihovny FlightGear blockset implementuje následující callback metody. Tyto metody jsou vykonávány během průchodu simulační smyčkou zobrazenou na obrázku 5.

mdlInitializeSizes	Definice vlastností bloku (počet vstupů/výstupů, stavů apod.)
mdlInitializesampleTimes	Specifikace periody vzorkování a souvisejících parametrů.
mdlStart	Volá se jednou při začátku simulace modelu.
mdlOutputs	Vyčíslení výstupů.
mdlUpdate	Obvykle aktualizace diskrétních stavů.
mdlRTW	Zde se předávají parametry do RTW souboru.
mdlTerminate	Uvolnění prostředků využívaných S-funkcí (zavření socketů, uvolnění paměti apod.).

Tabulka 4.1.: Použité callback metody.

Více informací o S-funkcích a související problematice lze nalézt v (9) v sekci *Simulink/Writing S-functions in C*.

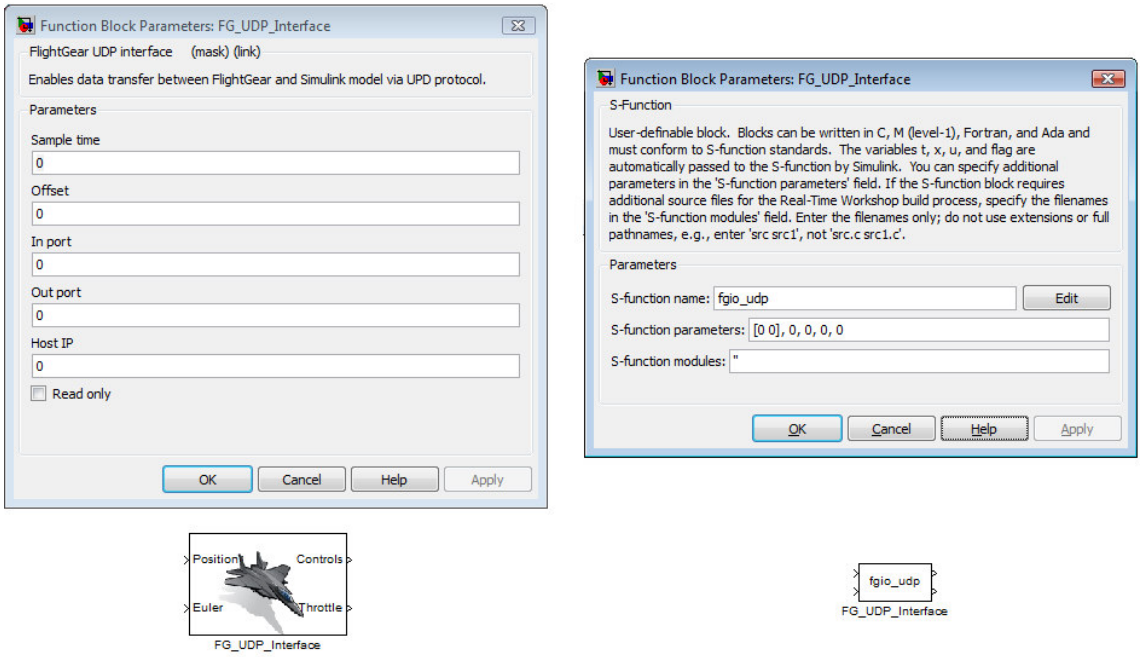
4.2.2. Vytvoření masky pro nastavení parametrů

Takzvaná *maska* je grafické rozhraní Simulinkových bloků, které uživateli přehledně zobrazuje vstupní parametry bloku (S-funkce) a jejich aktuální hodnoty. Na níže uvedeném obrázku ukážeme rozdíl bloku S-funkce s maskou a bez masky. Na obrázku jasně vidíme výhody použití masky.

Pokud nevyužijeme masku, uživatel musí přesně vědět, co jaký vstupní parametr S-funkce znamená, aby všechny parametry nastavil správně. To samé platí pro vstupy a výstupy bloku, protože bez aplikace masky jsou všechny porty bloku bez popisu. Vytvoření masky uživateli práci s blokem velmi zjednoduší.

Součástí masky je i *dokumentace* k danému bloku, k níž lze přistoupit přes tlačítko *Help* v konfiguračním dialogu nebo přes kontextové menu. Dokumentace s popisem daného bloku se vytváří formou HTML souboru pojmenovaného stejně jako S-funkce bloku. Soubor obsahující dokumentaci je vytvořen v rámci implementace bloku, a k bloku se připojí při vytváření masky.

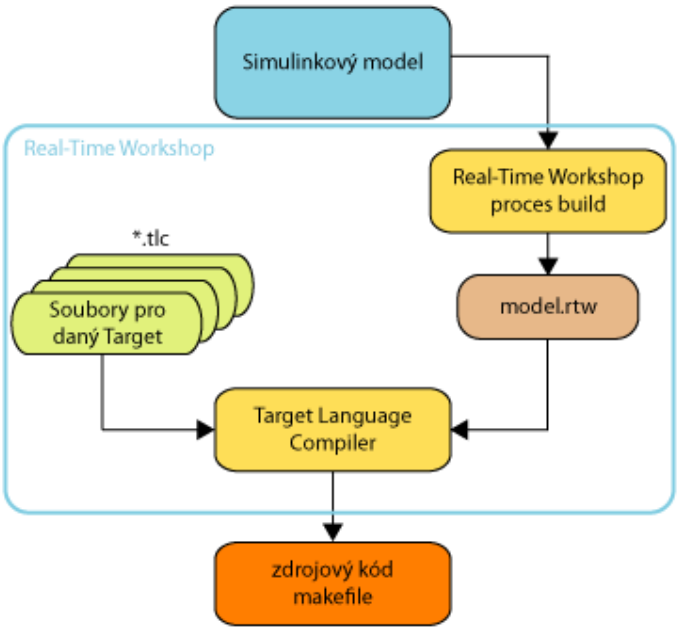
Postup vytvoření masky je podrobně popsán v dokumentaci k Matlabu (9) v sekci *Simulink/Creating Block Masks*.



Obrázek 6: Blok S-funkce s maskou (vlevo) a blok S-funkce bez masky (vpravo).

4.2.3. Implementace TLC skriptu

Na začátku této kapitoly bylo řečeno, že generování kódu v jazyce C pro cílovou platformu (v našem případě BOA5200 (1)) provádí Real-Time Workshop. Podívejme se nyní stručně, jak proces generování probíhá. Celý tento proces generování kódu je znázorněn na obrázku 7.



Obrázek 7: Přehled procesu generování kódu

Prvním krokem tohoto automatizovaného procesu je vygenerování souboru `model.rtw` respektive `název_modelu.rtw`. Tento soubor obsahuje všechny informace o modelu, které jsou nezbytné pro generování kódu ze Simulinkového modelu. Tento soubor je předán interní části Real-Time Workshopu, Target Language Compileru (TLC).

Výstupem Target Language Compileru jsou zdrojové soubory v jazyce C a makefile. Target Language Compiler před vytvořením těchto souborů zpracuje soubor `model.rtw` spolu s *TLC* soubory upravující generovaný kód pro cílovou platformu a *TLC* soubory obsahující funkcionalitu daného bloku. A právě implementace *TLC* souborů obsahujících funkcionalitu byl další krok při tvorbě bloků zahrnutých ve FlightGear blocksetu. TLC soubory jsou uloženy v adresáři `tlc_c`.

Více informací o S-funkcích a související problematice lze nalézt v (9) v sekci *Real-Time Workshop*.

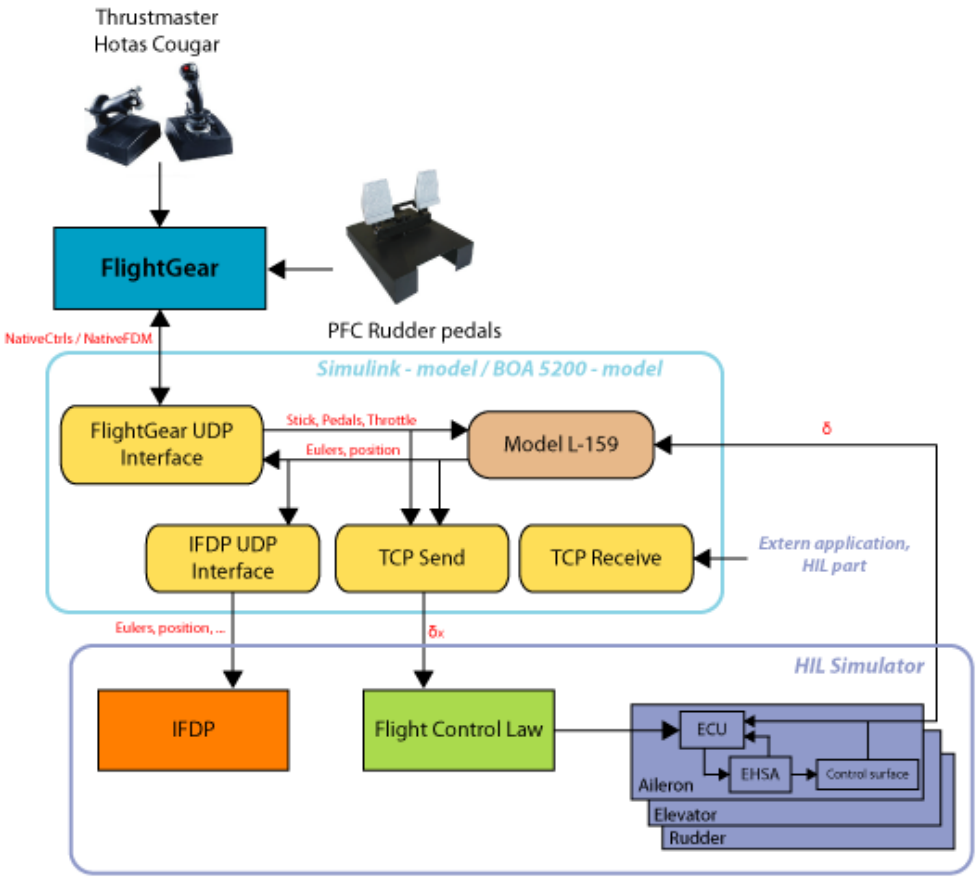
5. FlightGear Blockset

FlightGear blockset je knihovna bloků umožňující zapojení specifických externích aplikací a zařízení do modelů vytvářených v prostředí Simulink. Blockset umožňuje integrovat do modelu letecký simulátor FlightGear, integrovaný systém zpracování letových dat (IFDP) firmy Speel s.r.o., a další externí aplikace. Externími aplikacemi jsou myšleny takové aplikace, které zpracovávají či distribuují prostřednictvím speciálního protokolu letecká data. Všechny níže uvedené bloky jsou definovány v knihovně `fg_blockset.mdl`.

Instalací knihovny do prostředí Simulink získáme tyto hlavní funkce. První je čtení aktuálních hodnot stavových veličin z leteckého simulátoru. Dále je to možnost distribuovat data stavových veličin leteckému simulátoru a vývoj těchto dat vizualizovat ve formě klasického letu. Třetí významnou funkcí je možnost vizualizace specifické palety leteckých dat prostřednictvím IFDP systému či poskytovat tyto data aplikacím třetích stran.

Obrázek 8 názorně ukazuje začlenění jednotlivých bloků do modelu v Simulinku a tím i do HIL simulátoru. Na obrázku jsou bloky z knihovny FlightGear blockset vybarveny žlutou barvou. Každý blok tvoří rozhraní jiného typu mezi různými částmi HIL simulátoru.

Bloky ECU a EHSA, znázorněné na obrázku 8, ovládají řídící plochy. Blok ECU je elektronická řídící jednotka a blok EHSA je elektrohydraulický akční člen (servomechanismus).



Obrázek 8: Začlenění FlightGear blocksetu do HIL simulátoru

5.1. Instalace knihovny

Knihovnu instalujeme jednoduchým způsobem. Do zvoleného adresáře stačí rozbalit archiv, ve kterém je obsažená a z příkazové řádky Matlabu spustit skript `fg_blockset_install.m`. Instalace spočívá v přidání cesty ke knihovně do `Matlab path`. Postup pro instalaci BOA5200 a prostředí pro generování a překlad kódu lze nalézt v (8) v kapitole 2.

5.2. Společné vlastnosti pro všechny bloky

Všechny bloky jsou implementovány S-funkcemi typu *C MEX S-function* úrovně 2. S-funkce úrovně 2 na rozdíl od funkcí úrovně 1 mohou využívat větší část API Simulinku.

5.2.1. Komunikační rozhraní

Všechny bloky knihovny komunikují s ostatními bloky nebo externími aplikacemi (modely) pomocí protokolu TCP nebo UDP. Pro tuto síťovou komunikaci S-funkce využívají knihovnou `wsock32.dll` obsahující API pro síťovou komunikaci.

5.2.2. Generování kódu – předávání parametrů

Všechny vstupní parametry S-funkce se předávají mezi blokem v Simulinkovém modelu a vygenerovanými zdrojovými soubory. Předání probíhá vytvořením parametrů v `název_modelu.rtw`, do kterých se zapíšou hodnoty vstupních parametrů S-funkce. Ty jsou do RTW souboru zapsány v callback metodě `mdlRTW`. Tato metoda se volá pouze při generování kódu. Z TLC skriptu můžeme poté jednoduše přistupovat k parametrům vytvořeným v RTW souboru pomocí TLC příkazů.

5.2.3. Generování kódu – společné funkce

U všech bloků kromě `IFDP UDP Interface` jsou při zpracování souboru `název_bloku.tlc` generovány soubory `additionals_functions.c` a `additionals_functions.h`. Tyto soubory obsahují deklaraci pomocných funkcí využitých při zasílání dat po síti (změna z big endian na little endian apod.). Při generování kódu pro bloky z FlightGear blocksetu je kontrolováno, zda soubory s pomocnými funkcemi `additionals_functions.c` a `additionals_functions.h` nebyly již vygenerovány jiným blokem obsaženým v daném Simulinkovém modelu. Pokud ano, již se znova negenerují.

5.3. Blok FlightGear Interface

Blok `FlightGear Interface` je základem knihovny `FlightGear blockset`. Tento blok umožňuje obousměrnou komunikaci a cyklickou výměnu dat mezi modelem v Simulinku a leteckým simulátorem FlightGear spuštěným na PC prostřednictvím sítě Ethernet. Blok existuje ve dvou variantách lišících se komunikačním protokolem. První variantou je implementace využívající protokol TCP. Naproti tomu druhá varianta využívá při komunikaci protokol UDP. V ostatních ohledech je funkčnost obou variant stejná až na drobné detaily vyplívající z rozdílných vlastností TCP a UDP protokolu. Soubory tvořící tyto bloky jsou uvedeny v tabulce 1.

S-funkce	fgio_tcp.cpp fgio_udp.cpp
S-funkce MEX	fgio_tcp.mexw32 fgio_udp.mexw32
TLC soubory	fgio_tcp.tlc fgio_udp.tlc
Help soubor	fgio_tcp_help.html fgio_udp_help.html

Tabulka 1: Soubory, které jsou součástí bloků FlightGear Interface.

5.3.1. Vstupy bloku

Blok má dva vstupy, a to pozici letounu vzhledem k zemskému povrchu a úhly natočení podle jednotlivých os určující orientaci letadla ve vzduchu. Data z těchto vstupů jsou posléze zasílána do leteckého simulátoru.

Pozice vzhledem k zemskému povrchu

Tento vstupní parametr je reprezentován vektorem obsahujícím tři veličiny určující pozici letounu.

Veličina	Jednotka
Zeměpisná šířka (latitude)	rad
Zeměpisná délka (longitude)	rad
Nadmořská výška (altitude)	m

Orientace ve vzduchu

Tento vstupní parametr je reprezentován vektorem obsahujícím tři veličiny určující orientaci letounu ve vzduchu.

Veličina	Jednotka
Podélný náklon (roll)	rad
Příčný náklon (pitch)	rad
Kurz letounu (yaw)	rad

5.3.2. Výstupy bloku

Blok má dva výstupy. Výstupem bloku jsou řídící zásahy uživatele zadané prostřednictvím leteckého simulátoru. Ty reprezentují požadavky na nastavení řídících ploch letounu, které jsou sloučeny do jednoho vektoru tvořící první výstup, a velikosti aplikovaného tahu, jež tvoří druhý výstup.

Nastavení řídících ploch

Tento výstupní parametr je reprezentován vektorem obsahujícím hodnoty nastavení řídících ploch letounu.

Veličina	Jednotka	Rozsah
Nastavení křidélek (aileron)	-	-1 : 1
Nastavení výškového kormidla (elevator)	-	-1 : 1
Nastavení směrového kormidla (rudder)	-	-1 : 1

Tah

Bezrozměrná veličina v rozsahu 0 : 1.

5.3.3. Vstupní parametry S-funkce

S-funkce `fgio_udp` respektive `fgio_tcp`, která implementuje funkcionalitu tohoto bloku má pět vstupních parametrů. Všechny vstupní parametry jsou povinné a je potřeba je zadat pro správnou funkci bloku.

Sample time

Tento parametr je dvouprvkový vektor. Prvním prvkem je *perioda vzorkování* a druhým prvkem je *offset* periody vzorkování.

In port

Port, na kterém blok přijímá data od leteckého simulátoru FlightGear.

Out port

Port, na kterém blok vysílá data směrem do leteckého simulátoru FlightGear.

Host IP

IP adresa stroje, na kterém je spuštěn letecký simulátor FlightGear.

Read only

Pokud je tento parametr nastaven na hodnotu 1 (true), tak se deaktivují vstupní porty bloku `FlightGear Interface`, a tento blok poté slouží pouze ke čtení dat distribuovaných leteckým simulátorem. Pokud je parametr nastaven na hodnotu 0 (false), blok přijímá i vysílá data.

5.3.4. Princip funkce

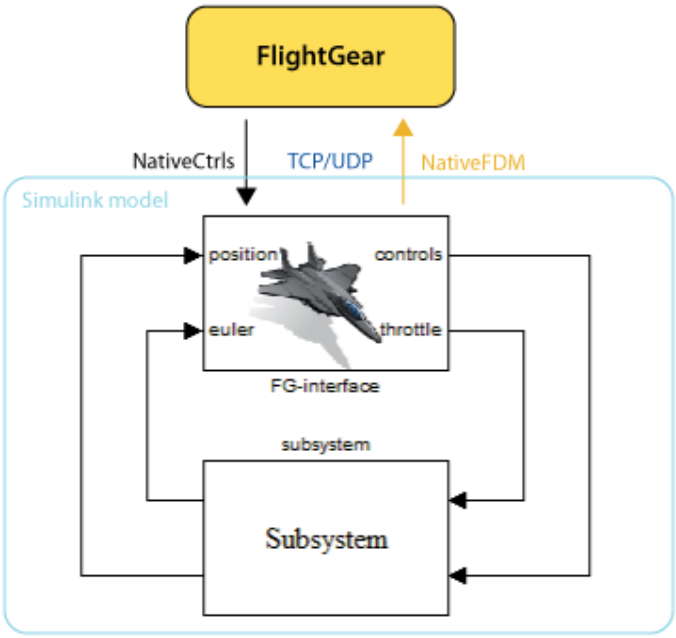
Základní úlohou tohoto bloku je cyklická výměna dat mezi leteckým simulátorem FlightGear a modelem v prostředí Simulink. Ta, jak již bylo řečeno, probíhá po síti Ethernet za použití protokolu TCP respektive UDP.

Letecký simulátor FlightGear definuje sadu různých protokolů pro datovou výměnu s externími aplikacemi. Při implementaci tohoto bloku byly použity protokoly `NativeCtrls` a `NativeFDM`. Pakety těchto protokolů mají pevně daný formát a jsou realizovány strukturami v jazyce C/C++. `NativeCtrls` obsahuje aktuální hodnoty řídících veličin plus

hodnoty dalších souvisejících veličin, a slouží k přenosu dat z leteckého simulátoru do externí aplikace. Naproti tomu `NativeFDM` obsahuje hodnoty veličin dodaných externím modelem/aplikací, a slouží k přenosu dat do simulátoru

V rámci integrace do prostředí Simulink není v simulátoru FlightGear využit interní model letounu a simulátor je přepnut do režimu, kdy očekává data od externího modelu. Externím modelem je v tomto případě například HIL simulátor respektive jeho reprezentace modelem v prostředí Simulink. Postup nastavení simulátoru FlightGear pro spolupráci s tímto blokem je dále popsán v příloze B.

Následující obrázek názorně zobrazuje schéma komunikace rozhraní tvořeného blokem `FlightGear Interface`. Datové pakety jsou ze simulátoru FlightGear zasílány s neměnnou frekvencí, která je nastavena při spuštění simulátoru. S touto frekvencí jsou také simulátorem přijímány pakety zaslané blokem `FlightGear Interface`.

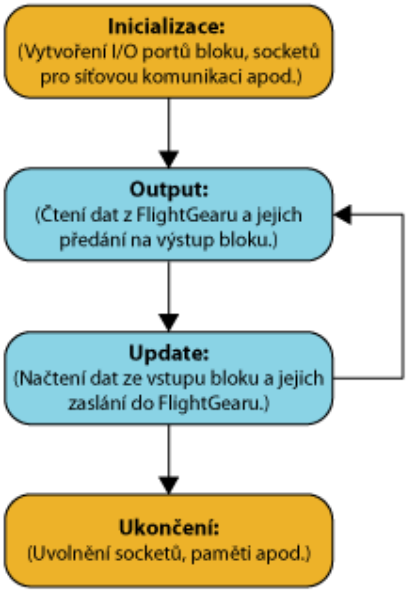


Obrázek 9: Komunikace mezi blokem FlightGear Interface a simulátorem FlightGear

5.3.5. Implementace bloku

Implementace tohoto bloku se nachází v souborech `fgio_tcp.cpp` a `fgio_udp.cpp`. Jak je vidět již podle názvu, první soubor implementuje blok komunikující po TCP a druhý blok komunikující po UDP.

S funkce `fgio_xxx` implementuje všechny standardní inicializační a ukončovací callback metody, které jsou v S-funkci vyžadovány. Po inicializaci bloku se spustí cyklická datová výměna se simulátorem FlightGear, která je implementována v callback metodách `mdlOutputs` a `mdlUpdate`. Následující obrázek přehledně ilustruje princip funkce bloku `FlightGear Interface`.



Obrázek 10: Funkční schéma bloku FlightGear Interface

Při inicializaci S-funkce jsou vytvořeny sockety pro oba směry komunikace, přičemž socket pro příjem dat z FlightGearu je v blokovacím režimu. FlightGear totiž zasílá data s nastavitelnou frekvencí a tímto zajistíme správné časování simulace modelu v Simulinku i bez bloku reálného času. Simulace zpracovává postupně jednotlivé bloky z modelu v jednom vlákně, a pokud je tedy v bloku `FlightGear Interface` blokovací operace příjmu dat, simulace se na této operaci zastaví do té doby, než přijde další datový paket z FlightGearu. Tím zabráníme tomu, aby čas v rámci simulace plynul rychleji než reálný čas. Simulink totiž za normálních okolností neprovádí simulace v reálném čase, ale čas v simulaci se vyvíjí v závislosti na použité vzorkovací frekvenci.

Poznámka:

TCP varianta bloku přijímá data od simulátoru FlightGear jako server a zpět do simulátoru je odesílá jako klient. Důvod této implementace je jednoduchý. FlightGear očekává a vyžaduje tento způsob komunikace.

5.3.6. Komunikační rozhraní

Jak již bylo zmíněno dříve datová výměna mezi blokem a simulátorem FlightGear probíhá prostřednictvím protokolu `NativeCtrls` a `NativeFDM`. Nyní se podíváme podrobněji na tyto protokoly.

Důležitá věc, kterou je potřeba při implementaci brát v potaz, je verze protokolu. Stará verze komunikačního protokolu nebude fungovat s novou verzí simulátoru. Síťová komunikace z hlediska bloku `FlightGear Interface` bude probíhat v pořádku, ale FlightGear nebude pakety staré verze zpracovávat. Při komunikaci směrem z FlightGearu, zase budeme dostávat irelevantní data nebo spojení nebude fungovat. Aktuální číslo verze protokolů je 27 pro `NativeCtrls` a 24 pro `NativeFDM`. Tyto verze jsou funkční od FlightGearu verze 1.0.0 až po aktuální verzi. Ta je nyní 1.9.1b. Aktuální čísla verze protokolů

(a souborů `hxx` definujících pakety) a další souvisejících věcí lze získat z CVS repositury simulátoru (10).

Další neméně podstatný problém, který je potřeba ošetřit, je nutnost provést drobné úpravy v souborech definujících pakety, pokud chceme generovat C kód pro cílovou platformu BOA5200 (1). Na té je pouze kompiler podporující ANSI C, zatímco pakety jsou definovány jako C++ třídy. Toto se dá vyřešit velmi snadno. Třídy definující paket změníme na struktury a můžeme je využívat v dále nezměněné podobě. Poznámky pro přizpůsobení paketů jsou v souborech `net_ctrls.hxx` a `net_fdm.hxx`, ve kterých jsou tyto pakety definovány. Součástí definice jsou ještě soubory `stdint.hxx` a `universal.h`, které jsou odkazovány ze souborů definující datové pakety.

Pokud by bylo potřeba upgradovat komunikační protokoly například z důvodu instalace nové verze FlightGearu, která by vyžadovala novější verze protokolů, stačí nahradit soubory `net_ctrls.hxx`, `net_fdm.hxx`, `stdint.hxx` a `universal.h` jejich aktuálními verzemi a provést výše zmíněnou úpravu definice paketů na struktury.

NativeCtrls

Paket tohoto protokolu je definován v souboru `net_ctrls.hxx`. Je definován třídou respektive strukturou `FGNetCtrls` a obsahuje řadu proměnných, ze kterých jsou v rámci této práce využity *aileron, elevator, rudder* a *throttle*. Struktura paketu je názorně vidět přímo v souboru `net_ctrls.hxx`.

NativeFDM

Paket tohoto protokolu je definován v souboru `net_fdm.hxx`. Je definován třídou respektive strukturou `FGNetFDM` a obsahuje řadu proměnných, které reprezentují komplexní soubor stavových informací dodávaných simulátoru z externího modelu. Z těchto proměnných jsou v rámci této práce využity *longitude, latitude, altitude, phi, theta* a *psi*. Struktura paketu je názorně vidět přímo v souboru `net_fdm.hxx`.

Čtenáře možná napadne otázka, proč se používají tak velké struktury pro přenos pouze těchto základních informací. Důvodem je požadavek na možnost rozšiřitelnosti tohoto rozhraní. Nyní, pokud bude potřeba přenášet další data, stačí pouze přidat do zdrojového kódu plnění/čtení příslušné proměnné, případně port bloku nebo rozšířit velikost vstupních/výstupních vektorů na jednotlivých portech.

Dále nebude potřeba vůbec měnit konfiguraci leteckého simulátoru ani mechanismy spouštějící simulaci. Kdyby byl použit například protokol *generic*, musela by se při rozšíření komunikace o nové datové položky změnit konfigurace I/O komunikace simulátoru, zdrojový kód S-funkce a konfigurace mechanismů spouštějících simulaci. Znamenalo by to tedy zásah do tří systémů, zatímco nyní je potřeba provést zásah pouze v jednom.

5.3.7. Generování kódu

Při generování kódu pro tento blok se volá soubor `fgio_tcp.tlc` respektive `fgio_udp.tlc`. Při jeho zpracování je do souboru `název_modelu.c`, který obsahuje kód vygenerovaný pro celý model, vygenerován kód pro jednotlivé instance bloku v rámci modelu.

5.4. Blok IFDP UDP Interface

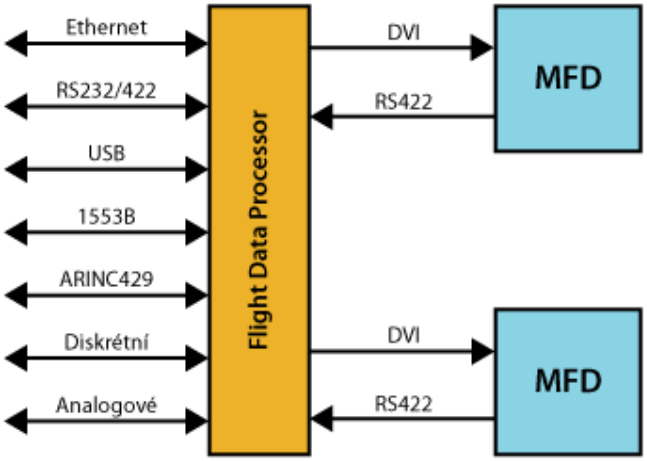
Blok `IFDP UDP Interface` implementuje komunikaci a datovou výměnu mezi HIL simulátorem a Integrovaným systémem zpracování letových dat (IFDP).

Integrovaný systém zpracování leteckých dat byl vyvinut firmou Speel Praha, s.r.o. jako moderní systém pro zpracování a zobrazení letových informací, který by byl využitelný v širokém spektru civilních i vojenských letounů. Pojem integrovaný systém je zde chápán ve smyslu zpracování dat z různých zdrojů tak, aby podoba výsledné informace umožňovala pilotovi její rychlé a správné vyhodnocení.

Systém IFDP umožňuje příjem a zpracování letových dat, jejich zobrazení na rastrových multifunkčních displejích s přímou interakcí obsluhy a přenos dat nutných pro zobrazování pohyblivých map či letových plánů vytvořených pilotem při plánování letu. Základní zobrazení obsahuje symboliku pro znázornění polohy letounu (ADI), horizontální situace (HSI) včetně pohyblivé mapy a informace o stavu vybraných palubních systémů – např. o množství paliva. Převzato z (11) str. 3.

Základní architektura systému

Základní architektura systém u zpracování letových dat je na obrázku 11. Základem systému IFDP je procesor letových dat (Flight Data Processor). Ten je schopen přijímat letecká data přes různá komunikační rozhraní. Data jsou dále tímto procesorem zpracována a výsledek tohoto procesu je zobrazen na dvou multifunkčních displejích (MFD). Každý MFD má na rámu 14 tlačítek, kterými lze ovládat IFDP a přepínat mezi různými módy zobrazení letových dat.



Obrázek 11: Základní architektura IFDP

V HIL simulátoru jsou data do IFDP distribuována přes síť Ethernet pomocí UDP protokolu. Podrobnější popis systému IFDP lze nalézt v dokumentu (11), jež je přílohou této práce.

Cyklická datová výměna implementovaná blokem IFDP UDP Interface probíhá pouze jedním směrem, a to z HIL simulátoru do IFDP. Soubory tvořící tento blok jako celek jsou vyjmenovány v tabulce 2.

S-funkce	mfd_udp_send.cpp
S-funkce MEX	mfd_udp_send.mexw32
TLC soubory	mfd_udp_send tlc
Help soubor	mfd_udp_send _help.html

Tabulka 2: Soubory, které jsou součástí bloku IFDP UDP Interface.

5.4.1. Vstupy bloku

Blok má dva vstupy, a to pozici letounu vzhledem k zemskému povrchu a úhly natočení podle jednotlivých os určující orientaci letadla ve vzduchu. Data z těchto vstupů jsou posléze zasílána do IFDP.

Pozice vzhledem k zemskému povrchu

Tento vstupní parametr je reprezentován vektorem obsahujícím tři veličiny určující pozici letounu.

Veličina	Jednotka
Zeměpisná šířka (latitude)	rad
Zeměpisná délka (longitude)	rad
Nadmořská výška (altitude)	m

Orientace ve vzduchu

Tento vstupní parametr je reprezentován vektorem obsahujícím tři veličiny určující orientaci letounu ve vzduchu.

Veličina	Jednotka
Podélný náklon (roll)	rad
Příčný náklon (pitch)	rad
Kurz letounu (yaw)	rad

5.4.2. Výstupy bloku

Blok nemá žádné výstupy.

5.4.3. Vstupní parametry S-funkce

Funkcionalita tohoto bloku je implementována v S-funkci `mfd_udp_send`. Tato funkce má 3 vstupní parametry. Všechny z nich jsou povinné a je potřeba je zadat pro správnou funkci bloku.

Sample time

Tento parametr je dvouprvkový vektor. Prvním prvkem je *perioda vzorkování* a druhým prvkem je *offset* periody vzorkování.

Out port

Port na IFDP, kam jsou tímto blokem zasílána data.

IFDP IP

IP adresa systému zpracování letových dat.

5.4.4. Princip funkce

Jak již vyplývá z názvu bloku, data jsou do IFDP zasílána prostřednictvím sítě Ethernet protokolem UDP. IFDP v současné verzi přijímá data na pevně stanovém portu. Číslo tohoto portu je 1602.

Procesor letových dat v IFDP je natolik výkonný, že neklade žádný horní limit na frekvenci zasílání paketů. Pakety ani nemusejí být zasílány s pevně danou frekvencí. IFDP jednoduše zpracuje přijaté pakety a zobrazí vyhodnocení jejich obsahu. Pokud nějakou dobu nejsou pakety zasílány, vizualizace se prostě zastaví, a po přijetí dalších paketů se opět rozběhne. Plynulá vizualizace letových dat je tedy závislá pouze na řízení datového toku nadřazeným počítačem. V současné implementaci IFDP nejsou přijatá data nijak potvrzována. Do budoucna se ale s touto variantou počítá, a v datovém paketu je pro ni vyhrazené místo.

5.4.5. Implementace bloku

Implementace S-funkce tohoto bloku se nachází v souboru `mfd_udp_send.cpp`. Cyklické zasílání dat do IFDP je realizováno v callback metodě *mdlOutputs*. Jak již bylo řečeno v předchozím bodě, IFDP nevyžaduje žádné speciální řízení či časování datového toku. Stávající implementace bloku tedy zasílá data do IFDP s frekvencí danou periodou vzorkování nastavenou v Simulinkovém modelu.

5.4.6. Komunikační rozhraní

V IFDP jsou data přijímány procesorem letových dat, který definuje strukturu datového rámce. Tento rámec je tvořen 111 byty.

Prvních 7 bytů je určeno pro potvrzování a identifikaci letových dat. Tato věc, ale ještě není v IFDP implementována, takže se do nich nevkládají žádné hodnoty, respektive je do nich v každém paketu vložena hodnota 0.

Zbylé byty rámce (8. až 111.) obsahují hodnoty letových proměnných v bytové reprezentaci. Datový typ všech letových proměnných je typ *float* podle normy IEEE-754. Každá hodnota letové proměnné je tedy v datovém rámci vyjádřena 8 byty. Rámec obsahuje 13 letových proměnných. Struktura paketu spolu s jednotkami jednotlivých letových proměnných je v tabulce 3. Pořadí bytů je little endian.

Pořadí bytu	Význam	Jednotka
1 - 7	Potvrzování a identifikace letových dat	[-]
8 - 15	Zeměpisná délka	[°]
16 - 23	Zeměpisná šířka	[°]
24 – 31	Nadmořská výška	[m]
32 – 39	Podélný náklon	[°]
40 – 47	Příčný náklon	[°]
48 – 55	Kurs	[°]
56 – 63	Přístrojová rychlost	[uzly]
64 – 71	Vysokotlaké otáčky	[%]
72 – 79	Vertikální zrychlení	[g]
80 – 87	Teplota výstupních plynů	[°C]
88 – 95	Spotřeba	[lb/h]
96 – 103	Množství paliva	[lb]
104 – 111	Nízkotlaké otáčky	[%]

Tabulka 3: Struktura paketu letových dat pro IFDP

5.4.7. Generování kódu

Při generování kódu pro tento blok se volá soubor `mfd_udp_send.tlc`. Při jeho zpracování je do souboru `název_modelu.c`, který obsahuje kód vygenerovaný pro celý model, vygenerován kód pro jednotlivé instance bloku v rámci modelu.

5.5. Blok FG Data Sender

Blok `FG Data sender` existuje ve dvou variantách lišících se tím, jaký používají protokol. První variantou bloku je implementace využívající protokol TCP. Druhá varianta využívá při komunikaci protokol UDP. Ve všech ostatních ohledech je funkčnost obou variant stejná až na drobné detaily vyplívající z rozdílných vlastností TCP a UDP protokolu.

Blok `FG Data Sender` slouží k zasílání letových dat k dílčím částem HIL simulátoru. Tyto letová data zahrnují pozici letounu vzhledem k zemskému povrchu, orientaci letounu v prostoru, nastavení řídích ploch a velikost aplikovaného tahu. Zařízení, které je příjemcem dat distribuovaných tímto blokem je například řídicí systém reprezentovaný v obrázku 8 blokem Flight Control Law. Soubory, které tvoří implementaci těchto bloků, jsou uvedeny v tabulce 4.

S-funkce	fg_tcp_send.cpp
	fg_udp_ send.cpp
S-funkce MEX	fg_tcp_ send.cpp.mexw32
	fg_udp_send.cpp.mexw32
TLC soubory	fg_tcp_ send.cpp.tlc
	fg_udp_ send.cpp.tlc
Help soubor	fg_tcp_ send.cpp.html
	fg_udp_ send.cpp.html

Tabulka 4: Soubory, které jsou součástí bloků FG Data Sender.

5.5.1. Vstupy bloku

Blok má 4 vstupy odpovídající zasílaným letovým datům. Obsahuje tedy následující vstupy:

Nastavení řídících ploch

Tento vstupní parametr je reprezentován vektorem obsahujícím hodnoty nastavení řídících ploch letounu.

Veličina	Jednotka	Rozsah
Nastavení křidélek (aileron)	-	-1 : 1
Nastavení výškového kormidla (elevator)	-	-1 : 1
Nastavení směrového kormidla (rudder)	-	-1 : 1

Orientace ve vzduchu

Tento vstupní parametr je reprezentován vektorem obsahujícím tři veličiny určující orientaci letounu ve vzduchu.

Veličina	Jednotka
Podélný náklon (roll)	rad
Příčný náklon (pitch)	rad
Kurz letounu (yaw)	rad

Pozice vzhledem k zemskému povrchu

Tento vstupní parametr je reprezentován vektorem obsahujícím tři veličiny určující pozici letounu.

Veličina	Jednotka
Zeměpisná šířka (latitude)	rad
Zeměpisná délka (longitude)	rad
Nadmořská výška (altitude)	m

Tah

Bezrozměrná veličina v rozsahu 0 : 1.

5.5.2. Výstupy bloku

Blok nemá žádné výstupy.

5.5.3. Vstupní parametry S-funkce

Sample time

Tento parametr je dvouprvkový vektor. Prvním prvkem je *perioda vzorkování* a druhým prvkem je *offset* periody vzorkování.

Host IP

IP adresa stroje, kde je spuštěná simulace modelu, který obsahuje blok `FG Data Receiver`.

Out port

Komunikační port, přes který probíhá výměna dat.

5.5.4. Princip funkce

Blok zasílá všechna vstupní data bloku `FG Data Receiver`, který je další komponentou této knihovny, prostřednictvím protokolu TCP respektive UDP. Data nejsou žádným způsobem potvrzována ani nijak kontrolována.

5.5.5. Implementace bloku

Implementace tohoto bloku je realizována v souboru `fg_tcp_send.cpp`. Cyklické zasílání dat do přes zadaný port je realizováno v callback metodě *mdlOutputs*. Frekvence zasílání dat se odvíjí od periody vzorkování nastavené v Simulinkovém modelu.

Poznámka:

TCP varianta tohoto bloku se chová jako klient ve vztahu k TCP verzi bloku `FG Data Receiver`.

5.5.6. Komunikační rozhraní

Data jsou zasílána paketem tvořeným strukturou v jazyce C/C++. Struktura popisující paket je definovaná v souboru *fgaddstruct.h*. Paket obsahuje všechny vstupní letová data a poté ještě další proměnné pro budoucí využití.

5.5.7. Generování kódu

Při generování kódu pro tento blok se volá soubor `fg_tcp_send.tlc`. Při jeho zpracování je do souboru `název_modelu.c`, který obsahuje kód vygenerovaný pro celý model, vygenerován kód pro jednotlivé instance bloku v rámci modelu.

5.6. Blok FG Data Receiver

Blok `FG Data Receiver` taktéž existuje ve dvou variantách, variantě využívající UDP protokol a variantě využívající TCP protokol. Tento blok je protějšek bloku `FG Data Sender`. Funkčnost obou variant bloku je v zásadě stejná. Jejich funkčnost se liší pouze drobnými detaily vyplívajícími z rozdílných vlastností TCP a UDP protokolu.

Blok `FG Data Receiver` slouží k příjmu dat zaslaných blokem `FG Data Sender`. Tento blok je součástí modelů, které potřebují přijímat data distribuovaná leteckým simulátorem FlightGear a modelem simulovaného letounu. Mezi tyto data patří pozice letounu vzhledem k zemskému povrchu, orientace letounu v prostoru, nastavení řídích ploch a velikost aplikovaného tahu. Soubory, které tvoří implementaci obou variant bloku, jsou uvedeny v tabulce 5.

S-funkce	fg_tcp_receive.cpp fg_udp_receive.cpp
S-funkce MEX	fg_tcp_receive.cpp.mexw32 fg_udp_receive.cpp.mexw32
TLC soubory	fg_tcp_receive.cpp.tlc fg_udp_receive.cpp.tlc
Help soubor	fg_tcp_receive.cpp.html fg_udp_receive.cpp.html

Tabulka 5: Soubory, které jsou součástí bloků FG Data Receiver.

5.6.1. Vstupy bloku

Blok nemá žádné vstupy.

5.6.2. Výstupy bloku

Blok má 4 výstupy. Tyto výstupy korespondují se vstupy bloku `FG Data Sender` a obsahují následující veličiny:

Nastavení řídících ploch

Tento výstupní parametr je reprezentován vektorem obsahujícím hodnoty nastavení řídících ploch letounu.

Veličina	Jednotka	Rozsah
Nastavení křidélek (aileron)	-	-1 : 1
Nastavení výškového kormidla (elevator)	-	-1 : 1
Nastavení směrového kormidla (rudder)	-	-1 : 1

Orientace ve vzduchu

Tento vstupní parametr je reprezentován vektorem obsahujícím tři veličiny určující orientaci letounu ve vzduchu.

Veličina	Jednotka
Podélný náklon (roll)	rad
Příčný náklon (pitch)	rad
Kurz letounu (yaw)	rad

Pozice vzhledem k zemskému povrchu

Tento vstupní parametr je reprezentován vektorem obsahujícím tři veličiny určující pozici letounu.

Veličina	Jednotka
Zeměpisná šířka (latitude)	rad
Zeměpisná délka (longitude)	rad
Nadmořská výška (altitude)	m (metry)

Tah

Bezrozměrná veličina v rozsahu 0 : 1.

5.6.3. Vstupní parametry S-funkce

S-funkce `fg_tcp_receive` implementující tento blok má 2 vstupní parametry.

Sample time

Tento parametr je dvouprvkový vektor. Prvním prvkem je *perioda vzorkování* a druhým prvkem je *offset* periody vzorkování.

In port

Port, na kterém blok přijímá data zaslaná blokem `FG Data Sender`.

5.6.4. Princip funkce

Blok přijímá letová data od bloku `FG Data Sender` prostřednictvím protokolu TCP. Data nejsou žádným způsobem potvrzována ani nijak kontrolována.

5.6.5. Implementace bloku

Implementace tohoto bloku je realizována v souboru `fg_tcp_receive.cpp`. Příjem dat je realizován v callback metodě *mdlOutputs*.

Poznámka:

TCP varianta tohoto bloku se chová jako server ve vztahu k TCP verzi bloku `FG Data Sender`.

5.6.6. Generování kódu

Při generování kódu pro tento blok se volá soubor `fg_tcp_receive.tlc`. Při jeho zpracování je do souboru `název_modelu.c`, který obsahuje kód vygenerovaný pro celý model, vygenerován kód pro jednotlivé instance bloku v rámci modelu.

5.7. Rozdíly v aplikaci bloků vyplývající z použitých protokolů

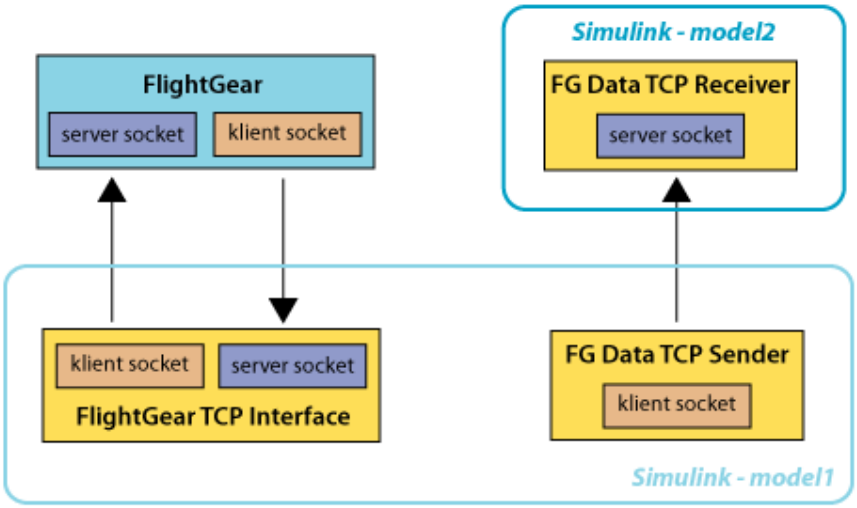
Bloky `FlightGear Interface`, `FG Data Sender` a `FG Data Receiver` existují ve dvou variantách. První varianta bloku využívá protokol TCP a druhá protokol UDP. Jak již bylo řečeno v předchozích bodech této kapitoly, funkcionalita těchto bloků zůstává stejná pro obě varianty až na drobné rozdíly vyplývající z vlastností TCP a UDP spojení. Jak ale nyní ukážeme, tyto drobné rozdíly mohou ovlivnit chování případně možnosti Simulinkového modelu, ve kterém jsou bloky začleněny. Omezení vyplívající z použití určité varianty bloků nemusí být v některých řešeních významné, ale v jiných můžou uživateli trochu komplikovat práci. Podívejme se nyní, co přesně obnáší použití různých variant bloků.

5.7.1. Bloky využívající TCP protokol

Mezi tyto bloky patří `FlightGear TCP Interface`, `FG Data TCP Sender`, `FG Data TCP Receiver`. Všechny bloky této kategorie používají pro komunikaci TCP spojení, které je realizováno na bázi klient – server. Výhody tohoto spojení jsou obecně známé. TCP je trvalé spojení mezi aplikacemi, které zaručuje spolehlivý přenos dat (data jsou přijímána v pořadí v jakém byla vyslána, nedochází ke ztrátě dat). Před započetím vlastního přenosu dat musí být nejdřív navázáno spojení mezi oběma účastníky přenosu.

Pokud tedy použijeme v Simulinkovém modelu bloky využívající TCP spojení musíme si uvědomit následující věc. Některé bloky se v rámci TCP spojení chovají jako server jiné zase jako klient (`FG TCP Interface` obsahuje oba typy socketů). Při spouštění modelu (simulace) musíme pro správnou funkčnost celku dodržet konvence navázaní TCP spojení. To znamená, že nejdřív musí být spuštěny komponenty (aplikace, Simulinkové modely), které obsahují sockety typu server čekající na příchozí spojení. Teprve poté mohou být spuštěny aplikace či Simulinkové modely, které se k těmto socketům připojují. Pokud provedeme spuštění jednotlivých aplikací v opačném pořadí, spojení nebude správně navázáno.

Nyní přejděme od obecného úvodu ke konkrétnímu problému, jakým je TCP spojení mezi leteckým simulátorem FlightGear a bloky tvořícími rozhraní mezi ním a Simulinkový modelem nebo dílčími částmi HIL simulátoru. Kompletní schéma tohoto problému z hlediska TCP komunikace ilustruje obrázek 12.



Obrázek 12: Schéma TCP komunikace mezi simulátorem FlightGear a jednotlivými bloky knihovny FlightGear blockset.

Pokud chceme přijímat data ze simulátoru FlightGear a zároveň předávat či přijímat data pomocí bloků `FG Data TPC Sender/Receiver`, situace při spouštění aplikací se již lehce komplikuje. Máme-li situaci jako na obrázku 12, je potřeba spustit jednotlivé modely a simulátor FlightGear v následujícím pořadí:

- Nejprve spustit ***model2***. Tím se vytvoří server socket čekající na příchozí spojení z modelu *model1*, které navazuje blok `FG Data TCP Sender`.
- Poté spustíme ***model1***. Naváže se spojení mezi modelem *model1* a *model2* respektive příslušnými bloky. Dále se vytvoří server socket, na který se může připojit FlightGear.
- Nakonec spustíme simulátor **FlightGear**. Ten se připojí na vytvořený server socket v modelu *model1*, a posléze vytvoří server socket, na který se připojí blok `FlightGear TCP Interface`.

Pro úspěšné navázání spojení mezi všemi účastníky TCP komunikace nelze pořadí výše zmíněných kroků měnit. Nelze měnit ani pořadí spuštění modelu *model1* a simulátoru FlightGear, protože FlightGear se v této konfiguraci při spuštění pokusí nejprve připojit na sever socket vytvořený blokem `FlightGear TCP Interface`. Pokud tento socket ještě není vytvořen, FlightGear se nespustí.

Pokud komunikace mezi modelem *model1* a modelem *model2* probíhá opačným směrem než na obrázku 12, tak je pořadí spouštění jednotlivých komponent následující:

- model1
- model2
- FlightGear

Další podstatnou věcí, kterou je potřeba si uvědomit při použití TCP spojení je ta, že **nelze přerušit simulaci** v hlavním modelu (*model1* na obrázku 12) **a poté v ní pokračovat**. Spojení mezi jednotlivými účastníky se totiž přeruší a je potřeba ho znovu navázat. To znamená restart všech modelů či aplikací dle výše uvedeného postupu.

Tato varianta se tedy více hodí pro dlouhotrvající simulace vyžadující kvalitní spojení mezi vzdálenými účastníky simulačního procesu zajišťující bezeztrátový přenos dat.

5.7.2. Bloky využívající UDP protokol

Mezi tyto bloky patří `FG UDP Interface`, `FG Data UDP Sender`, `FG Data UDP Receiver` a `IFDP UDP Interface`. Bloky této kategorie využívají UDP spojení, kde se nerozlišuje mezi klientem a serverem. UDP spojení není trvalé spojení a není ho nutné navazovat. Obě komunikující strany jsou si tedy rovnocenné. Přenos zpráv není nijak potvrzován, ani není nijak kontrolováno doručení odeslaných paketů.

Toho, že není potřeba navazovat spojení pro zasílání UDP zpráv, můžeme velmi dobře využít. Tato vlastnost UDP protokolu, totiž umožňuje odstranit výše zmíněné nevýhody TCP spojení plynoucí z nutnosti navázat spojení mezi jednotlivými účastníky před započetím vlastního přenosu.

Vybereme-li do Simulinkových modelů UDP varianty jednotlivých bloků, plynou z této volby následující výhody. **Aplikace či modely můžeme spustit v libovolném pořadí** bez toho, abychom narušili správnou funkci spojení mezi jednotlivými účastníky komunikace. Je ale potřeba upozornit, že díky využití blokujících socketů pro příjem dat v jednotlivých

blocích, je simulace v modelech obsahujících patřičné bloky pozastavena, pokud bloky nepřijímají žádná data.

Další výhodou plynoucí z použití UDP protokolu je **možnost pokračovat v pozastavené simulaci**. Protože spojení není trvalé, tak není tímto přerušeno a z hlediska účastníků, kteří nebyli „pozastaveni“ jednoduše nepřicházejí nová data. Tito účastníci tedy do doby, než jejich protějšci obnoví svou činnost, uchovávají poslední přijatá data a s nimi pracují.

Tato varianta je vhodnější pro krátkodobější simulace, kde tolik netrváme na kvalitě spojení nebo pro simulace, které je potřeba přerušovat. Pokud zvolíme dostatečnou frekvenci výměny dat, nepředstavuje žádný problém ani ztráta některých zpráv. Přenášené zprávy přenášejí pouze stavové informace. Pokud dojde ke ztrátě některé zprávy, znamená to pouze to, že u některé části systému dojde k malému zpoždění v aktualizaci stavové informace. Pokud bezchybná a včasná aktualizace není klíčovým kritériem v dané simulaci, nezpůsobí toto zmíněné zpoždění žádný problém.

Pokud pro nás výpadek i jediné stavové informace představuje problém, není nic jednoduššího než použít TCP variantu bloků z knihovny.

6. Závěr

Cílem této práce byla integrace leteckého simulátoru do hardwarového simulátoru typu *Hardware In the Loop (HIL)* vyvinutého Katedrou řídící techniky a využívaném ve firmě Aero Vodochody a.s. Integrace představovala návrh, vývoj, zprovoznění a testování uživatelského rozhraní sloužícího k řízení a vizualizaci průběhu simulace vykonávané HIL simulátorem.

Návrh a vývoj tohoto rozhraní obnášel výběr vhodného leteckého simulátoru představujícího vizualizační a řídící rozhraní, vhodného příslušenství a vytvoření knihovny bloků pro prostředí Simulink umožňující propojení leteckého simulátoru se Simulinkovým modelem a specifickými částmi HIL simulátoru.

Navazujícím úkolem bylo zprovoznění a otestování celého řešení přímo ve firmě Aero Vodochody a.s.

Pro vizualizaci byl vybrán simulátor FlightGear. Tento simulátor představoval nejjednodušší variantu pro implementaci cyklické výměny letových dat. Další výhodou bylo to, že s tímto simulátorem již pracovali lidé z Katedry řídicí techniky, a tudíž existovala možnost s někým osobně konzultovat případné problémy.

Dále byla vyvinuta knihovna FlightGear blockset pro prostředí Simulink. Tato knihovna obsahuje bloky realizující cyklickou výměnu letových a řídích dat mezi Simulinkovým modelem a simulátorem FlightGear. Dále knihovna obsahuje blok umožňující vizualizaci letových dat prostřednictvím Integrovaného systému zpracování letových dat, který byl vyvinut firmou Speel s.r.o. V neposlední řadě knihovna obsahuje bloky zajišťující výměnu letových dat mezi simulačním modelem v Simulinku a dílčími částmi HIL simulátoru.

Všechny implementované bloky podporují generování real-time kódu v jazyce C pro cílovou platformu BOA5200 (1) s operačním systémem Linux. Při generaci kódu je využit Linux ERT Target vyvinutý v rámci (8).

Dílčí části této práce byly průběžně testovány a zprovozněny v rámci HIL simulátoru přímo ve firmě Aero Vodochody a.s. Výsledkem je HMI rozhraní, které v kombinaci s HIL simulátorem představuje komplexní systém pro vývoj, měření a testování řídicího systému založeného na konceptu Flight By Wire zmíněného v úvodu.

Rozhraní, a tím pádem i možnosti HIL simulátoru jako celku, je možné v budoucnu jednoduše rozšířit o vizualizaci dalších letových dat či parametrizaci dílčích částí simulátoru pomocí nových externích vstupů. Vytvoření nových bloků podle postupu uvedeného v kapitole 4 umožní k HIL simulátoru připojit libovolné další periferie. Ty mohou sloužit jak pro vizualizaci dílčích částí simulačního procesu, tak pro jeho řízení.

Knihovna zároveň umožňuje generování kódu pro libovolnou cílovou platformu s operačním systémem Linux, tudíž je možné toto řešení snadno upravit i pro jiné koncepce HIL simulace.

7. Bibliografie

1. **Analogue and Micro.** Boa5200 specification. [Online] http://www.analogue-.
2. **Waszniowski, L., Z. Hanzálek, P. Hospodář, M. Hromčík and J. Doubrava.** Hardware in the Loop Simulation of FBW components. [Online] 2009. http://dce.felk.cvut.cz/waszniowski/HIL_FBW/HIL_FBW.htm.
3. Aeromodelling. *Model aircraft and Aerodynamics.* [Online] http://www.myaeromodelling.com/wp/.
4. **Laminar Research.** The House of X-Plane. *X-Plane SDK.* [Online] http://www.xsquawkbox.net/xpsdk/mediawiki/Main_Page.
5. **Microsoft.** Microsoft Developer Network. *SimConnect SDK Reference.* [Online] 2008. http://msdn.microsoft.com/en-us/library/cc526983.aspx#SimConnect_API_Reference.
6. Microsoft Developer Network. *Microsft ESP.* [Online] 2009. http://msdn.microsoft.com/en-us/library/aa306570.aspx.
7. **Jelínek, Pavel.** Podpora simulace s hardware ve smyčce. *Diplomová práce.* Praha : ČVUT FEL, 2008.
8. **Hamáček, Lukáš.** RTW target for Linux with CANopen support. *Diplomová práce.* Praha : ČVUT FEL, 2009.
9. **The MathWorks.** Documentation for MathWorks Products, R2009b. [Online] 2009. http://www.mathworks.com/access/helpdesk/help/helpdesk.html.
10. **FlightGear.** CVS repository. */source/src/Network.* [Online] http://cvs.flightgear.org/viewvc/source/src/Network.
11. **Zechovský, Zdeněk.** IFDP - Integrovaný systém zpracování letových dat. *Závěrečná zpráva.* 2008.
12. **Olson, Curtis L.** FlightGear Official Website. [Online] http://flightgear.org/.
13. **Michael Basler, Martin Spott a další.** The FlightGear Manual. [Online] 15. December 2007. http://www.flightgear.org/.
14. **FlightGear.** FlightGear Forums. *Joystick only operates left aileron.* [Online] http://www.flightgear.org/forums/viewtopic.php?f=11&t=3325&st=0&sk=t&sd=a&hilit=Joystick+only+operates+left+aileron&start=15.
15. Property Tree. *FlightGear Wiki.* [Online] 12 2009. http://wiki.flightgear.org/index.php/Property_Tree.

Příloha A
Instalace simulátoru FlightGear

Letecký simulátor v FlightGear je volně ke stažení na oficiálních stránkách tohoto simulátoru (12). Při začátku implementace této práce byla aktuální verze simulátoru 1.0.0 a knihovna FlightGear blockset byla vyvíjena pro tuto verzi. Aktuální verze simulátoru je nyní 1.9.1b. FlightGear blockset by měl bez problémů fungovat i s novou verzí simulátoru. Jediné, co je potřeba zkontrolovat při použití nové verze simulátoru je to, zda uživatel využívá aktuální verze protokolů NativeCtrls a NativeFDM. Způsob kontroly je popsán v 5.3.6.

Po instalaci do zvoleného adresáře je potřeba ošetřit několik věcí pro správnou funkci leteckého simulátoru v rámci HIL simulátoru. Jedná se hlavně o správné nastavení joysticků a nastavení leteckých přístrojů uvnitř simulátoru FlightGear. Nastavení přístrojů není nutné, ale pokud se neprovede, některé relevantní přístroje nebudou fungovat správně.

Nastavení joysticků

V HIL simulátoru je zapojen joystick s plynovou pákou Thrustmaster Hotas Cougar a pedály PFC Rudder Pedals. Nejdříve je potřeba zkontrolovat konfiguraci joysticku. Způsob jakým se konfigurují joysticky je popsán v manuálu k simulátoru FlightGear (13) v kapitole 3.6 *Joystick support*. Tento manuál je součástí instalace a je to soubor s tímto umístěním `/FG_inst_dir/docs/getstart.pdf`. Podrobný popis instalace joysticků lze nalézt taktéž v sekci `docs` v souboru `/FG_inst_dir/docs/README.Joystick.html`.

Thrustmaster Hotas Cougar

Instalace ovladačů

Joystick je distribuován s instalačním CD a manuálem. Toto CD obsahuje ovladače pro vlastní joystick s plynovou pákou, poté speciální programy pro konfiguraci a monitoring stavu joysticku. Popis práce s jednotlivými programy je popsán v uživatelském manuálu. Pro naše účely postačí základní instalace joysticku. Tu provedeme spuštěním a dokončením standardní instalace (setup.exe).

Konfigurace ve FlightGearu

Thrustmaster Hotas Cougar je zařízení standardně podporované simulátorem FlightGear verze 1.0.0 a vyšší. V instalačním adresáři pro něj najdeme konfigurační soubor, takže by po zapojení a instalaci ovladačů měl bez problémů fungovat.

Osobní zkušeností jsem ale zjistil, že za určitých okolností Hotas Cougar nefunguje podle očekávání, respektive za některých okolností se neprojeví chyba konfigurace. Konfigurace joysticků pro FlightGear je implementována formou XML souborů. Konfigurace pro joystick Thrustmaster Hotas Cougar je vytvořena v souboru `/FG_inst_dir/data/Input/Joysticks/ThrustMaster/HOTAS-Cougar.xml`.

Tento soubor obsahuje chybu, která způsobí, to že po spuštění simulátoru a vzletu letadla, má letadlo tendenci stále zatáčet doleva. Z nějakého důvodu, který se mi nepodařilo přesně určit, se tento problém projevil pod operačním systémem Windows XP, ale již se neprojevil pod Windows Vista.

Každopádně chyba je následujícího charakteru. Jedno z tlačítek ovládá *aileron trim*, což je veličina udávající rozdíl mezi vychýlením pravé a levé výškovky, a používá se pro kompenzaci dlouhodobých vlivů, které zapříčiňují menší zatáčení letadla. A právě konfigurace tohoto tlačítka obsahuje chybu. Konfigurace tlačítka ve zmíněném XML souboru se nachází na řádcích 119 až 151. Tlačítko funguje, tak že při stisknutí resetuje aileron trim, a jinak ho nastaví nějakou hodnotu. Pro nás důležitá část původního kódu vypadá takto:

```
<desc>aileron trim/reset aileron trim</desc>
        <low>
                <repeatable>true</repeatable>
                <binding>
                        <command>nasal</command>
                        <script>
                                mod = getprop("/input/joysticks/js[0]/button[2]");
                                if (mod == nil or mod == 0) {
                                        Tlačítko není stisknuté, nastavíme hodnotu trimu -
1
                                        controls.aileronTrim(-1)
                                } elsif (mod == 1) {
                                        Tlačítko stisknuté, resetuj trim
                                        setprop("/controls/flight/aileron-trim", 0);
                                }
                        </script>
                </binding>
        </low>
        Řeší aileron trim pro druhý směr zatáčení
        <high>
                <repeatable>true</repeatable>
                <binding>
                        <command>nasal</command>
                        <script>
                                mod = getprop("/input/joysticks/js[0]/button[2]");
                                if (mod == nil or mod == 0) {
                                Tlačítko není stisknuté, nastavíme hodnotu trimu 1
                                        controls.aileronTrim(1)
                                } elsif (mod == 1) {
                                        Tlačítko stisknuté, resetuj trim
                                        setprop("/controls/flight/aileron-trim", 0);
                                }
                        </script>
                </binding>
        </high>
```

Standardní aileron trim je nastaven na ±1. To znamená maximální výchylku křidélek na jednu či druhou stranu, takže pokud nedržíme dané tlačítko, v podstatě hned se nám nastaví maximální výchylka křidélek, kterou joystick nedokáže kompenzovat a letadlo se nedá pořádně ovládat. Důvod je jednoduchý. Výchylka křidélek je ±1, a je uložená v příslušné stavové proměnné. Joystick může měnit výchylku křidélek v rozsahu také ±1, a přistupuje k přímo ke stejné stavové proměnné. K této proměnné se ještě přičítá hodnota aileron trim. Výsledkem je součet hodnot 1 nebo -1 dodaných od aileron trim a hodnoty dodané z joysticku. Rozsah, ve kterém se proměnná může pohybovat, je ±1. Takže po aplikaci operace součtu se výsledek může pohybovat v rozsahu -1 až 0 nebo 0 až 1. To znamená, že na jednu stranu nelze vůbec zatáčet. Řádky definující toto chování jsou ve výše uvedeném kódu vyznačeny červeně. Těžko určit, co je správná konfigurace tohoto tlačítka. Pokud chceme mít standardně nastavený nějaký aileron trim. Stačí změnit parametry funkcí `controls.aileronTrim` na nějakou rozumnou hodnotu (menší než ±0.5). Pokud standardně nechceme žádný aileron trim a zmáčknutím nějaký nastavit, může konfigurace vypadat stejně jako níže uvedený kód. Takto je také v současné době nastaven Hotas Cougar v rámci HIL simulátoru.

```
<desc>aileron trim/reset aileron trim</desc>
        <low>
                <repeatable>true</repeatable>
                <binding>
                        <command>nasal</command>
                        <script>
                                mod = getprop("/input/joysticks/js[0]/button[2]");
                                if (mod == nil or mod == 0) {
                                        controls.aileronTrim(0)
                                } elsif (mod == 1) {
                                        setprop("/controls/flight/aileron-trim", -0.3);
                                }
                        </script>
                </binding>
        </low>
        <high>
                <repeatable>true</repeatable>
                <binding>
                        <command>nasal</command>
                        <script>
                                mod = getprop("/input/joysticks/js[0]/button[2]");
                                if (mod == nil or mod == 0) {
                                        controls.aileronTrim(0)
                                } elsif (mod == 1) {
                                        setprop("/controls/flight/aileron-trim", 0.3);
                                }
                        </script>
                </binding>
        </high>
```

Tento problém byl řešen na fóru věnovaném FlightGearu ve vlákně (14). Zbylá XML konfigurace pro Hotas Cougar by měla být v pořádku.

PFC Rudder pedals

Pedály od firmy PFC se vzhledem k simulátoru FlightGear chovají jako běžný joystick. Standardně FlightGear neobsahuje konfigurační XML pro tento typ pedálů, a proto je potřeba je „identifikovat“ (rozpoznat, který pohyb pedálů odpovídá jakým osám apod.) Tento postup zde nebudu podrobně rozebírat, protože je dobře popsán v (13) a v souboru `FG_inst_dir/docs/README.Joystick.html`.

Uvedu zde pouze výsledek této identifikace, jíž je konfigurační soubor `pfc-pedals-usb.xml`. Ten je potřeba nakopírovat do adresáře (vytvořit pokud neexistuje) `FG_inst_dir/data/Input/Joysticks/PFC/`. Dále je nutné uvědomit FlightGear, že přibyla nová konfigurace, kterou má zkoušet, pokud bylo připojeno nové zařízení. To se provede přidáním následujícího řádku do souboru `/FG_inst_dir/data/joysticks.xml` do tagu `PropertyList`.

```
<js-named include="Input/Joysticks/PFC/pfc-pedals-usb.xml" />
```

Tímto by mělo být všechno týkající se FlightGearu nainstalováno.

Konfigurace simulátoru FlightGear

Konfigurace provozních parametrů

Všechna potřebná konfigurace simulátoru FlightGear, pro využití v simulacích v rámci HIL simulátoru, lze nastavit při spuštění formou parametrů z příkazové řádky. Pro tyto účely je vytvořen spouštěcí dávkový soubor *fgio.bat*, který je přílohou této práce.

Při editaci tohoto souboru v něm najdeme sedm následujících parametrů, uvozených klíčovým slovem `set`.

FlightGearInstallDir
Adresář, kde je nainstalovaný simulátor FlightGear.

Native-fdm_IP
IP stroje, který zasílá do simulátoru FlightGear data. Pokud tento parametr zůstane prázdný FlightGear přijímá data z libovolné IP adresy. Pokud zadáme specifickou adresu, FlightGear přijímá data pouze z této adresy.

Native-fdm_port
Port, na kterém FlightGear přijímá data.

Native-ctrls_IP
IP stroje, kam FlightGear zasílá data, a který je příjemcem řídících dat.

Native-ctrls_port
Port pro zasílání dat.

Protocol
Protokol přenosu. Platné varianty jsou `udp` a `tcp`.

Frequency
Frekvence přenosu (zasílání/příjmu) dat.

Aby datová výměna mezi simulátorem FlightGear a blokem `FlightGear Interface` fungovala správně, je potřeba tyto parametry nastavit tak, aby se shodovaly s parametry nastavenými v bloku `FlightGear Interface`. Pro správnou funkci by měla frekvence zasílání dat odpovídat periodě vzorkování Simulinkového modelu.

Všechny použité parametry příkazové řádky jsou podrobně vysvětleny v (13). Zmíníme tedy jen ty základní. FlightGear je příkazem `--fdm=external` přepnut do externího režimu. To znamená, že na rozdíl od interního režimu, nezpracovává a nezobrazuje data dodaná interním leteckým modelem jako v běžném případě, ale zpracovává a zobrazuje data dodaná externím modelem/aplikací. Tyto data jsou předávána pomocí TCP nebo UDP protokolu (příkazy `--native-fdm` a `--native-ctrls`).

Konfigurace přístrojů

Poslední věcí, kterou je potřeba nastavit pro správnou funkci přístrojů během simulace, je přenastavení zdrojových dat pro některé přístroje, jako je například umělý horizont. Tuto změnu provedeme tak, že v souboru pro daný přístroj (nyní umělý horizont) `FG_inst_dir\data\Aircraft\Instruments\attitude-indicator.xml` změníme zdroje dat pro přístroj z kategorie `instrumentation` na kategorii `orientation`. Důvod je ten, že data (pozice a orientace letounu), která zasíláme do simulátoru FlightGear neplní proměnné v kategorii `instrumentation` odkud je čte umělý horizont apod., ale plní proměnné v kategorii `orientation`. Proto musíme konfigurací ve zmíněném souboru přenastavit zdroj dat na kategorii `orientation`. Všechny XML tagy typu

```
<property>/instrumentation/attitude-indicator/indicated-roll-deg</property>
```

změníme na

```
<property>/orientation/roll-deg</property>
```

Musíme změnit všechny tyto tagy, které obsahují roll, pitch, yaw. Jména potřebných proměnných lze nalézt v Property Tree v příslušných kategoriích (`orientation`, `instrumentation`). K Property Tree se dá přistoupit způsobem popsaným v (15) nebo po spuštění FlightGearu v menu `File/Browse Internal Properties` . Tím dosáhneme požadované změny a přístroj bude při simulaci fungovat.

Stejné úpravy jsou možné pro všechny relevantní přístroje zobrazující nějakým způsobem data o pozici či orientaci letadla. Konfigurační soubory jednotlivých přístrojů se nacházejí v adresáři `FG_inst_dir\data\Aircraft\Instruments`. Upravené konfigurace přístrojů jsou přílohou této práce.

Příloha B
Příklad použití knihovny FlightGear blockset

Tento návod popisuje podrobně zprovoznění cyklické výměny dat mezi Simulinkovým modelem, leteckým simulátorem FlightGear, IFDP a externí aplikací či Simulinokovým modelem.

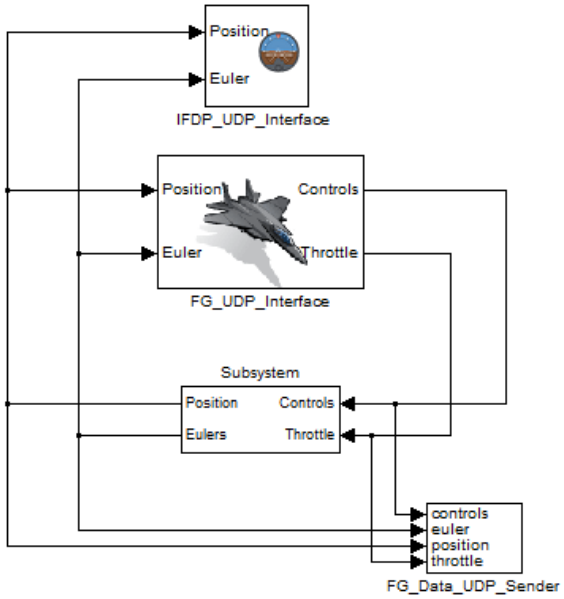
Sestavení Simulinkových modelů

V tomto příkladu využijeme všechny typy bloků z knihovny FlightGear blockset. Pro předvedení možností blocksetu využijeme dva Simulinkové modely. Celý tento příklad včetně modelů je součástí CD s přílohami. Příklad obsahuje UDP verze jednotlivých bloků. Pokud bychom chtěli využít TCP variantu bloků, nastavení se provádí stejným způsobem. První část příkladu je věnována simulaci v Matlabu, zatímco druhá část je věnována spuštění simulace na počítači BOA52000 (1).

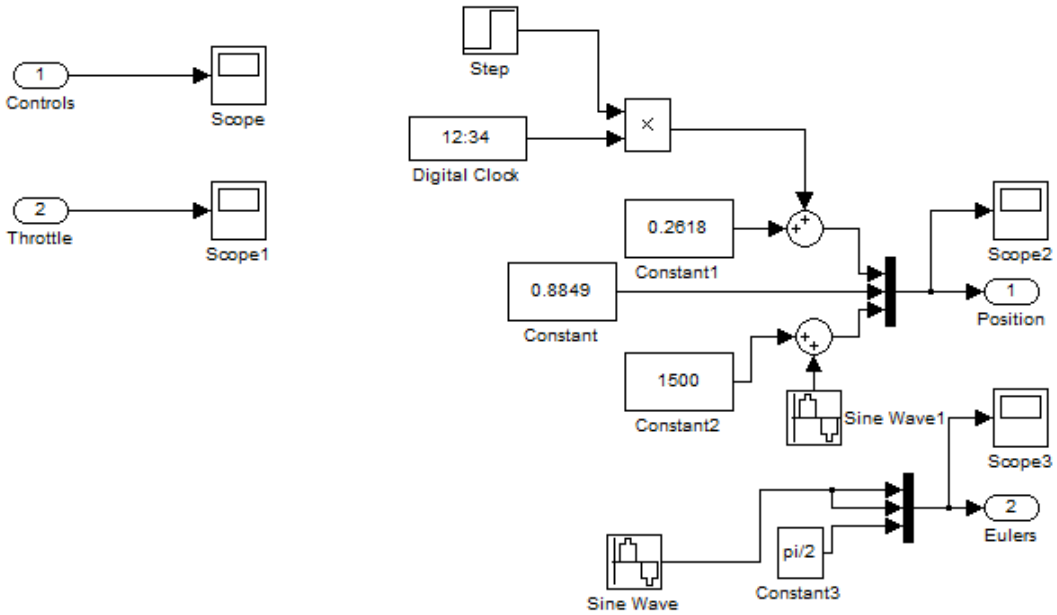
První model obsahuje bloky komunikující s leteckým simulátorem FlightGear, s integrovaným systémem zpracování leteckých dat a externím modelem, kterému přeposílá aktuální letecká data. *Druhý model* představuje distribuovanou část simulačního systému. Zapojení prvního modelu je na obrázku 13.

Blok `Subsystem` v tomto případě reprezentuje část modelu, která obsahuje model simulace navržené pro HIL simulátor. Ten přijímá řídící data od simulátoru FlightGear prostřednictvím bloku `FlightGear Interface`, a předává mu zpět data pro vizualizaci prostřednictvím stejného bloku. V tomto ukázkovém příkladě máme jednoduchý subsystém, který zasílá simulátoru FlightGear fiktivní generovaná data nevázající se nijak na vstupní řídící data. Vstupní data jsou uvnitř subsystému pohlceny respektive přivedeny na bloky `Scope`. Zapojení subsystému je na obrázku 14. Blok generuje na své výstupy následující data. Do výstupního vektoru `Position` je generována rampa do složky `latitude`, konstantní hodnota do složky `longitude` a sinusový průběh nadmořské výšky do složky `altitude`.

Do výstupního vektoru `Eulers` je generován stejný sinusový průběh do složek `roll` i `pitch` a konstantní hodnota do složky `yaw`. Toto zapojení simuluje pohyb letounu vzhledem k zemskému povrchu a zároveň změny jeho orientace v prostoru. Průběh všech vstupních i výstupních veličin lze zobrazit pomocí bloků `Scope`.

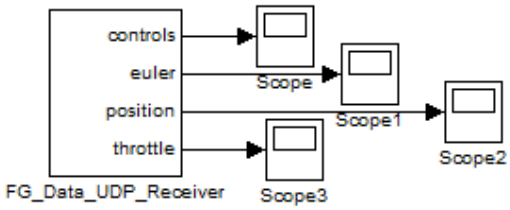


Obrázek 13: Zapojení prvního modelu z ukázkového příkladu



Obrázek 14: Zapojení bloku Subsystem

Druhý Simulinkový model je vcelku jednoduchý a je tvořen pouze blokem `FG Data Receiver` a bloky `Scope`. Zapojení tohoto modelu je na obrázku 15. V tomto modelu pouze ověříme příjem zaslaných dat.



Obrázek 15: Zapojení druhého modelu z ukázkového příkladu

Nastavení parametrů komunikace – Simulace v Matlabu

Před spuštěním simulace je potřeba nejprve správně nastavit všechny komunikační parametry (nebo ověřit již nastavené). Nastavení bude popsáno zvlášť pro každý blok.

Nejdřív nastavíme periodu vzorkování, kterou bude používat jak oba modely, tak jednotlivé bloky v nich obsažené. V příkazové řádce Matlabu nastavíme periodu vzorkování (např. 20ms) příkazem

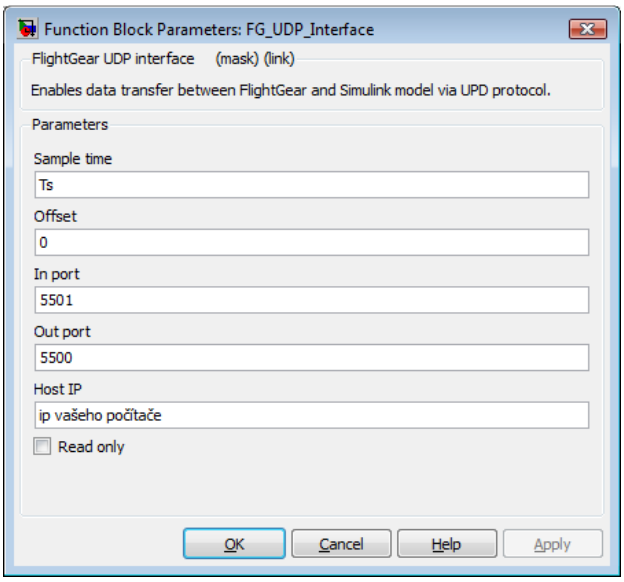
```
Ts = 0.02
```

Tento parametr poté použijeme pro nastavení periody vzorkování obou modelů. V obou modelech zvolíme v záložce `Simulation/Solver/Solver options` nastavení `Type` na `Fixed-step` a po té nastavíme do `Fixed-step size` parametr `Ts`. Nyní můžeme přejít k nastavení jednotlivých bloků.

FlightGear Interface + FlightGear

Nejprve nastavíme parametry pro komunikaci se simulátorem FlightGear. Je potřeba nastavit parametry bloku `FlightGear Interface` a poté správně parametry ve spouštěcím souboru simulátoru `fgio.bat`. Pro účely ukázkové aplikace nastavíme parametry bloku FlightGear podle obrázku 16. Nyní je potřeba správně nastavit parametry v `fgio.bat`. Důležité je, aby se shodovaly patřičné porty, protokol, a frekvence zasílání dat. Parametry ve spouštěcím souboru nastavíme takto:

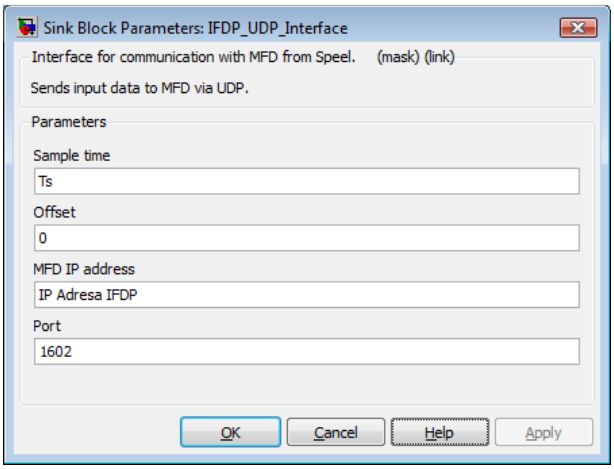
```
set FlightGearInstallDir=instalační adresář FlightGearu
set native-fdm_IP=
set native-fdm_port=5500
set native-ctrls_IP=IP vašeho počítače
set native-ctrls_port=5501
set protocol=udp
set frequency=50
```



Obrázek 16: Nastavení bloku FlightGear interface

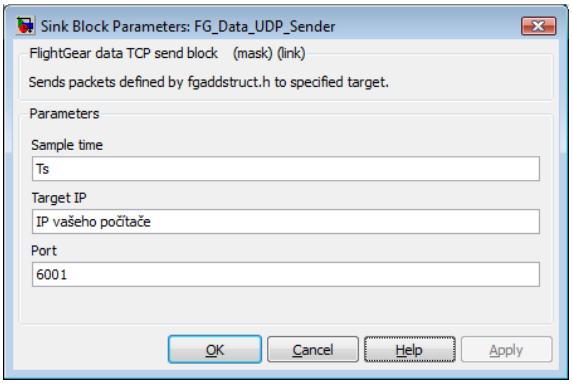
Nastavení bloku IFDP UDP Interface

Pokračujeme nastavením parametrů bloku `IFDP UDP Interface`. IFDP zatím komunikuje na pevno na portu 1602.



Obrázek 17: Nastavení bloku IFDP Interface

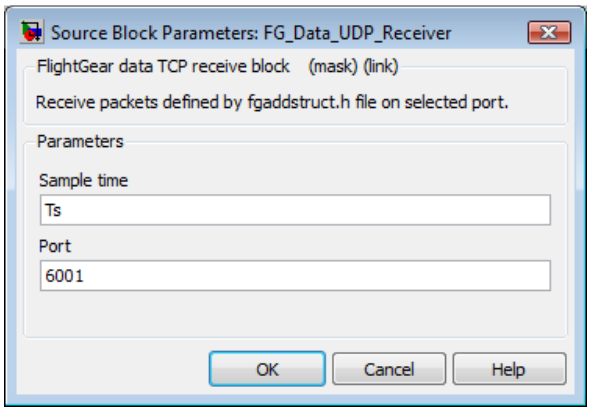
Nastavení bloku FG Data Sender



Obrázek 18:Nastavení bloku FG Data Sender

Nastavení bloku FG Data Receiver

Jako poslední nastavíme parametry bloku `FG Data Receiver` podle obrázku 19, a tím máme nastaveny všude potřebné parametry.



Obrázek 19: Nastavení bloku FG Data Receiver v druhém modelu

Spuštění simulace

Při použití UDP protokolu můžeme spustit FlightGear a modely v libovolném pořadí. Pokud spustíme nejdřív simulace, tak ty budou pozastavené, dokud nezačnou přicházet data od FlightGearu (blokovací sockety na čtení dat, zmíněno v 5.7.2).

Spustíme tedy nejdřív například FlightGear pomocí `fgio.bat` a poté simulaci v obou modelech, a vše by mělo správně fungovat. To znamená, že v simulátoru by měl být vidět vývoj letu a na scopech v modelech přijímaná data.

Poznámka

Při použití TCP bloků je potřeba v `fgio.bat` nastavit protokol na TPC (`set protocol=udp`). Ostatní parametry zůstanou nastaveny stejně. Při spouštění je potřeba dodržet toto pořadí. Nejdříve spustit simulaci v **druhém modelu** s přijímacím blokem, poté simulaci v **prvním modelu** a nakonec spustit simulátor FlightGear pomocí `fgio.bat`.

Nastavení parametrů komunikace – Simulace na počítači BOA 5200

Při spouštění simulace na BOA5200 je postup nastavení do určité míry shodný s předchozím případem. Zde uvedeme pouze rozdíly a parametry, které je potřeba nastavit navíc. Ostatní lze nastavit podle výše uvedeného postupu.

Na počítači BOA budeme spouštět **první model** obsahující blok `FlightGear Interface`.

Změny v fgio.bat

Při simulaci na počítači BOA5200 je potřeba změnit v `fgio.bat`, IP adresu native-ctrls_IP na

`native-ctrls_IP=IP BOA5200.`

Nyní se simulace vykonává na jiném stroji než našem PC, a proto potřebujeme na tento stroj přesměrovat i zasílání dat ze simulátoru. Kontrola IP na BOA5200 se provede následujícím způsobem. Připojíme se na BOA5200 pomocí rozhraní RS232 (nebo pomocí SSH pokud je IP adresa nastavena v rámci používané sítě). Pomocí následujících příkazů přejdeme do adresáře s konfiguračním souborem Ethernetového připojení a vypíšeme jeho obsah.

```
cd etc/init.d
cat config-eth0
```

Pokud je nastavena špatná IP adresa, přepíšeme ji v tomto souboru na správnou. K dispozici je editor `vi`. Pokud ho z nějakého důvodu nemůžete použít (například použitý terminál nepodporuje práci s editorem) můžete použít následující sadu příkazů, které napíšete do terminálu.

```
sed -e "s/aaa\.bbb\.ccc\.ddd/www.xxx.yyy.zzz/" config-eth0 >
config-eth0_new
cp config-eth0_new config-eth0
rm config-eth0_new
```

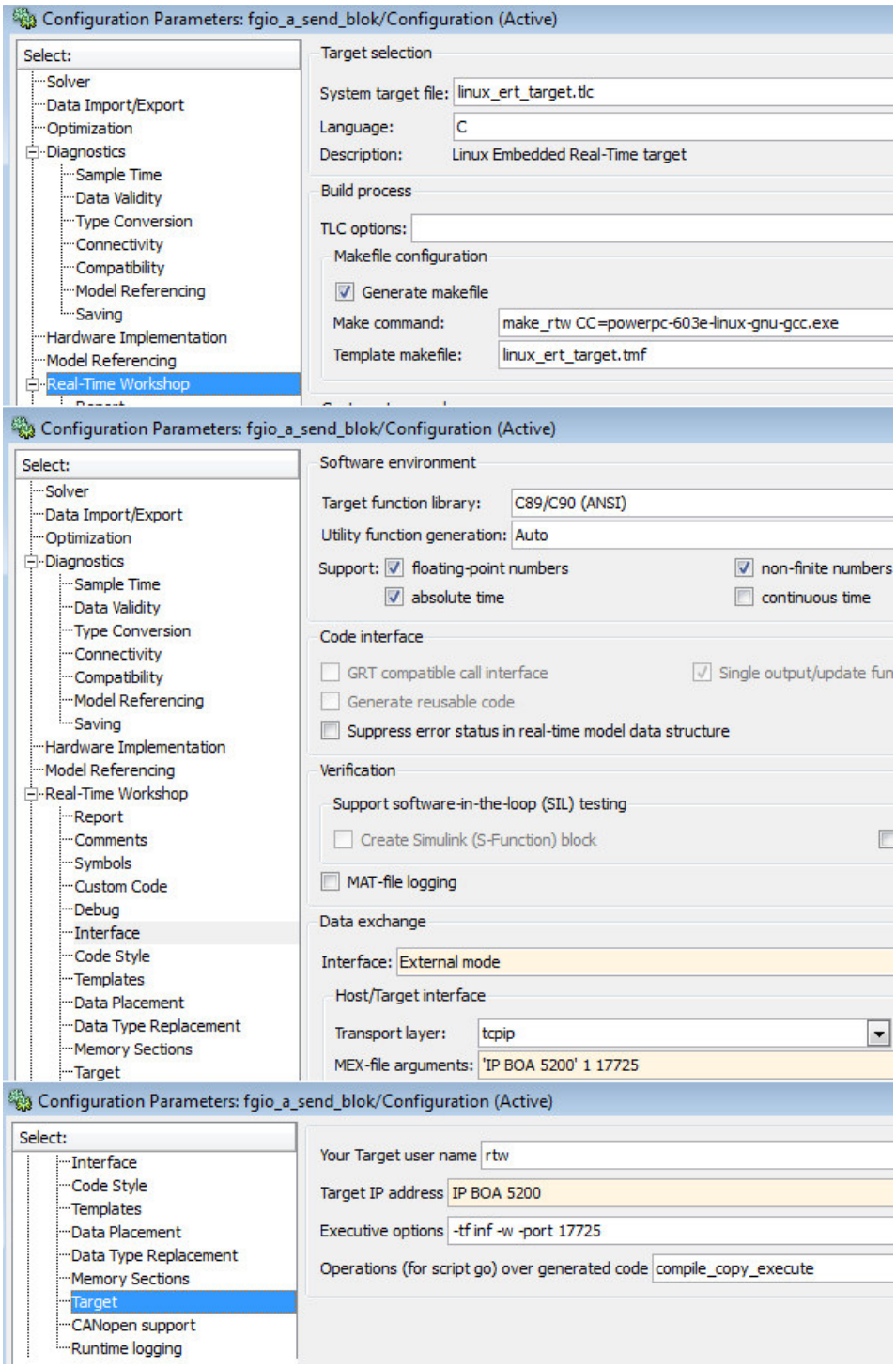
Kde `aaa.bbb.ccc.ddd` je stávající IP adresa a `www.xxx.yyy.zzz` je nová IP adresa. Tato sekvence překopíruje `configh-eth0` do souboru `configh-eth0_new` a nahradí starou adresu novou. Poté zkopíruje novu konfiguraci zpět do `configh-eth0` a `configh-eth0_new` smaže.

Změny v konfiguraci bloků

V parametrech bloků není potřeba provádět žádné změny.

Nastavení parametrů modelu pro spuštění na BOA5200

Před započetím generování kódu pro daný model a spuštěním simulace je potřeba v prvním modelu nastavit parametry v *Simulation/Configuration parameters* podle následujícího obrázku.



Obrázek 20: Nastavení modelu pro spuštění na BOA5200

Poté už zbývá jen nastavit externí mód simulace (*Simulation/External*) a vygenerovat kód. Ten vygenerujeme v okně *Configuration parameters* v záložce *Real-Time Workshop* tlačítkem *Generate code*.

Spuštění simulace

Po vygenerování máme v pracovním adresáři skript `go`. Do pracovního adresáře je potřeba nakopírovat z adresáře, kde je uložen FlightGear blockset, soubory `net_ctrls.hxx`, `net_fdm.hxx`, `stdint.hxx`, `universal.h` a `fgaddstruct.h`. Poté spustíme MSys , přejdeme do pracovního adresáře a spustíme skript `go`. Po kompilaci a zadání hesla se na BOA5200 zkopíruje a spustí spustitelný soubor.

Nyní spustíme FlightGear a oba modely. Postupujeme stejně jako v předchozím případě (pořadí spuštění je opět závislé na použitém protokolu). Jediný rozdíl je při spouštění **prvního modelu** s blokem **FlightGear Interface**. V tomto modelu se nejdříve musíme připojit k BOA5200 pomocí tlačítka *Connect to Target* nebo pomocí stejnojmenné položky v menu *Simulation* a poté spustit simulaci. Po spuštění modelů a leteckého simulátoru by měla začít cyklická výměna letových dat. Vývoj simulace lze opět sledovat v simulátoru nebo na scopech v bloku `Subsystem`.

Příloha C
Použité zkratky

ADI	Attitude Direction Indicator
API	Application Programming Interface
CAN	Controller Area Network
DLL	Dynamic-link library
ECU	Electronic control unit
EHSA	Electro-Hydraulic Servo Actuator
HIL	Hardware In the Loop
HMI	Human Machine Interface
HSI	Horizontal Situation Indicator
HTML	HyperText Markup Language
IFDP	Integrated Flight Data Processor
MEX	Matlab Executable
MFD	Multifunction display
RTW	Real-Time Workshop
SDK	Software Development Kit
TLC	Target Language Compiler
XML	Extensible Markup Language

Příloha D
Obsah přiloženého CD

Přiložené CD obsahuje následující soubory a adresáře:

- `boa5200` – memory images potřebné pro instalaci BOA5200
- `demo` – adresář obsahující ukázkový příklad práce s FlightGear blocksetem.
- `FlightGear` – adresář obsahující instalátor FlightGearu, manuál a konfigurační soubory
- `FlightGear_blockset` – adresář obsahující kompletní FlightGear blockset
- `linux_ert_target` – adreář obsahující Linux ERT Target
- `IFDP` – adresář obsahující specifikaci komunikačních protokolů použitých v IFDP.
- `software` – adresář obsahující software potřebný pro překlad vygenerovaného kódu a jeho spuštění na BOA5200
- `Sending Data to X-Plane.html` – soubor popisující rozhraní simulátoru X-plane
- `dp_2009_cerny_jan.pdf` – vlastní text diplomové práce
- `readme.txt` – krátký souhrn obsahu CD