**ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE**

**Fakulta elektrotechnická**
**Katedra měření**

# Simulovaná rozhraní testovací stanice avioniky

# Simulated Interface of an Avionics Test-Bench

Diplomová práce

Studijní program: Kybernetika a robotika

Studijní obor: Letecké a kosmické systémy

Vedoucí práce: doc. Ing. Pavel Pačes, Ph.D.

Konzultant: Ing. Michal Závišek, Honeywell

**Bc. Jan Frank**

**Praha 2016**

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
Fakulta elektrotechnická
Katedra měření

Akademický rok **2015-16**

# ZADÁNÍ DIPLOMOVÉ PRÁCE

| | |
|---|---|
| Student: | **Bc. Jan Frank** |
| Studijní program: | **Kybernetika a robotika** |
| Obor: | **Letecké a kosmické systémy** |
| Název tématu česky: | **Simulovaná rozhraní testovací stanice avioniky** |
| Název tématu anglicky: | **Simulated Interface of an Avionics Test-bench** |

**Pokyny pro vypracování:**

Cílem je vyvinout softwarové řešení umožňující simulovat reálné ovládací prvky pro zadávání pilotážně navigačních dat, a to buď ze vzdáleně připojeného počítače, nebo skriptem pro automatizaci. Základními úkoly jsou:

- Nakonfigurovat TIU server pro letouny Dassault a Agusta. Vytvořit konfigurační soubor, který nastaví všechny potřebné signály. Zapojit PC a HW ovladače tak, aby se mezi nimi dalo přepínat elektronickými přepínači.
- Vytvořit SW, který umožní zápis a vyčítání dat k SW generovaným ovladačům. SW zapisuje data na sběrnici a jako zpětnou vazbu čte hodnoty znaků, které se zobrazují na displeji. Součástí práce je vytvořit grafickou nadstavbu klienta pro uživatelsky snadné ovládání.
- Vyvinout procedury pro testování vybraných scénářů ovládání bez zásahu uživatele. Procedury budou realizovány ve formě dávkových příkazů (skriptu).

**Seznam odborné literatury:**

[1] Cary R. Spitzer: Digital Avionics Handbook, 2006, CRC Press, ISBN-10: 0849384419

[2] Blasch E., Bosse E., Lambert D.: High-Level Information Fusion. 2012, CRC Press, ISBN-10: 1608071510

| | |
|---|---|
| Vedoucí diplomové práce: | doc. Ing. Pavel Pačes, Ph.D. |
| Datum zadání diplomové práce: | 10. prosince 2015 |
| Platnost zadání do[1]: | 30. září 2017 |

L. S.

Doc. Ing. Jan Holub, Ph.D.
vedoucí katedry

Prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 10. 12. 2015

[1] Platnost zadání je omezena na dobu tří následujících semestrů.

# OBSAH


Anotace ..... 7
Annotation ..... 7
Prohlášení ..... 8
Poděkování ..... 9
1. Úvod ..... 11
2. Stav před řešením práce ..... 12
3. Popis prostředí ..... 13
3.1. Letadla ..... 13
3.2. Testovací stanice avioniky SITS (test bench) ..... 15
3.2.1. Modular avionics unit (MAU) [3] ..... 16
3.2.2. Racal multiplexer a ITA ..... 17
3.2.3. Ovládací prvky a zobrazovače ..... 17
3.2.4. Palubní klávesnice KB-600[5] ..... 18
3.3. Test interface unit (TIU) [4] ..... 19
3.3.1. TIU klienti ..... 20
3.3.2. TATS (TIU-based automated test system) ..... 20
3.4. Zapojení testovací stanice ..... 21
3.4.1. ASCB-D [8] ..... 22
3.4.2. Sběrnice RS 422 ..... 23
3.5. Flight management system - FMS ..... 24
4. Popis problému a možnosti řešení ..... 25
4.1. Využití ..... 25
4.2. Stávající řešení ..... 25
4.3. Navržené řešení ..... 25
4.4. Další prostředky ..... 26
5. Nastavení serveru a úpravy hardwaru ..... 28
5.1. TIU konfigurační soubor pro KB-600 ..... 28
5.1.1. Periodický blok ..... 30
5.1.2. Pseudoperiodický blok ..... 30
5.1.3. Nastavení serveru ..... 31
5.2. Změna zapojení stanice ..... 32
6. Grafická aplikace MKB remote simulation ..... 33

6.1. Framework a knihovny ... 33
6.2. Komunikace ... 33
6.3. Po spuštění ... 35
6.4. Stisknutí klávesy ... 37
6.5. Uživatelské ovládání ... 38
7. Testování aplikace ... 40
7.1. Zkušenosti z užívání ... 40
7.2. Navržená vylepšení ... 41
8. Ovládání kurzoru ... 42
8.1. Úvod k fungování CCD ... 42
8.2. Softwarová simulace CCD ... 44
8.3. Problémy s pohybem kurzoru ... 45
9. Automatizované testování ... 46
9.1. Úvod do testování ... 46
9.2. Automatizace MKB ... 46
9.3. Automatizace CCD ... 47
9.4. Snímání obrazovky ... 48
9.5. Porovnávání obrazu ... 49
9.6. Tvorba automatického scénáře ... 50
9.7. Záznam ovládání uživatelem ... 52
10. Praktické využití skriptů ... 53
10.1. Snímání a porovnávání obrazu ... 53
10.2. Testování databází ... 54
10.3. Testovací scénáře ... 56
10.4. Zhodnocení funkčnosti skriptů ... 58
10.5. Výhody a nevýhody testování se skripty ... 59
10.6. Návrhy na rozšíření ... 60
11. Závěr ... 61
Seznam zkratek ... 63
Seznam obrázků ... 64
Seznam tabulek ... 65
Použitá literatura ... 66
Seznam příloh ... 67

# ANOTACE

Tato práce pojednává o systému umožňujícím dálkové ovládání klávesnice testovací stanice přes podnikovou síť a skriptech sloužících k automatizaci testů. K tomu slouží testovací server simulující signály generované ovládacím rozhraním a speciální skriptovací jazyk TATS vycházející z Visual Basic skriptovacího jazyka. Testovací stanice slouží k verifikaci požadavků softwaru systému Primus Epic. Práce popisuje vznik a funkci uživatelské aplikace dálkového ovládání a několika automatických skriptů, které slouží ke kontrole funkce softwaru avioniky letadla. Závěr práce pojednává o testování softwaru avioniky s využitím automatických testů.

**Klíčová slova:** Kokpit, Primus Epic, dálkové ovládání, Visual Basic skript, automatizace testování, systém správy letu FMS, avionika, systémy zajišťující bezpečný let.

# ANNOTATION

This diploma thesis discuss development of the system capable of keyboard remote controlling at test bench using corporate network and scripts allowing automated tests. There is used test server simulating signals generated by user interface and special scripting language called TATS originating in Visual Basic script language. Test bench serves for verification of software requirements of Primus Epic system. This work describes development of user application for remote control and few automated scripts, which are used to control avionical software of airplane. At the end there is discussed testing of avionical software using automated tests.

**Key words:** Cockpit, Primus Epic, remote control, Visual Basic script, test automatization, flight management system FMS, avionics, safety critical systems.

# PROHLÁŠENÍ

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

Bc. Jan Frank V Praze dne

# PODĚKOVÁNÍ

Chtěl bych touto cestou poděkovat mému konzultantovi, Ing. Michalu Záviškovi a všem kolegům za nejen odborné rady. Firmě Honeywell vděčím za příležitost a technické zázemí.

Nesmím také zapomenout na mou rodinu a přátele, kteří mi poskytovali vytrvalou podporu během celého studia.

# 1. ÚVOD

Integrovaný kokpit Primus Epic, vyvíjený firmou Honeywell, se skládá z mnoha komplexních, redundantních systémů, které prochází průběžnými aktualizacemi. Z toho plynou náročné požadavky na testování, které je často velmi komplexní. Každá nová verze softwaru musí být verifikována – to znamená ověřit, že chování systému je v souladu se schválenými požadavky.

K testování slouží testovací stanice. Tyto komplikované stanice plné avioniky a jiných elektronických systémů jsou nejen náročné na údržbu, ale také velmi drahé. Proto je žádoucí, aby se využily co nejefektivněji. Jedním ze způsobů, jak stanici efektivněji využít je umožnit k ní dálkový přístup bez nutnosti nacházet se na místě stanice. To otevírá možnost vývojářům z jiných poboček firmy, které se nacházejí i na jiných kontinentech, testovat jejich nejnovější software přes internet okamžitě po vydání.

Prvním cílem mé práce je upravit stávající stanici tak, aby umožňovala vzdáleně simulovat uživatelské vstupy z palubní klávesnice. K tomu je potřeba změnit zapojení stanice, nastavit strukturu datového stromu serveru a také vyvinout uživatelsky jednoduchou aplikaci. Celé řešení bude zakomponováno do systému dálkového ovládání testovacích stanic různých typů letadel.

Druhým cílem je vytvořit knihovny příkazů pro skriptovací jazyk, které umožní z jednotlivých příkazů složit scénáře pro testování bez zásahu uživatele. Řešení by mělo umožnit uživateli také vytvářet snímky obrazovky pro kontrolu funkce po ukončení testu. Přínosem je uvolnění kapacity pracovníků, kteří nemusí vykonávat zdlouhavý test a tedy opět úspora nákladů spojených s testováním.

Výstupem mé práce tedy budou nástroje umožňující rychlejší a levnější testování a vývoj nových aplikací pro letadla, což je v dnešním soupeřivém prostředí konkurenční výhodou.

Na úvod této práce upozorňuji, že všechny použité snímky obrazovky systému Primus Epic pochází z vývojové verze systému a neodpovídají produkční verzi softwaru pro žádné z existujících letadel.

# 2. STAV PŘED ŘEŠENÍM PRÁCE

Před zahájením mé práce již simulovaná rozhraní testovací stanice sloužící k dálkovému ovládání existovaly. Aplikace, které k tomuto účelu slouží, jsou nainstalované přímo v počítači u testovací stanice, protože musí využívat jeho hardware. Uživatel k tomuto počítači má přístup pomocí vzdálené plochy. Aplikace generují signály odcházející po sériové lince do testovací stanice podle jeho pokynů. Nazvěme tyto aplikace dálkovým ovládáním první generace.

První generace ovládání má sice své výhody, jako je jednoduchost systému, ovšem i řadu nevýhod. Mezi ty patří především nutnost dalšího hardware a nemožnost využít skriptovacího jazyka k jeho automatizaci.

Proto neexistuje žádné řešení umožňující automatické testy zahrnující používání klávesnice nebo kurzoru. Používané skripty slouží k nastavení letounu či avioniky do požadované konfigurace, aby bylo možné ověřit, že se aplikace chovají jak je požadováno. Poloautomatické testy využívající skriptů poté vyzvou uživatele k ověření jejich výsledku, neumožňují udělat snímek obrazovky.

Testovací stanici nestačí pouze ovládat, uživatel potřebuje také sledovat displeje. K tomu již delší dobu slouží pohyblivá IP kamera. Ta se ukázala jako velmi vhodná pro uživatele, ovšem nedostačuje pro využití k automatizaci, pokud by se mělo zapojit jakékoli rozpoznávání obrazu.

Jak je vidět z odstavců výše, vývoj dálkového ovládání druhé generace a automatizace testování se může opírat o existující řešení a zkušenosti. Přesto bude v některých bodech řešení naprosto nové a nevyzkoušené.

**Tab. 1 – Souhrn existujících a předpokládaných nových řešení**

| | Dálkové ovládání 1. generace | Dálkové ovládání 2. generace | Automatizace testování |
|---|---|---|---|
| Klávesnice (MKB) | Existuje | **Cílem DP** | **Cílem DP** |
| Ovladač kurzoru (CCD) | Existuje | Řeší jiná DP | **Cílem DP** |
| Kamera | Existuje | Shodné s 1. generací | - |
| Video grabber | - | - | **Cílem DP** |

# 3. POPIS PROSTŘEDÍ

## 3.1.LETADLA

Obě letadla, pro která je plánované využít simulované rozhraní využívají integrovaný digitální kokpit Primus Epic firmy Honeywell. Jde o modulární digitální kokpit, který je možné uzpůsobit přání zákazníka. Přizpůsobitelné je nejen využívané přístrojové vybavení, ale také počet displejů či ovládacím prvky.

**Dassault Falcon 7X** - letoun společnosti Dassault patří mezi letouny z kategorie business jet. Jeho délka je přes 23m a rozpětí 26m. Se třemi motory dosahuje rychlosti až mach 0,9 a dolet má téměř 6000 mil. Letoun je vybaven technologií fly-by-wire a umožňuje přistání na krátkých drahách a za nepříznivých meteorologických podmínek. Integrovaný digitální kokpit klade velký důraz na redundanci a přepojitelnost systémů, což komplikuje testování a vyžaduje poměrně komplexní verifikace. [1]


**Obr. 1 – Letoun F7X za letu [1]**

**AgustaWestland AW 139** - vrtulník vyvinutý společnostmi Agusta a Bell patří mezi větší stroje. Jeho dolet je kolem jednoho tisíce kilometrů a maximální vzletová hmotnost až 6400kg s možností přepravovat až 15 cestujících. Má široké využití v civilní i vojenské sféře - k přepravě lidí, nákladů, ale také záchranářským, hasičským či SAR (search and rescue) operacím. Předpokladem k takovému využití je komplexní vybavení kokpitu umožňující operovat v široké škále podmínek se zachováním vysoké bezpečnosti. [2]

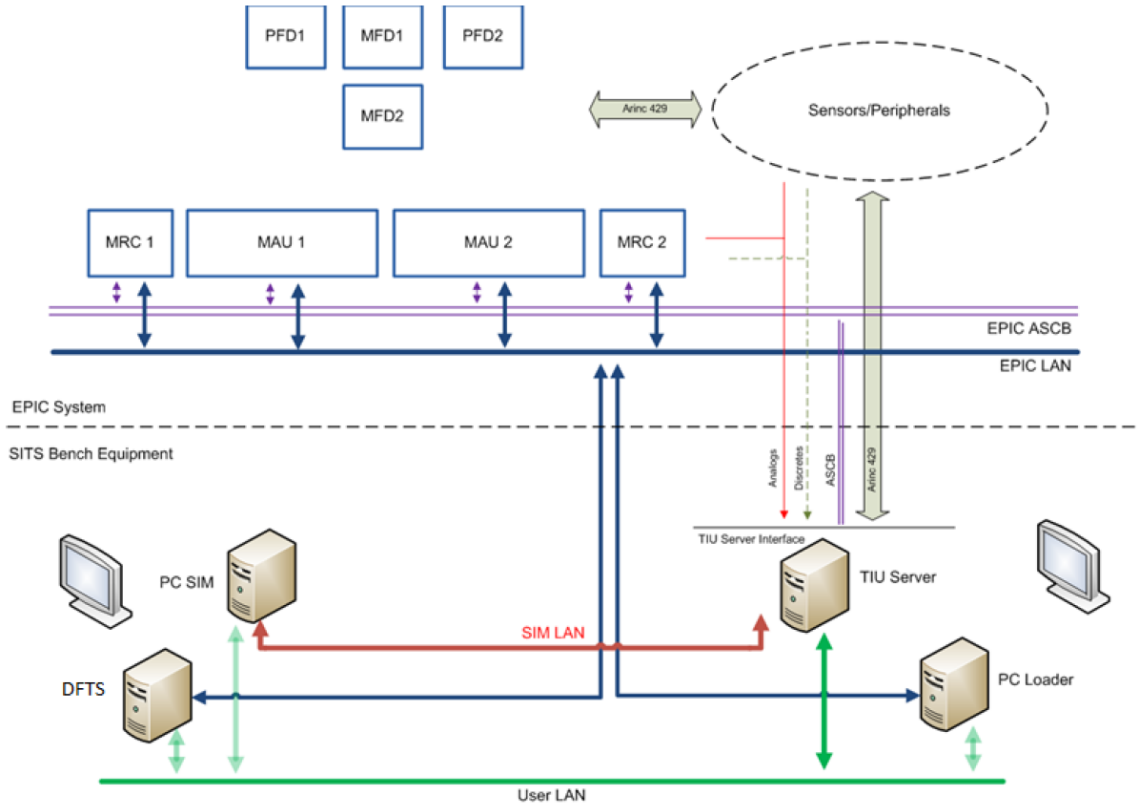

**Obr. 2 - Vrtulník AW139 [10]**

Hlavní rozdíl v ovládání integrovaného kokpitu těchto letadel je jiná použitá klávesnice. Během vývoje jsem se zaměřil na klávesnici letounu F7X na které jsem celý systém testoval. Klávesnice vrtulníku AW139 využívá ke komunikaci Arinc 429 na rozdíl od RS 422 u F7X. Bohužel projekt dálkového ovládání AW139 se v průběhu práce na diplomové práci z projektových důvodů zastavil a nemohl jsem dokončit konfiguraci serveru pro tento vrtulník.

## 3.2. TESTOVACÍ STANICE AVIONIKY SITS (TEST BENCH)

Testovací stanice neboli SITS (system integration test station, testovací stanice pro systémovou integraci) slouží k testování a certifikaci nového softwaru pro digitální kokpity, v tomto případě kokpitu Primus Epic. Fyzicky se jedná o rozměrnou stanici obsahující napájení, kabeláž, sloty pro komponenty, přepínače a jiné. To, jaké komponenty stanice obsahuje, záleží mimo typu letadla také na tom, jaké testy se na ní provádí. V nejvyšší možné konfiguraci je plně vybavená skutečnou avionikou a dalším přístrojovým vybavením letadla.

Stanici můžeme rozdělit na palubní část a část testovací, její základní schéma najdete na Obr. 3. Palubní část se nachází v horní části (označena EPIC system) a testovací ve spodní.




**Obr. 3 - Schéma testovací stanice [3]**

Nejdůležitější prvky stanice pro mou práci jsou vyjmenované níže, jejich popisu se budou věnovat následující kapitoly.

- MAU (modular avionics unit) - palubní část
- Ovládací prvky a zobrazovače - palubní část
- Racal multiplexer a ITA - testovací část
- TIU (test interface unit) - testovací část

Na Obr. 4 je pohled na celou testovací stanici z kamery diskutované v kapitole 4.4. V levé části jsou čtyři displeje avioniky obklopené ovládacími prvky a čtyřmi monitory počítačů, sloužících k ovládání stanice. Bílé moduly jsou racal multiplexer a napravo od nich dvě černé plochy ukrývají ITA (detailní popis obou zařízení v kapitole 3.2.2). Zlaté moduly úplně napravo jsou dvě jednotky MAU diskutované v následující kapitole.


**Obr. 4 - Testovací stanice SITS**

### 3.2.1. Modular avionics unit (MAU) [3]

Jde o kabinet pro různé funkční karty. Vyrábí se v jedno- a dvoukanálové verzi, což společně s výběrem jednotek a počtem slotů od 4 do 24 zajišťuje vysokou modularitu a umožňuje přizpůsobit MAU do nejrůznějších typů letadel. Výměna modulů je jednoduchá, modul stačí zasunout a připevnit šroubem. Všechny konektory karet jsou dostupné z čelní strany.

Karty se po zasunutí připojí vzadu na páteřní 32 bitovou paralelní sběrnici jednotky, která se nazývá VbPCI. Tato komunikuje pomocí modulů NIC (network interface circuit) přes sběrnici ASCB-D s dalšími prvky systému (jiné MAU, PFD, MFD, MRC…). To jaké karty jsou v MAU zapojeny plně určuje její funkčnost a možnosti.

Mezi základní karty patří napájecí zdroj (PSM), již zmíněný NIC, výpočetní modul (PROC), vstupně výstupní karty (I/O), databázové karty (DBM) nebo grafická karta (AGM). Každá karta je vybavena jednotkou BIC (backplane interface circuit), která zajišťuje komunikaci po páteřní sběrnici a je ovládána modulem NIC.

### 3.2.2. Racal multiplexer a ITA

Racal multiplexer je reléový přepínač, který může uživatel ovládat přes počítačový program. Umožňuje tak nastavit, jestli do MAU vstupují vodiče a sběrnice z reálných periferií (ovladače, avionika, rádio, snímače atd.) nebo ze simulovaných zdrojů – tedy TIU (viz kapitola 3.3), simulace letu atd.

ITA je podobné nepájivému poli. Je to rozdělovač, kde lze hardwarovou změnou spojů nastavit, které uzly jsou propojené a tím změnit konfiguraci stanice pro jiné letadlo. Přes něj vedou všechny spoje z hardwaru i simulovaných zdrojů do MAU.

### 3.2.3. Ovládací prvky a zobrazovače

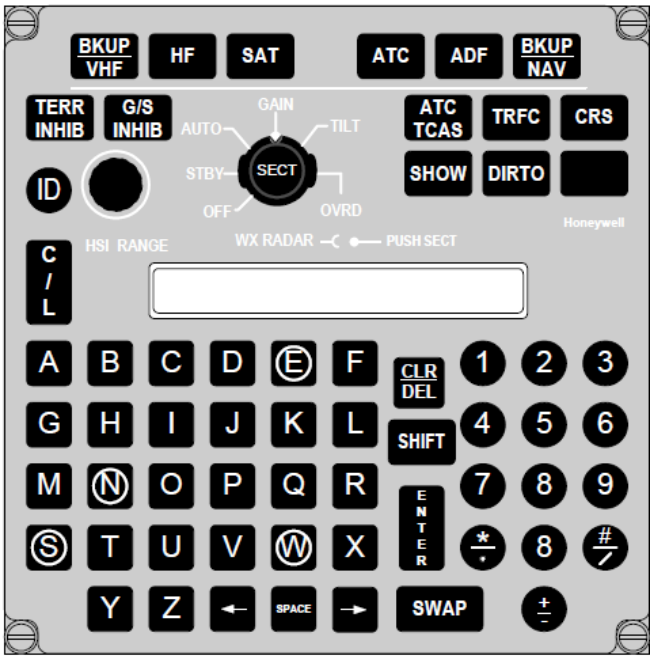
K ovládání integrovaného kokpitu je potřeba především klávesnice a ovladač kurzoru. Tyto ovládací prvky se liší letadlo od letadla a jejich podoba závisí na zákazníkovi. Ovladač kurzoru může být ve formě trackballu, touchpadu, joysticku apod. Klávesnice slouží k zadávání alfanumerických dat k ovládání FMS (flight management system) a rádia či radionavigačních přístrojů. Klávesnice dále nabízí uživateli funkční klávesy s pevně definovanou funkcí.

Tyto ovládací prvky bývají dále doplněny různými specializovanými panely, které umožňují ovládat vybavení jako je autopilot či navigační přístroje. Údaje se zobrazují na dvou typech displejů – MFD (multifunctional flight display) a PFD (primary flight display). Tyto displeje jsou pro mou práci důležité, protože mi předávají zpětnou vazbu o pohybu kurzoru a o zadaném řetězci z klávesnice.


**Obr. 5 - Pohled do kokpitu letounu Falcon 7X [6]**

### 3.2.4. Palubní klávesnice KB-600[5]

Tento typ palubní klávesnice využívá letoun Dassault. Je to jednoduchá klávesnice s jednořádkovým displejem zobrazujícím maximálně 14 znaků. Znaky se zadávají do paměti klávesnice (přitom se zobrazují na displeji) a jejich potvrzení se provádí klávesou Enter. Do paměti je možné umístit maximálně 32 znaků. Rozložení klávesnice je k vidění na Obr. 6.




**Obr. 6 - Rozložení klávesnice KB-600 [5]**

Palubní klávesnice komunikuje pomocí asynchronní sériové sběrnice RS-422 s MAU. Odesílá periodický paket velikosti 48 bytů s periodou 25ms. Tento paket je blíže popsán v kapitole 5.1.1. Pokud je stisknuta funkční klávesa, je ihned nebo po odeslání právě odesílaného periodického paketu odeslán jeden aperiodický byte. Přesný popis komunikace obsahuje interní dokument [5] HRD for KB-600.

Na palubní klávesnici je 5 druhů tlačítek nebo přepínačů:

- Alfanumerické znaky – přidají do paměti nový znak.
- Funkční klávesy – po stisknutí klávesnice pošle paket s informací, která klávesa byla stisknuta.
- Otočné knoflíky – ovládají 7 bitové čítače, jedna otočka má 16 pozic.
- WX radar přepínač – má celkem 6 poloh.
- Další klávesy (Enter, Shift, Del, atd.) – funkce záleží na klávese.

## 3.3. TEST INTERFACE UNIT (TIU) [4]

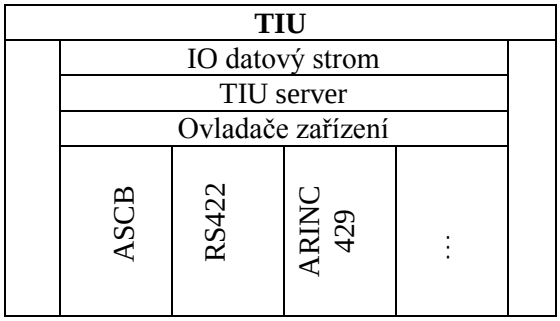
TIU je ve své podstatě vstupně výstupní (IO) server. Slouží ke generování signálů místo hardwaru nebo spravování toků dat, generovaných hardwarem umístěným na testovací stanici. Zároveň zpřístupňuje tyto data uživateli, čímž mu umožňuje zaměřit se na testování na vyšší úrovni bez nutnosti zabývat se ovladači nižších úrovní. TIU je server běžící na Windows (aktuálně se využívají Windows XP), ale možnost připojení přes Ethernet umožňuje vysokou nezávislost na platformách a softwaru.

TIU poskytuje náhradu hardwarových signálů s využitím unikátních jmen, která jim přiřadí. Jako příklad uveďme signál:

```
ASCB D/Irs1/irs12ms/pitchAngle
```

První úroveň ASCB-D značí, že se jedná o sběrnici ASCB D (ta je blíže diskutovaná v kapitole 3.4.1, jde o jeden ze základních stavebních kamenů systému Primus Epic). Druhá úroveň udává funkční jméno, třetí úroveň značí periodu odesílání signálu a až poslední úroveň je názvem samotného parametru. Každý signál má svou unikátní cestu a ne všechny cesty musí mít tento konkrétní formát.

**Tab. 2 - Schéma TIU serveru**




V tomto a dalších odstavcích jsou diskutované jednotlivé vrstvy TIU zobrazené v Tab. 2. IO datový strom definuje vlastnosti každé IO karty a obsahuje proměnné, které reprezentují jednotlivé hardwarové signály proměnné v čase. Každá proměnná má vlastní adresu a je uzlem datového stromu. Různé typy proměnných můžou mít různé vlastnosti. Mezi základní společné vlastnosti patří: datový typ, pozice v hierarchii datového stromu, jméno, popis, jednotka apod.

TIU server je program, komunikující s pomocí ovladačů se vstupně výstupními kartami a umožňuje uživateli přístup k informacím v datovém stromu. Server nemá uživatelské rozhraní, protože se jedná o systémovou službu. Je nastavován skrze konfigurační program nainstalovaný se serverem. Ten mimo nastavení serveru umožňuje také nahrání a změnu struktury datového stromu a nastavení ovladačů. Sever umožňuje 3 módy funkce – real, virtual a off. V reálném módu využívá server IO karet k odesílání a přijímání dat, což je normální funkce. Ve virtuálním módu server data ukládá, ale neposílá je ven, proto nepotřebuje připojené IO karty.

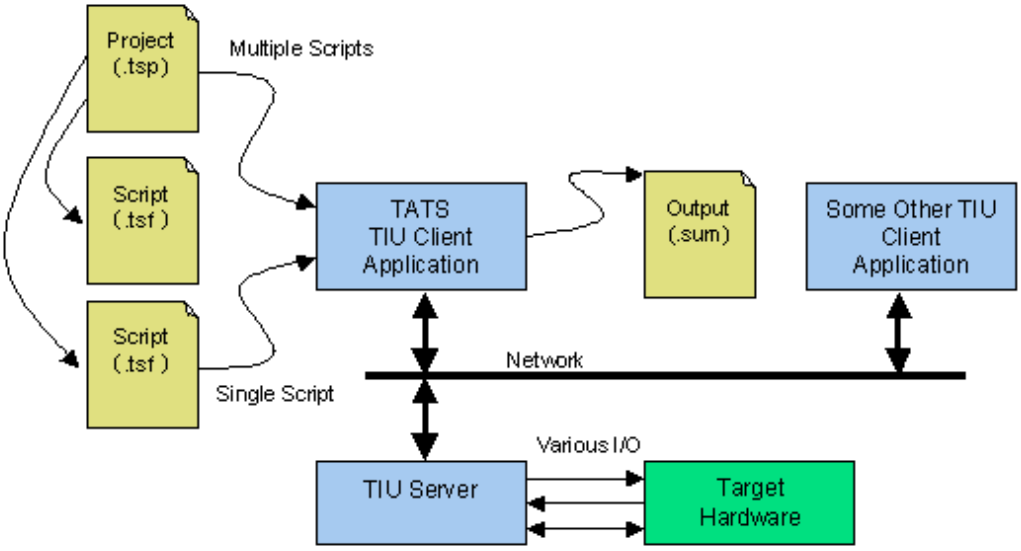
K serveru lze připojit v podstatě libovolnou IO kartu se kterou komunikuje pomocí ovladačů. Ovladače sdílí se serverem přenosový buffer, ve kterém se shromažďují data k odeslání resp. přijatá data, která tak je možné monitorovat.

### 3.3.1. TIU klienti

Klienty se připojují k serveru přes BSD socket pomocí Ethernetu a využívají protokol UDP/IP k přenosu dat a TCP/IP k přenosu povelů. Klienti mají různý účel, v režimu čtení slouží především k zobrazování a nahrávání průběhů dat a nebo naopak v režimu zápisu k jejich emulaci. Umožňují také přehrávání nahraných záznamů do datového stromu nebo spouštění předdefinovaných průběhů (harmonické signály, šum...).

### 3.3.2. TATS (TIU-based automated test system)

TATS skripty jsou nástroj využívaný k automatizovanému testování a ověřování hardwaru a softwaru. Jsou založené na využití TIU serveru k abstrakci hardwarových signálů různých typů za jména (uzly datového stromu). Skript umožňuje číst a zapisovat s pomocí těchto jmen na server. Schéma fungování skriptů je na Obr. 7.



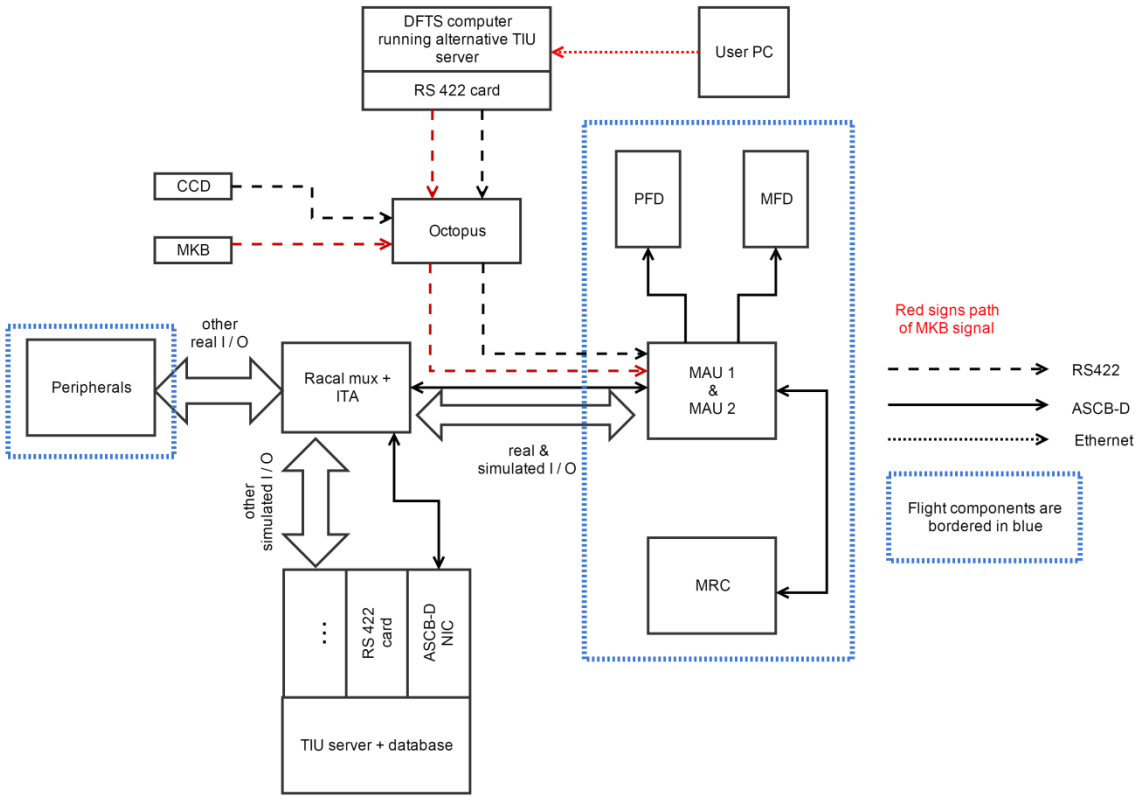

**Obr. 7 - Schéma TATS skriptů [3]**

TATS je také zvána aplikace s integrovaným vývojovým rozhraním těchto skriptů. Jde zároveň o TIU klient a tak umožňuje nejen úpravu skriptů, ale také monitorování jejich vykonávání a debuggování.

TATS skripty jsou psané v Microsoft Visual skriptovacím jazyce, doplněném o rozšíření umožňující komunikaci s TIU serverem. Jejich zdrojové kódu mají příponu .tsf. Ukázku kódu včetně rozšíření najdete v příloze č. 2.

## 3.4.ZAPOJENÍ TESTOVACÍ STANICE

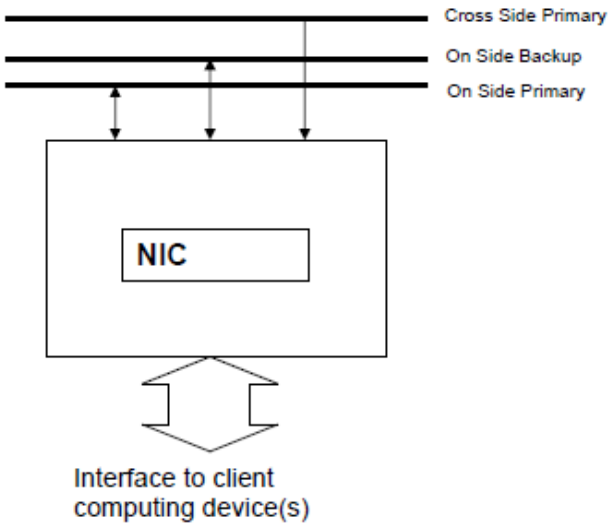
Ústředním prvkem každého integrovaného kokpitu Primus Epic, stejně jako testovací stanice, je MAU. Ta pomocí modulárních karet zajišťuje všechny funkce integrovaného kokpitu. Díky modularitě umožňuje pracovat se vstupy a výstupy téměř libovolného druhu od analogových přes digitální až po pokročilé sběrnice. Příkladem můžou být sběrnice ARINC-429 a RS422 využívané ke komunikaci s ovládacími prvky. Speciálním případem potom je sběrnice ASCB-D, která tvoří páteřní komunikační linku systému Primus Epic. Do MAU vstupují údaje ze senzorů a jiných periferií a z ní vystupují povely či jiná data pro aktuátory, záznamníky, zobrazovače a mnohé jiné.




**Obr. 8 - Schéma původního zapojení testovací stanice**

### 3.4.1. ASCB-D [8]

ASCB-D je sériová, duálně redundantní, polo-duplexní sběrnice s vysokou mírou spolehlivosti využívaná v letectví. Byla vyvinuta firmou Honeywell. Každá sběrnice se dělí na dvě strany (on-side a cross-side), přičemž každá strana má dvě linky. Umožňuje přenosové rychlosti až 10Mbit/s resp. 16Mbit/s.



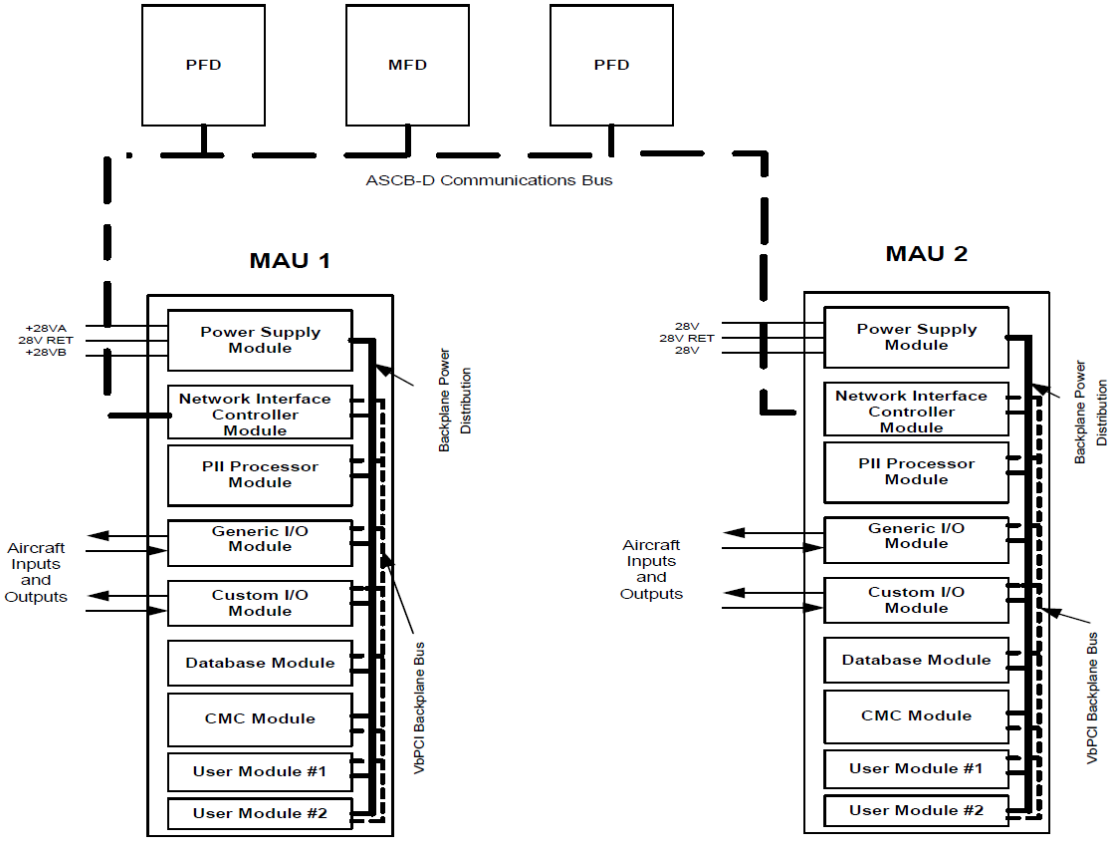

**Obr. 9 - Schéma připojení NIC ke sběrnici ASCB-D [8]**

Každé zařízení připojené k této sběrnici pro komunikaci po ní musí využívat NIC (network interface controller). Každý NIC je připojen ke třem sběrnicím, dvě jsou jeho primární a jedna sekundární. Toto zapojení vidíte na Obr. 9, na tomto ovšem chybí ještě jedna sběrnice – cross side backup. K té není tento konkrétní NIC připojený, protože je to jeho sekundární sběrnice.

Sběrnice pracuje s frekvencí 80Hz. Každý rámec začíná synchronizační zprávou a skládá se z N zpráv. Zprávy jsou přenášeny ve třech taktech – v prvním jsou data generována ve vysílači, ve druhém vysílána a ve třetím zpracována v přijímači. Rámce se skládají z periodických datových balíčků, každý NIC může vysílat 0 nebo více datových balíčků. Každý balíček začíná hlavičkou, následují data a končí CRC kontrolou.

Myšlenkou ASCB-D není poskytovat komunikační kanál mezi dvěma zařízeními. Jejím hlavním úkolem je tvořit síť mezi všemi MAU a dalšími připojenými jednotkami (např. displeji či rádii). Moduly v MAU jsou připojené pomocí BIC na páteřní síť, která využívá NIC ke komunikaci s dalšími MAU (viz kapitola 3.2.1). Ukázkový diagram tohoto zapojení vidíte na Obr. 10.

Pro software jsou potom všechny jednotky zdánlivě propojené v jediné síti, ve které každá jednotka přijímá všechna data. Připojení k této síti zajišťuje všem softwarovým funkcím rovnocený přístup k datům bez ohledu na jejich fyzickou alokaci.




**Obr. 10 - Diagram systému Primus Epic [8]**

### 3.4.2. Sběrnice RS 422

Jde o sériovou, napěťově vyváženou, simplexní linku, která může mít pouze jeden vysílač a až 10 příjemců. Pokud potřebujeme obousměrný přenos, musíme využít dvou linek. Každá linka se skládá ze dvou vodičů, přijímací zařízení detekuje diferenciální napětí. Rozdíl musí být větší než 0,2V a logická hodnota přenášená po vedení je určena jako polarita tohoto rozdílu. Přijímací zařízení musí být impedančně přizpůsobeno vedení. Přenosová rychlost může na krátké vzdálenosti dosahovat až 10Mbit/s.

Palubní klávesnice KB-600 využívá pouze jednosměrnou asynchronní komunikaci (z MKB do MAU). Přenos probíhá s následujícím nastavením:

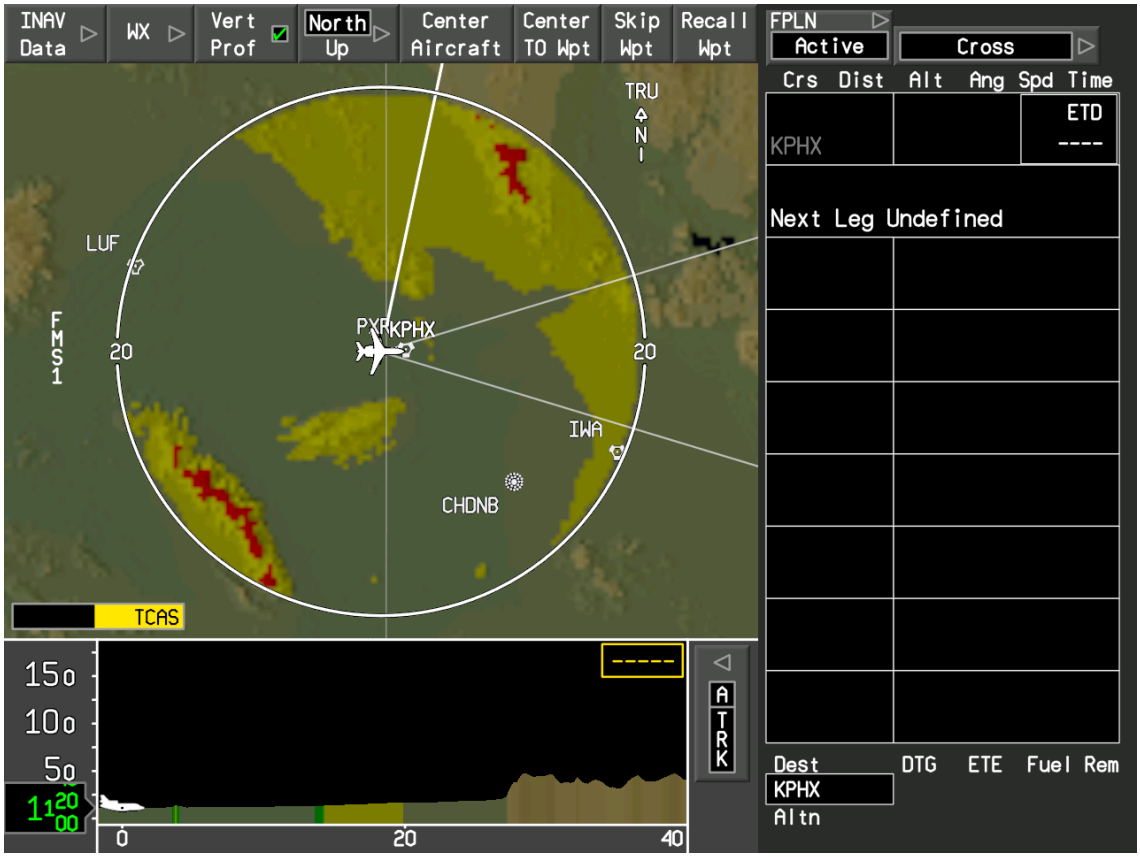
**Tab. 3 - Vlastnosti přenosu sériové linky**

| | |
|---|---|
| Přenosová rychlost (baud rate) | 112,5K |
| Počet datových bitů | 8 |
| Počet start bitů | 1 |
| Počet stop bitů | 1 |
| Paritní bit | není |

## 3.5.FLIGHT MANAGEMENT SYSTEM - FMS

Flight management system (neboli FMS) je systém umožňující posádce pracovat za letu s letovým plánem a navigaci po trati. Obsahuje navigační databázi letových bodů, jako jsou letiště, letové trasy, radionavigační majáky či příletové a odletové trasy. FMS využívá různých systémů (například GPS či INS) k zjištění pozice letounu a zobrazení na mapě. K ovládání slouží palubní klávesnice a ovladač kurzoru (CCD).

FMS je v systému Primus Epic součástí integrovaného kokpitu. Jde o softwarovou funkci běžící v několika instancích (až třech), která je alokovaná do několika procesorových modulů. FMS sdílí zobrazovací prvky (displeje) i ovládací prvky (MKB a CCD) s EFIS. Uživatel zadává textové údaje z klávesnice a číselné buď z klávesnice, nebo otočným knoflíkem na CCD. Výhodou je, že FMS po zadání jedné hodnoty a jejím potvrzení přeskočí kurzorem sám na další požadovanou hodnotu.




**Obr. 11 - Navigační mapa a seznam letových bodů FMS**

Každý displej je rozdělen na šestiny a každá aplikace zabírá celistvý násobek šestiny displeje. Na Obr. 11 vidíte 4/6 mapu a 2/6 seznam letových bodů. Nastavení je flexibilní a umožňuje tak pilotovi nastavit vzhled displejů podle jeho představ.

# 4. POPIS PROBLÉMU A MOŽNOSTI ŘEŠENÍ

## 4.1. VYUŽITÍ

Na testovací stanici se provádí různé druhy testů. Může jít o ověření funkcí EFIS systému, či komplexní rozsáhlé testy pro verifikaci a ověření letuschopnosti. Palubní klávesnicí se zadávají různé hodnoty, zkratky letových bodů, frekvence, souřadnice apod. Částečně umožňuje také přepínání oken synoptiky (paralelních oken) a ovládá srážkový radar.

Tato práce se zaměřuje dvěma směry: vzdálené ovládání integrovaného kokpitu pomocí simulované MKB k umožnění uživatelského ovládání z libovolného počítače v síti k drobnému testování či ověření funkce. Druhým směrem je plně automatizované testování. Tím je například ověření databází FMS, toto testování je náročné na zadávání různých údajů uživatelem.

## 4.2. STÁVAJÍCÍ ŘEŠENÍ

Než jsem začal pracovat na této práci, bylo již implementováno řešení, které umožňuje simulování MKB ze vzdáleného počítače. Ovšem toto řešení není ideální z několika důvodů: jde o komerční a zpoplatněné řešení, uživatel se musí připojit přes vzdálenou plochu k určenému počítači a především vyžaduje hardware, který není běžnou součástí testovací stanice. Toto řešení navíc nepodporuje skriptování vstupů z klávesnice.

Stávající řešení funguje na principu připojení zařízení na sběrnici mezi MKB a MAU. Toto zařízení je interně nazýváno Octopus (název nese podle množství kabelů připomínajících chapadla). Počítačový program umožňuje přepínat Octopus tak, že je s MAU propojeno buď reálné MKB nebo sériový výstup z počítače. Komunikaci po sériové lince z počítače zajišťuje program simulující klávesnici. Jedním z nedostatků je, že klávesnici nejde přímo simulovat vzdáleně, ale je nutné se připojit přes vzdálenou plochu k počítači v místě testovací stanice. Další nevýhody tohoto řešení byly shrnuty v minulém odstavci.

## 4.3. NAVRŽENÉ ŘEŠENÍ

Mnou navržené řešení oproti původnímu nevyžaduje žádný dodatečný hardware, pouze úpravu stávajícího propojení a také nový software. Řešení využije stávající TIU server ke generování sériové komunikace mezi MKB a MAU. Serveru je potřeba změnit konfiguraci tak, aby umožňoval tuto novou funkci. Dále je nutné upravit zapojení stanice tak aby se dalo přepínat mezi reálným zdrojem (hardwarovou MKB) a simulovaným (TIU).

Ve druhé části své práce potom navrhnu dvě možnosti jak instruovat server k tomu, co má po sériové lince vysílat. Prakticky jde o zapisování do uzlů datového stromu serveru, podle které se vytváří vysílaný řetězec. První možností je aplikace, která se přes počítačovou síť připojí k serveru a bude do jeho datového stromu zapisovat potřebné údaje. Tato aplikace s grafickou nástavbou bude uživateli simulovat reálný hardware funkčně i vizuálně a zároveň umožní zadávat uživateli znaky pomocí klasické počítačové klávesnice.

Druhou možností, které také využiji, je použití skriptovacího jazyka TATS popsaného výše. Skripty umožní vytvořit plně automatizované testy, které nejenom budou ovládat integrovaný kokpit s pomocí simulování MKB a CCD, ale také zaznamenávat průběh testu pomocí snímků obrazovek PFD a MFD. Záznam obsahu displejů je důležitý pro zpětné ověření správné funkčnosti EFIS a vyhodnocení testů.

## 4.4.DALŠÍ PROSTŘEDKY

Ovládání integrovaného kokpitu není možné pouze se simulovanou MKB. Kromě klávesnice uživatel potřebuje vizuální zpětnou vazbu a možnost pohybovat kurzorem. Toho bude při dálkovém ovládání dosaženo těmito způsoby:

Kurzor bude ovládán samostatnou aplikací nebo skriptem. Aplikace pro ovládání kurzoru (simulace CCD) je vyvíjena souběžně s mou prací. Obě řešení sdílí některé společné prvky systému – především jde o jeden TIU server a tedy jednu sériovou RS422 kartu. V budoucnu se uvažuje o spojení obou aplikací (simulace MKB a CCD) do jedné.


**Obr. 12 - Pohled na PFD z kamery SNC 550**

Ke sledování dění na displejích bylo zvoleno jednoduché řešení – snímá je pohyblivá IP kamera, která uživateli umožňuje sledovat i jemné detaily (především číst text). Uživatel se k jejímu rozhraní připojí přes internetový prohlížeč zadáním její IP adresy.

Na stanici v Brně využíváme kameru Sony SNC 550 s rozlišení 1280x720 a 28 násobným optickým zoomem. [7] Na Obr. 12 vidíte pohled z této kamery na displej při asi 3 / 4 maximálního optického přiblížení. Je z něj zřejmé, že všechny potřebné údaje jsou dobře čitelné a navíc se v případě potřeby může pohled dále přiblížit.

Dění na obrazovkách je možné snímat také pomocí DVI → USB video grabberu připojenému k palubní grafické kartě. Ten se ale ukázal pro online sledování nepraktický a přenos obrazu málo plynulý. Toto řešení bude využito pouze pro záznam jednotlivých snímků z průběhu automatizovaných testů.

# 5. NASTAVENÍ SERVERU A ÚPRAVY HARDWARU

## 5.1. TIU KONFIGURAČNÍ SOUBOR PRO KB-600

Prvním z řešených úkolů musela být konfigurace serveru TIU. Tento server po spuštění vytváří datový strom s jeho uzly na základě textových souborů. Tyto soubory jsou specifické pro každou periferii (IO kartu) a v jejich těle je zakódovaná struktura datového stromu včetně kompletních cest a názvů proměnných. Datový strom poté využívá dané zařízení (k fungování viz kapitola 3.3).

Mnou editovaný soubor tedy obsahuje konfiguraci datového stromu pro komunikační kartu sériové sběrnice RS-422. Protože TIU nevyužívá sériovou komunikaci pouze pro MKB musí tento soubor obsahovat také jiná nastavení.

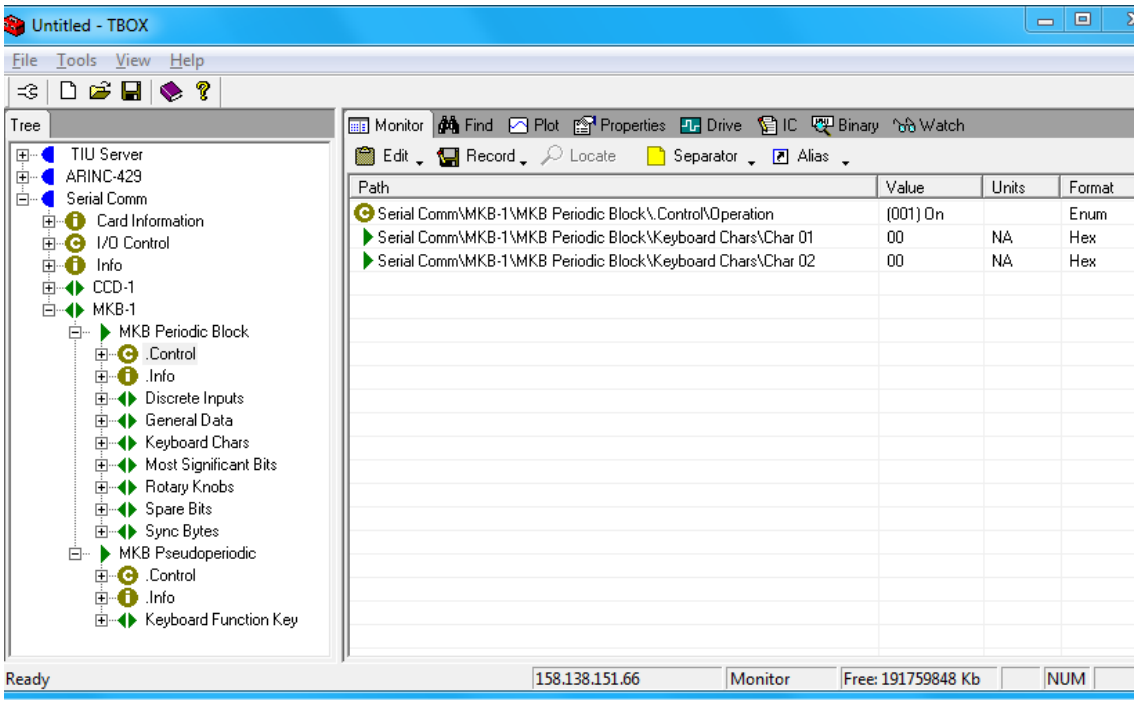
V konfiguračním souboru je potřeba nastavit následující:

- Kanál – určuje vlastnosti sériové linky: číslo zařízení (RS karty), používaný port, baud rate, stopbit, počet datových bitů a paritu.
- Blok – definuje vlastnosti bloku, především: periodu, počet elementů, velikost elementů v bytech, tx/rx, periodický/aperiodický, endianitu, aj. Na jednom kanálu můžeme vysílat více bloků.

Dále se v konfiguračním souboru objevují klíčová slova GROUP a DATA. První z nich vytváří novou skupinu, což je vlastně uzel (větvení) datového stromu. Klíčové slovo DATA udává cestu k paměti v datovém stromu. Nastavení pro DATA jsou: počáteční bit ve zprávě, velikost, rozlišení, offset, počáteční hodnota, jednotka, aj.

V příloze č. 1 najdete ukázku formátu konfiguračního souboru.

Na Obr. 13 můžete vidět okno programu TBOX. Tento interní program firmy Honeywell je jedním z TIU klientů a umožňuje náhled na datový strom uložený v TIU. V levém podokně vidíte strukturu stromu a v pravém uživatelem vybrané proměnné. S pomocí tohoto obrázku bych rád vysvětlil zvolenou strukturu vytvořeného datového stromu.




**Obr. 13 - Okno programu TBOX zobrazující strukturu datového stromu**

Nejvyšším uzlem je Serial Comm. Název tohoto uzlu nelze nijak změnit, slouží k rozlišení IO karty a spadají pod něj všechny sériové linky. Pod ním vidíte uzly CCD-1 a MKB-1, zde jde o jednotlivá nastavení kanálů. Kanál CCD slouží k ovládání kurzoru, MKB je mnou nakonfigurovaný kanál. Jednička značí první klávesnici tak, aby bylo možné v budoucnu případně přidat druhou pro kopilota (ve většině kokpitů jsou klávesnice dvě).

Dále vidíte dva bloky - MKB Periodic block a MKB Aperiodic block. Tyto popisují následující podkapitoly. Kompletní konfigurační soubor je součástí příloh. Řádky začínající apostrofem jsou ignorovány (jde o komentáře), v těchto řádcích je podrobně vysvětlena syntaxe konfiguračního souboru.

### 5.1.1. Periodický blok

Nastavení tohoto bloku vychází z hardwarových vlastností palubní klávesnice. Především tedy perioda 25ms a délka 48bytů. V tomto bloku se v každé periodě vyšle jedna zpráva obsahující všechny byty:

**Tab. 4 - Přehled bytů vysílaných ve zprávě**

| Č. bytu | Obsah bytu |
|---|---|
| 0 | Počáteční synchronizace |
| 1-2 | ID zařízení a verze dat |
| 3-6 | Čítače otočných tlačítek |
| 7-8 | Data z diskrétních vstupů |
| 9 | Stavové bity |
| 10-41 | Obsah displeje |
| 42 | Rezervováno pro budoucí rozšíření |
| 43-46 | Kontrolní součet |
| 47 | Koncová synchronizace |

Jak vidíte na Obr. 13, je blok rozdělen do sedmi skupin, které jsem nadefinoval v konfiguračním souboru. Uživatel si může libovolně zvolit počet skupin a jejich pořadí v databázi. Toto dělení nemá žádný vliv na strukturu vysílané zprávy, určující je pořadí bitu ve zprávě, které je definováno pro každou skupinu.

### 5.1.2. Pseudoperiodický blok

V průběhu práce na konfiguraci TIU jsem zjistil, že jeho aktuální verze sice umožňuje nastavit aperiodický blok, ale nenašel jsem způsob jak jej odeslat. Z důvodu chybějící dokumentace se mi nepodařilo zjistit, zda je odesílání aperiodických dat nemožné a nebo zda potřebuje jen nějaké další nastavení pro správnou funkci.

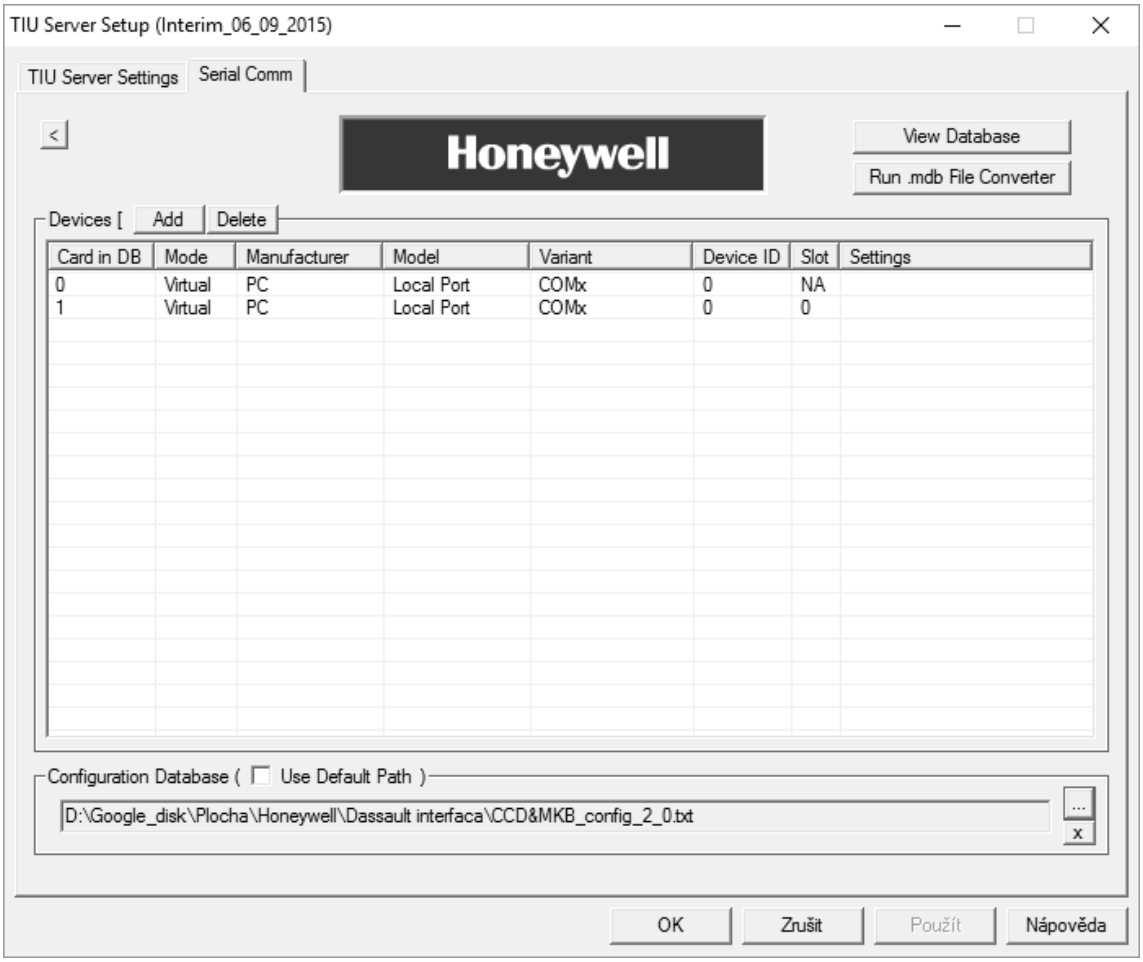
Z tohoto důvodu jsem musel najít náhradní řešení pro odesílání aperiodických dat. Jako nejjednodušší variantu jsem zvolil tuto: v databázi jsem nakonfiguroval druhý periodický blok s velkou periodou (1000ms). Odesílání tohoto bloku je ihned po zapnutí aplikace vypnuto a zapíná se pouze v případě stisknutí funkční klávesy na dobu 1025ms. To zajistí, že se tento paket odešle přesně jednou a potom je opět odesílání zastaveno.

Zapínání a vypínání odesílání pseudoperiodického bloku musí zajistit uživatelská aplikace, protože server TIU neumožňuje spustit odesílání jen na omezenou dobu. Je tedy implementováno v dále diskutované aplikaci, případně ve skriptu.

### 5.1.3. Nastavení serveru

Před spuštěním serveru je potřeba nastavit používanou sériovou kartu a přidat do jejího nastavení konfigurační soubor. Ten při spuštění serveru nastaví jeho databázi do požadované podoby.

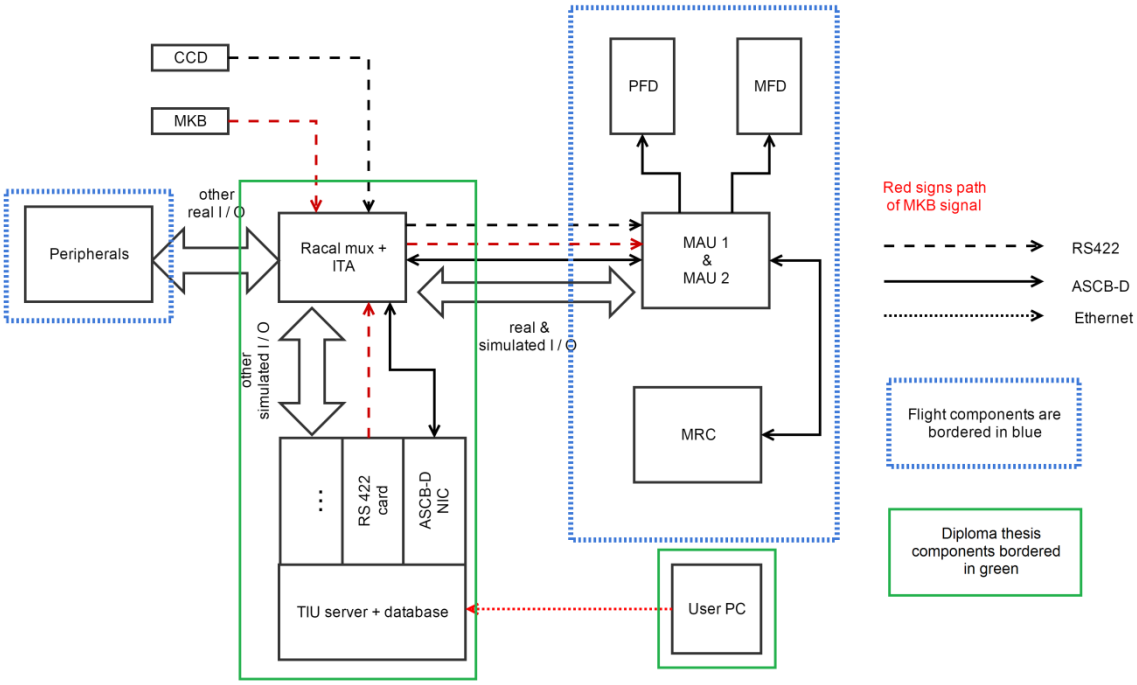
Příklad nastavení pro stanici Dassault F7X je na Obr. 14. V tomto případě se používá jedna sériová karta (s ID 0), která má více portů. První z nich se používá pro simulování CCD a druhý pro simulování MKB. Datové stromy pro obě zařízení jsou nastaveny v jednom konfiguračním souboru.




**Obr. 14 - Nastavení TIU serveru**

## 5.2. ZMĚNA ZAPOJENÍ STANICE

Nové zapojení stanice je na Obr. 15, pokud ho porovnáme s Obr. 8, změna je jasná na první pohled. Není potřeba zařízení Octopus, které není běžnou součástí stanice. Signály z MKB a CCD míří nyní společně se signály ze stávajícího TIU do přepínače racal multiplexer. Ten je ovládaný uživatelem z počítače a umožňuje zvolit, které ze vstupujících signálů pokračují do MAU.




**Obr. 15 - Schéma upraveného zapojení testovací stanice**

# 6. GRAFICKÁ APLIKACE MKB REMOTE SIMULATION

Prvním řešením, jak komunikovat s TIU je vyvinout novou aplikaci pro Windows. Cílem je klient, který se připojí k serveru z libovolného počítače přes Ethernet a zapisuje přímo do jeho datového stromu požadované hodnoty. Uživatel komunikuje přes grafické rozhraní, které vizuálně připomíná palubní klávesnici.

## 6.1. FRAMEWORK A KNIHOVNY

Aplikaci jsem vyvíjel v prostředí Microsoft Visual Studio 2015 na platformě .NET 4.0. Tato platforma umožňuje využít mnoho jazyků, já jsem zvolil managed C++, protože jsem s ním měl dřívější zkušenost. Platforma obsahuje grafickou nadstavbu Windows form, která umožňuje jednoduše vytvářet grafické rozhraní. Jde o událostmi řízené prostředí.

Jak se v průběhu vývoje ukázalo, volba této kombinace nebyla nejvhodnější, protože Microsoft podporuje především Windows form aplikace v C#. Navíc existuje požadavek na využití aplikace pod Windows XP, které mají s kompatibilitou .NET ve verzi 4.0 problém a vyšší verze dokonce nepodporují vůbec.

Aplikace využívá dvě existující knihovny vytvořené firmou Honeywell pro komunikaci s TIU serverem. Jsou to ns_socket, která vytváří API pro knihovnu tiuiface. Druhá z knihoven implementuje funkce pro práci se serverem, jako jsou funkce k připojení, nahrání a stažení dat nebo přehrání nadefinovaných průběhů proměnných. Definuje také datové typy s tímto spojené.

## 6.2. KOMUNIKACE

Stěžejním úkolem aplikace je komunikace s TIU serverem. Ta probíhá přes Ethernet na protokolu UDP/IP resp. TCP/IP. Protokol UDP slouží pro přenos dat (není garantováno doručení paketu), zatímco protokolem TCP se přenáší povely. Firemní knihovna ns_socket kompletně zajišťuje transportní vrstvu. Knihovna tiuiface mi poskytuje část funkcí aplikační vrstvy. Například funkce pro připojení a odpojení od serveru a také pro zápis a čtení obecné proměnné. Nižšími vrstvami komunikace jsem se nezabýval.

Poznámka: server TIU používá vlastní datové typy, mezi ty základní patří bezznaménkový byte, float nebo enum. Pro ukládání v databázi však využívám pouze bezznaménkový byte. Před odesláním a po přijetí je tedy nutné v aplikaci klienta převést data do tohoto typu. To také zajišťují funkce pro odesílání a příjem.

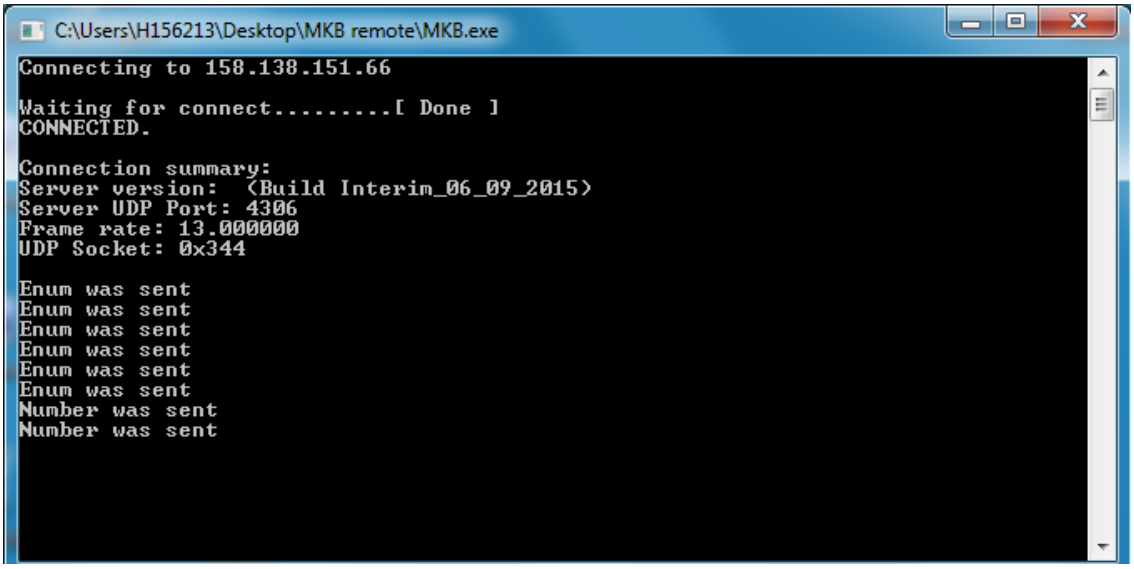
Nad knihovnou tiuiface jsem vystavěl svou uživatelskou vrstvu, která implementuje funkce specifické pro mé využití. Jsou to funkce:

- Pro nastavení připojení – nastavuje IP adresu, port, typ spojení. Ale také vytváří seznam všech uzlů datového stromu, do kterých bude aplikace zasahovat.
- Pro připojení k serveru – otevře kanál mezi klientem a serverem, nastaví vlastnosti komunikace a vyčte verzi serveru.
- Pro odpojení od serveru – uzavře komunikační kanál.
- Pro odeslání dat – jedná se o několik různých funkcí, přičemž každá implementuje odeslání jiného datového typu. Jsou to typy – celočíselné číslo, hodnota z výčtu (enum) či řetězec znaků.
- Pro příjem dat – opět více funkcí pro datové typy číslo a řetězec.

Příkladem odeslání dat budiž odeslání obsahu paměti:

```
SendStack(&aCp, par_inf, message.stack, message.length);
```

Tato ukázka kódu volá funkci pro odeslání textového řetězce. Jejími parametry jsou struktura aCp, která uchovává nastavení serveru, par_inf což je vektor uchovávající informace o uzlech stromu, samotný řetězec a jeho délka.




**Obr. 16 - Okno aplikace MKB remote simulation při spouštění**

Na Obr. 16 vidíte okno aplikace těsně po jejím spuštění. V konzoli se nejprve vypíše průběh připojení k serveru a jeho vlastnosti. Každý řádek poté značí zapsání jedné proměnné do datového stromu serveru – vždy se vypíše datový typ, a zda se podařilo hodnotu zapsat. Po připojení k serveru a zapsání inicializačních hodnot se otevře okno grafické nadstavby a konzole je skryta.

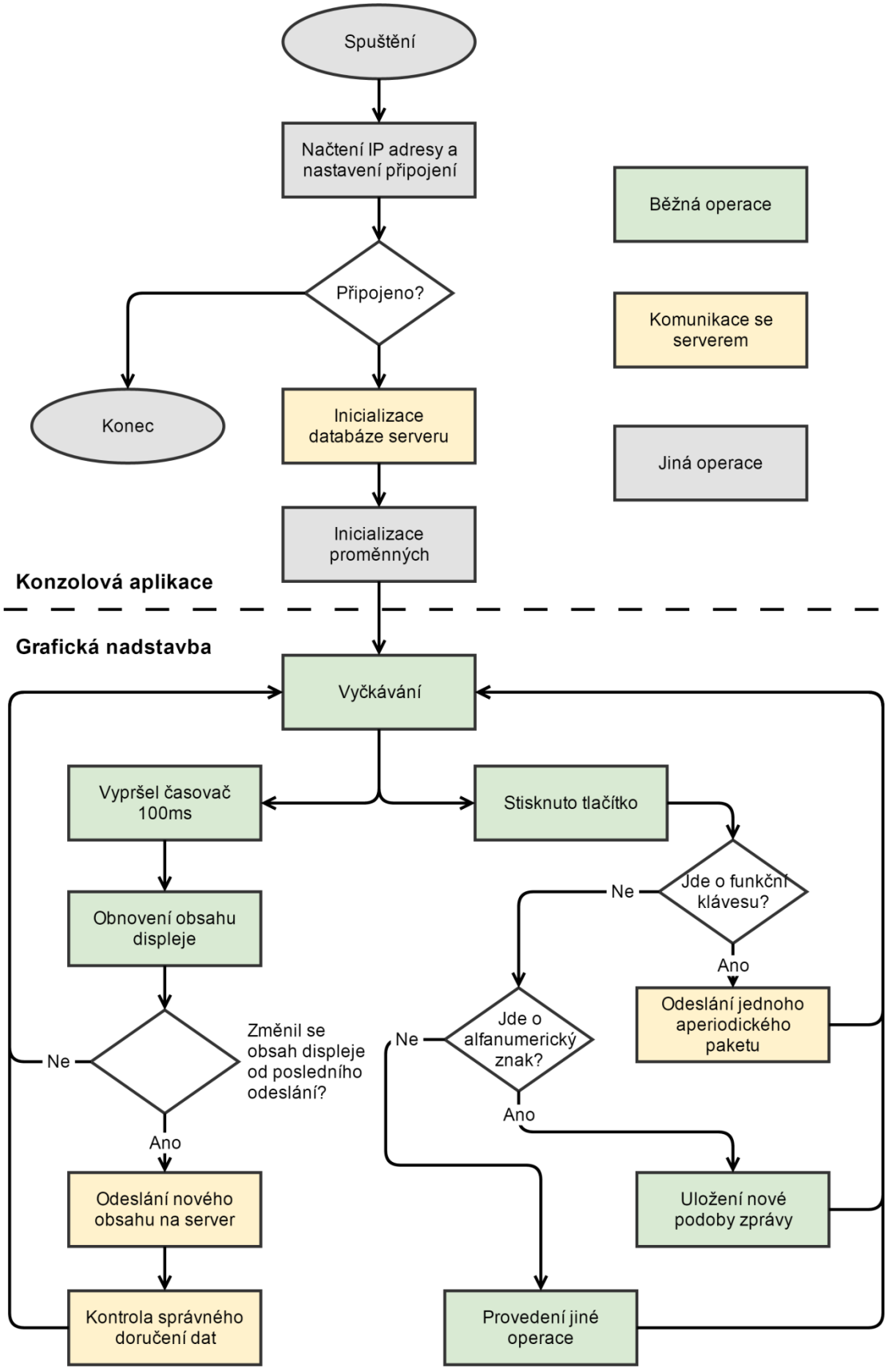
## 6.3.PO SPUŠTĚNÍ

Po zapnutí aplikace nejdříve načte IP adresu z konfiguračního souboru config.txt, který musí být uložený ve stejné složce jako aplikace, a pokusí se o připojení k serveru na dané adrese. Pokud se připojení nezdaří, aplikace o tom informuje a je ukončena. Po připojení je server inicializován – proměnné v datovém stromu jsou nastaveny do takového stavu, aby simulovaly MKB po zapnutí. To znamená, že display je prázdný, přepínač na počátku, čítače vynulované.

Po připojení k serveru se inicializují proměnné v paměti aplikace. Jde především o strukturu datablock_t, která v sobě uchovává stejné údaje jako MKB. Jsou to tedy: řetězec zadaný uživatelem, hodnoty čítačů a polohu přepínače WX radar. Řetězec je inicializován jako prázdný, číselné hodnoty na nulu a WX radar je inicializován do polohy off.

Struktura datablock_t dále obsahuje pomocné proměnné: délku řetězce, pozici kurzoru, a informaci zda se řetězec od posledního odeslání změnil. Jakmile toto všechno proběhne je konzole skryta a objeví se okno na Obr. 19.

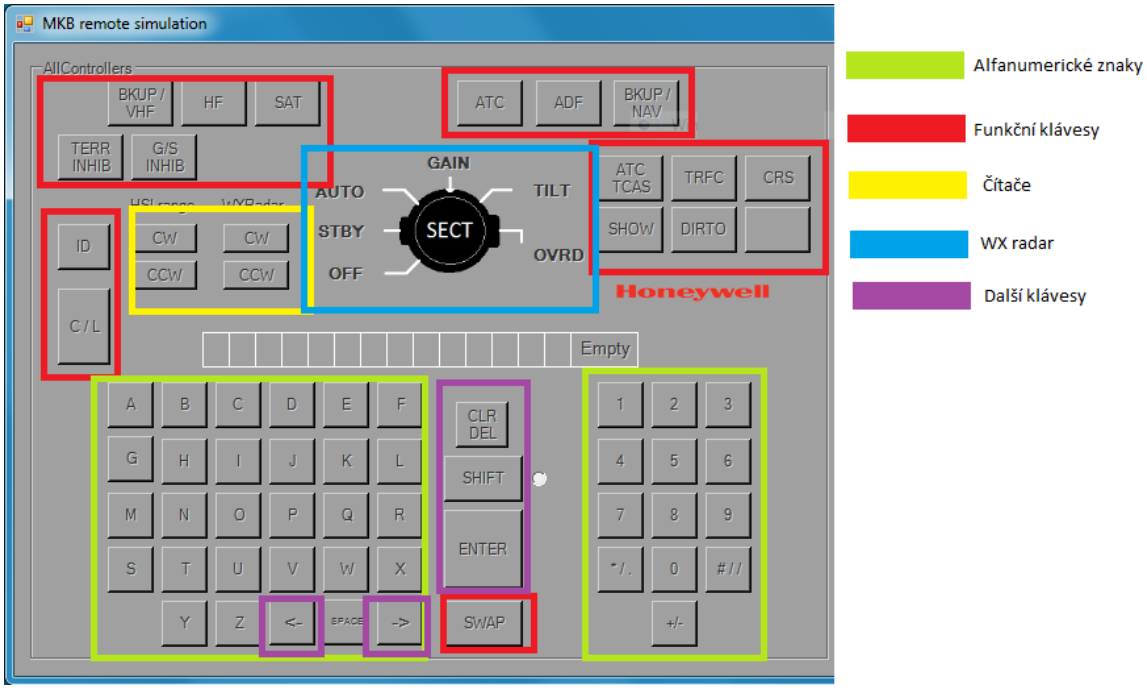
Nyní je klient připojený k serveru a připraven k fungování. Zjednodušený vývojový diagram aplikace vidíte na Obr. 17. Tento nemohl postihnout všechny větve programu, a proto zobrazuje jen ty nejdůležitější. Nepokrývá především chybové stavy a také při stisknutí některých tlačítek se vykonává více operací. Např. uživatel ukončuje program stisknutím klávesy Esc, tato větev není v diagramu zobrazena.




**Obr. 17 - Zjednodušený vývojový diagram aplikace MKB remote simulation**

## 6.4.STISKNUTÍ KLÁVESY

Uživatel má dvě možnosti jak stisknout klávesu – buď myší v okně aplikace, nebo na standardní počítačové klávesnici. Obě možnosti vyvolají stejnou reakci a to v závislosti na typu stisknuté klávesy. Typy kláves jsou rozděleny na Obr. 18.




**Obr. 18 - Rozdělení typů kláves**

Pokud je stisknut alfanumerický znak, je tento přidán do řetězce v paměti klávesnice. Posune se kurzor a změní se délka zprávy, to vše se ukládá ve struktuře datablock_t. Ovšem nový znak není ihned odeslán na server.

Odesílání probíhá jednou za 100ms po vypršení čítače, pokud od posledního vypršení proběhla změna. Zvolil jsem odesílání v definovaných intervalech, protože jinak by se mohlo stát, že uživatel by rychlým zadáváním hodnot mohl přetížit komunikaci.

Pokud je stisknuta funkční klávesa, je její hodnota ihned zapsána na server a odeslána v aperiodickém paketu. Tzn. že je spuštěno odesílání pseudoperiodického paketu na dobu takovou, aby se odeslala zpráva přesně jednou. Tato operace je podrobně vysvětlená v kapitole 5.1.2.

Stisknutím tlačítka ovládajícího čítač se buď přičte (po směru hodinových ručiček) nebo odečte jednotka a hodnota se zapíše do paměti klávesnice. Hodnota se odesílá okamžitě.

Otočení přepínače WX radar zapíše aktuálně zvolenou hodnotu do paměti a okamžitě ji odešle. Pozice OVRD tohoto přepínače není stabilní, ale po 1s se přepínač vrátí do pozice TILT. K tomu slouží čítač.

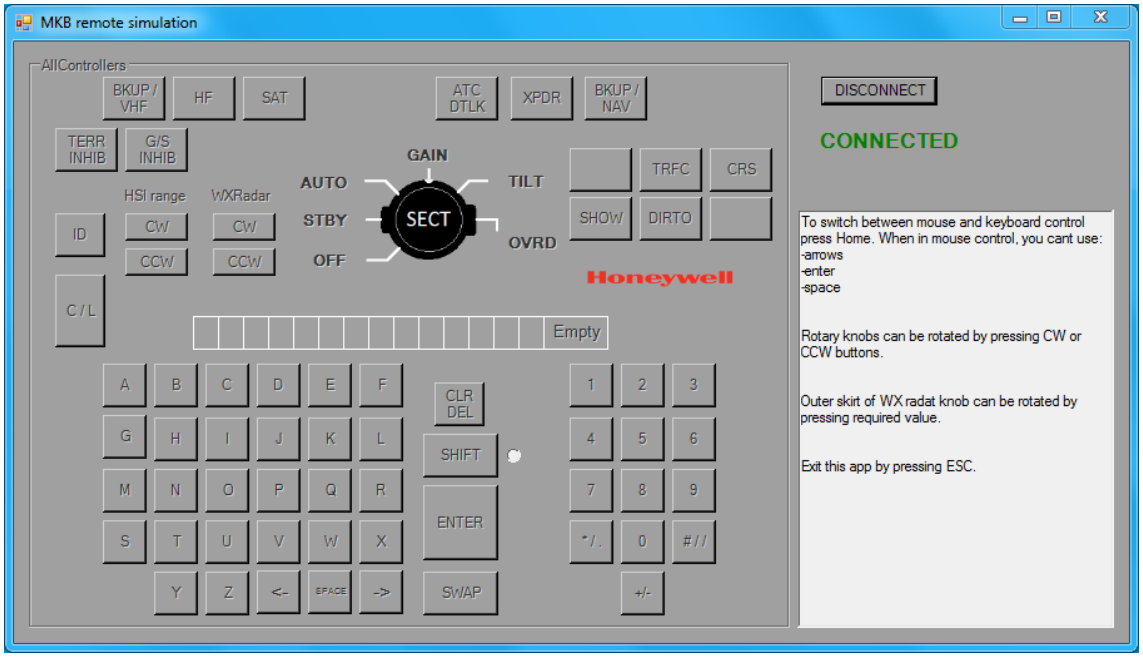
Stisknutím dalších tlačítek se vykonávají různé úkony. Enter pošle informaci o ukončení editace řetězce (funguje na stejném principu jako funkční klávesa) a paměť se vynuluje. Klávesa Shift přepíná mezi spodní a horní funkcí kláves. To se projeví např. u klávesy Delete smaže poslední znak, zatímco horní funkce stejné klávesy (CLR) smaže celý řetězec. Shift zůstává přepnutý i po puštění klávesy, jestli je aktivní indikuje přepínač vedle klávesy Shift v grafickém rozhraní. Šipky posouvají kurzor na displeji a tak umožňují editovat řetězec na kterékoli pozici.

## 6.5. UŽIVATELSKÉ OVLÁDÁNÍ

Aplikace se nemusí instalovat, podmínkou ovšem je mít v počítači Microsoft Visual C++ 2015 redistributable knihovny, které se dají stáhnout z webových stránek Microsoftu. Ve stejné složce, ve které se nachází aplikace je nutné mít textový soubor config.txt, který obsahuje řetězec: „IP: 000.000.000.000“ kde místo řetězce nul zapíše uživatel IP adresu TIU serveru.

Před spuštěním aplikace je nutné mít spuštěný TIU server nakonfigurovaný podle kapitoly 5.1.3 a přepínače na racal multiplexeru přepnuté na simulovaný zdroj dat.

Pokud se po zapnutí nepodaří připojit, aplikace se nezapne. V takovém případě je nutné ověřit IP adresu, a zda server běží a je správně nakonfigurován. Po spuštění se uživateli zobrazí okno aplikace, které je zobrazené na Obr. 19.




**Obr. 19 - Okno aplikace MKB remote simulation**

Nyní má uživatel dvě možnosti jak aplikaci ovládat. Zadáváním znaků na klasické počítačové klávesnici, nebo stisknutím tlačítka v okně aplikace myší. Pro využití funkčních kláves musí uživatel použít okno aplikace, protože počítačová klávesnice neobsahuje funkční klávesy.

Pro umožnění ovládání myší, je nutné stisknout tlačítko Home na počítačové klávesnici. Pokud je stisknuto, tlačítka se stanou aktivními ovšem šipky, mezerník a Enter na počítačové

klávesnici dostanou jinou funkci. Ovládání myší lze libovolně zapínat a vypínat opakovaným stiskem klávesy Home.

Knoflík WX radar uprostřed klávesnice má 3 funkce: vnější otočný lem se otáčí stisknutím některého z nápisů umístěných kolem něj. Vnitřní otočný lem (posunující čítač) se ovládá stisknutím tlačítek CW (clockwise = po směru hodinových ručiček) a CCW (counterclockwise = proti směru). Stisknutí SECT s funkcí funkční klávesy se provede stisknutím samotného obrázku tlačítka.

Stisknutím tlačítka disconnect se aplikace odpojí od serveru, lze ji znovu připojit stiskem tlačítka connect. Toho může uživatel s výhodou využít, pokud potřebuje restartovat TIU server. Pokud by aplikaci neodpojil, spojení by se stalo neplatným a aplikaci by se již nepodařilo připojit. Aplikaci lze vypnout stisknutím ESC nebo křížku.

V pravé části okna se nachází pole se základní nápovědou, která by uživateli měla usnadnit ovládání.

Řetězec uložený v paměti se zobrazuje v poli, které vypadá jako displej na reálné palubní klávesnici KB-600. Aplikace má shodný vzhled i chování displeje, pouze místo kurzoru na hardwaru symbolizovaného obdélníčkem, je v aplikaci zobrazeno podtržítko.

# 7. TESTOVÁNÍ APLIKACE

## 7.1.ZKUŠENOSTI Z UŽÍVÁNÍ

Při testování aplikace jsem byl připojený přes počítačovou síť v rámci jedné budovy, ovšem nacházel jsem se v jiné místnosti. Pro sledování displejů jsem využíval obraz kamery zobrazený přes webové rozhraní v prohlížeči Internet Explorer. Jak vypadá ovládání ze vzdáleného počítače můžete vidět na Obr. 20.

Po spuštění je aplikace připravena k užívání za méně než 4s. Její odezva je plynulá a svižná pro krátké řetězce. Ve srovnání s reálnou palubní klávesnicí má ovládání delší odezvu, ale nejde o výrazný rozdíl. Pro řetězce delší než 10 znaků je z důvodu odesílání velkého množství dat na server již poznat prodleva. Běžně se ovšem zadávají řetězce kratší. Tato prodleva se u reálné klávesnice vůbec neobjevuje. Odesílání příkazů funkčních kláves je spolehlivé, ovšem jejich odezva je pomalejší.

Spojení je stabilní a k serveru se podaří připojit pokaždé. Ani jednou se mi nestalo, že by se znak na server neodeslal nebo byl odeslán špatný znak.


**Obr. 20 - Testování aplikace**

I nezkušení uživatelé se v aplikaci rychle zorientují a jsou schopni ji používat. V pravé části okna se zobrazuje základní nápověda pro práci s aplikací, která vysvětluje některé prvky ovládání, které nemusí být na první pohled jasné. V kombinaci s aplikací pro dálkové ovládání kurzoru (viz následující kapitola) umožňují tyto dva prvky ovládat kompletní integrovaný kokpit Primus Epic.

## 7.2.NAVRŽENÁ VYLEPŠENÍ

Při testování jsem objevil několik příležitostí ke zlepšení. Jednou z možností, jak zrychlit odezvu je snížit periodu pseudoperiodického paketu. Nyní je nastavena na 1s, což je zpoždění, které sice nevadí práci, ale uživatel jej pozná. Navrhuji snížit jeho periodu až pod 100ms.

Drobnou uživatelskou úpravou by mohlo být namapování funkčních kláves na klávesy F počítačové klávesnice. Tato úprava by umožňovala rychlejší ovládání bez nutnosti přepínat do módu ovládání myší.

Dalším vylepšením aplikace je umožnit načítat jména uzlů datového stromu ze souboru config.txt, který nyní uchovává pouze IP adresu serveru. Umožní to pružnější změnu struktury datového stromu a také usnadní využití stávající aplikace i pro jiné typy klávesnic. Úplná změna typu klávesnice (např. přidání dalších kláves nebo změna principu fungování) ovšem bude vyžadovat širší úpravy kódu.

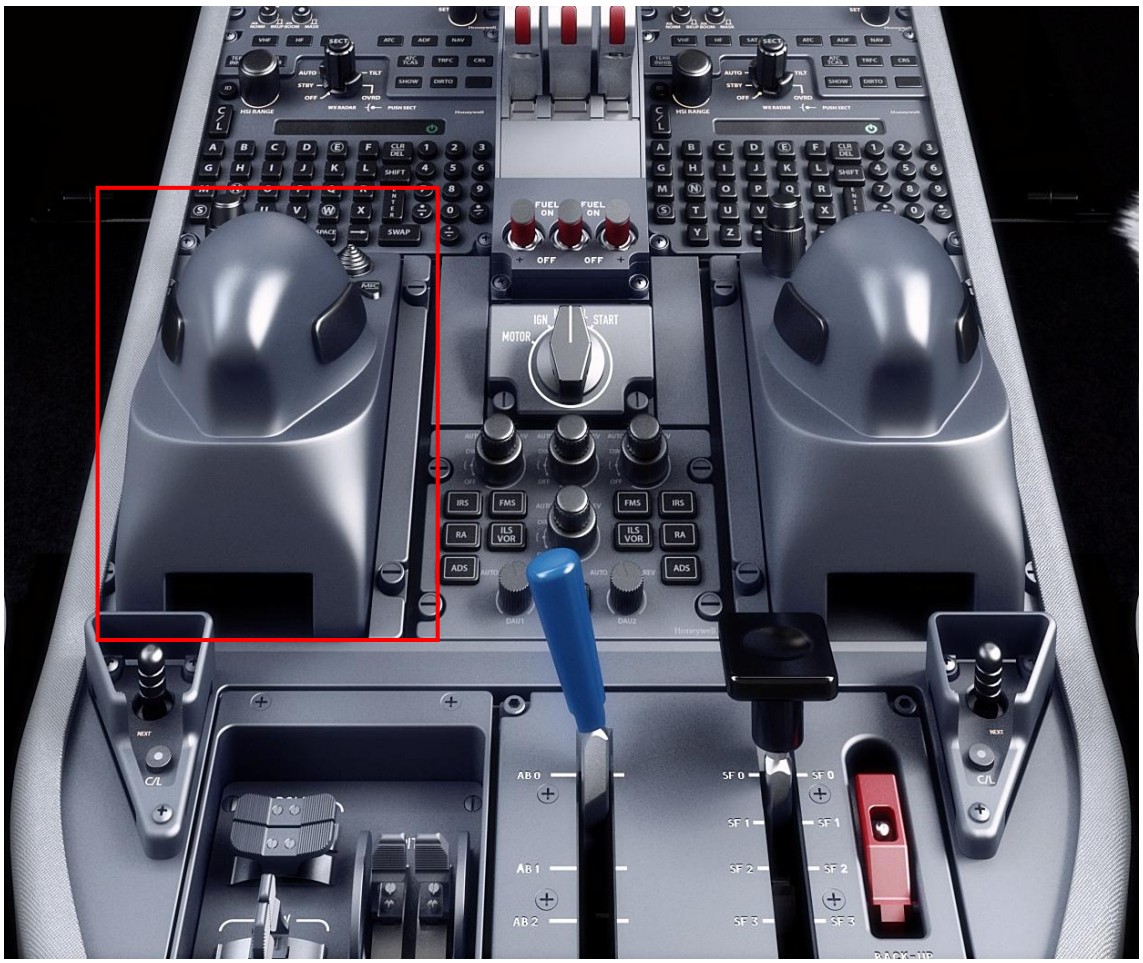
V kombinaci dálkového ovládání CCD a MKB se skrývá problém pokud si uživatel přeje využívat funkční klávesy. Ovládání CCD běží na pozadí a omezuje pohyb myši pouze na oblast svého okna, zatímco aplikace MKB využívá klávesnice. Pokud chce uživatel využít funkční klávesu (potřebuje myš), musí nejprve přepnout okna a pozastavit vzdálené ovládání CCD. Toto řešení je poněkud nepohodlné a proto se v budoucnu počítá s integrací obou aplikací do jedné.

# 8. OVLÁDÁNÍ KURZORU

Pro vzdálené ovládání kurzoru je též nastavený TIU a existuje aplikace podobná mé MKB remote. Práci na tomto ovládání odvedl můj kolega, Filip Černý v rámci své diplomové práce. Automatizaci pohybu kurzoru jsem již zpracoval sám s využitím jeho konfigurace TIU.

## 8.1. ÚVOD K FUNGOVÁNÍ CCD

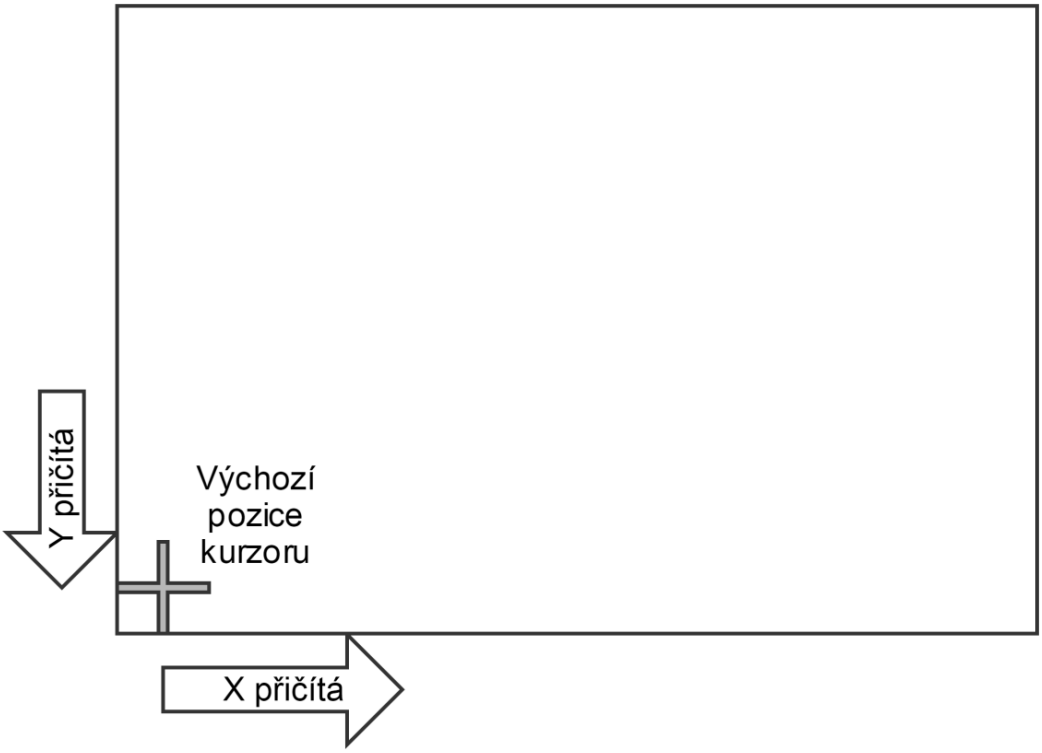
Kurzor pilot ovládá pomocí zařízení zvaného CCD, zobrazeného na Obr. 21. Jde o trackball s opěrkou pro ruku, která zajišťuje stabilitu při turbulencích. Mimo ovládací kuličku a potvrzovací tlačítka obsahuje zařízení také otočný knoflík pro navyšování (resp. snižování) vybraných hodnot, čtyřsměrové tlačítko pro skok kurzoru mezi displeji a tlačítko pro vyvolání menu.




**Obr. 21 - Detail CCD [12]**

Zařízení generuje absolutní a relativní pozici kurzoru (o kolik se kurzor posunul od poslední zprávy) a předává ji spolu s dalšími hodnotami periodicky odesílaným řetězcem přes sériovou linku RS422 do MAU. Obě pozice jsou předávané ve formě 10 bitového čítače, pro X a Y souřadnici zvlášť. Pro nahrazení zařízení softwarem stačí posílat absolutní pozici, protože relativní si z ní vypočítá přímo aplikace, která tyto data využívá.

K posunutí přes celý displej je potřeba několik přetečení čítače, to znamená, že pokud se čítač naplní od 0 do 1023 posune se kurzor jen přes část displeje. Navíc záleží na rychlosti změny v čítači, pokud se hodnoty mění rychleji kurzor se posunuje po větších krocích. Čítač se po dosažení limitní hodnoty přetočí a pokračuje v přičítání, resp. odečítání, od druhé limitní hodnoty. Počáteční pozice kurzoru (po restartu) a směr přičítání jsou zobrazeny na Obr. 22.




**Obr. 22 - Vlastnosti kurzoru**

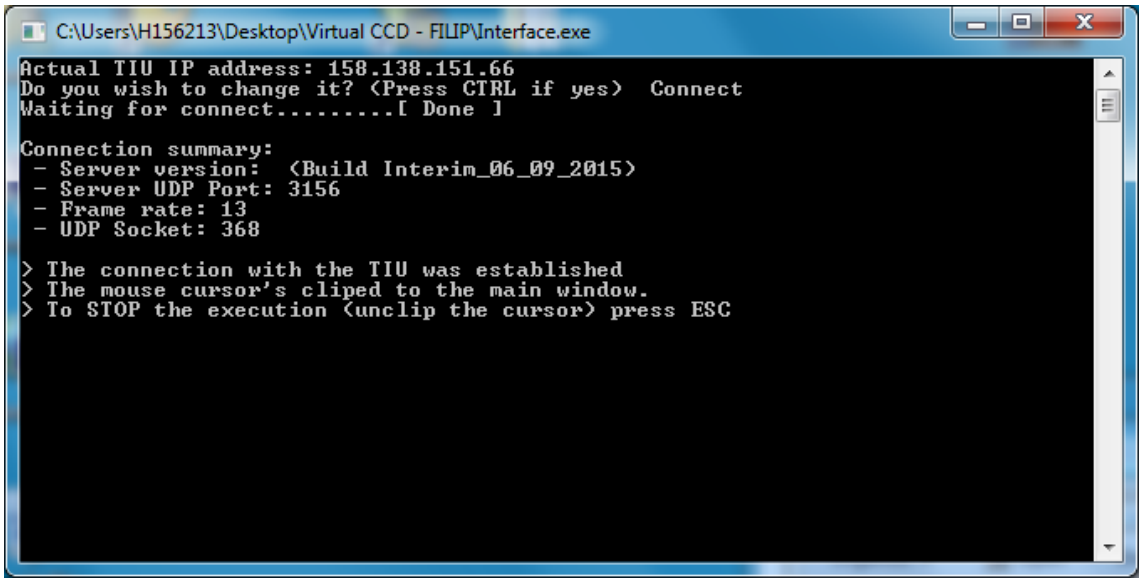
Otočný knoflík na CCD reprezentuje dva 7 bitové čítače. Knoflík má vnější a vnitřní lem, každý s vlastním čítačem. Oba čítače v aplikacích ovládají stejnou hodnotu (tu nad kterou se právě nachází kurzor), ovšem vnější knoflík po větších krocích. Jeho fungování je stejné jako otočných knoflíků na MKB, pouze slouží k ovládání jiných prvků.

Tlačítka menu a čtyřsměrové tlačítko vkládají do sériového řetězce bity, které signalizují jejich stisknutí. Menu vyvolává nabídku pro přepnutí na okno jiné aplikace a čtyřsměrové tlačítko posílá kurzor přímo na výchozí pozici na jiném displeji bez nutnosti točit ovládací kuličkou.

## 8.2.SOFTWAROVÁ SIMULACE CCD

Softwarovou simulaci CCD sloužící k dálkovému ovládání testovací stanice vyvinul Filip Černý v rámci své diplomové práce. V principu jde o stejné řešení jako u MKB. CCD komunikuje s MAU po sériové sběrnici RS422, takže hardware simuluje TIU server. Využíváme jeden server s jedním konfiguračním souborem, který obsahuje oddělené kanály pro CCD a MKB. Struktura TIU je jasná z Obr. 13.

Konfigurace TIU je podobná té pro MKB popsané v kapitole 5.1 s tím rozdílem, že CCD nevysílá žádné aperiodické pakety. Možnosti přístupu do TIU jsou také shodné – buď přes aplikace nebo pomocí TATS skriptů. Aplikaci jsem využil pouze pro testování MKB remote, při automatizování je nutné kurzor ovládat pomocí TATS skriptů.




**Obr. 23 - Okno aplikace Interface, umožňující vzdálené ovládání CCD**

Na Obr. 23 vidíte aplikaci Interface, která umožňuje vzdálené ovládání CCD. Kurzor se pohybuje podle pohybů myši, levé tlačítko myši slouží jako potvrzovací, pravé tlačítko myši slouží místo tlačítka menu a kolečko slouží jako otočný knoflík. Aplikace se připojuje k TIU serveru po zadání IP adresy. TIU server umožňuje připojení více klientů zároveň a proto lze používat aplikace Interface a MKB remote najednou.

## 8.3. PROBLÉMY S POHYBEM KURZORU

Pohyb kurzoru po displejích není vždy přímočarý. Aby aplikace usnadnily uživateli práci, jejich nabídky (tlačítka nebo pole pro hodnoty) přitahují kurzor k sobě.

Z toho vyplývá zásadní komplikace při automatizování kurzoru – pohyb není jednoduše reprodukovatelný. Při jiném rozložení obrazovky (otevřené jiné aplikace) a jiných aktivních položkách se kurzor dostane na odlišné místo, přestože skript dává stejné příkazy. Je to dáno tím, že nezadáváme cílovou souřadnici na displeji, ale musíme emulovat posun kurzoru uživatelem.

Druhou komplikací je rychlost odesílání příkazů. Pokud totiž skript přičítá v serveru k čítačům jinou rychlostí, pohybuje se kurzor v jiných krocích a tlačítka jej přitahují jindy a z jiného místa. Pokud bychom měli dokonale rychlý server, tento problém nenastane. Ovšem v reálné situaci se stává, že serveru jeden z tisíců zápisů dat trvá o něco déle a v tom okamžiku se pohyb rozsynchronizuje a kurzor se nedostane na požadovanou pozici.

Oba výše zmíněné problémy jsem chtěl vyřešit zavedením zpětné vazby z MAU do skriptu. Skript ovšem nemůže přímo přistupovat do paměti karet či na sběrnice MAU a proto je odkázán na data, která MAU odesílá do TIU serveru. Ukázalo se, že pozice kurzoru mezi nimi není a proto nemůžu zpětnou vazbu realizovat.

Ze softwarového hlediska není vyvedení signálu do TIU problém, ovšem z procesního ano. Za letu je nežádoucí jakýkoli mrtvý kód, který se nevyužívá a zapínat tuto část kódu pouze na testovací stanici jde proti myšlence testování letového software.

# 9. AUTOMATIZOVANÉ TESTOVÁNÍ

## 9.1. ÚVOD DO TESTOVÁNÍ

Automatizované testování na stanici umožňují TATS skripty. Ty jsou odvozené z Visual basic skriptu (VBS), doplněním klíčových slov zabezpečujících funkce TIU serveru. Rozšíření umožňuje mimo jiné zápis a čtení do a z datového stromu, ladění, monitorování a synchronizaci s proměnnými, generování průběhů v TIU či vkládání zpoždění.

VBS samotný umožňuje ovládání operačního systému a dalších aplikací běžících vedle TIU serveru. Bohužel TATS neumožňuje využít všechny možnosti VBS a také je optimalizovaný pro Windows XP, a některé jeho funkce nefungují na Windows 7.

Základní myšlenka je taková, že uživatel spustí v aplikaci TATS (vývojové a ladící prostředí) skript sloužící k otestování bezproblémového fungování nové verze softwaru FMS. Různé skripty budou testovat různé scénáře ovšem budou k tomu využívat společných funkcí zajišťující operace s kurzorem nebo klávesnicí. Scénářem se rozumí testová procedura složená z jednotlivých operací a úkonů ověřující vybrané požadavky.

Aby uživatel nemusel být testu přítomen, skript musí umožnit automatické volby z nabídek kurzorem (posunování kurzoru po obrazovce) a zadávání údajů z klávesnice (alfanumerický řetězec, nebo krokové nastavování otáčením knoflíkem). Tyto operace jsou vykonány zavoláním funkcí, které přiblížím v dalších kapitolách.

Pro kontrolu výsledků bude skript zároveň sám snímat obrazovku v okamžicích kontroly. Uživatel si tak po ukončení testu bude moci projít sadu snímků obrazovky označených časem pořízení a zkontroluje, že test proběhl bez problémů. Pokud to bude potřeba, lze také ukládat hodnoty z datového stromu TIU do výstupního souboru.

## 9.2. AUTOMATIZACE MKB

K automatickému používání MKB jsem vytvořil sadu funkcí nazvaných MKB_functions. Ta implementuje funkce pro: zapsání řetězce, stisknutí Enteru, stisknutí funkční klávesy, přepnutí WX přepínače a otáčení s otočným knoflíkem.

Funkce *SendStack(text)* slouží k zapsání řetězce a je základní funkcí. Jejím jediným parametrem je textový řetězec, který se má odeslat. Funkce do proměných stromu zapíše řetězec, což je stejné jako by jej uživatel napsal na reálné palubní klávesnici. Poté musí uživatel potvrdit zadání stiskem klávesy Enter, což řetězec odstraní z displeje klávesnice. K tomu slouží funkce *Enter*.

Funkce *RotateKnob(number_of_dents,knob)* umožňuje přičíst či odečíst od čítačů danou hodnotu a tím simuluje otáčení knoflíků na klávesnici po směru resp. proti směru hodinových ručiček. Prvním parametrem je počet zarážek, o které se má knoflík otočit (celá otáčka činí 16 zarážek) a druhým parametrem je číslo knoflíku, kterým se otáčí (na klávesnici jsou knoflíky dva, očíslované 1 a 2).

Dalšími funkcemi jsou *FunctionKey(key)* a *WxPosition(number_of_position)*. Obě funkce přebírají jeden parametr a tím je klávesa, která má být stisknuta resp. pozice na kterou se má přepínač otočit.

## 9.3. AUTOMATIZACE CCD

Pro automatizaci CCD jsem vytvořil sadu funkcí CCD_functions. Ta implementuje funkce pro: posun kurzoru do výchozí pozice, posunutí kurzoru všemi směry, stisknutí Enteru, stisknutí tlačítka menu a otáčení otočného knoflíku.

Funkce *ToZero(DU)* zajišťuje skok kurzoru do definovaného bodu na počátku zvoleného displeje. Ten se nachází v levém spodním rohu (viz Obr. 22). Při vykonávání skriptu je potřeba se občas vrátit do tohoto bodu a z něj pokračovat, abychom měli jistotu, že se kurzor nachází na požadovaném místě. Skok je také rychlejší než přesun o určitý počet kroků. Jediný parametr funkce udává, na který displej se chceme přesunout.

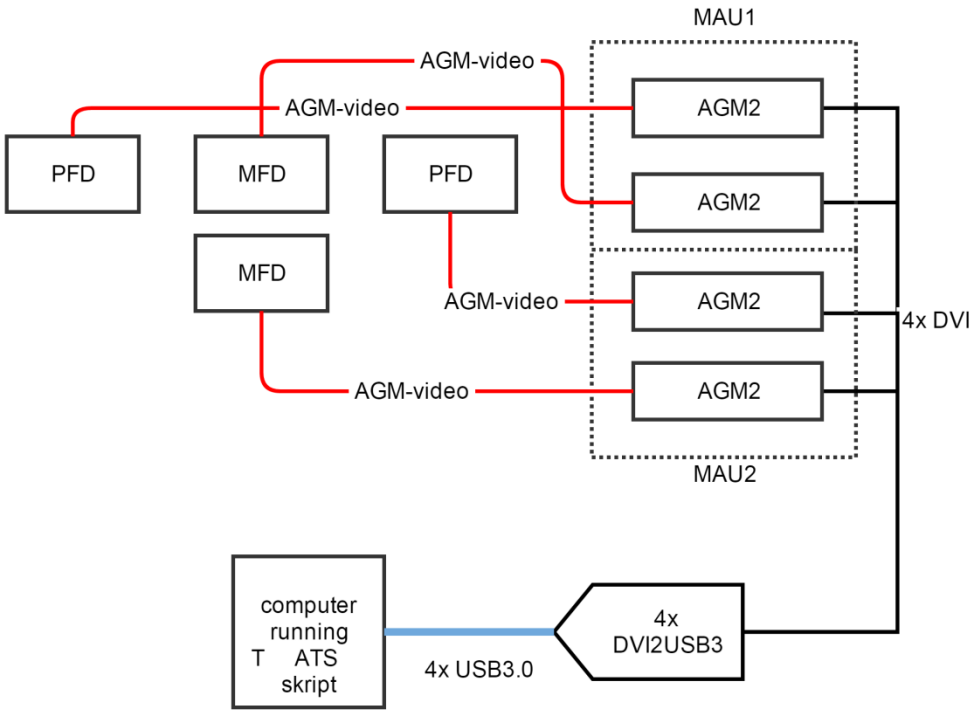
Funkce *ToRight(rough,fine)*, resp. *ToLeft, Upwards* a *Downwards* posunují kurzor ve směru odpovídajícím jejich názvu. Posunují ho o počet kroků, který je specifikovaný v parametrech. Aby funkce umožňovali rychlé i přesné přesouvání kurzoru je toto rozděleno do dvou stupňů – hrubý a jemný pohyb. Nejprve hrubý pohyb posunuje s kurzorem rychleji tím, že přičítá k čítači s větším krokem. Následuje jemný pohyb, který přičítá s malým krokem. Rychlost přičítání je vždy stejná a to nejvyšší možná, omezená pouze možnostmi zápisu na server.

*ClickEnter* a *ClickMenu* jsou jednoduché funkce zajišťující kliknutí na levé tlačítko Enter (kliknutí na pravé není potřeba, protože má stejnou funkci) nebo na tlačítko menu. *Funkce RotateKnobCCD(number_of_dents, knob)* otočí knoflíkem o počet zoubků specifikovaný v parametru. Druhý parametr určuje, jestli se má otočit vnitřním nebo vnějším knoflíkem. Vnější tlačítko má buď stejnou funkci jako vnitřní nebo u některých aplikací zajišťuje hrubější krok.

## 9.4.SNÍMÁNÍ OBRAZOVKY

Součástí avioniky, v kabinetech MAU, jsou také čtyři grafické adaptéry AGM2 (advanced grafic module druhé generace), každý z nich generuje data pro jeden display. AGM2 má dva paralelní výstupy, z nichž jeden je typu DVI a není běžně využitý – můžu jej tedy využít ke sledování obrazovky. DVI výstupu se nevyužívá u dálkového ovládání, protože je uživatelsky hůře přístupný a méně plynulý než využívaná kamera.

Ke snímání obrazovky využívám převodník Epiphan DVI2USB3.0, převádějící signál z DVI na USB 3.0. Převodník umožňuje snímat obraz o rozlišení až 1920x1200 s frekvencí 60Hz. Výrobce dodává k zařízení uživatelskou aplikaci umožňující vytvářet snímky obrazovky i nahrávat videa. [9]




**Obr. 24 - Zapojení video grabberů**

Funkce *SaveScreen(path, name)* uloží snímek požadovaného displeje pod zvoleným jménem do složky na kterou odkazuje cesta v parametru předaném při zavolání funkce. Funkce spustí aplikaci DVI2USB3, která umožňuje udělat snímek vstupního signálu z převodníku Epiphan. Snímek pojmenuje časovou značkou pořízení, nebo jménem předaným v druhém parametru.

Pokud se snímá obrazovka periodicky zatěžuje opakované spouštění Epiphan neúměrně počítač. Proto jsem se rozhodl přidat funkci *StartEpiphan*, která aplikaci spustí a pozastaví snímání. Potom lze používat funkci *MakeScreenshot(path,name)*, která funguje stejně jako SaveScreenshot s tím rozdílem, že při každém snímání nepouští a nezavírá znovu program, ale pouze spustí a opět pozastaví nahrávání.

V současné době funguje převodník DVI2USB3 pouze na jednom displeji, protože počítač nemá dostatek USB portů. Používám sledování spodního displeje MFD což pro ukázkové scénáře stačí, pro sledování jiného displeje je potřeba připojit ke grabberu jiný vstup. Po doplnění dalších zařízení bude nutné doplnit funkci pro přepínání vstupu mezi různými grabbery v aplikaci DVI2USB3.

V testovacím skriptu bude celý scénář definován včetně okamžiků, kdy se má udělat snímek obrazovky. Stačí tedy zavolat funkci SaveScreen a říci, snímek kterého displeje chceme vytvořit a kam jej uložit.

## 9.5. POROVNÁVÁNÍ OBRAZU

Až v průběhu testování automatizovaných skriptů vznikl požadavek na automatickou detekci některých událostí zobrazených na MFD. Mezi ně patří např. detekování zaplněné paměti či kontrola úplnosti uložených položek. Protože není k dispozici zpětná vazba z MAU, musím vycházet ze snímků obrazovky. Rozhodl jsem se k tomu využít software třetí strany, který umožňuje porovnání dvou obrázků.

Jde o ImageMagick® software, který je k dispozici jako freeware a umožňuje práci s mnoha grafickými formáty. Jeho možnosti jsou rozsáhlé, já využiji především schopnost oříznout a porovnat dva obrázky. K tomu slouží příkazy *convert –crop* a *compare*, které lze zavolat z příkazového řádku. Výstup z příkazového řádku je přesměrován do textového souboru, ze kterého se čte výsledek.

Porovnání obrázku funguje na principu hledání střední barevné vzdálenosti. To je metrika, která porovnává barevnost obou obrázků a vrací průměrnou vzdálenost mezi barvami. Funkce compare (součástí software ImageMagick) tedy vrací nulu pro stejné obrázky a čím rozdílnější mají obrázky barevnost, tím větší číslo vrátí. Tuto metriku jsem zvolil, protože v případě změny zobrazených hodnot na displeji se barevnost výrazně nemění, ale v případě chybové hlášky je barevnost velmi změněna.

Snímek obrazovky tedy bude nejprve uložen funkcí SaveScreen a poté oříznut a porovnán se vzorem. K tomu slouží mnou vytvořená funkce v TATS, která sama zavolá výše uvedené funkce třetí strany v příkazovém řádku.

K porovnání obrázků slouží TATS funkce *Compare(path,picture,pattern,crop)*, jejíž parametry jsou cesta k obrázkům, vstupní obrázek, vzorový obrázek a řetězec určující jak má být vstupní obrázek oříznut. Pokud je řetězec crop roven nule, ořezávání neprobíhá. Výstup této funkce je buď řetězec „identical“, který značí, že obrázky jsou shodné, nebo návratová hodnota funkce compare, která udává jak moc jsou obrázky rozdílné.

## 9.6. TVORBA AUTOMATICKÉHO SCÉNÁŘE

Uživatel scénář skládá z jednotlivých funkcí, jejich seznam se stručným popisem najdete v Tab. 5. TATS skripty neumožňují vytvoření knihoven, kvůli lepší orientaci jsem přesto funkce rozdělil do souborů, podle jejich využití. Každý scénář si musí funkce z těchto souborů zahrnout do kódu aby je mohl využívat. Soubory shromážděné v projektu Automation_project.tsp jsou tyto:

- MKB_functions – funkce spojené s klávesnicí
- CCD_function – funkce spojené s pohybem kurzoru
- Screen_capture – funkce spojené se záznamem a porovnáním obrazu

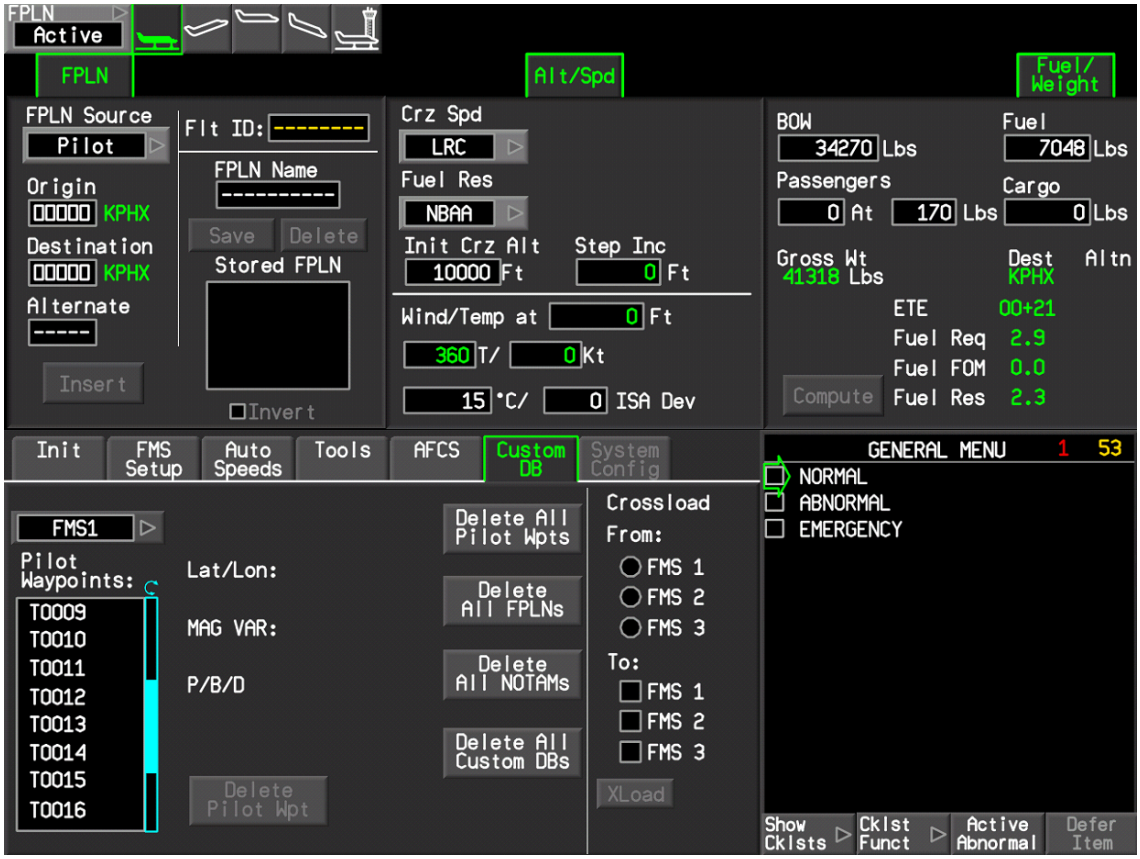
**Tab. 5 - Tabulka funkcí použitelných při tvorbě scénářů**

| Funkce | Popis | Soubor |
|---|---|---|
| SetPath(default_path) | Vyzve uživatele k nastavení cesty | Screen_capture |
| SaveScreen(path, name) | Uloží snímek aktuální obrazovky se jménem „name“ do složky „path“ | Screen_capture |
| Compare(path, picture, pattern, crop) | Porovná obrázek se vzorem, pokud crop není „0“ ořízne předtím obrázek | Screen_capture |
| StartEpiphan( ) | Zapne program Epiphan | Screen_capture |
| MakeScreenshot(path, name) | Vytvoří snímek obrazovky s využitím zapnutého soft. Epiphan | Screen_capture |
| InitializeMKB( ) | Připraví skript k práci s klávesnicí | MKB_functions |
| SendStack (text) | Zapíše na klávesnici text v parametru | MKB_functions |
| Enter( ) | Stiskne Enter | MKB_functions |
| FunctionKey(key) | Stiskne funkční klávesu v parametru | MKB_functions |
| WxPosition(number_of_position) | Natočí tlačítko WX radaru do pozice předané parametrem | MKB_functions |
| RotateKnob(number_of_dents, knob) | Otočí knoflík o daný počet zoubků | MKB_functions |
| InitializeCCD( ) | Připraví skript k práci s CCD | CCD_functions |
| ToZero(DU) | Přesune kurzor do nuly displeje „DU“ | CCD_functions |
| ToRight(rought,fine) | Posune kurzor o daný počet hrubých a jemných kroků směrem podle názvu zvolené funkce | CCD_functions |
| ToLeft(rought,fine) | | CCD_functions |
| Upwards(rought,fine) | | CCD_functions |
| Downwards(rought,fine) | | CCD_functions |
| ClickEnter( ) | Stiskne potvrzovací tlačítko na CCD | CCD_functions |
| ClickMenu( ) | Stiskne tlačítko menu na CCD | CCD_functions |
| RotateKnobCCD (number_of_dents,knob) | Otočí knoflík o daný počet zoubků | CCD_functions |

Aby scénář proběhl správně, je před jeho spuštěním potřeba nastavit displeje vždy do stejné konfigurace a nepoužívaný kurzor přesunout na nepoužívaný displej (pokud automatizuji pilotův CCD a MKB, je to kopilotův PFD). To proto aby měl všechny nabídky na stejném místě.

Při používání kurzoru je potřeba vždy vycházet z definované a známé pozice. Touto jsem zvolil levý spodní roh displeje, který má souřadnice [0,0]. Z této pozice je možné kurzor směrovat k jednotlivým položkám podle uživatelem definovaného počtu kroků. Funkce pro posouvání kurzoru mají hrubý a jemný krok a tak umožňují přesné nastavení do požadované pozice.

Bohužel kvůli výše uvedeným problémům s přitahováním kurzoru různými nabídkami jsem nenašel způsob jak spočítat dopředu počet kroků k přesunutí mezi dvěma body. Každý posun se musí empiricky určit při vytváření nového scénáře a záleží tak na zkušenostech uživatele jak rychle je schopný scénář sestavit.




**Obr. 25 - Počáteční rozložení spodního MFD pro práci s databázemi FMS**

## 9.7. ZÁZNAM OVLÁDÁNÍ UŽIVATELEM

Druhou možností jak provést automatizovaný scénář je neskládat jej z jednotlivých kroků, ale naučit se jej od člověka – uživatele. Nejjednodušší možností jak tohoto dosáhnout je nahrát záznam databází TIU serveru během vykonávání operace uživatelem a následně jej přehrát.

Vyvinul jsem skript, který dokáže nahrát vývoj obsahu databází v čase a zpětně jej přehrát. Ovšem výsledný pohyb kurzoru nereprodukoval přesně ten původní. Domnívám se, že to je způsobeno neschopností zapisovat do databází TIU serveru s obnovovací frekvencí serveru.

Našel jsem ovšem alternativní řešení, jak vytvořit záznam. Klient TBOX umožňuje sledování, přepisování a také nahrávání proměnných v databázích TIU serveru. Vše, co je potřeba udělat, je otevřít si proměnné, které chci sledovat. Seznam těchto proměnných lze uložit do souboru s příponou .tbx a z něj je také načíst.

Pro vytvoření záznamu scénáře tedy stačí v programu TBOX otevřít .tbx soubor a stisknout tlačítko pro záznam. Pokud uživatel používá virtuální ovladače stanice (MKB remote a Interface aplikace), všechny změny proměnných jsou zaznamenány a uloženy do souboru s příponou .tbr. Pro zpětné přehrání platí jediná podmínka – rozložení aplikací na displejích před spuštěním záznamu musí být shodné, jinak se kurzor nebude pohybovat shodně se vzorem. K přehrávání záznamu disponuje TBOX nástrojem data player.

Nevýhodou nahrávání scénářů pomocí programu TBOX je neschopnost kombinovat je s TATS skripty a dát je tak do smyčky se změnou některých zadávaných hodnot v každém opakování. Právě takové scénáře jsou časově náročné a jejich automatizace ušetří uživateli nejvíce času.

Nahrané scénáře také nijak nezrychlí práci – z principu se přehrají stejně rychle, jako je uživatel nahrál. Jsou tudíž vhodné například na prezentace funkcí kokpitu nebo k výukovým účelům. Je například možné, aby nezkušený uživatel spustil záznam, který mu přivede systém do požadovaného stavu bez znalosti ovládání. Pro ostatní využití je vhodnější složit scénář z připravených TATS příkazů.

# 10. PRAKTICKÉ VYUŽITÍ SKRIPTŮ

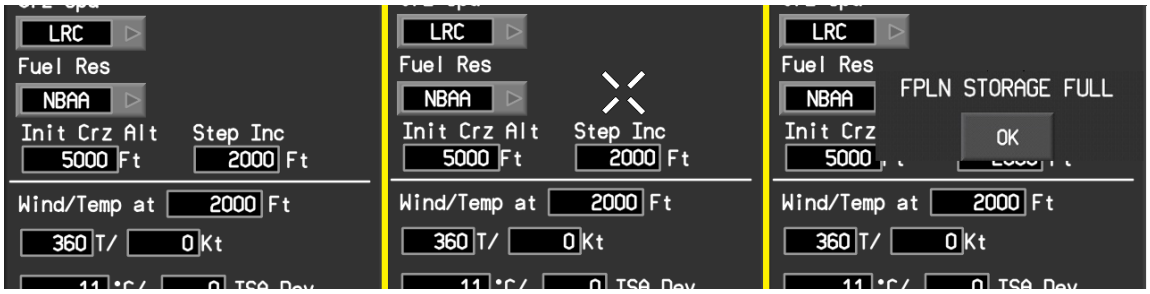
## 10.1. SNÍMÁNÍ A POROVNÁVÁNÍ OBRAZU

Při snímání obrazovky jsem narazil na problémy s kapacitou paměti a výkonem počítače. To omezovalo TATS klienta, který nestíhal odesílat data na server. Nakonec jsem ale snížil hardwarovou zátěž od snímání obrazovky. Především aplikace zůstává stále otevřená, pouze se spouští a zastavuje snímání. Aplikace také nesnímá celou oblast displeje, ale pouze požadovaný výřez. Tyto úpravy jsou dostatečné pro zajištění nepřerušovaného a bezchybného vykonávání skriptu.

Snímání snímků je tedy plynulé, snímky jsou vždy bez potíží uloženy do složky zvolené uživatelem. Při běhu skriptu je potřeba zajistit, aby se počítač na kterém běží skript nepoužíval, protože posílá příkazy programům stejnou cestou jako uživatel. Pokud by někdo přepnul okno, skript by neproběhl podle předpokladů.

Po sejmutí snímku obrazovky může být tento porovnán se vzorem. I porovnání probíhá bez potíží a vrací vždy správnou hodnotu. Nastavený práh umožňuje při porovnání ignorovat drobné změny obrazu, jako je změna jednoho čísla či kurzor ve výřezu. Přesto není vhodné používat avioniku v době kdy na ní probíhá test.

Detekce chybových hlášek v obraze funguje bez chyb a funkce vrací upozornění, že se na obrazovce něco změnilo. Ukázku porovnání vidíte na Obr. 26. Levý obrázek ukazuje vzor se kterým jsou ostatní porovnávány. Prostřední obrázek je detekován jako shodný, třetí je detekován jako rozdílný a tudíž se předpokládá, že je na něm chybová hláška – což je správný výsledek.




**Obr. 26 - Porovnání snímků displeje**

## 10.2. TESTOVÁNÍ DATABÁZÍ

FMS integruje několik uživatelských databází, které mají specifikovanou minimální kapacitu. Mezi databáze patří databáze letových bodů a databáze letových plánů. Podobně je specifikován též minimální počet letových bodů, které se dají vložit do jednoho letového plánu.

Testy databází jsem si vybral jako vhodné k prezentování automatického vyhodnocení výsledků. Aby se kapacita těchto databází ověřila, musí uživatel vytvořit nejméně tolik položek, jaká je kapacita databáze. To jsou stovky položek, jejichž vytvoření je jednotvárné a časově náročné. Výstupem je potom binární odpověď – databáze má / nemá dostatečnou kapacitu.

Právě kvůli neustále se opakujícímu postupu vytvoření jedné položky je toto ideální test k automatizování. Skript stačí naprogramovat pro jedno opakování, které se uzavře do smyčky a měnit se bude pouze název ukládané položky – např. s rostoucím pořadovým číslem. Před spuštěním testu se skript zeptá uživatele jaký test a jakým způsobem chce provést a také ho instruuje ke správnému nastavení stanice a počítače. Poté skript sám smaže databáze, nechá uživatele ověřit že je vše připraveno a spustí se plně automatická část zadávající položky do databáze.

První možností vyhodnocení je nechat skript vložit tolik položek kolik je kapacita a po jeho skončení uživatel ověří, že jsou všechny vložené. Je potřeba zkontrolovat, že jsou uložené všechny body, nebo že se některý neuložil dvakrát apod.

Komplikovanější variantou je po každém opakování přidat kontrolu naplněnosti paměti porovnáním obrazu. Tato varianta však již může test částečně sama vyhodnotit, jejím výstupem je počet dostupných paměťových buněk. Nedokáže ovšem ověřit, že se uložily správně všechny prvky, jejich správné pojmenování atd.

Při spuštění této varianty si skript uloží aktuální podobu obrazovky v místě, kde se objevuje hláška o plné paměti. Vždy po zadání nového prvku se udělá snímek obrazovky a porovná se s původním. Pokud jsou snímky rozdílné, znamená to že se pravděpodobně objevila chybová hláška, test se zastaví a skript oznámí uživateli jaká je kapacita paměti.

Celkem jsem vytvořil 3 testy tohoto typu:

- Test kapacity databáze letových plánů
- Test kapacity databáze letových bodů
- Test kapacity letových bodů v jednom letového plánu

V Tab. 6 můžete vidět, kolik minimálně času zabere test uživateli a za jaký čas je schopný stejný test provést počítač. Navíc je potřeba zohlednit, že uživatel automatický test pouze spustí a vyhodnotí, nemusí mu být přítomný po celou dobu. Nejlepších výsledků se dosahuje u zadávání letových bodů, protože v tomto testu musí uživatel ručně zadávat velké množství dat na palubní klávesnici. Automatické testování je v takovém případě rychlejší o 69% oproti ručnímu.

**Tab. 6 - Doba trvání testů**

| Test | Délka trvání testu v minutách | | |
|---|---|---|---|
| | **Ruční test (minimálně)** | **Automatický test (přibližně)** | **Rozdíl** |
| Letové plány | 29 | 23 | 6 (-21%) |
| Letové body | 268 | 85 | 184 (-69%) |
| Letové body v jednom plánu | 29 | 17 | 12 (-41%) |

Testy databází jsme ve spolupráci s dalšími zaměstnanci firmy Honeywell využili k testování nové verze softwaru FMS. Ukázalo se, že kapacita letových plánu není dostatečná – databáze má mít kapacitu 255 letových plánu, ale již po 102 uložených FMS hlásí zaplněnou paměť.

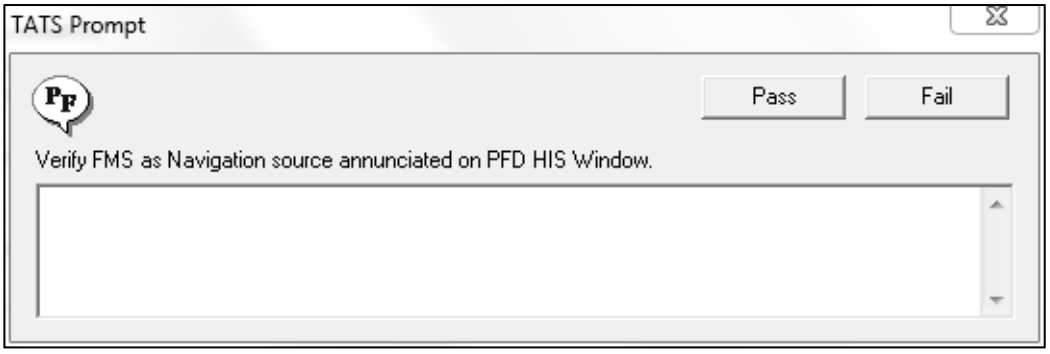
Ještě horší problém byl odhalen u databáze letových bodů. Ta má mít kapacitu 1000 bodů, ale po vložení 45 bodů se začaly objevovat potíže s jejich seznamem. Seznam nezobrazoval všechny body, nezobrazoval informace o některých bodech, případně neumožnil projít v seznamu níže a sám se vracel na jeho začátek. Největší problém nastal když bylo vloženo více než 100 letových bodů. Okno s jejich seznamem přestalo fungovat a neustále se restartovalo, dokud uživatel nesmazal celou dotabázi letových bodů.

Toto okno slouží k nastavení avioniky či FMS a například umožňuje přepínat zdroj informací o pozici letounu. Okno je za letu pro piloty velmi důležité a jeho ztráta má vliv na bezpečnost letu. Toto je zajímavý příklad toho, jaké chyby se dají odhalit v průběhu vývoje softwaru a ukazuje na to, jak je důležité je odstranit před distribucí.

## 10.3. TESTOVACÍ SCÉNÁŘE

Všechny testy FMS se řídí procedurami, které uživatele instruují, jak postupovat a stanovují indikátory podle kterých uživatel rozhodne, jestli jsou splněny požadavky nebo ne. Rozhodli jsme se tyto procedury přepracovat do formy skriptu a očekáváme zrychlení testování. Skript bude využívat simulované ovládací prvky pro zadávání dat a nastavení displejů tak, aby uživatel pouze kontroloval jejich výsledky a nemusel zasahovat do ovládání. Zde prozatím není cílem automatické vyhodnocení.

Pokud je skript spuštěný, na displeji počítače se zobrazují výzvy pro uživatele, kde má zkontrolovat jaký parametr. Výzva zároveň obsahuje pole, kam uživatel může zapsat komentář a tlačítka prošel / neprošel (pass / fail) jako to vidíte na Obr. 27. Výstupem skriptu je potom textový dokument, ve kterém se shromažďují výsledky všech kroků a komentáře k nim. Ukázka výstupního dokumentu je vidět v příloze 3. Každý řádek výstupního dokumentu obsahuje číslo testu, větu která se ověřuje (indikátor), výsledek testu (prošel / neprošel) a komentář.




**Obr. 27 - Výzva pro uživatele**

Aby byl test flexibilní, přidal jsem do něj možnost začít od libovolného kroku – pokud tedy chce uživatel zopakovat pouze jeden krok, nemusí procházet celý test od začátku. Výsledky testu se ukládají průběžně, není problém test vypnout za běhu. K vypnutí stačí místo výsledku napsat do komentáře „exit“ a stisknout pass či fail.

Podle jedné z procedur jsem připravil test FMS power UP, Database & Redundancy, který je časově nejnáročnější z testů prováděných na stanici v Brně. Očekáváme, že na něm se nejvýrazněji projeví úspora času.

**Tab. 7 – Část instrukcí k FMS power UP, Database & Redundancy testu**

| ID testu | Postup testu | Očekávaný výsledek | Prošel / neprošel | Komentář |
|---|---|---|---|---|
| 1 | Zapněte stanici. | Ověřte že „Current position“ je „Not initialized“ v okně avionika. | | |
| 2 | | Ověřte, že zdroj navigace oznamovaný na PFD je FMS. | | |
| 3 | V okně senzory, záložce navigace zvolte FMS. | Ověřte, že standartní zvolený mód pro všechny FMS je synchronní. | | |

V Tab. 7 vidíte zápis instrukcí k testu, pro manuální provedení. Test má celkem 46 kroků, zde vidíte pouze tři první. Tyto instrukce byly převedeny do formy skriptu, část kódu ve kterém jsou kroky nadefinovány vidíte na obrázku Obr. 28. Instrukce jsou uchovávány ve formě matice o n řádcích (n je počet kroků testu) a 3 sloupcích. Každý sloupec má jiný účel a jak je z obrázku vidět, ne každý krok využívá všechny sloupce.

Postup testu (třetí sloupec matice) určuje, jestli je definována procedura s instrukcemi. Ta nastavuje potřebné parametry, nebo uživateli přepne zobrazení na displeji na potřebné aplikace. Proceduru je nutné pro každý krok zvlášť složit z příkazů skriptu používaných k automatizaci. Každý krok testu má buď svou proceduru nebo nemá žádnou. Po provedení procedury se zobrazí výzva (viz Obr. 27) pro uživatele s větou ze sloupce očekávaný výsledek (uloženo v prvním sloupci matice) a skript uloží zpětnou vazbu od uživatele – odpověď prošel / neprošel a komentář. Navíc mezi postupem testu a výzvou ke kontrole se může zobrazit dialogové okno vyzívající uživatele k činnosti nebo upozorňující na nějaký děj. Tento text je uložen ve druhém sloupci matice. Ověřovací hláška je povinná pro každý krok testu, zbylé dva sloupce jsou nepovinné a lze je v daném kroku vynechat.

```
ReDim s(46,3)
'Definition of test procedure
's(x,0)= "string"    string which is displayed when user is prompted to check whether test passed or failed
's(x,1)= "string2"   string2 is displayed in message box before pass/fail dialog
's(x,2)= bool        If 1, there exists sub called PreTstx (x is test ID) which is called prior to any
'                    dialog apears. If 0, there is no sub.

s(0,0)=""  'skip first cell, just for clarity - number of cell is equal to number of test now
'_____
s(1,0)="Verify that current position is *Not Initialized* in Avionics window Init Tab (lower MFD)."
s(1,1)="Power on the bench and verify, that all custom databases are empty."
s(1,2)=0
'_____
s(2,0)="Verify FMS as Navigation source annunciated on PFD HIS Window."
s(2,1)=""
s(2,2)=0
'_____
s(3,0)="Verify on left PDU that the default selected mode for all FMS is Synchronous."
s(3,1)=""
s(3,2)=1
'_____
s(4,0)="Verify on PDU different FMS is driving each PDU. (By default FMS1 pilot side and FMS2 co-pilot side)
s(4,1)=""
s(4,2)=0
'_____
s(5,0)="Verify on MDU FMS data is driven by pilot side FMS (Default FMS1)."& vbNewLine & "Note:This can be v
s(5,1)=""
s(5,2)=0
'_____
s(6,0)="Make sure Active Mode in all three FMSs is Synchronous - In triplex mode all data on FMS will be Syn
s(6,1)=""
s(6,2)=0
'_____
s(7,0)="Verify FMS driving MDU has been changed accordingly."
s(7,1)=""
s(7,2)=1
```

**Obr. 28 - Definice scénáře v kódu skriptu**

Zásadním nedostatkem při automatizování scénářů je, že v některých okamžicích je potřeba využít také další ovládací prvky mimo CCD a MKB. Tyto není možné ovládat s pomocí TIU serveru a tedy automatizovaně za použití TATS. Pokud se Honeywell rozhodne využívat automatizované testy v budoucnu v běžné praxi, bude tento problém vyřešen změnou zapojení testovací stanice a rozšířením konfigurace TIU serveru. Prozatím v takovém případě instruuje skript uživatele k provedení operace ručně. V okamžiku, kdy je potřeba stisknout nějaké tlačítko, zobrazí skript výzvu ke stisknutí tlačítka a následné potvrzení tlačítkem ok.

## 10.4. ZHODNOCENÍ FUNKČNOSTI SKRIPTŮ

U každého nového příkazu jsem ihned ověřoval, zda-li splňuje mé požadavky na něj. Požadavky jsou především bezchybné vykonání operace, vysoká opakovatelnost a univerzální použití. Díky tomu jsem se mohl při vytváření testů spolehnout na plně funkční knihovny příkazů k ovládání MKB, CCD a snímání obrazu. V této kapitole tedy hodnotím pouze výkony samotných testů.

Pro automatizování stále se opakujících příkazů se skripty ukázaly jako velmi vhodné. Testování databází je bezproblémové a přineslo výsledky ještě během vytváření a ladění scénářů. Při zadávání nových prvků v opakující se smyčce jsem narazil na jedinou potíž, kterou bylo příliš rychlé zadávání. To jsem vyřešil přidáním zpoždění mezi jednotlivé příkazy, které sice vykonávání testu zpomalilo, ale odstranilo chyby.

S pomocí testů databází jsem odhalil nedostatečnou kapacitu databází, která v jednom případě neměla ani polovinu požadované kapacity a dokonce po překročení určitého počtu bodů všechny body smazala. Také jsem zjistil, že grafické rozhraní FMS není připravené na plné seznamy letových bodů a objevují se chyby v zobrazování, nelze procházet seznamy atd.

Při testování scénářů se neobjevily žádné problémy. Po restartu stanice si uživatel zvolí test, kterým chce začít a ten se okamžitě začne vykonávat. Uživatel pouze volí, jestli je požadavek splněn nebo ne a případně zapisuje komentáře.

Během testů se ukázalo, že pokud má uživatel sledovat změnu hodnoty po stisknutí tlačítka, je pro něj příjemnější toto tlačítko stisknout manuálně. Jednak ví, kdy očekávat změnu a také má dostatek času k přesunutí pohledu na dané místo. Není tedy vhodné za každé situace trvat na ovládání pomocí příkazů skriptu.

Skript může být v některých okamžicích pomalejší než ruční ovládání, ale celkový průběh je rychlejší. Nejvyšší úspora je podle předpokladů u zadávání řetězců znaků na klávesnici. Také v automatické verzi naprosto chybí rozmýšlení uživatele, jaký je další krok a zapisování výsledků jednotlivých kroků. Konkrétní data o kolik je procedura rychlejší nemám, protože proceduru jsem nikdy netestoval v kompletním rozsahu se všemi náležitostmi. Můžu pouze odhadnout, že test bude přibližně o 50% rychlejší.

## 10.5. VÝHODY A NEVÝHODY TESTOVÁNÍ SE SKRIPTY

Ovládání integrovaného kokpitu pomocí skriptu je rychlejší. Posun kurzoru je přibližně shodný s možnostmi uživatele nebo mírně pomalejší. V používání klávesnice je skript výrazně rychlejší, protože je schopný poslat řetězec prakticky okamžitě. Dosažené zrychlení u testů databází je vidět v Tab. 6. Např. test paměti letových bodů je kratší až o 70%, což znamená úsporu asi tři hodiny.

Výhodou automatického testu je lepší reprodukovatelnost, protože je z procesu vyřazen uživatele, který může vnášet chyby. Skript zreprodukuje naprosto stejné ovládání při každém spuštění. To umožňuje test zopakovat na jiné testovací stanici v jiné pobočce firmy Honeywell a tím ověřit, že chyba není na stanici.

Uživateli obsluhujícímu automatizovaný test stačí nižší kvalifikace. Nemusí systém znát podrobně a obejde se dokonce bez znalosti ovládání systému. Stačí mu následovat instrukce zobrazené na počítači a pokud si není jistý může udělat snímek obrazovky pro vyhodnocení zkušenějším uživatelem.

V neposlední řadě automatizované testování umožňuje také vzdálené testování. Je možné se ke stanici připojit ze vzdáleného počítače a test spustit. Pokud jde o test, který nevyžaduje zásah uživatele do ovládacích prvků lze jej vzdáleně vykonat bez omezení. Uživatel potom ověření výzev provádí s pomocí IP kamery.

Mezi nevýhody automatizovaného testování bych zařadil především malou flexibilitu. Scénář se dá využít pouze pro jeden typ testu a jeho úpravu nezvládne uživatel bez znalosti skriptovacího jazyka a všech používaných funkcí. Navíc pro novou verzi FMS je potřeba často i nový skript, protože chování některých ovládacích prvků mezi verzemi se mění.

Další slabostí automatického testu je, že neodhalí neočekávané situace. Uživatelé, kteří doposud prováděli testování ručně, uvádí, že část problémů, které zachytí, se neobjeví v okamžicích předepsané kontroly (tedy při ověření podle indikátorů), ale v jiných okamžicích. Je pro ně důležité sledovat chování systému a toho, co se zobrazuje na displejích, aby získali celkový přehled a mohli říct, že systém se chová tak, jak se očekává. Toto automatický test neumožní.

S pomocí porovnávání obrazu je schopný skript ověřit, že všechno funguje jak má. Ale pro rozlišení neočekávaných stavů je vhodnější, pokud je během vykonávání skriptu přítomný uživatel který si může všimnout, co se s aplikací během testu děje. Stále zde ale zůstává výhoda rychlosti – skript dokáže ovládat systém rychleji než uživatel.

## 10.6. NÁVRHY NA ROZŠÍŘENÍ

První z mých návrhů jak zjednodušit tvorbu scénářů je vytvořit standard, kterým by se dal test zapsat do .xls dokumentů. Na tomto principu už ve firmě Honeywell funguje jiný typ testování. Skript potom slouží k načítání jednotlivých kroků ve smyčce z .xls dokumentu. Každý řádek dokumentu reprezentuje jeden krok testu a uchovává tak data místo matice zmíněné v kapitole 10.3.

Pokud se všechny kroky a povely píšou přímo do kódu, stává se kód nepřehledným a kvůli každé změně testu se do něj musí zasahovat. Načítání povelů z dokumentu umožňuje nejen rychleji vytvářet nové scénáře, ale také upravovat existující.

Druhým návrhem je integrovat do procesu rozšířené rozpoznávání obrazu pro účely vyhodnocování průběhu a výsledků testů. V základní variantě by mohlo jít například o rozpoznávání textu, neboli OCR. Pokud bude skript ve snímku obrazovky schopný v daném výřezu detekovat text a porovnat jej s očekávaným textem, odpadne v dalších případech nutnost přítomnosti uživatele u testování. Často totiž spočívá ověření v kontrole, že se na určité pozici nachází požadovaný text.

Velkým zjednodušením by bylo detekování skutečné pozice kurzoru na displejích z obrazu video grabberu. K tomu bude nutné implementovat rozpoznávání obrazu, které by zjišťovalo souřadnice kurzoru. Tyto souřadnice by sloužily jako zpětná vazba a po porovnání s požadovanými by se rozdílem řídil pohyb kurzoru tak, aby se vždy nacházel ve vytyčené oblasti.

Je pravděpodobné, že tímto směrem se bude ubírat další vývoj automatizovaného testování. Bude ovšem nutné ověřit, jaká je nejistota při rozpoznávání textu a zda-li je takový postup vůbec možný. Pokud se takové postupy ověří, je možné že se v budoucnu přepracují některé testovací procedury tak, aby vyhovovaly více automatizovanému testování, než manuálnímu.

# 11. ZÁVĚR

Prvním úkolem mé práce bylo nakonfigurovat testovací stanici do takového stavu, aby se daly softwarově simulovat výstupy z hardwaru. To zahrnovalo připojení počítače na kterém běží TIU server ke sběrnicím, které vstupují do řídící jednotky a nakonfigurování TIU severu. Využíval jsem stávající zapojení ale také jsem navrhl nové zapojení, které odstraní nutnost pořizovat na dalších testovacích stanicích nový hardware.

Pro TIU server jsem napsal konfigurační soubor, který nastaví proměnné pro všechny potřebné signály. Odesílají se z něj periodické zprávy ale také jednotlivé aperiodické pakety vyvolané akcí uživatele. Právě odesílání aperiodických paketů bylo nutné vyřešit, protože TIU sever tuto možnost běžně neumožňuje. Konfigurační soubor byl zkombinován s konfiguračním souborem pro další ovládací prvky a je nyní běžně používán v praxi při testování softwaru ve firmě Honeywell.

Následovalo řešení simulace ovládacích prvků ze vzdáleně připojeného počítače. Cílem bylo nahradit palubní klávesnici KB-600 aplikací pro Windows. Aplikace je klient pro server TIU využívající grafické nadstavby. Ta se připojí přes podnikovou síť k TIU serveru a mění hodnoty proměnných uložených v jeho databázi. Server následně generuje signály odesílané po sběrnici RS422, které vstupují do řídící jednotky. Aplikace obsahuje také zpětnou vazbu umožňující kontrolu, jestli se data uložená na serveru shodují s odeslanými.

Ke komunikaci se serverem jsem využil knihovny vyvinuté v rámci firmy Honeywell, nad kterými jsem postavil vlastní vrstvu pracující s daty. Grafická nadstavba aplikace využívá .NET frameworku a věrně připomíná vzhled palubní klávesnice.

Řešení se ukázalo jako plně funkční, bylo otestováno v rámci jedné budovy na testovací stanici letounu F7X. Uživatelé jsou s funkcí a vzhledem aplikace spokojeni a hodnotí její užívání jako plynulé a přirozené. Aplikace slouží příležitostně k testování softwaru a ovládání testovací stanice v Brně z pobočky v Phoenixu, USA.

V poslední fázi jsem vyvinul procedury pro automatické testování. Procedury jsou psané v jazyce TATS, který vychází z Visual Basic skriptu a umožňuje komunikaci s TIU serverem. Připravil jsem knihovny, obsahující sadu příkazů ovládajících vstupy z klávesnice a ovladače kurzoru. Z jednotlivých příkazů uživatel sestaví TATS skript vykonávající požadovanou činnost.

Také jsem vytvořil příkazy k sejmutí snímku obrazovky a porovnání obrazu. Tyto snímky slouží buď k automatickému vyhodnocení testu nebo k uložení snímku pro vyhodnocení uživatelem.

Za účelem prezentování automatizačních příkazů jsem vytvořil také dva ukázkové skripty, které implementují testy běžně prováděné na stanici. Je to test databází a test spuštění systému a redundance. První zmíněný je specifický tím, že jde o neustále se opakující smyčku, která díky automatizaci proběhne až o 70% rychleji. Druhý oproti tomu vykonává mnoho různých činností a jeho úkolem je uživateli „připravit půdu“ k ověření výsledků.

Oba skripty slouží pouze jako ukázka a předtím, než se začnou prakticky využívat při testování, je potřeba tuto metodu dále rozvíjet a ladit. Přesto se mi s využitím testů databází podařilo odhalit některé nedostatky databází FMS. Právě vyvíjená verze softwaru nesplňuje všechny požadavky na kapacitu paměti a její překročení způsobí kolaps FMS (blíže diskutováno na konci kapitoly 10.2). V oblasti automatizace testů se očekává další vývoj a na závěr své práce jsem přidal několik doporučení, kterým směrem by se měl vývoj ubírat.

Můj pohled na problematiku automatických testů se v průběhu práce měnil. Z počátku se zdálo, že největším problémem bude vytvořit funkce umožňující ovládání. Ukázalo se ovšem, že tou největší výzvou je ovládat rozhraní vytvořené pro lidského uživatele automaticky bez zpětné vazby.

Na závěr bych chtěl shrnout, že jsem dosáhl cílů vytyčených v zadání a dokázal jsem přidat i další rozšíření. Těmi bylo především automatizované snímání a rozpoznávání obrazu. Práce pro mě byla inspirativní a během její tvorby jsem se naučil mnoho nového.

# SEZNAM ZKRATEK

| | |
|---|---|
| AGM | Advanced grafics module (pokročilí grafický modul) |
| BIC | Backplane interface circuit (propojovací obvody) |
| CCD | Cursor control device (zařízení ovládající kurzor) |
| EFIS | Electronic flight information system (elektronický letový informační systém) |
| FMS | Flight management system (systém plánování a optimalizace letu) |
| IO | Input / output (vstup / výstup) |
| INS | Inertial navigation system (inerciální navigační systém) |
| MAU | Modular avionics unit (modulární jednotka pro avioniku) |
| MFD | Multifunctional flight display (multifunkční displej) |
| MRC | Modular radio kabinet (modulární jednotka pro rádia) |
| NIC | Network interface controller (ovladač síťového rozhraní) |
| PFD | Primary flight display (primární letový displej) |
| PSM | Power supply module (napájecí modul) |
| SAR | Search and rescue (operace pátrání a záchrany) |
| SITS | System integration test station (testovací stanice pro systémovou integraci) |
| TATS | TIU-based automated test system (automatický testovací systém založení na TIU serveru) |
| TIU | Test interface unit (jednotka testovacího rozhraní) |
| VBS | Visual Basic script (Visual Basic skript) |
| WX radar | Weather radar (srážkový radar) |

# SEZNAM OBRÁZKŮ

Obr. 1 – Letoun F7X za letu [1] ..... 13
Obr. 2 - Vrtulník AW139 [10] ..... 14
Obr. 3 - Schéma testovací stanice [3] ..... 15
Obr. 4 - Testovací stanice SITS ..... 16
Obr. 5 - Pohled do kokpitu letounu Falcon 7X [6] ..... 17
Obr. 6 - Rozložení klávesnice KB-600 [5] ..... 18
Obr. 7 - Schéma TATS skriptů [3] ..... 20
Obr. 8 - Schéma původního zapojení testovací stanice ..... 21
Obr. 9 - Schéma připojení NIC ke sběrnici ASCB-D [8] ..... 22
Obr. 10 - Diagram systému Primus Epic [8] ..... 23
Obr. 11 - Navigační mapa a seznam letových bodů FMS ..... 24
Obr. 12 - Pohled na PFD z kamery SNC 550 ..... 26
Obr. 13 - Okno programu TBOX zobrazující strukturu datového stromu ..... 29
Obr. 14 - Nastavení TIU serveru ..... 31
Obr. 15 - Schéma upraveného zapojení testovací stanice ..... 32
Obr. 16 - Okno aplikace MKB remote simulation při spouštění ..... 34
Obr. 17 - Zjednodušený vývojový diagram aplikace MKB remote simulation ..... 36
Obr. 18 - Rozdělení typů kláves ..... 37
Obr. 19 - Okno aplikace MKB remote simulation ..... 38
Obr. 20 - Testování aplikace ..... 40
Obr. 21 - Detail CCD [12] ..... 42
Obr. 22 - Vlastnosti kurzoru ..... 43
Obr. 23 - Okno aplikace Interface, umožňující vzdálené ovládání CCD ..... 44
Obr. 24 - Zapojení video grabberů ..... 48
Obr. 25 - Počáteční rozložení spodního MFD pro práci s databázemi FMS ..... 51
Obr. 26 - Porovnání snímků displeje ..... 53
Obr. 27 - Výzva pro uživatele ..... 56
Obr. 28 - Definice scénáře v kódu skriptu ..... 57

# SEZNAM TABULEK

Tab. 1 – Souhrn existujících a předpokládaných nových řešení ..... 12
Tab. 2 - Schéma TIU serveru ..... 19
Tab. 3 - Vlastnosti přenosu sériové linky ..... 23
Tab. 4 - Přehled bytů vysílaných ve zprávě ..... 30
Tab. 5 - Tabulka funkcí použitelných při tvorbě scénářů ..... 50
Tab. 6 - Doba trvání testů ..... 55
Tab. 7 – Část instrukcí k FMS power UP, Database & Redundancy testu ..... 56

# POUŽITÁ LITERATURA


1. *Falcon 7X* [online]. [cit. 2016-01-07]. Dostupné z: http://www.dassaultfalcon.com/en/Aircraft/Models/7X/Pages/overview.aspx

2. *Agusta westland AW139* [online]. [cit. 2016-02-23]. Dostupné z: http://www.helis.com/database/model/262/

3. *HONEYWELL. SBE specific Products and Subsystems: Primus Epic system overview. 2003.*

4. *HONEYWELL. TIU user's guide [firemní úložiště]. 2001 [cit. 2015-11-04].*

5. *HONEYWELL. Hardware requirement document for the keyboard KB-600. 2005.*

6. *AIRLINERS* [online]. 2011 [cit. 2015-12-17]. Dostupné z: http://www.airliners.net/

7. SNC 550 product page [online]. [cit. 2016-01-06]. Dostupné z: https://pro.sony.com/bbsc/ssr/micro-government/cat-securitycameras/product-SNCEP550/

8. *HONEYWELL. Modular avionics unit specification for Primus Epic*. 2002.

9. *Epiphan DVI2USB3* [online]. [cit. 2016-02-23]. Dostupné z: http://www.epiphan.com/products/dvi2usb-3-0/

10. *AW139_balder* [online]. [cit. 2016-02-23]. Dostupné z: https://commons.wikimedia.org/wiki/File:AgustaWestland_AW139_at_Balder.JPG

11. *Image Magick* [online]. [cit. 2016-03-01]. Dostupné z: http://www.imagemagick.org/script/index.php

12. *Dassaultfalcon* [online]. [cit. 2016-03-08]. Dostupné z: www.dassaultfalcon.com

# SEZNAM PŘÍLOH


1. Příloha Ukázka formátu konfiguračního souboru
2. Příloha Ukázka skriptovacího jazyka TATS
3. Příloha Část výstupu skriptu po spuštění scénáře FMS power up
4. Příloha Konfigurační soubor CCD&MKB_config_2_0.txt
5. Příloha Komprimované zdrojové kódy aplikace MKB remote
6. Příloha Komprimované zdrojové kódy skriptů

# 1. Příloha

Ukázka formátu konfiguračního souboru

```
'***********************************
'MKB periodical frame
CHANNEL, MKB-1, Channel Node, 1, COM4, 115200, NONE, 1, 8

BLOCK, MKB Periodic Block, Block Node, 25.0, 48, 1, TX, PERIODIC, RUN,
NONE, CCDCHECKSUM16, 0x0

SYNC, Sync Node, 0x000000ff, 0x00000002, 0x000000ff, 0x00000003
GROUP, Discrete Inputs, Parameter Group


'ENUM values - either active or inactive (defined at the end)
'<key: DATA>,<data name>,<desc>,<type>,<start bit>,<bit size>,<enum
ID>,0,<init value>,<units>,<element seq>
DATA, Enter Keypad, Data Node, ENUM, 56, 1, 1, 0, 0, NA, 0
'bits 01:06 from byte 7 Spare?
DATA, WX RADAR (OFF), Data Node, ENUM, 64, 1, 1, 0, 0, NA, 0
DATA, WX RADAR (STBY), Data Node, ENUM, 65, 1, 1, 0, 0, NA, 0
DATA, WX RADAR (AUTO), Data Node, ENUM, 66, 1, 1, 0, 0, NA, 0
DATA, WX RADAR (GAIN), Data Node, ENUM, 67, 1, 1, 0, 0, NA, 0
DATA, WX RADAR (TILT), Data Node, ENUM, 68, 1, 1, 0, 0, NA, 0
DATA, WX RADAR (OVRD), Data Node, ENUM, 69, 1, 1, 0, 0, NA, 0


'*****************************************************************
****
'Pseudoperiodic frame with long period
'Is turned on just for short time and immediately turned off to
simulate aperiodic frame

BLOCK, MKB Pseudoperiodic, Block Node, 1000, 1, 1, TX, PERIODIC, RUN,
NONE, CCDCHECKSUM16, 0x0

GROUP, Keyboard Function Key, Parameter Group

DATA, Function Keys byte, Aperiodic byte coding function keys, UINT,
0, 8, 0, 0, 0, NA, 0
```

## 2. Příloha

Ukázka skriptovacího jazyka TATS

```
''Function SendStack
'Sets char nodes according to string defined by input parameter "text"
Function SendStack (text)
   Dim char
   Dim j

   'Go throught string and write char by char to the server
   For i = 1 To Len(text)
      char=Asc(Mid(text,i,1))
      If char > 90 Then
         char=char-32
      End If
      Write Chars(i-1), char
      j=i
   Next

   Read(Chars(j)), char

   'Go through nodes behind last used node and set them to zero,
   'until there is a zero charakter
   Do While char <>0
      Write Chars(j), 0
      j=j+1
      Read(Chars(j)), char
   Loop

   SendStack = 1
End Function

''Function Enter
'Sends enter to one for one second and clears stack (as if user hits enter)
Function Enter
   Write Enter_Keypad, 1
   Delay 1.0 's

   ClearStack
   Write Enter_Keypad, 0
End Function
```

## 3. Příloha

Část výstupu skriptu po spuštění scénáře FMS power up

```
1     Verify that current position is *Not Initialized* in
      Avionics window Init Tab (lower MFD).

      PASS

2     Verify FMS as Navigation source annunciated on PFD HIS
Window

      PASS

3     Verify on left PDU that the default selected mode for all
      FMS is Synchronous.

      FAIL  Second FMS is single

4     Verify on PDU different FMS is driving each PDU. (By
      default FMS1 pilot side and FMS2 co-pilot side).

      PASS

5     Verify on MDU FMS data is driven by pilot side FMS
      (Default FMS1).
      Note:This can be verified by annunciations/Indications On
      the INAV Window

      FAIL  Driven by copilot side (FMS2)

6     Make sure Active Mode in all three FMSs is Synchronous -
      In triplex mode all data on FMS will be Synchronized

      PASS

7     Verify FMS driving MDU has been changed accordingly.

      PASS
```

4. **Příloha - elektronická**
   Konfigurační soubor CCD&MKB_config_2_0.txt

5. **Příloha - elektronická**
   Komprimované zdrojové kódy aplikace MKB remote

6. **Příloha - elektronická**
   Komprimované zdrojové kódy skriptů