VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

RADAROVÝ VÝŠKOMĚR PRO ULTRALEHKÝ LETOUN

RADAR ALTIMETER FOR ULTRALIGHT AIRCRAFT

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **JIŘÍ ZAHRADNÍK**
AUTHOR

VEDOUCÍ PRÁCE **Ing. LUKÁŠ MARŠÍK,**
SUPERVISOR

BRNO 2017

Zadání bakalářské práce/19279/2016/xzahra22

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačové grafiky a multimédií Akademický rok 2016/2017

Zadání bakalářské práce

Řešitel: **Zahradník Jiří**
Obor: Informační technologie
Téma: **Radarový výškoměr pro ultralehký letoun**
Radar Altimeter for Ultralight Aircraft
Kategorie: Zpracování signálů

Pokyny:

1. Seznamte se s radarovými výškoměry, programováním embedded systémů a zaměřte se také na rozhraní vhodná pro jednoduchou interpretaci informací uživateli.
2. Vytipujte vhodnou embedded platformu umožňující sběr a zpracování dat z FM-CW radaru.
3. Navrhněte systém zpracovávající radarová data a vhodně indikující změřenou výšku. Promyslete také umístění radarového senzoru pod trupem či křídlem letadla.
4. Implementujte funkční systém navržený v předchozím bodě.
5. Otestujte systém v simulovaném a případně reálném prostředí.
6. Diskutujte dosažené výsledky a navrhněte možné pokračování práce.

Literatura:

- M. Skolnik: Radar Handbook, 3rd edition, McGraw-Hill Professional, 2008
- M. Skolnik: Introduction to Radar Systems, McGraw-Hill Science, 3rd edition, 2002
- M. A. Richards: Fundamentals of Radar Signal Processing, 1st edition, McGraw-Hill, 2005
- B. R. Mahafza: Radar Signal Analysis and Processing Using MATLAB, Chapman and Hall, 2008

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 až 3 zadání.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Maršík Lukáš, Ing.**, UPGM FIT VUT
Konzultant: Zemčík Pavel, prof. Dr. Ing., UPGM FIT VUT
Datum zadání: 1. listopadu 2016
Datum odevzdání: 17. května 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačové grafiky a multimédií
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Jan Černocký
vedoucí ústavu

Abstrakt

V této bakalářské práci se autor zabývá návrhem a částečnou implementací radarového výškoměru. V této práci je kladen důraz na modulární architekturu a proto je tento výškoměr navržen jako soubor samostatných modulů komunikujících pomocí BSD schránek. Implementace programového vybavení je v C++ a pro generování zvuku je použita knihovna PulseAudio. Dále je zde řešena bezpečnost mezivláknové fronty a zásobníku pomocí třídy implementované jako šablona pro zachování jednoduchosti a obecnosti pro další využití.

Abstract

In this bachelor thesis author designs a radar altimeter. In this thesis the emphasis is placed onto modular architecture andthat is why this altimeter is designed as a group of independent modules communicating through BSD sockets. Software implementation is made in C++ and for sound generator is used library PulseAudio. Next topic is multithreading safety implementation of queue and stack which was made by template class to keep simplicity and generality.

Klíčová slova

radar, výškoměr, PulseAudio, C++, vícevláknové zpracování, BSD schránky, avionika, generování zvuku, vestavěné systémy, Xilinx Zynq, FPGA, USB zvuková karta, přistání

Keywords

radar, altimeter, PulseAudio, C++, multithreading, BSD sockets, avionics, sound generating, embedded systems, Zynq, FPGA, USB soundcard, landing

Citace

ZAHRADNÍK, Jiří. *Radarový výškoměr pro ultralehký letoun*. Brno, 2017. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Maršík Lukáš.

Radarový výškoměr pro ultralehký letoun

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Lukáše Maršíka. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Jiří Zahradník
16. května 2017

Poděkování

Tímto chci poděkovat mému vedoucímu, panu Ing. Lukáši Maršíkovi za odborné vedení práce a kolegům z práce za poskytnutí doplňujících informací.

Obsah

Kapitola 1

Úvod

V posledních letech se letectví stává čím dál tím důležitějším a běžnějším. Ať už se jedná o bezmotorové jednomístné kluzáky, akrobatické či vyhlídkové ultralehké letouny anebo velké dopravní letadla. Rozmachu se dočkalo hlavně sportovní letectví, kdy s postupným vývojem nových technologií dochází ke zpřístupňování těch stávajících běžným občanům. S tímto rozmachem ruku v ruce přichází tlak na zefektivnění výuky létání. Jedním z možných způsobů je vyvinutí radarového výškoměru rozumných rozměrů s rozumným výkonem za rozumnou cenu pro výuku přistání.

Výškoměr je jedním z nepostradatelných přístrojů pro navigaci za letu. Je důležité, aby fungoval správně ve všech situacích, jelikož chyba v měření může znamenat havárii letadla. V lepším případě dojde "pouze"ke škodám na majetku, v horším případě může dojít i ke ztrátám na životech. Plní nezastupitelnou úlohu, díky níž mohou piloti bezpečně létat i za špatné viditelnosti.

Motivace pro tuto práci je zvýšení bezpečnosti a ulehčení výcviku začínajících pilotů. Jde hlavně o ulehčení výuky přistání, která začátečníkovi může způsobovat potíže. Jedná se primárně o výuku dosednutí(anglicky "flare"), kdy musí student stáhnout výkon motoru a přitáhnutím uvést letoun do horizontálního letu. Pokud student udržuje letadlo v horizontálním letu nad dráhou ve velké výšce, může být dosednutí tvrdší, než na které je stavěn podvozek, a může dojít k poškození podvozku či zranění posádky i cestujících.

Cílem této práce není, jak se může zdát, zpracování signálů. Prvním cílem této práce je úvod historie vývoje avioniky, hlavně výškoměrů. Druhým cílem je úvod do problematiky měření výšky již od jejího počátku. Dále bude čtenář stručně seznámen s vývojem radarů, dozví se, jak se radary dělí a k čemu se jednotlivé typy využívají. Třetím cílem je popis problému výuky přistání a důvod pro zadání této práce, jakožto i jejího řešení. Dalším cílem je návrh takového systému, který splní veškeré požadavky na bezpečnost letového provozu na letištích, zvýší bezpečnost výuky přistání a samozřejmě zjednoduší samotnou výuku. V ideálním případě dojde i k implementaci a vzniku samotného systému, jeho integrace do letadla a úspěšnému otestování.

V následujících kapitolách budeme plnit cíle vytyčené v předchozím odstavci. Nejdříve se podíváme na vývoj přístrojů a jejich použití, vývoj měření tlaku a s jeho pomocí i měření výšky. Uvedeme si něco existujících radarových výškoměrech a jejich využití a zběžně si ukážeme vývoj radaru. Po tomto zařazení do historického kontextu si popíšeme a vysvětlíme fungování jednotlivých technologií a přiblížíme si problematiku přistání. Následuje návrh řešení našeho problému, kde analyzujeme zadání a pokusíme se dobrat optimálního řešení. Následně bude vysvětlena implementace navrženého řešení a v závěru shrneme výsledky této práce, její přínos a možnost dalšího pokračování.

Kapitola 2

Historie avioniky

2.1 Vývoj přístrojů

V roce 1903 bratři Wrightové uskutečnili první řízený motorový let. Trval jen chvíli, ale byl to přelom v technologii dopravy. Tímto začala éra letectví. Za první světové války, kdy došlo k prvnímu masivnímu rozšíření letadel, ať již k průzkumným účelům, bombardování nebo vzdušným soubojům byli piloti omezeni počasím a za špatného počasí nemohli létat, jelikož neexistovala technologie, která by umožnila pilotovi nespoléhat pouze na svůj zrak a navigovat za letu i jinak, než pouze vizuálně. Toto vedlo ke snahám vyvinout systém, který by umožňoval navigaci za jakýchkoliv podmínek. Za jasného slunečného počasí nebo třeba v bouřce.

Začaly se objevovat první přístrojové desky a na nich první přístroje. Kvůli omezené ploše palubní desky se na ni moc přístrojů nevlezlo, proto na ni bylo umístěno pouze několik nejdůležitějších přístrojů. Mezi tyto přístroje patřil kompas, teploměr oleje, otáčkoměr, ukazatel indikované rychlosti a samozřejmě výškoměr. S tímto primitivním vybavením se museli spokojit piloti během první světové války i v období míru po ní. Postupem času toto omezené vybavení přestalo splňovat podmínky pro přístrojové lety a s nástupem nových technologií se na palubní desku muselo vejít mnohem více zařízení. Každé letectvo prodělalo vývoj, bohužel, nejrychlejší byl v dobách druhé světové války, kdy došlo k nejrychlejšímu technologickému pokroku na obou stranách. Rozdíl mezi přístrojovými deskami jednotlivých národů můžeme pozorovat na obrázcích 2.1 a 2.2. Podoba kokpitu se podstatně neměnila až do příchodu digitální technologie, kdy analogové přístroje nahradily sofistikované systémy zobrazující veškeré údaje potřebné pro všechny fáze letu a v případě vojenských letounů i pro bojové situace na displeje v kokpitu.

Obrázek 2.1: Přístrojová deska stíhacího letounu Bf109-G2[1]

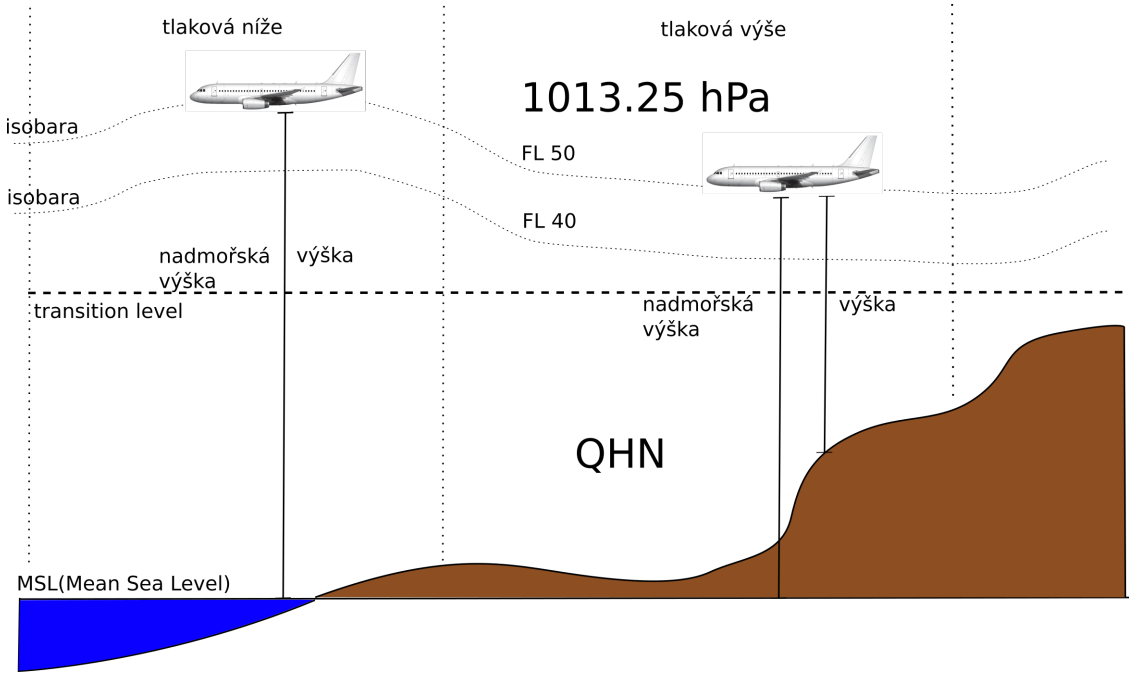
Obrázek 2.2: Kokpit doprovodného stíhacího letounu P51[2]

[1] zdroj: http://forum.largescaleplanes.com/index.php?showtopic=8725
[2] zdroj: http://www.warbirdalley.com/p51.htm

2.2 Vývoj měření výšky a úvod do této problematiky

Barometry byly lidstvu známy již od 17. století, tyto bohužel obsahovaly vodu. Aby mohl být tlak vzduchu úspěšně změřen, barometr vyžadoval přibližně 10 metrů vysokou trubici. Avšak italskému matematikovi a fyzikovi Evangelistovi Torricellimu se podařilo zkrátit délku trubice na 80cm nahrazením vody rtutí, která je cca třináctkrát hustší než voda. I přes své relativně malé rozměry je tento barometr pořád příliš velký pro masové rozšíření. Za další zmenšení přístroje je zodpovědný francouzský vědec Lucien Vidi, který vynalezl aneroidní barometr, přezdívaný aneroid[5].

Díky této změně se barometr zmenšil na rozumnou velikost a již se dal použít v omezeném prostoru letadel. Ve skutečnosti je barometrický výškoměr pouze překalibrovaný aneroid. Bohužel stejně jako ukazatel indikované rychlosti je závislý na mnoha faktorech. Mimo jiné můžeme jmenovat teplotu a vlhkost vzduchu. Zásadním problémem je, že barometrický výškoměr už z principu neukazuje relativní výšku nad terénem, ale výškud AMSL(height Above Mean Sea Level). Reálně tedy dvě letadla letící podle výškoměru ve výšce 6000 stop několik desítek námořních mil od sebe mohou být v jiné nadmořské výšce. Na obrázku 2.3 je názorná ukázka. Zde může čtenáře napadnout, zda se letadla kvůli rozdílným indikovaným hodnotám nesrazí. Barometrické výškoměry sice ukazují nadmořskou výšku špatně, ale když se letadla přiblíží, tak ukazují stejně špatně a letadla se tedy minou.



Obrázek 2.3: Rozdíl mezi barometrickou(nadmořskou) výškou a aktuální výškou

Další potíže nastávají s rozdílnými atmosférickými tlaky mezi počátečním a cílovým letištěm. Mějme následující situaci: Letadlo letí z letiště Shiphol v Amsterdamu, které je tři metry pod hladinou moře, do Himalájí. Zde se v Tibetu nachází letiště Bangda ve výšce 4334 metrů[2]. Rychlou úvahou dojdeme k závěru, že si pilot bude muset dávat pozor v jaké výšce nad terénem letí a na jakou výšku bude přistávat. Toto bylo vyřešeno přidáním možnosti kalibrace. Pilot před vzletem nastaví hodnotu QNH, kterou pilotovi sdělí řídící letového

provozu, taktéž před přistáním učiní řídící v cílové oblasti. Na obrázku 2.4 můžeme vidět bíle okénko ukazující tlak QNH a kolečko pro jeho nastavení. Do přechodové hladiny se používá tento výškoměr a nad touto hladinou pilot letí podle druhého výškoměru, který je pevně nastaven na hodnotu 1013.25 hPa. Toto je pevně zavedený standard pro veškeré civilní lety.

Obrázek 2.4: Výškoměr z letounu Bf109[3]

S nástupem satelitních technologií se začaly využívat systémy satelitní navigace. Toto upřesnilo navigaci a již se nemusíme spoléhat pouze na pozemní kontrolu při určovaní pozice letadla. Pomocí systému GPS je schopna posádka určit přesnou pozici na Zemi, rychlost a azimut během několika sekund, aniž by se musela spoléhat na nepřesné pozemní radary.

Existující radarové výškoměry Dalším zdokonalením bylo uvedení radarových výškoměrů. Ty jsou využívány jako hlavní součást GPWS(Ground Proximity Warning System), kde měří absolutní výšku nad terénem. Toto se využívá ve velkých letadlech vážících desítky tun. Našim cílem je vyvinout dostatečně malý, ale výkonný vestavěný systém, který by radarové výškoměry zpřístupnil i pro segment lehkých a ultralehkých letadel.

2.3 Vývoj radaru

Zrození radaru se datuje do počátku 20. století, kdy si mocnosti nezávisle na sobě uvědomily význam této technologie. Mezi lety 1934 a 1939 začaly USA, Velká Británie, Německo, Sovětský svaz, Japonsko, Nizozemsko, Francie a Itálie nezávisle na sobě vyvíjet systém detekce pomocí radiových vln[9]. Vývoj radaru výrazně pomohl při bitvě o Británii, kdy britové detekovali formující se letadla Luftwaffe nad Francouzským územím, a tímto umožnil rychlou odezvu Britských pilotů.

Během studené války se radary výrazně zdokonalily. Zvětšil se jejich výkon a zmenšila se jejich velikost. V současné době se radary houfně využívají k mírovým účelům, od sledování počasí přes letectví či posuny zemské kůry až po mapování.

[3]zdroj: http://www.warbirdsite.com/Collection.html

Kapitola 3

Použité technologie a teorie přistání

3.1 Radary a jejich dělení

Radar (**RA**dio **D**etection **A**nd **R**anging) je systém pro detekci objektů, u kterých je pomocí rádiových vln schopen určit vzdálenost, azimut a rychlost sledovaných objektů. Využití radarů je široké, od mapování terénu, detekce osob přes měření rychlosti vozidel a využití v civilním letectví až po vojenské účely.

Klasifikace radarů Radary se dělí podle typu vysílání a podle použití[10]. Základní dělení je:

- **Primární radar** - Slouží k zobrazování objektů ve vzdušném prostoru. Vysílač vysílá vysokofrekvenční signál a přijímač zachytává ozvěnu vysílaného signálu. Na příslušném místě na obrazovce posléze vykreslí, v profesním slangu řečeno, knedlík, podle pozice objektu ve vzdušném prostoru vůči radarové stanici.
- **Sekundární radar** - Slouží k přenosu informací mezi radarovou stanicí a letadlem. Aby toto spojení fungovalo, musí mít cílový objekt zapnutý transpondér v příslušném módu, který po přijetí signálu zpracuje dotaz a odpoví požadovanými informacemi např. výška, rychlost, souřadnice GPS a kurz.

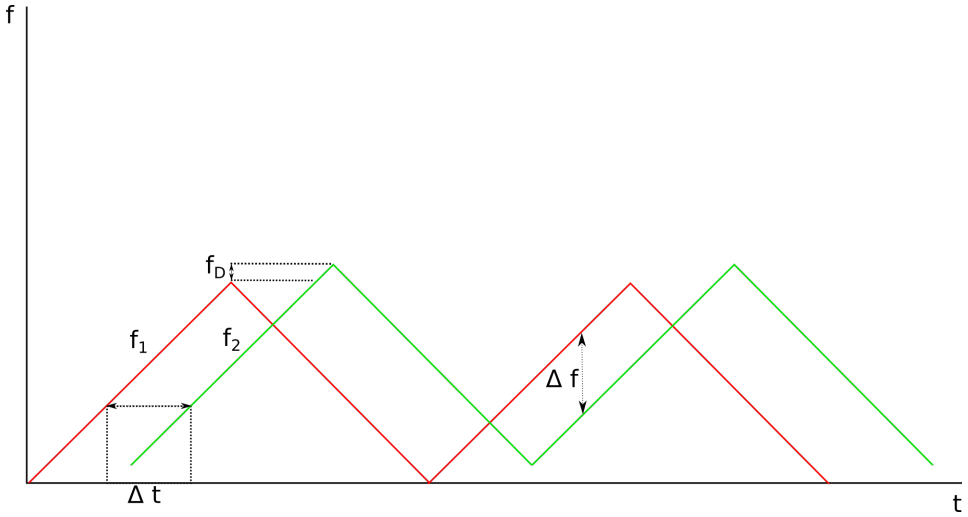
Primární radary se dále dělí na:

- **Pulsní** - Tomuto typu radarů postačuje pouze jedna anténa a to z toho důvodu, že se radar periodicky přepíná mezi vysílacím režimem a přijímacím režimem. Z tohoto pramení nevýhoda tohoto typu radarů: minimální a bez modulací i maximální dosah. Minimální dosah radaru je určen rychlostí přepínaní mezi jednotlivými režimy. Pokud se vrátí ozvěna dříve, než se radar přepne do přijímacího režimu, radar objekt nedetekuje a informace je ztracena. Obdobně to funguje i u maximálního dosahu. Radar musí zachytit ozvěnu před vysláním dalšího impulsu.
- **Kontinuálně vysílající** - Radary tohoto typu vysílají nepřerušovaně a zároveň nepřerušovaně zpracovávají přijatou ozvěnu, proto potřebují jak vysílací, tak přijímací anténu. Tyto radary se dělí na:

– Modulované - Tyto radary využívají frekvenční modulace signálu, díky které jsme schopni vypočítat vzdálenost z ozvěny vysílaného signálu. Toto je typ radaru, kterým bude vybaveno naše zařízení.
– Nemodulované - Tyto radary vysílají konstantní signál, který umožňuje pouze určení rychlosti zachycených objektů.

U FM-CW[1] radarů je oproti pulsním radarům měření vzdálenosti podstatně komplikovanější.

Jak je zmíněno výše, FM-CW radar potřebuje pro detekci vzdálenosti frekvenční modulaci. V reálu se využívá více modulací, zde si uvedeme pouze několik. Modulaci pilovitou, a modulaci trojúhelníkovou. Díky těmto modulacím jsme schopni zjistit vzdálenost překážky. Nejdříve odstraníme dopplerův posuv a posléze můžeme vzdálenost vypočítat pomocí časového rozdílu mezi odeslaným signálem a přijatým signálem se stejnou frekvencí [8].



Obrázek 3.1: Frekvenční modulace trojúhelníkem

Dopplerovu frekvenci vypočítáme s pomocí vzorce

$$f_d = \frac{\Delta f_1 + \Delta f_2}{2}$$

přijatou frekvenci vypočítáme ze vzorce

$$f_r = \frac{\Delta f_1 - \Delta f_2}{2}$$

a výslednou vzdálenost určíme pomocí naši znalosti rychlosti s jakou modulujeme vysílaný signál.

[1] Frequency-Modulated Continuous-Wave

3.2 Vestavěné systémy

Jelikož náš systém má jediný účel, a to zjistit výšku letadla nad přistávací dráhou, využívat víceúčelová zařízení je pro nás zbytečně robustní a neforemné. Potřebujeme jednoduché, rychlé a elegantní zařízení, které bude plnit pouze jediný úkol. Zde přicházejí do hry vestavěné systémy.

Definice 1 *Vestavěné systémy VS jsou systémy, ve kterých je zpracování dat vestavěno/vloženo do většího systému a ve kterém není zpracování dat viditelné uživateli prostřednictvím např. PC počítače. Anglicky se vestavěné systémy označují jako* ***embedded system****[7].*

Vestavěné zařízení je jednoduchý a elegantní způsob řešení jednotvárných algoritmických úloh. Naše úloha musí být vykonávána v reálném čase, tzn. máme určenou dobu odezvy a tu musíme splnit. Nutnost mít co nejnižší odezvu určuje nároky na zvolené zařízení. Na tomto zařízení poběží pouze jediná aplikace. To nám umožňuje optimalizovat aplikaci na míru systému pro dosažení co největšího výkonu. Bohužel optimalizace aplikace není vše. Aby se algoritmus vykonával co nejrychleji, musíme pro ni zvolit odpovídající zařízení. Pro náš systém potřebujeme zajistit dostatečně rychlé vzorkování abychom zabránili aliasingu. Toto je nejlepší řešit na hardwaru, jelikož rychlost zpracování bude vyšší a budeme schopni zaručit splnění podmínky Nyquistova teorému, kdy vzorkovací frekvence musí být alespoň dvakrát větší než maximální frekvence signálu:

$$f_v > 2f_{max}$$

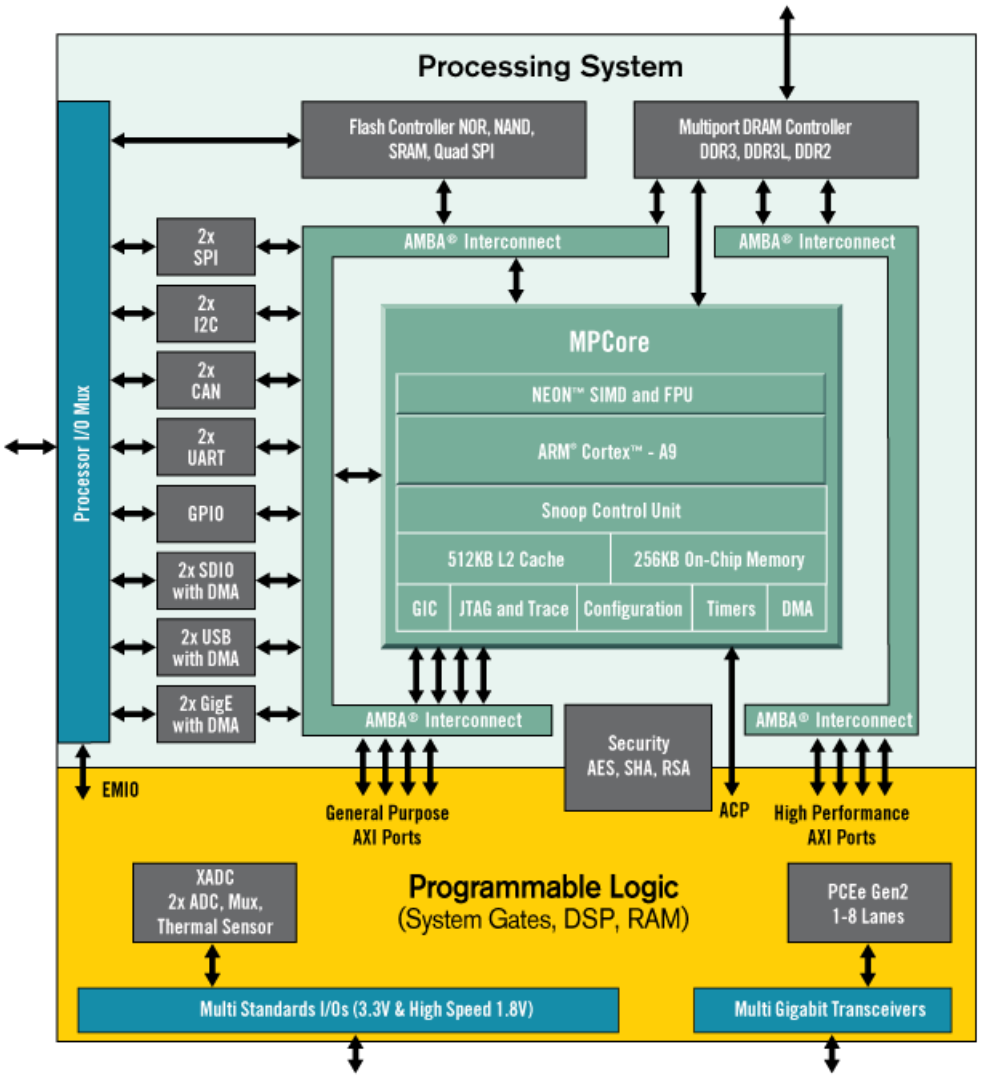
Pro tuto úlohu se hodí programovatelné hradlové pole FPGA. Toto pole bude vzorkovat signál s dostatečnou frekvencí a po sběrnici jednotlivé vzorky posílat dále do procesoru pro zpracování.

Xilinx Zynq Po určení požadavků na hardware v úvahu připadají zařízení od výrobce programovatelných hradlových polí Xilinx. Konkrétně rodina procesorů Zynq, která kombinuje FPGA s procesorovým jádrem ARM Cortex. [4] Jak již bylo zníněno výše, FPGA navzorkuje signál a pošle jej k dalšímu zpracování přez AXI porty jádru procesoru ARM, na kterém poběží odlehčená verze Linuxu. Na tomto operačním systému poběží aplikace provádějící potřebné výpočty a bude posílat výsledek na výstup. Architekturu procesorů rodiny Zynq můžeme vidět na obrázku 3.2.

Řídící modul Řídící modul obsahující výše zmíněný procesor je dodán společností CAMEA s.r.o. Modul obsahuje vstupně výstupní porty, které můžeme využít v náš prospěch. Jako výstupní porty můžeme využít USB sběrnici s přímým přístupen do paměti, GPIO porty nebo Serial Peripheral Interface.

- USB můžeme využít pro přenos přesných informací obrazovou formou na displej, kde zobrazujeme výšku, případně využít externí USB zvukovou kartu pro generování zvukových signálů.
- Pomocí GPIO portů, případně rozhraní SPI, můžeme ovládat panel s LE Diodami, který barevnými LED seřazenými vertikálně. Tento panel bude mít diody vyzařující světlo od zeleného spektra, přes žlutou, oranžovou a sytě červenou, kdy bude začínat fáze dosednutí.

- Zvuková komunikace může probíhat generováním tónů a odesílání konektorem typu 3.5mm Jack. Bude se generovat stále stejný tón konstantní délky, přičemž zkracující se perioda bude signalizovat snižující se výšku. Pokud bude letadlo těsně nad zemí, bude se generovat tón bez přerušení.



Obrázek 3.2: Architektura procesoru Zynq-7000[2]

Kalibrace modulu Další částí naší úlohy je nutnost kalibrace zařízení, jelikož každé letadlo může mít tento modul umístěno v jiné výšce nad zemí. Kalibrace může být statická - v kódu bude přičtena konstanta, nebo dynamická, kdy se před startem modul zkalibruje. Statická kalibrace má výhodu jednoduššího chování modulu a bude vyžadovat o jeden ovládací prvek méně, avšak bude nutno zasahovat do zdrojových kódů při instalaci na

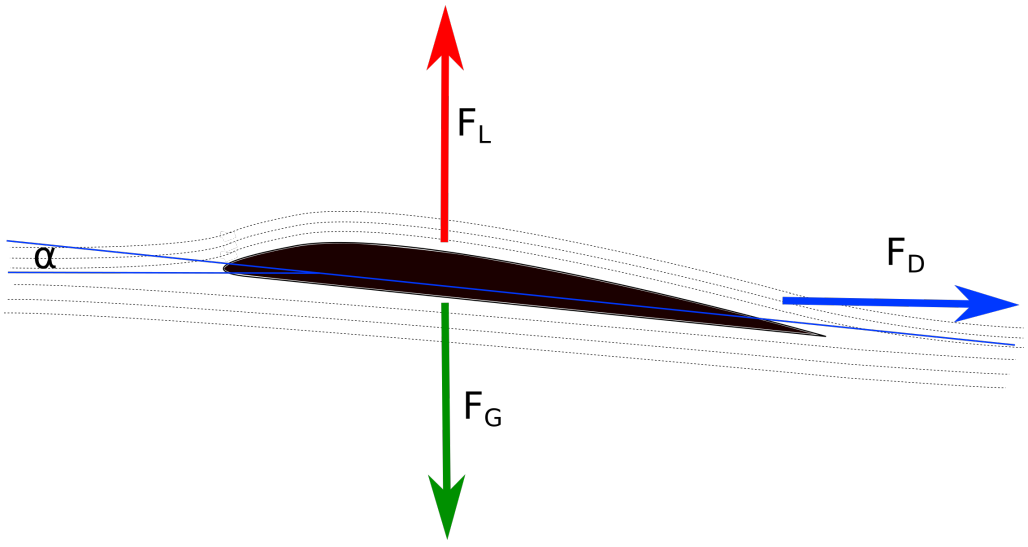
[2]zdroj: https://www.xilinx.com/products/silicon-devices/soc/zynq-7000.html

jiné letadlo. Dynamická kalibrace, na rozdíl od statické, má nevýhodu složitějšího řídícího programu, ale bude možno mít jeden program pro všechny typy letadel. Přidaný ovládací prvek nebude téměř znát. V tomto případě se autor přiklání k dynamické kalibraci.

Ovládací panel Nutnost kalibrace nás přivádí k ovládacímu panelu modulu. Přístroj je třeba nějak zapnout, toto bude první přepínač. Dále vyvstává otázka nutnosti mít vždy zapnutou zvukovou signalizaci. Jelikož bude využití tohoto výškoměru při vzletu mizivé, bude dobré, když bude mít pilot možnost vypnout zvukovou indikaci, která v situaci, kdy ji není potřeba může působit rušivě. Proto může být přítomen přepínač ovládající tuto funkci. Jako třetí kontrolní prvek zvolíme tlačítko spouštějící kalibraci přístroje na referenční hodnotu země. Je nutné, abychom zabránili kalibraci výškoměru ve vzduchu, proto by mělo toto tlačítko být překryté víčkem. V případě, že bude koncipováno jako ovládací prvek na dotykovém displeji, za pohybu letadla by mělo být deaktivované.

3.3 Teorie přistání

Letadlo, které vzlétne, musí zpět na zem. Ideálně rozumným a hlavně bezpečným způsobem. Není žádoucí, aby se jak letadlu, ale hlavně pasažérům cokoliv přihodilo. Způsobilo by to nejen finanční ztráty. Proto je kladen velký důraz na výuku a správné naučení přistávání.



Obrázek 3.3: Proudění vzduchu podél křídla.

Když se blíže podíváme na přistání, je to doslova řízený pád. Na obrázku 3.3 můžeme vidět příklad proudění vzduchu podél křídla konvenční konstrukce. F_L značí vztlakovou sílu(lift), F_G značí gravitační sílu(gravity), F_D značí odpor vzduchu(drag) a α značí úhel náběhu. Díky tomuto tvaru nad křídlem proudí vzduch rychleji než pod křídlem, podle Bernoulliho rovnice to znamená, že nad křídlem má vzduch menší hustotu než pod a dochází ke vzniku vztlaku. Čím větší je úhel náběhu, tím větší je vztlak a zároveň se zvětšuje odpor vzduchu. Toto platí pouze za podmínky, kdy je úhel náběhu menší než je jeho kritická hodnota. Od této kritické hodnoty, kterou má každý tvar křídla jinou, dochází ke snižování generovaného vztlaku až k nule, ale odpor vzduchu stále stoupá. To znamená, že letadlo ztrácí

rychlost a výšku. Na pilotovi je, aby tento řízený pád udržel v rozumné míře a nepropadl se do nekontrolovatelného pádu, jelikož při přistání není mnoho prostoru pro znovuzískání kontroly nad letadlem.

Přistání lze rozdělit do několika fází[1][6].

1. STAR(**S**tandard **T**erminal **A**rrival **R**oute) - Po opuštění letové trasy letadlo provede prvotní přiblížení po trase určené v letecké informační příručce(AIP).

2. ILS Přiblížení (**I**nstrument **L**anding **S**ystem **A**pproach) - Po ukončení STARu pilot na základě uděleného povolení provede přiblížení pomocí přístrojů, které ho přesně navádějí na dráhu. V této fázi si pilot připraví letadlo do přistávací konfigurace.

3. Závěrečné přiblížení (Final approach) - Během této fáze pilot musí udržovat přibližovací rychlost danou výrobcem letadla a letět po kurzu osy dráhy pod sklonem určeným přibližovacími pravidly(glidepath). Zde do hry vstupuje naše zařízení, které bude pilotovi oznamovat výšku stroje nad terénem.

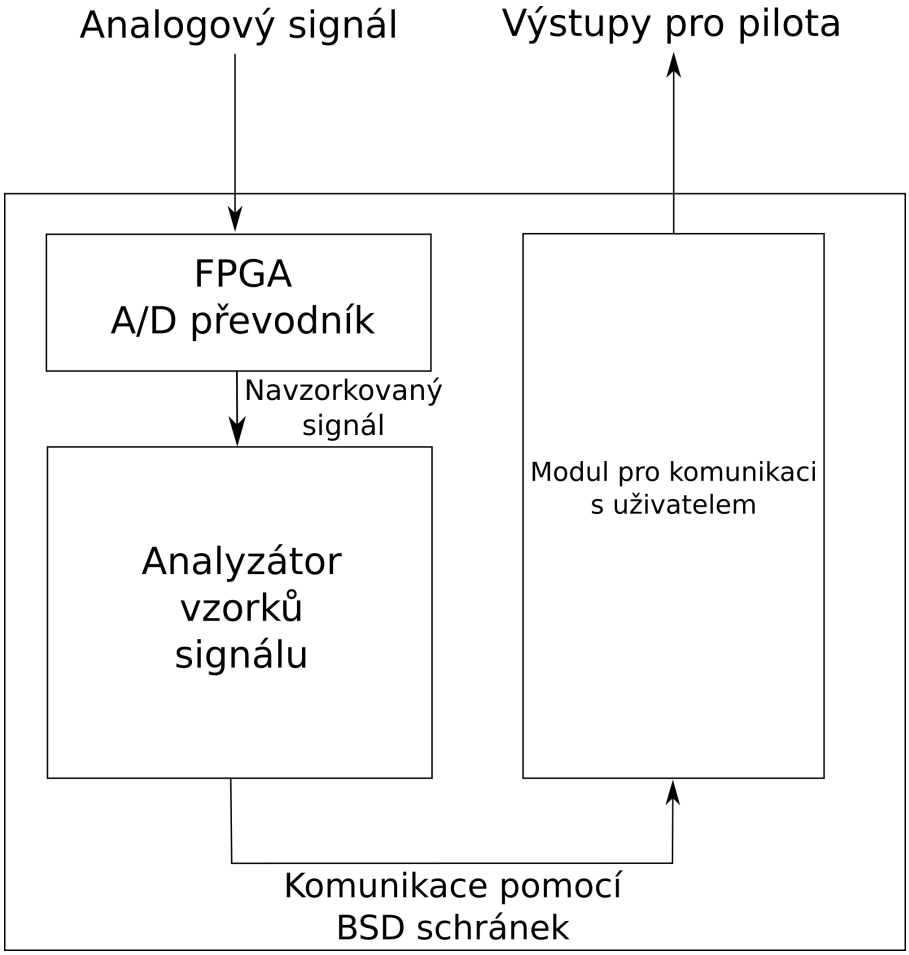
4. Výdrž - Když se pilot přibližuje k dráze, musí přitáhnout knipl, tím zvýší úhel náběhu a jak jsme si již vysvětlili výše, způsobí zpomalení stroje. Pilotovým úkolem v této fázi letu je s vypnutým motorem udržovat letadlo ve vodorovném letu těsně nad dráhou, na kterou pak díky ztrátě vztlaku dosedne. Je důležité, aby letadlo bylo těsně nad dráhou, čím výše se v okamžiku ztráty vztlaku letadlo nachází, tím tvrdší přistání může čekat. A tím větší škody mohou vzniknout.

5. Dosednutí a rolování (Touchdown and taxi) - Po dosednutí pilot zpomalí letadlo na požadovanou rychlost pro rolování a roluje na místo určené řídícím letového provozu.

Kapitola 4

Návrh řešení

Nejdříve si musíme uvědomit, co je našim cílem, analyzovat zadání a podle něj si pak navrhnout vhodné řešení. Vhodným se rozumí jednoduché, efektivní a rozšířitelné pro další využití.



Obrázek 4.1: Moduly

Kvůli omezenému prostoru naše zařízení musí být malé, kvůli zachování aktuálnosti předané informace rychlé, kvůli spotřebě elektrické energie energeticky nenáročné a kvůli podmínkám, ve kterých bude pracovat odolné vůči vnějším vlivům. Naštěstí nároky na rychlost zpracování nejsou příliš přísné. Průměrný člověk má reakční dobu přibližně sekundu, v závislosti na tréningu a požadované reakci. Zároveň ale platí, že čím rychlejší náš systém bude, tím přesnější informace budeme schopni dodávat pilotovi. Předané informace musí být ve vhodném formátu pro rychlou a bezchybnou interpretaci pilotem, aniž by působily rušivě a odváděly pozornost.

4.1 Platforma

Jak již bylo zmíněno v úvodu této kapitoly, potřebujeme malé, ale rychlé zařízení, které bude v reálném čase poskytovat údaje o aktuální výšce. V našem případě, vzhledem k reakční době člověka okolo sekundy, bude stačit, když systém stihne spočítat výsledek a předat jej pilotovi do ∼250ms.

Kvůli nárokům na velikost a spotřebu můžeme vyřadit objemné osobní počítače s architekturou x86 případně AMD64 a spíše se ponořit do oblasti vestavěných systémů. V současnosti trh nabízí dostatečně výkonné a zároveň dostatečně kompaktní zařízení, která jsou pro náš účel vhodná. Jako možný kandidát se jeví mikropočítač Raspberry Pi, který už je dostatečně výkonný a zároveň malý, aby mohl splnit požadavky na naši úlohu. Bohužel, kvůli konektivitě je tato platforma nevhodná. Jako další se nabízí Arduino. Tato platforma nabízí solidní výkon za rozumné peníze. Bohužel tento výkon nedostačuje našim potřebám a hodí se spíše do nově rozvíjejícího se oboru Internet of Things. Vhodnější alternativa k tomuto řešení se jeví procesor rodiny Zynq od výrobce Xilinx, který je určen přesně pro naše potřeby, a díky integrovaným programovatelným hradlovým polím FPGA výrazně urychluje zpracování ozvěny signálu byť už jen pouhým vzorkováním daného signálu. Přístroj s procesorem dodala společnost CAMEA s.r.o. Na tomto procesoru bude spuštěn operační systém OpenSUSE, který bude obsluhovat naši aplikaci.

4.2 Hardware

Radarový transciever, se kterým pravděpodobně budeme v této práci pracovat je IVS-948 firmy InnoSenT dodaný firmou CAMEA s.r.o. Jedná se o kontinuálně vysílající a přijímající radar s frekvenční modulací. Díky frekvenční modulaci vysílaného signálu jsme schopni změřit nejenom rychlost překážky, ale i jeho vzdálenost, což je přesně to, co pro naše účely potřebujeme.

Výstupní zařízení Náš systém musí pilotovi nějak interpretovat sesbíraná data, jinak by byl zbytečný. Způsob komunikace musí být tak detailní, aby dokázal předat informace o výšce nad zemí a zároveň dostatečně jednoduchý, aby pilot mohl data ze systému zpracovat rychle a efektivně, aniž by výstupní zařízení odvádělo pozornost od právě prováděných úkonů. Jedna z možností je použití zvukových signálů, druhá je zobrazování aktuálních informací na přístrojové desce letounu. Zde si musíme položit otázku, jaký z těchto způsobů zvolíme. Zobrazení dat na přístrojové desce nabízí možnost jejich nejpřesnější interpretace, bohužel bude odvádět pozornost pilota a zvyšuje se riziko nehody. Jako další nevýhodu autor vidí zbytečně zabraný prostor, jehož přidaná hodnota po osazení přístrojem pro zobrazovaní dat není dostatečná pro naše použití.

Vstupní zařízení Pilot musí být schopen zařízení také ovládat. Proto se musíme zamyslet nad možnými ovládacími prvky. Zařízení budeme muset nějak zapnout a kalibrovat. Máme dvě možnosti. Buď se zařízení bude zapínat a vypínat automaticky zároveň s motorem, nebo jej bude moci pilot ovládat. Autor se přiklání k možnosti ovládání pilotem, jelikož v pozdějších fázích výuky může být asistence našeho systému nežádoucí. Přesně si to určí až zákazník.

Zákazník jistě bude chtít modul také kalibrovat. Zde si musíme uvědomit, že kalibrace se může úspěšně provést pouze, když letadlo stojí na zemi v klidu. Dochází totiž k nastavení referenční hodnoty výšky zařízení nad dráhou a není žádoucí aby se kalibrace prováděla ve vzduchu. Možné následky jsme si již uváděli. Spuštění musí být jednoduché a zároveň k němu nesmí dojít náhodou. Proto se nabízí možnost krytého tlačítka. Inspirovat se můžeme u stíhacích letounů, kdy při odklopení krytky dochází k odjištění zbraní. Finální podobu si opět určí zákazník.

4.3 Software

Zde si musíme připomenout, že našim úkolem není zpracování signálu. Program pro obsluhu generování informací pro pilota musí být rychlý, spolehlivý a relativně nenáročný na výpočetní výkon.

Komunikace Moduly mezi sebou musí komunikovat, přičemž nesmí docházet ke ztrátě dat, zpoždění nebo pádu aplikace. Nejdříve se musíme zamyslet nad vhodným způsobem komunikace.

Jako první se podívejme na komunikační rozhraní CORBA(**C**ommon **O**bject **R**equest **B**roker **A**rchitecture). Toto rozhraní poskytuje dobrý komunikační kanál v případě, že je třeba komunikovat mezi procesy na jednom stroji případně mezi procesy na více strojích. Aby CORBA fungovala, potřebuje spuštěný CORBA Name Server. Bohužel tato aplikace by zabírala potřebné prostředky a nemusel by zbýt výkon pro nutné věci. Další důvod pro zamítnutí tohoto řešení je jeho robustnost nevhodná pro vestavěné systémy.

Jako další možnost se jeví využití sdílené paměti. Ta umožňuje velice rychlé předávání dat za nízkého úbytku výkonu. Toto řešení ovšem vyžaduje zabezpečovací mechanismus pro zachování konzistence. Kvůli složitosti tohoto mechanismu se od řešení pomocí sdílené paměti odkloníme a spíše se přikloníme k modelu klient-server implementovaným pomocí BSD schránek. BSD schránky poskytují již hotovou implementaci komunikace, které je dostatečně rychlá a nezabírá příliš mnoho systémových prostředků, toto je pro naše účely ideální.

Nyní se musíme rozhodnout zda použijeme spojovanou či nespojovanou službu. U spojované služby nehrozí ztráty paketů, jelikož se v případě jejich nedoručení posílají znovu. Bohužel toto si nemůžeme dovolit, jelikož tato informace již nebude aktuální. Na rozdíl od spojované služby TCP, nespojovaná služba UDP sice nezajišťuje opětovné poslání paketu, ale zaručí nám, že odesílaná data budou aktuální i v případě, že se nějaký paket ztratí. Tato ztráta nás vzhledem k počtu odeslaných paketů nijak neomezí.

Architektura modulu Vzhledem k modelu klient-server musíme zabezpečit paralelní příjímání zpráv a následné vykonání události na základě přijaté zprávy. Kvůli tomuto požadavku nemůžeme využít přímé jedno vláknové zpracování, jelikož obsluha hardware je

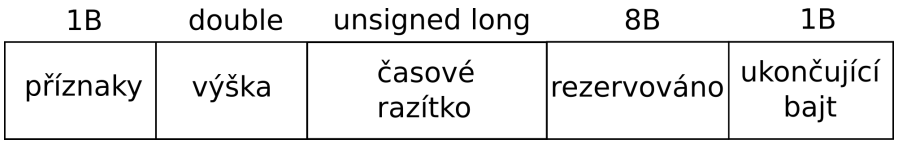
blokující a docházelo by ke zpoždění, které by akumulací starých paketů vyústilo v neaktuální až nebezpečné informace. Paket zpožděný o půl sekundy může mít katastrofální následky. Uveďme příklad. Je léto, slunné počasí a ve stínu je teplota 37 stupňů Celsia. Pilot provádí přiblížení před přistáním na betonové, případně asfaltové dráze. Vzhledem k rozdílným teplotám vzduchu nad trávou a betonem, případně asfaltem, může při příletu nad dráhu dojít k prudkému propadu letadla kvůli rozdílným hustotám teplého a studenějšího vzduchu. Tyto propady bývají rychlé a je nutné na ně rychle reagovat a nezmatkovat. Při špatné reakci může dojít k poškození stroje v tom lepším případě, a k zranění posádky a pasažérů v tom horším případě. Ani jedna možnost není žádoucí, proto potřebujeme pracovat pouze s aktuálními informacemi. V případě blokujících operací, by mohlo dojít k značnému zpoždění a tedy i škodám. Proto přistoupíme k více vláknovému řešení.

Při práci s více vlákny si musíme dávat pozor na zabezpečení komunikace a musíme zachovat konzistenci dat. Data se předávají mezi vlákny pomocí fronty. Aby bylo jasné, kdo k ní může přistupovat, musí být vhodně zabezpečena a tudíž využijeme mutexy. Jedno vlákno ve bude ve smyčce přijímat zprávy a vkládat je do fronty. Druhé vlákno je z fronty vyjme a provede náležité operace.

4.4 Komunikační protokol

Aby byly dva moduly schopny komunikovat nějakým rozumným způsobem, potřebujeme komunikační protokol. U návrhu komunikačního protokolu je třeba zjistit, co přesně potřebujeme a pozastavit se nad možnými rozšířeními do budoucna. Mohlo by se zdát, že jedinou potřebnou informací je výška letadla nad terénem. Bohužel tomu tak není. Ano, výška nad terénem je nejdůležitější přenášená informace, ale sama o sobě je nám k ničemu. S danou výškou potřebujeme přenést i časovou informaci, kdy byla výška odeslána abychom mohli zajistit práci pouze s aktuálními informacemi, jak již bylo zmíněno výše. Nyní se ovšem musíme zamyslet, jakým směrem se vývoj zařízení bude ubírat. Tato práce není pouze pro uspokojení akademiků, ale v jejím závěru a v následném vývoji má vzniknout konkurenceschopný výrobek pro trh. Proto můžeme předpokládat výrazný a rychlý vývoj. Pravděpodobně v některé následující iteraci vývoje dojde k rozšíření o informace, v jakém stavu se radar nachází. Jako dobrý krok se jeví také přidání několika bajtů pro zatím neznámé účely.

Finální verze protokolu bude obsahovat příznaky pro přenos stavových informací o velikosti jednoho bajtu a výšku nad terénem, která bude v datovém typu `double`, jeho velikost se bude lišit podle platformy, na které budou programy spuštěny. Nesmíme zapomenout na časové razítko v mikrosekundách pro identifikaci neaktuálních zpráv, rezervovaný prostor o velikosti osmi bajtů a nakonec ukončující bajt nastaven na nulu. Výsledek můžeme vidět na obrázku 4.2.

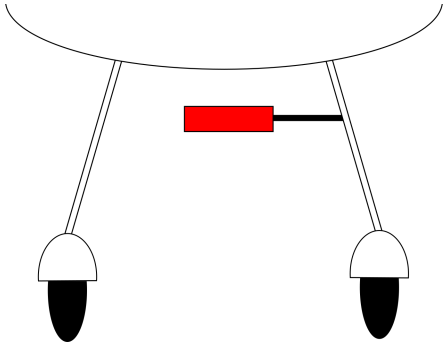


Obrázek 4.2: Komunikační protokol včetně velikostí jednotlivých částí

4.5 Umístění na letadle

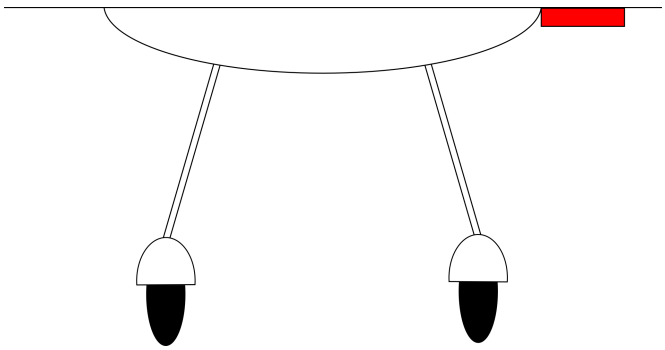
Nyní se zamysleme nad možnostmi umístění zařízení na letadle. Zařízení musí být na letadle umístěno tak, ať je šance poškození přístroje při tvrdším přistání minimální, zároveň musí být měření konzistentní a musí být chráněno i proti povětrnostním podmínkám. Toto vylučuje umístění přístroje na konci křídel. Může dojít k odření křídla o zem při přílišném naklonění, navíc bude muset být provedena velká korekce výšky a může dojít k změnám detekované výšky při náklonech letadla, taky dojde k narušení aerodynamiky křídla, kdy letadlo bude mít tendence stáčet se za křídlem, na kterém se zařízení nachází. Ideální by bylo umístit senzor na pneumatiky jednoho z kol hlavního podvozku, bohužel velice rychle zjistíme, že je to nevhodné řešení z důvodů jednorázového použití radaru. Jako další vhodné možnosti se jeví umístění:

- s odsazením na nohu podvozku za předpokladu, že se na ni zařízení vejde(v případě zatažitelného podvozku). Při umístění na pevný podvozek je třeba zamezit možnosti vzniku poškození následkem tvrdého přistání a špatného počasí.



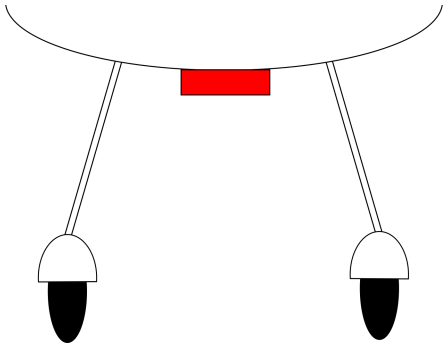
Obrázek 4.3: Montáž na podvozku

- na spodní část kořene křídla, kde nebude moment síly způsobující rotaci letadla ve vodorovné ose tak velký, jako na konci křídla. Otřesy, které bude muset zařízení snášet nebudou tak velké jako na noze hlavního podvozku. Stále je třeba myslet na ochranu proti špatnému počasí. Další nevýhoda je nutnost vrtání do trupu či křídla kvůli ukotvení. Bohužel zde zůstává nutnost korekce podle výšky kořene letadla nad rovinou podvozku.



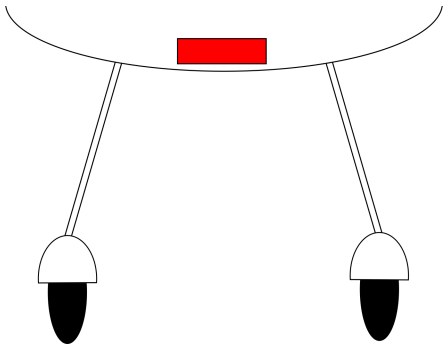
Obrázek 4.4: Montáž na spodní části křídla

- na trup letadla, toto bohužel bude vyžadovat vrtání do trupu a tudíž úpravu konstrukce letadla. Samozřejmostí pak musí být kalibrace modulu, stejně jako výše.



Obrázek 4.5: Montáž na trup letadla

- do trupu letadla, ideálně do odpružené schránky ať se zařízení nepoškodí. Zde je třeba vzít v úvahu materiál trupu letadla a jeho propustnost elektromagnetického záření případně. Zároveň je třeba zachovat možnost snadného přístupu k zařízení. S touto možností je třeba počítat již při návrhu letadla. Tudíž bude finančně nákladná.



Obrázek 4.6: Montáž do trupu letadla

Pro testování bude vhodné uchycení na nohu s odsazením, jelikož bude vyžadovat nejmenší zásahy do konstrukce letadla, bude k zařízení snadný přístup a hlavně bude nejlevnější. Naopak pro produkční modely již bude vhodné zakomponovat tento výškoměr do letadla. To už záleží na potenciálním zákazníkovi.

Kapitola 5

Implementace

V předchozí kapitole jsme si pověděli co budeme dělat. Nyní si řekneme jak to budeme dělat, co k tomu budeme potřebovat a proč to tak budeme dělat.

5.1 Systémové prerekvizity

Abychom aplikaci vůbec spustili, musí na dodané platformě běžet operační systém. Mohli bychom dlouze polemizovat nad výhodami jednotlivých linuxových distrubucí, jestli vybrat binární nebo distribuovanou pomocí zdrojových kódů či vybrat podle balíčkovacího systému. Naštěstí toto nemusíme řešit, jelikož na cílové platformě je předinstalovaný operační systém OpenSUSE. Pro naše účely zcela postačuje.

Jako další potřebujeme ovladač pro externí zvukovou kartu připojenou pomocí USB. Kvůli jednoduchosti a jednostrannosti použijeme ovladač ALSA. Ovšem aby ALSA detekovala externí zvukovou kartu, musí ji správně detekovat systém. Proto je důležité zavedení modulu snd-usb-audio do jádra operačního systému. Bez tohoto modulu jádro vidí externí zvukovou kartu pouze jako neznámé USB zařízení.

5.2 Jazyk

Abychom mohli vybrat konkrétní jazyk, musíme stanovit požadavky aplikace. Jak víme, každý jazyk má svá pro a proti.

Můžeme vybírat z několika paradigmat a z nich odvíjejících se vlastností a míry abstrakce. Také se musíme rozhodnout, jestli chceme jazyk překládaný, či interpretovaný. V našem případě potřebujeme jazyk, který umožňuje psát programy, které nespotřebovávají příliš mnoho výpočetního výkonu, poskytuje vysokou míru abstrakce a zároveň umožňuje psát nízko běžící programy. Dále musí poskytovat dobře odladěné knihovny a podporu komunity. Do tohoto rámce zapadá jen několik jazyků. Pro jazyk Java mluví jeho schopnost fungovat na mnoha platformách, vysoká úroveň abstrakce a relativně jednoduchý vývoj software v tomto jazyce. Bohužel, jeho největší přednost fungovat na mnoha platformách je v našem případě jeho největší nevýhodou. Platformu známe a virtuální stroj potřebný ke spouštění bajtkódu(bytecode) má příliš vysoké nároky na procesorový čas. Dalším vhodným kandidátem se zdá být Python. Ve prospěch Pythonu mluví možnost neuvěřitelně rychlého vývoje. Další výhodou je velké množství knihoven poskytující výbornou podporu pro vývoj. Proti mluví fakt, že se jedná o interpretovaný jazyk, který sice lze přeložit do C/C++, ale stále

nedosahuje rychlostí C/C++. Jeho další nevýhodou je Global Interpreter Lock, který omezuje paralelní zpracování dat, nemluvě o jeho vysoké spotřebě paměti, protože všechno je objekt.

Přikloňme se tedy k překládaným jazykům. Na první příčce rychlosti vykonávaného kódu sedí Assembler. Díky své nízké úrovni abstrakce a nízkoúrovňovému běhu poskytuje bezkonkurenční rychlost za předpokladu, že kód je správně napsán. Bohužel právě kvůli své nízké úrovni abstrakce je vývoj v tomto jazyce neúměrně náročný získanému výkonu. Další nevýhodou je jeho nepřenositelnost mezi platformami a v případě přechodu na novější verzi procesoru by se musely provést neúměrně velké zásahy do kódu.

Přikloňme se tedy k tzv. vysokoúrovňovému Assembleru, jazyku C/C++. Pro jazyk C/C++ svědčí jejich relativně vysoká úroveň abstrakce, relativně rychlý vývoj za zachování nízkých hardwarových nároků a podpora v podobě mnoha knihoven, modulů a frameworků. Tento jazyk mimo jiné podporuje pohodlnou implementaci vícevláknového zpracování dat, které jsme si navrhli v předchozí kapitole.

5.3 Použité knihovny

Na naprostou většinu operací nám stačí knihovny, které již jsou obsaženy v systému. Dále ale potřebujeme knihovny, které nám umožní ovládat framework ALSA a s jeho pomocí generovat zvukový signál. V tomto konkrétním případě použijeme knihovnu PortAudio[3]. PortAudio je multiplatformní otevřená knihovna pro vstupně/výstupní zvukové operace. Toto API poskytuje možnosti asynchronního volání funkcí a další nástroje pro generování zvukového signálu. Knihovna je implementována v jazyce C a je možno ji použít i v jazyce C++. Také na ní existuje nástavba v jazyce Python. Ten ale v této práci nevyužíváme.

5.4 Vícevláknová obsluha požadavků

Kvůli vícevláknové implementaci, kdy jedno vlákno zprávy přijímá a předává druhému, které na základě těchto zpráv generuje výstup, se musíme zamyslet nad zabezpečením mezivláknové komunikace.

U přenosu dat mezi vlákny je důležité zachovat jejich konzistenci a zabránit nesynchronizovanému přístupu ke sdíleným zdrojům. V našem případě je tím zdrojem fronta sdílená mezi dvěma vlákny.

Systém Linux poskytuje několik způsobů zabezpečení paralelních operací. Pro naše použití bude bohatě postačovat binární semafor mutex. Aby se autor nemusel zaobírat řešením synchronizace při vstupu do kritické sekce, implementoval si vlastní vláknově bezpečnou frontu(FIFO) a zásobník(FILO). Fronta a zásobník jsou implementovány pomocí šablonových tříd umístěných ve jmenném prostoru `threadSafety` v souboru `threadSafety.h`. Implementace pomocí šablon nám umožňuje vytvoření instancí pro jakékoliv jednoduché či složené datové typy a tím zaručuje obecnost použití. Rozhraní tříd je jednoduché a neobsahuje zbytečnosti. Vnitřní implementace tříd obsahuje frontu, do které ukládáme data, mutex, kterým zabezpečujeme kritickou sekci, `std::condition_variable` pro detekci vložení prvku do fronty, atomický příznak(`std::atomic`) datového typu `bool` pro signalizaci prázdné fronty. Pro obsluhu mutexu se využívá vymoženost standardu C++11 `std::unique_lock`, který při vytvoření v konstruktoru správně inicializuje a uzamkne mutex, a při zrušení proměnné daný mutex ve svém destruktoru správně uvolní. Mezi frontou a zásobníkem jsou dva rozdíly. Prvním rozdílem je využití fronty s dvojitým koncem pro implementaci zásob-

níku a druhým je modifikace rozhraní nahrazením veřejné položky `T &front()` položkou `T &top()`. Z pohledu uživatele těchto tříd je jejich chování stejné. Obě položky vrací referenci na prvek, který bude po zavolání metody `T pop()` z objektu vyjmut jako první a bude zkopírován do uživatelem zvolené proměnné.

Po zajištění konzistence přístupu ke sdílenému zdroji musíme zajistit zpracování pouze aktuálních informací. Toto je zajištěno tříděním zpráv podle časového razítka přijímaného z modulu zpracovávající radarová data. Komunikační modul si vygeneruje své časové razítko, porovná jej s přijatým razítkem a za předpokladu, že je v povoleném limitu 150 milisekund generuje zvuk. V případě, že není, je zpráva zahozena a na řadu přichází aktuálnější.

5.5 Princip generování zvuku

Zvuk je generován pomocí API poskytovaným knihovnou PortAudio. Nejdříve si povíme o obecném principu fungování tohoto API a následně se zaměříme na konkrétní implementaci v této práci.

Nejdříve musí dojít i inicializaci. Pro to potřebujeme získat informace o výstupních zařízeních a otevřít zvukový kanál. Při otevírání dáváme API k dispozici adresu statické funkce, která bude asynchronně volána během běhu programu(callback). Tato funkce generuje signál, který je dále posílán do ALSA API. Programátor tak je odstíněn od nízkoúrovňového programování. Komunikace mezi programátorem callbackem probíhá pomocí statické struktury definované jako globální proměnná. Jelikož je globální, hodnoty ve struktuře jsou překladačem inicializovány na nuly. Díky tomu program začíná generovat zvuk až po nastavení frekvence ve vlákně generující výstup.

Nyní nám vyvstává problém generování správného signálu. Po kontrole konzistence přijatých dat generujeme obdélníkový signál. Zvukový signál musí indikovat vertikální přibližování k dráze a tudíž se pauza mezi tóny a délka tónu musí vhodně zkracovat. Musíme také počítat s konfigurací, kdy každé letadlo má výšku výdrže jinou. Z toho důvodu musíme použít vhodný vztah pro výpočet. Experimentálně bylo zjištěno, že koeficient pro násobení aktuální výšky nad dráhou:

$$K \ = \ 25000 \ / \ h$$

je vhodný pro výšky výdrže pod 3 metry a nad tři metry je vhodný výraz níže.

$$K \ = \ 25000 \ / \ (h \ / \ 2)$$

K značí koeficient a h značí výšku výdrže daného letadla.

5.6 Konfigurace a spuštění

Konfigurace probíhá modifikací konfiguračního souboru `/opt/cfg/radar.conf`. Vzor můžeme vidět na obrázku 5.1. Položka označující typ letadla je určen pro budoucí použití v rámci rozšíření práce a v současné době nemá na běh programu žádný vliv. Další položky jej ovšem ovlivňují výrazně. Položka `FLARE_INIT_HEIGHT` definuje výšku nad dráhou, kdy má pilot začít přitahovat. Pod touto výškou program generuje vyšší frekvenci než v předchozích fázích přistání. Položka `FLARE_HEIGHT` označuje výšku výdrže, ve které má pilot držet letadlo do doby než ztratí vztlak a dosedne na dráhu. Poslední položka `CTRL_PORT` označuje port, na kterém bude program generující zvuk přijímat zprávy.

Spouštění všech programových modulů probíhá automaticky ihned po startu. Cesta ke spustitelným binárním souborům je uvedena ve skriptu `/root/boot/launcher.sh`, který se spouští ihned po startu a startuje jednotlivé služby.

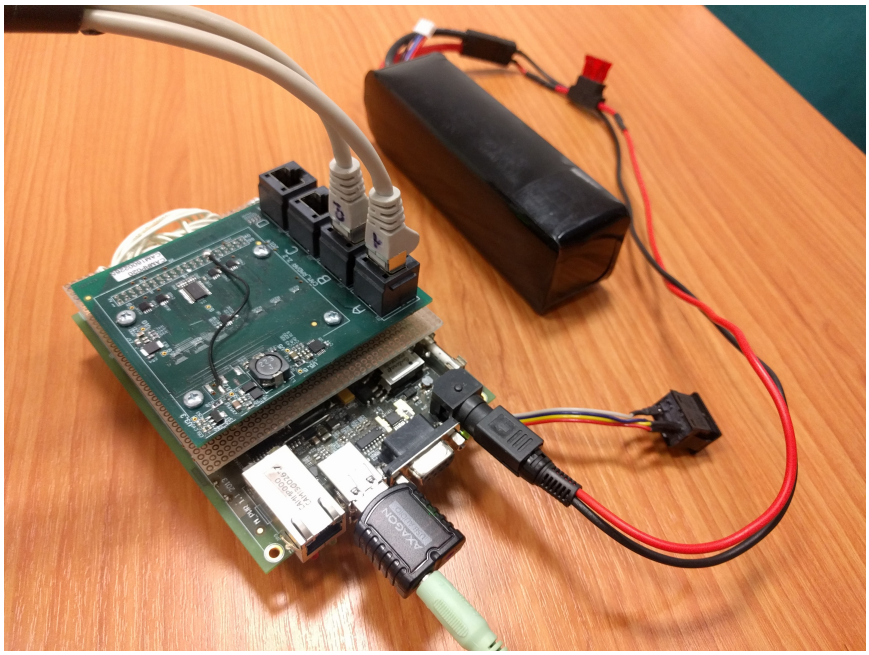
```
1 #Typ letadla
2 PLANE_TYPE=A320
3 #Vyska zahajeni flare [m]
4 FLARE_INIT_HEIGHT=50
5 #Vyska flare pro dosednuti [m]
6 FLARE_HEIGHT=10
7 #port pro komunikaci s aplikaci
8 CTRL_PORT=1337
```

Obrázek 5.1: ukázka konfiguračního souboru

5.7 Testování

Pro testování funkčnosti jednotlivých modulů autor napsal testovací aplikaci, která iterativně vyplňuje pakety potřebnými daty a ty následně odesílá modulu pro generování zvuku.

Nejdříve autor otestoval komunikaci mezi těmito dvěma moduly ručním spuštěním modulů na cílové platformě, test proběhl v pořádku s očekávanými výsledky. Následně autor otestoval automatické spouštění po nastartování zařízení a spuštění operačního systému. I tento test se zdařil a ihned po spuštění operačního systému začalo zařízení generovat zvuk. Fotografie z testování může čtenář vidět na obrázcích 5.2 a 5.3.



Obrázek 5.2: Detail zařízení s externí zvukovou kartou

Obrázek 5.3: Zařízení se zapojeným radarem

Tyto testy ověřily celkovou funkčnost modulu pro generování zvuku, včetně kontroly a zahazování neaktuálních zpráv. Ovšem při překladu je třeba dávat pozor na správné nastavení systémového data, jinak nejdou zdrojové soubory přeložit. Tímto je zařízení připraveno pro zavedení modulu zpracovávající radarový signál a cíl naší práce byl splněn. Další testování bude třeba provést po rozšíření této práce zavedením dříve zmíněného modulu nebo přesunutím modulu zpracovávající radarová data do FPGA.

Kapitola 6

Závěr

V úvodu této práce jsme si vytyčili jisté cíle. Čtenář se seznámil se stručnou historií měření výšky a byl relativně detailně uveden do této problematiky. Vysvětlili jsme si významy pojmů QHN, AMSL a transition level. Dále byla čtenáři vysvětlena nepřesnost barometrických výškoměrů, která byla zdůvodněna a čtenáři bylo vysvětleno, proč chyba měření výšky pomocí aneroidu nezpůsobí srážku letadel ve vzduchu. Čtenáři také byl stručně přiblížen vývoj radaru a byly popsány použité technologie v našem řešení. Následně byl čtenář stručně uveden do problematiky přistání.

Naším hlavním cílem ovšem byl návrh systému, který by usnadnil výuku přistání začínajícím pilotům a zvýšil její bezpečnost. Tento cíl byl splněn. V této kapitole jsme se zamysleli nad vhodnými platformami. Dále jsme se zamysleli nad podobou vstupního a výstupního hardware s analýzou dopadů na pozornost pilota, důrazem na jednoduchost ovládaní. Následně jsme si ukázali návrh komunikace mezi jednotlivými částmi se stručnou analýzou možných technologií pro implementaci komunikace. V té stejné sekci jsme si navrhli možnou architekturu programového vybavení. Také jsme vytvořili komunikační protokol. Na konci této kapitoly jsme se zamysleli nad umístěním radaru na letadle.

Po návrhu celého systému jsme přikročili k řešení implementace. Nejdříve jsme si vyjmenovali potřebné systémové prerekvizity pro zajištění správné funkcionality programového vybavení. Dále jsme se analyzovali jednotlivé jazyky pro implementaci navrženého systému. Samozřejmostí je výběr vhodných knihoven a zdůvodnění daného výběru. Popsali jsme si implementaci vláknově bezpečné fronty pomocí šablon. Taky jsme si ukázali kontrolu aktuálnosti přijatých dat ke zpracování. Následně jsme si vysvětlili princip generování zvuku. Samozřejmostí je i popis konfigurace zařízení pro různé modely letadel, spuštění systému a na závěr jsme si vysvětlili postup testování systému.

Autor oceňuje, že díky této práci získal důležité zkušenosti týkající se návrhu kritických systémů běžícím v reálném čase. Dále oceňuje zkušenosti získané návrhem zařízení jako celku skládajícího se z jednotlivých modulů, které mezi sebou musí bezchybně a rychle komunikovat. Další důležitá zkušenost je kladení důrazu na výběr lidí pro spolupráci, konkrétně na jejich spolehlivost a důslednost.

V práci je možno pokračovat doplněním algoritmu pro zpracování radarového signálu, čímž se tento systém stane plně funkčním. Dále bude záležet na potenciálním zákazníkovi, jak se postaví k ovládacím prvkům hardware a jakou implementaci bude požadovat. Také může dojít k přesunutí algoritmu pro zpracování radarového signálu na FPGA.

Literatura

[1] *Flying techniques*. [Online; navštíveno 25.01.2017].
URL http://www.aeroskytech.com/english/final/final.html

[2] *Great Circle Mapper*. [Online; navštíveno 22.4.2017].
URL http://www.gcmap.com/airport/BPX

[3] *PortAudio - Portable Cross-platform Audio I/O*. [Online; navštíveno 15.05.2017].
URL http://www.portaudio.com/

[4] *Zynq-7000 All Programmable SoC Overview* . [Online; navštíveno 15.05.2017].
URL https://www.xilinx.com/support/documentation/data_sheets/ds190-Zynq-7000-Overview.pdf

[5] Belville, J. H.: *Mid-nineteenth century barometers: being a reprint of Manual of the mercurial and aneroid barometers (1858) by John Henry Belville, to which is appended The aneroid barometer, how to buy and use it (1849) by a Fellow of the Meteorological Society*. Turner and Devereux, 1975, ISBN 0950255742.

[6] ICAO: *Aircraft Operations - Doc 8168*. International Civil Aviation Organisation, 2006.

[7] Schwarz, J.; Růžička, R.; Strnadel, J.: *Mikroprocesorové a vestavěné systémy IMP, Studijní opora* . VUT v Brně, leden 2006.

[8] Skolnik, M.: *Radar Handbook*. McGraw-Hill Education, 2008, ISBN 978-0071485470.

[9] Watson, R. C.: *Radar Origins Worldwide*. Trafford Publishing, 2009, ISBN 1426921101.

[10] Wolff, C.: *Classification of Radar systems*. [Online; navštíveno 24.01.2017].
URL http://www.radartutorial.eu/02.basics/Classification%20of%20Radar%20systems%20(1).en.html

Přílohy

Příloha A

Obsah přiloženého paměťového média

- `src` - složka se zdrojovými kódy pro překlad spustitelných binárních souborů
- `text` - složka se zdrojovou podobou textu bakalářské práce
- `README` - manuál pro instalaci a spuštění