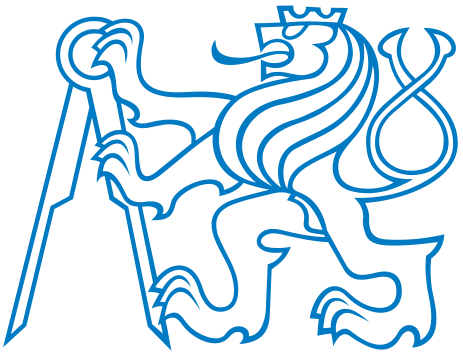


ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

Fakulta elektrotechnická

Katedra mikroelektroniky

Modul rozhraní mezi leteckým simulátorem a ovládacím panelem

Bakalářská práce

Autor práce:	Andrey Nosov
Vedoucí práce:	Ing. Tomáš Teplý
Studijní program:	Elektronika a Komunikace

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Nosov** Jméno: **Andrey** Osobní číslo: **495639**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra mikroelektroniky**

Studijní program: **Elektronika a komunikace**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Modul rozhraní mezi leteckým simulátorem a ovládacím panelem

Název bakalářské práce anglicky:

Flight Simulator Interface Module

Pokyny pro vypracování:

Navrhněte a realizujte elektronický modul, který bude zprostředkovávat přenos signálů mezi počítačovou aplikací leteckého simulátoru a ovládacím panelem pro malá a středně velká letadla. Pro komunikaci s počítačem použijte standardní rozhraní.
Postup řešení:
1) Prostudujte nabídku aktuálně komerčně dostupných rozhraní pro připojení ovládacích panelů k PC a simulátorů ovládacích panelů pro malá a středně velká letadla, zhodnoťte jejich vlastnosti. Na základu typických ovládacích prvků ultralehkých a lehkých vrtulových letounů a snímačů/indikátorů používaných pro jejích osazení definujte vlastnosti, které by mělo splňovat vámi realizované rozhraní.
2) Proveďte návrh vlastního řešení, které bude splňovat požadavky z předchozího bodu.
3) Zařízení realizujte.
4) Otestujte funkčnost vašeho řešení.
5) Shrňte dosažené výsledky a uveďte srovnání s komerčními výrobky.

Seznam doporučené literatury:

[1] Developing Plugins [online]. X-Plane Developer. February 2019. Dostupné z https://developer.x-plane.com/article/developing-plugins/
[2] Yiu, J.: The Definitive Guide to ARM Cortex-M3 and Cortex-M4 Processors Third Edition, Elsevier, 2014
[3] Záhlava, V.: Návrh a konstrukce desek plošných spojů, BEN, Praha, 2011

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Tomáš Teplý katedra mikroelektroniky FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **26.01.2022** Termín odevzdání bakalářské práce: **20.05.2022**

Platnost zadání bakalářské práce: **30.09.2023**

Ing. Tomáš Teplý podpis vedoucí(ho) práce	prof. Ing. Pavel Hazdra, CSc. podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

. Datum převzetí zadání

Podpis studenta

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem č. 1/2009 o etické přípravě vysokoškolských závěrečných prací.

V Praze, 20. května 2022

Poděkování

Děkuji vedoucímu práce Ing. Tomáši Teplému za cenné rady a konstruktivní kritiku. Také děkuji pracovníkům společnosti Elsaco, především Ing. Jindřichovi Francovi a Ing. Michalovi Grunclovi, za jejich pomoc a odborné konzultace. Velké poděkování patří mé rodině a mým přátelům za podporu, kterou mi poskytli.

Abstrakt

Tato práce je zaměřena na návrh modulu rozhraní mezi leteckým simulátorem a ovládacím panelem letadla, složeného z vlastnoručně vyrobených ovládacích prvků ultralehkých a lehkých letounů. V teoretické části se práce věnuje popisu ovládacích prvků a současně dostupným možnostem jejich připojení k simulátorům. Jsou stanoveny požadované parametry modulu a popsána komunikační cesta mezi ovládacími prvky a simulátorem. Podrobněji je popsána komunikace prostřednictvím rozhraní USB, zvoleného pro propojení modulu a počítače. Praktická část práce se věnuje vyzkoušení funkčnosti rozhraní na prototypu a následné realizaci výsledného zařízení.

Klíčová slova: komunikační rozhraní, letecký simulátor, USB HID, I/O modul

Abstract

This work focuses on the design of an interface module connecting a flight simulator and a control panel of an airplane. The control panel is composed out of hand-made control elements for ultra-light and light aircraft. The theoretical part concerns itself with the description of the control elements as well as the possible options of connecting them to the simulator. Required parameters are stated and the communications route between the control elements and the simulator is described. The communication through the USB interface, which was chosen for connecting the module and the computer, is described in greater detail. The practical part covers testing of the functionality of the interface on a prototype and the following implementation of the final device.

Key words: communication interface, flight simulator, USB HID, I/O module

Obsah

Obrázky

Tabulky

Seznam použitých zkratek

I/O	–	Input / output
UDP	–	user datagram protocol
LED	–	light-emitting diode
USB	–	universal serial bus
LCD	–	liquid-crystal display
PWM	–	pulse-width modulation
PC	–	personal computer
UART	–	universal asynchronous receiver-transmitter
HID	–	human interface device
XPLM	–	X-Plane plugin manager
DLL	–	dynamic-link library
SPI	–	serial peripheral interface
DPS	–	deska plošných spojů

1 Úvod

Tato práce se zabývá návrhem elektronického I/O-modulu pro přenos dat mezi počítačovou aplikací leteckého simulátoru a ovládacím panelem pro ultralehké a lehké vrtulové letouny s použitím standardního komunikačního rozhraní.

I/O-modul je určený pro uživatele leteckých simulátorů, především profesionálního simulátoru X-Plane 11, kteří si chtějí vyzkoušet ovládání ultralehkých a lehkých vrtulových letounů, zároveň ale dávají přednost vlastnoručně vyrobeným ovládacím prvkům před hotovými výrobky. Modul bude sloužit pro připojení základních ovládacích prvků letadla.

Cílem práce je navrhnout jediné zařízení, řízené mikrokontrolérem, které bude ovládat panel s ovládacími prvky a provádět komunikaci s počítačem, následně zařízení realizovat a otestovat jeho funkčnost.

V první části se práce věnuje teoretickému popisu celého projektu. Nejprve je uvedena motivace pro zpracování daného projektu. Dále jsou popsány a zhodnoceny možnosti, které jsou momentálně dostupné pro připojení vlastnoručně vyrobených ovládacích prvků k leteckým simulátorům. Následně jsou popsány typické ovládací prvky letounů a jejich osazovací členy, které si bude moci uživatel k simulátoru připojit. Na základě těchto prvků a plánované funkcionality modulu jsou definovány požadavky na platformu, na jejíž osnově se bude zařízení stavět. Nakonec je uveden popis komunikační cesty mezi modulem a simulátorem.

Druhá část popisuje praktickou realizaci projektu. Nejprve je popsán návrh prototypu zařízení. Začíná zprovozněním komunikace mezi modulem a počítačem. Dále následují popis hlavního programu řídicího I/O-modul a způsoby připojení jednotlivých ovládacích prvků. K tomu je uveden popis plug-inu, zprovozňujícího komunikaci na straně leteckého simulátoru. Po vyhodnocení prototypu následují popis změn provedených ve výsledném zařízení a testování jeho funkčnosti. Na závěr je výsledný I/O-modul porovnán s konkurenčními výrobky.

2 Motivace

Téma práce vzniklo z projektu, který vede můj otec ve spolupráci se společností Elsaco. Jedná se o fyzický simulátor letadla typu Cessna, umístěný na pohyblivé platformě, určené pro simulaci přetížení při letu. Na platformě jsou umístěny křeslo pilota, palubní deska s ovládacími prvky a monitor, na který je vyveden obraz z leteckého simulátoru X-Plane 11. Současný prototyp je vybaven ovládacími prvky od různých výrobců, ale časem se má část těchto hotových prvků vyměnit za vlastnoručně vyrobené. Pro jejich připojení k simulátoru se využívá komunikačního modulu GSA-055 a I/O-modulu GSA-010 firmy FlightIllusion. Dají se využít jak pro připojení výrobků firmy Flightillsuion, tak i vlastnoručně vyrobených ovládacích prvků. Tato zařízení dávají mnohem větší možnosti, než panel letadla typu Cessna vyžaduje. I/O-modul GSA-010 obsahuje 48 digitálních vstupů, 34 výstupů a 8 vstupů s analogově-digitálním převodníkem s možností rozšíření [1]. Komunikační modul GSA-055 umožňuje připojení až 64 takových I/O-modulů [2]. To se odehrává na ceně zařízení: oba moduly se dohromady prodávají za 450 eur.

Při vyhledávání levnějších alternativ těchto dvou modulů se nepovedlo nalézt zařízení s podobnou funkcionalitou a menším počtem vstupů a výstupů, které by vystačilo pro připojení ovládacích prvků menšího letadla. Tak vznikl nápad vytvořit vlastní zařízení pro jejich propojení se simulátorem. Mimo cenu byly významnými důvody k vytvoření vlastního zařízení možnosti uplatnit znalosti nabyté během studia a získat zkušenost s vytvořením hotového produktu.

Následně se téma projektu zobecnilo na návrh zařízení určeného pro připojení ovládacích prvků neurčitých ultralehkých a lehkých vrtulových letounů, tedy nejen letadel typu Cessna.

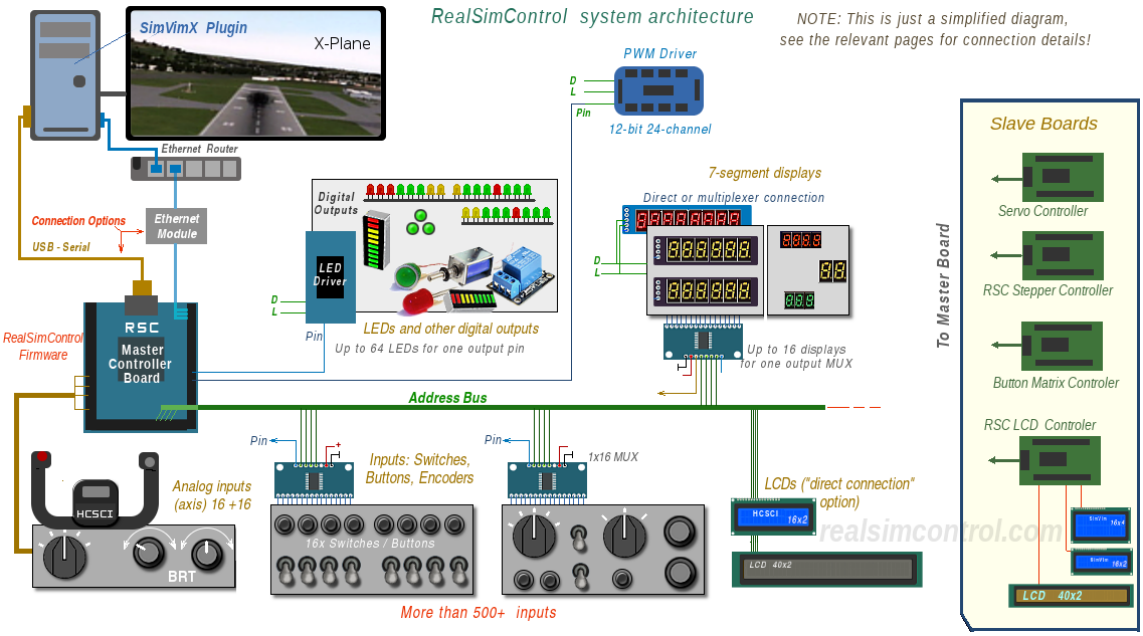
Obrázek 1: Letecký simulátor na pohyblivé platformě

3 Současné možnosti připojení

V současnosti existuje několik možností pro připojení vlastnoručně vyrobených ovládacích prvků k leteckým simulátorům.

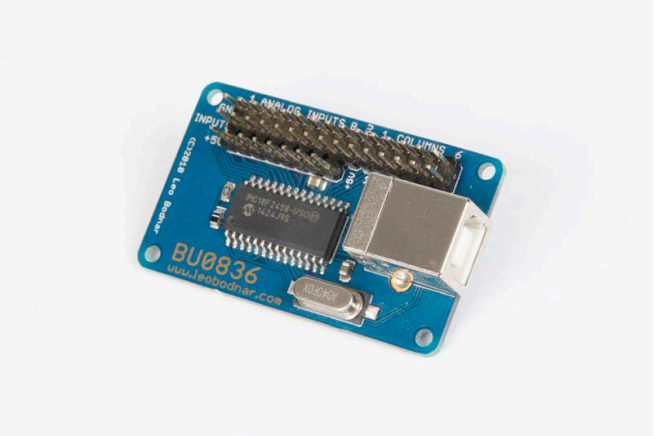
První možností pro člověka, který se rozhodne pro výrobu vlastních ovládacích prvků, je vytvoření i vlastního zařízení pro komunikaci prvků s počítačem. Taková možnost je nejflexibilnější, vyžaduje ale určité znalosti z oblasti elektroniky, sériové komunikace i programování a pro běžného uživatele je spíše nedostupná.

Pro usnadnění vlastnoručního návrhu byly vytvořeny otevřené platformy. Nejlepším příkladem takové platformy je RealSimControl. Umožňuje uživateli propojení jeho ovládacích prvků a simulátoru X-Plane prostřednictvím Arduino-desek. Podle webových stránek projektu [3] tento přístup nevyžaduje po uživateli žádné znalosti v oblasti programování a nabízí velkou flexibilitu v počtu a typech připojených zařízení. Nicméně z hlediska elektronického zapojení může být takové řešení pro běžného uživatele stále poměrně komplikované. Hlavní nevýhodou této platformy je ale omezení jedním jediným simulátorem X-Plane, se kterým bude moci zařízení komunikovat. Tento projekt totiž provádí komunikaci s počítačem prostřednictvím rozhraní Ethernet a protokolu UDP [3]. Data se vysílají ve speciálním formátu, určeném pro komunikaci pouze s aplikací X-Plane. Jedná se sice o jeden z nejlepších současně dostupných simulátorů, ale rozhodně nejde o jedinou možnost pro realistické lety v domácím prostředí. V případě, že by si uživatel chtěl vyzkoušet let v jiné aplikaci, stane se jeho panel plně nepoužitelný.



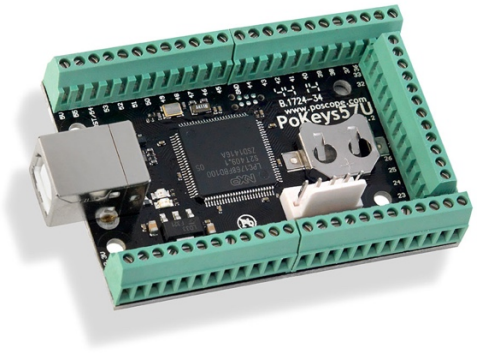
Obrázek 2: Schematické zapojení platformy RealSimControl [3]

Další možností je použití cenově dostupných I/O-modulů bez podpory aktuátorů [4, 5]. Tato možnost bývá velmi často doporučována na leteckých fórech. Jedná se o zařízení s USB-výstupem, mající jistý počet digitálních a analogových vstupů pro připojení ovládacích prvků. Pro počítačové aplikace se takové zařízení jeví jako herní ovládač. Uživatel si tak může samostatně vevnitř aplikace simulátoru připojit různé funkce letadla ke konkrétním digitálním a analogovým vstupům na desce. Nevýhodou těchto zařízení je absence výstupů pro ovládání aktuátorů, které bývají velmi často součástí ovládacích prvků (např. LED-indikátory a ukazatele řízené elektrickými motory).



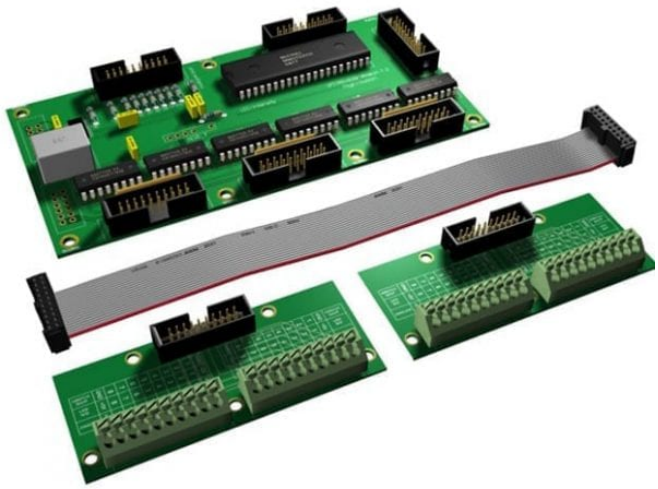
Obrázek 3: I/O modul BU0836 [4]

Určitou podporu indikačních prvků má například zařízení ProKeys57U [6]. Pro počítač se také jeví jako herní ovládač, navíc ale výrobce nabízí speciální plug-in pro simulátor X-Plane, který nabízí lepší integraci s touto aplikací. Modul obsahuje sedm vstupů s digitálně-analogovými převodníky a 55 pinů pro digitální vstupy i výstupy. Díky plug-inu lze tyto výstupy využít pro připojení jednotlivých LED, alfanumerických LCD a sedmisegmentových LED-displejů [7]. Nicméně přestože by toho bylo zařízení schopno, daný plug-in neumožňuje připojení elektrických motorů.

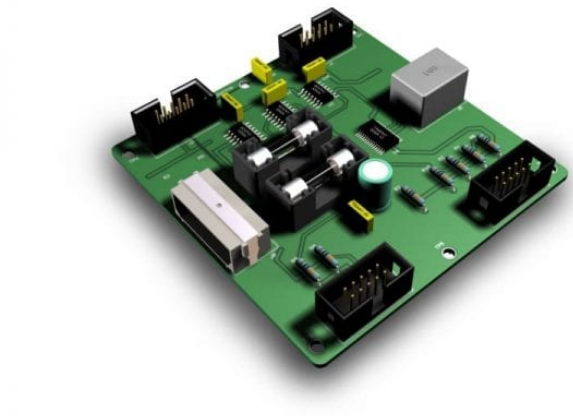


Obrázek 4: ProKeys57U [6]

Pokročilé ovládání indikátorů, včetně motorů, je realizováno v profesionálních zařízeních firmy FlightIllusion GSA-055 a GSA-010 [1, 2], zmíněných v kapitole o motivaci tohoto projektu. Podporují veškerou funkcionalitu k sestavení vlastního ovládacího panelu, nejsou omezeny konkrétním simulátorem a připojují se přes standardní USB-rozhraní. Jejich hlavní nevýhodou je ale příliš vysoká cena (450 eur za obě zařízení).



Obrázek 5: I/O modul GSA-010 [1]



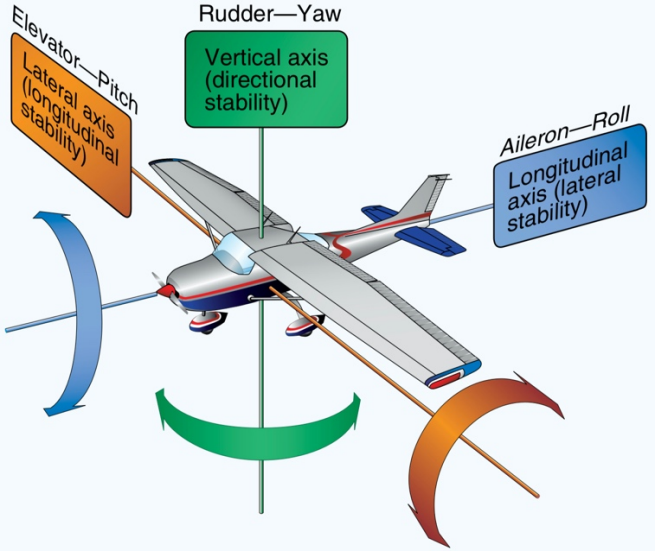
Obrázek 6:Komunikační rozhraní GSA-055 [2]

4 Ovládací prvky letounů

V této kapitole budou vyjmenovány základní ovládací prvky, které se využívají pro ovládání ultralehkých a lehkých letounů. Zaměříme se na funkce těchto prvků, na elektrické komponenty, jejichž pomocí se dají při vlastnoruční výrobě tyto prvky osadit, a na elektrické signály, jež bude třeba pro práci s těmito prvky zpracovávat. Souhrnný, maximální počet stejných prvků, které se mohou současně objevit na jednom panelu, je dále uveden v tabulce 1.

4.1 Primární řízení letounů

Primární řídicí prvky letounů jsou určeny pro změnu jeho polohy ve třech základních osách: podélném sklonu (pitch), příčném náklonu (roll) a kurzu (yaw) [8]. Za polohu letadla v těchto osách odpovídají dva ovládací prvky: řídící páka (nebo berany) a pedály.



Obrázek 7: Vizualizace základních polohových os letounu [8]

4.1.1 Řídící páka/Berany

Daný ovládací prvek má dvě osy pohybu: k sobě / od sebe a pohyb do stran. Těmito pohyby se mění podélný sklon a příčný náklon letadla [9]. V závislosti na typu letadla se může ovládat jak jednou rukou (v tomto případě se jedná o řídící páku) tak i oběma (takovému prvku se říká berany) [10].

V případě leteckého simulátoru je pro nás podstatná absolutní poloha řídící páky na obou osách. Při vlastnoruční výrobě beranů se pro určení polohy nejčastěji využívají dva jednootáčkové potenciometry [11, 12]. Ty generují lineární napěťový signál úměrný absolutní poloze řídící páky podél os. Pro osazení řídící páky se může využít i Hallových senzorů a pevných magnetů [13]. Hallovy senzory generují lineární napěťový signál úměrný magnetické indukci, která se mění při pohybu řídící páky.

4.1.2 Pedály

Tímto ovládacím prvkem se mění kurz letounu, a to jak ve vzduchu, tak i na zemi. Pedály mají jednu až tři osy pohybu. Základní osa umožňuje sešlápnutí jednoho z pedálů, jež určí směr, do kterého se má letoun pohybovat [9]. Dvě další možné osy jsou realizovány pouze na některých typech letadel. Sešlápnutím pedálů špičkami nohou se pedály nakloní k zemi a aktivují brzdy na kolech [10].

Opět je podstatná absolutní poloha ve všech třech osách. Jelikož je pohyb pedálů relativně malý, používají se pro detekci jejich polohy opět nejen jednootáčkové potenciometry, ale i Hallovy senzory v kombinaci s pevnými magnety [14, 15]. Výstupem takového prvku budou jeden až tři napěťové signály, úměrné absolutní poloze pedálů.

4.2 Sekundární řízení letounů

Za sekundární se považují ovládací prvky usnadňující řízení letounu [8]. Část těchto prvků zajišťuje jeho stabilizaci ve třech základních osách, která je realizována pomocí vyvažovacích prvků (trimů). Ze tří os se převážně realizuje vyvážení podélného sklonu. Druhá část mění vzdušný odpor a vztlak letounu, které se ovládají pomocí klapek, slotů a spoilerů. Poslední dva prvky se v menších letadlech téměř neobjevují, a proto se jimi v rámci daného projektu zabývat nebudeme.

4.2.1 Vyvažovací kolo

Vyvažovací kolo (v praxi a literatuře je častěji používán název trimovací kolo) slouží k vyvážení letadla v ose podélného sklonu. Jedná se o kolo, které může pilot otáčet do obou směrů (nahoru nebo dolů) a takto měnit polohu špičky letadla vůči vlastní pozici. Při otáčení kola nahoru se pohybuje špička letadla dolů a opačně při změně směru otáčení. Slouží k usnadnění ovládání letadla, aby pilot nemusel po dobu celého letu tahat za řídící páku, ale mohl jeho pozici uvést do neutrální polohy [9]. Při zapnutí autopilota se vyvažovací kolo ovládá počítačem a točí se samostatně.

V tomto případě není zásadní absolutní poloha kola, ale relativní změna jeho polohy. Při vlastnoruční realizaci se pro jeho osazení nejčastěji využívá inkrementálních rotačních enkodérů a víceotáčkových potenciometrů [16, 17, 18]. Pro simulaci autopilota a otočení kola se využívá unipolárního nebo bipolárního krokového motoru. Navíc je u daného ovládacího prvku často realizován ukazatel indikující míru vyvážení. Ten bývá často realizován pomocí modelářského servomotoru [16, 18]. Výstupem takového kola mohou být jak lineární napěťový signál z víceotáčkového potenciometru, úměrný poloze kola, tak i dva číslicové signály z rotačního enkodéru, udávající směr a rychlost pohybu kola. Servomotor se řídí PWM-signálem (signálem s pulzně šířkovou modulací) a pro ovládání krokového motoru bude třeba motorového ovládače.

4.2.2 Vyvážení příčného náklonu a kurzu

Vyvážení příčného náklonu a kurzu jsou mnohem méně časté než vyvážení podélného sklonu. Pro vyrobení vlastnoručních prvků určených k vyvážení letounu v těchto osách se nepodařilo najít žádné přehledné návody. Ani v reálných lehkých a ultralehkých letounech nejsou podobné ovládací prvky příliš časté.

Nicméně povedlo se najít hotový panel, který obsahuje prvek pro vyvážení kurzu [19]. Je – stejně jako je tomu v případě podélného sklonu – realizován pomocí vyvažovacího kola, akorát umístěného horizontálně. Proto budeme předpokládat, že pokud by se uživatel rozhodl pro realizaci prvku určeného k vyvážení kurzu na svém ovládacím panelu, pak opět využije vyvažovacího kola.

4.2.3 Ovládání klapek

Klapky slouží ke zpomalení letounu a zvýšení vztlaku na křídlech [8]. Nejčastěji se využívají při vzletech a přistáních. Ve většině případů se jedná o páčku, která může být nastavena do několika poloh (typicky čtyř), kde každá poloha indikuje míru vysunutí klapek. Vedle páčky bývá umístěn ukazatel, udávající reálnou polohu klapek.

Pro zjištění polohy páčky se využívá jednootáčkových potenciometrů a vícepolohových přepínačů [20, 21]. Ukazatel se – stejně jako v případě vyvažovacího kola – ovládá pomocí modelářského servomotoru [20]. V případě potenciometru dostáváme na výstup lineární změnu napětí v závislosti na změně polohy páčky. V případě přepínače je to několik digitálních výstupů indikujících polohu přepínače. Ukazatel se bude ovládat PWM-signálem.

4.3 Ostatní ovládací prvky

Dosud jsme se věnovali prvkům zajišťujícím změnu polohy letounu. Nicméně pro plnou kontrolu nad letadlem si s nimi nevystačíme. Pilot musí mít možnost ovládání motoru, podvozku, polohy vrtule a využití paliva.

4.3.1 Plynový kvadrant

Jedná se o jednotku, která je složena ze tří pák – plynové přípusti, nastavení vrtule a směsi. Páka přípusti reguluje množství vzduchu přiváděného do motoru, páka nastavení vrtule ovládá úhel náběhu vrtule a směs reguluje množství paliva, přidávaného do motoru [22].

Stejně jako u většiny ostatních prvků, i v tomto případě je důležitá absolutní poloha všech tří pák. Realizují se pomocí potenciometrů [23, 24], které lineárně mění výstupní napětí v závislosti na posuvu jednotlivých pák.

4.3.2 Ovládání podvozku

Představuje velmi jednoduchý prvek ve tvaru přepínače stavu podvozku v letadlech se zatahovacím podvozkem. Přepínač má dvě polohy – vytažený podvozek a zatažený podvozek. Navíc často obsahuje tři světelné indikátory, udávající stav podvozku v přední, levé a pravé části letounu [25].

Pro osazení se používá dvoustavový spínač, generující logické úrovně 0 a 1 [26, 27]. Indikátory jsou realizovány pomocí světelných diod [26].

4.3.3 Přepínání nádrží

Palivo se ve většině letounů uchovává ve dvou nádržích, umístěných v jejich křídlech. Pilot má možnost volit nádrž, ze které se bude palivo čerpat. To se provádí pomocí přepínače nádrží. V závislosti na konkrétním typu letounu může takový přepínač mít dvě až čtyři polohy. Základní dvě – levá nádrž a pravá nádrž –, další dvě polohy – obě nádrže zároveň a žádná – [28].

Realizuje se pomocí dvou- až čtyřstavového otočného přepínače, který má odpovídající počet logických výstupů, udávajících jeho aktuální polohu [29].

4.3.4 Brzdová páčka

Na kolech letounu jsou umístěny brzdy, určené k jeho zastavení při pojíždění po zemi. Jak bylo zmíněno dříve, mohou být brzdy součástí pedálů, což bývá velmi častý případ. Stejně tak se může jednat o samostatnou páčku.

Nepovedlo se najít vlastnoruční realizace brzdové páčky. Nicméně na ultralehkých letadlech se tento prvek objevuje velmi často, a proto bylo rozhodnuto ho do základních ovládacích prvků zahrnout. Realizace by byla možná pomocí tahového potenciometru. Změna napětí na výstupu by udávala absolutní polohu brzdové páčky.

4.3.5 Startovací přepínač

Pomocí tohoto prvku se zapínají dvě magneta, která napájí motor a provádí samotné startování motoru. Jedná se o pětistavový otočný přepínač s polohami OFF, R, L, BOTH a START. Motor se zapaluje přepnutím do polohy START. BOTH, L a R odpovídají za zapnutí magnet, z nichž se poslední dvě využívají pro kontrolu zapalovacího systému při předletové kontrole letounu [30].

I na simulačních ovládacích panelech se používá pětistavový otočný přepínač s pěti logickými výstupy, indikujícími jeho polohu [31].

4.3.6 Spínače

V tomto případě se nejedná o jeden ovládací prvek. Každý panel má velké množství různých spínačů a a spínacích prvků, , odpovídajících za nejrůznější funkce letounu. Je těžké odhadnout jejich přesný počet, jelikož závisí pouze na konfiguraci letounu. Zatímco spínač baterie se objeví v každém letadle, například spínač nočních světel jen v některých. Pro odhad počtu spínačů, který by mohl uživatel chtít do svého panelu

nainstalovat, se podíváme na již hotové panely, určené pro letecké simulátory [32, 33, 34]. Z počtu spínačů, které tyto panely mají, bylo usouzeno, že pokud budeme počítat s více než 16 spínači, pak by tento počet měl uživateli vystačit i se zálohou.

Tabulka 1: Maximální počet současně připojených prvků

Součástka	Počet kusů
Potenciometr	12
Hallův senzor	5
Inkrementální rotační enkodér	2
Vícepolohový přepínač	3
Spínače	18
LED	3
Krokový motor	2
Modelářský servomotor	3

5 I/O modul

V této kapitole jsou finalizovány požadavky na navrhovaný modul a platformu, na jejímž základě se bude stavět. Je třeba určit počet požadovaných vstupů a výstupů, jimiž by měl řídicí mikrokontrolér disponovat, a zvolit komunikační rozhraní, které bude použito pro komunikaci s počítačem.

5.1 Vstupy a výstupy

Z popisu součástek v kapitole 4 určíme celkový počet a typy vstupů a výstupů, které by mělo zařízení obsahovat. Použijeme takový počet vstupů a výstupů, který vystačí pro připojení všech možných konfigurací panelu, nikoliv pro zapojení všech uvedených součástek najednou.

Začneme digitálními vstupy, vyhodnocujícími logickou úroveň vnějších signálů. Tyto vstupy využijeme pro spínače, vícepolohové přepínače a inkrementální rotační enkodéry. Celkově to vychází na 35 digitálních vstupů. Vstupy pro spínače a přepínače se dají realizovat pomocí posuvných registrů s paralelními vstupy. Při použití čtyř osmibitových registrů snížíme počet digitálních vstupů mikrokontroléru na pět, z nichž bude jeden sloužit pro čtení dat z registrů a další čtyři jako vstupy pro rotační enkodéry. Dále bude zařízení potřebovat 12 vstupů s možností analogově-digitálního převodu, které využijeme pro vyhodnocení výstupního napětí potenciometrů a Hallových senzorů. U parametrů převodníku vyjdeme z již komerčně dostupných modulů, které využívají 12bitové rozlišení [4, 6].

Aby měl uživatel možnost zapojení světelných diod, můžeme využít posuvného registru s paralelními výstupy, pro jehož ovládání budeme potřebovat tři digitální výstupy mikrokontroléru. Dalších 5 digitálních výstupů využijeme pro řízení vstupních posuvných registrů. Pro ovládání servomotorů bude modul používat výstup z generátoru PWM-signálu. Takové výstupy budou zapotřebí celkem tři. Ovládání krokových motorů se bude provádět pomocí motorových ovládačů. Jelikož se nedá předem určit, jaký typ krokových motorů uživatel zvolí, je vhodné mít ovládače jak pro unipolární, tak i pro bipolární motory. Možnou volbou těchto ovládačů jsou ULN2003 pro pohon unipolárních motorů a A4988 pro pohon bipolárních motorů. V takovém případě bude mikrokontrolér potřebovat dalších 12 digitálních výstupů pro řízení motorových ovládačů.

5.2 Komunikace s PC

Pro komunikaci s PC budeme volit ze sériových asynchronních rozhraní UART a USB, jež se nejčastěji využívají pro komunikaci mezi počítačem a jeho periferiemi [35, 36]. Jsou podporovány většinou výrobců mikrokontrolérů a často již mají implementovány knihovny a ovládače pro obousměrný přenos dat.

5.2.1 UART

Jedná se o zařízení umožňující sériovou asynchronní komunikaci, která podporuje řadu komunikačních standardů. Jeho vlastnosti jsou v této kapitole popsány na základě informací ze zdrojů [35, 36].

Ve své základní podobě vyžaduje UART pro komunikaci dva komunikační kanály: TX (pro odesílání dat) a RX (pro přijímání dat). Přenos dat lze provádět v režimech poloduplexu i plného duplexu. Často se ke dvěma základním vodičům přidává vodič pro napájecí napětí VDD a vodič pro společné uzemnění dvou zařízení GND.

Pro synchronizaci dvou zařízení se využívá tzv. startovních bitů na začátku každého datového rámce. Dále následují samotná data, která chceme přenést (až osm bitů), a na konci vkládáme koncový bit. Pro kontrolu kontrolu přenesené informace je možné před koncovým bitem vložit bit paritní.

Jednotlivé bity se při plném duplexu nejčastěji reprezentují pomocí úrovní napětí ve vodiči vztaženému k zemi. Při poloduplexním přenosu se určuje rozdíl napětí na dvou přenosových vodičích a ten reprezentuje jednotlivé bity.

Samotné úrovně napětí závisí na zvoleném komunikačním standardu a nejčastěji se pohybují v rozmezí –15 V až 15 V. Je možné ale zvolit i standard, který by vyžadoval napájení pouze 5 V, což by mohlo poskytnout USB-rozhraní počítače.

Aby dvě zařízení mohla vzájemně komunikovat, je třeba, aby se dohodla na společné přenosové rychlosti. Typickou přenosovou rychlostí je 115 200 bit/s [37]. Většina mikrokontrolérů vyžaduje takt hodin, který bude šestnáctinásobkem přenosové rychlosti. Zároveň by se u obou komunikačních zařízení neměla tato hodnota odchylovat o více než 3 %.

Celkově je standard velmi flexibilní. Umožňuje volbu přenosové rychlosti, velikosti přenášených balíků a kontrolu nad chybovostí přenášené informace. Díky jeho jednoduchosti bude ho možné realizovat na téměř libovolném druhu mikrokontroléru a mít pod kontrolou celý přenos mezi modulem a počítačem. Nevýhody se ale objevují na uživatelské straně. Žádný z populárních operačních systémů, například Windows nebo MacOS, nemá předinstalované ovládače pro automatickou komunikaci s UART-zařízeními. Uživatel si bude muset nainstalovat ovládač buď od výrobce mikrokontroléru, nebo od jiného vydavatele, aby jeho počítač mohl zahájit komunikaci se zařízením. Funkčnost produktu se tak stává závislá na vydavateli takového softwaru.

Mohla by nastat situace, kdy by při obnovení operačního systému včas nevyšla nová verze ovládače a zařízení by se na neurčitou dobu stalo nefunkčním. Případně by se pro zajištění i základní funkcionality modulu musel vytvořit plug-in pro každý konkrétní simulátor, který by komunikaci s modulem prováděl. V jiných simulátorech by byl ovládací panel nepoužitelný.

5.2.2 USB

USB je industriální standard, který zahrnuje specifikace pro kabely, konektory, protokoly komunikace a napájení mezi počítačem a jeho periferiemi. Jeho vlastnosti jsou v této kapitole popsány na základě informací ze zdrojů [36, 38].

V základní konfiguraci se připojení provádí pomocí čtyř vodičů: D– a D+ pro samotnou komunikaci, VBUS pro napájení periferních zařízení a GND pro společné uzemnění. Jak naznačují názvy, komunikační kabely jsou komplementární a vedou diferenciální signál, který zajišťuje lepší odstup signálu od šumu.

Komunikace mezi zařízeními probíhá ve formátu host-device. Host je zařízení, jež komunikaci vede, a pouze host může komunikaci inicializovat. Komunikace probíhá v režimu interrupt transfer. Po pravidelných časových intervalech, určených při počáteční konfiguraci komunikace mezi zařízeními, vysílá host zprávy typu OUT, které slouží jako indikátory pro zařízení device. Tyto časové intervaly nesmí být kratší než 1 ms. Ve zprávách typu OUT se můžou, ale nemusí přenášet doplňková data. Zařízení device na zprávy typu OUT odpovídá zprávami typu IN, v nichž přenáší informace o současném stavu zařízení. Zprávy mohou být maximálně 64 bytů dlouhé, a spolu s minimální délkou intervalu tak dostáváme maximální rychlost přenosu 64 kB/s.

Pro různé typy zařízení probíhá komunikace pomocí USB odlišně. Při připojení USB-periferie k počítači probíhá konfigurace zařízení, při které se určí jeho typ a způsob komunikace. Pro tento projekt se zaměříme na typ zařízení HID (Human Interface Device), česky zařízení lidského rozhraní.

HID je třída zařízení určených pro přímou interakci s člověkem, například myš, klávesnice nebo i palubní deska letadla. Typ zařízení se určuje deskriptory, stručnými popisky, které zařízení device sděluje zařízení host při prvním připojení. Deskriptory obsahují třídu zařízení (v tomto případě HID), informaci o výrobci, verzi operačního systému, a zejména definují strukturu přenášených dat, jež si budou dvě zařízení posílat.

Na rozdíl od jiných USB-tříd není u zařízení typu HID nutné volit jeho podtřídu z nějaké omezené množiny možností. Je to velká výhoda, jelikož pomocí HID-deskriptorů můžeme flexibilně určit počet fyzických vstupů zařízení a jejich formát. Formát vstupů lze nastavit z již předurčené množiny HID Usage, která obsahuje kategorii (Usage Page) Simulation Controls Page a podkategorii (Usage ID) Flight simulation device. V této podkategorii lze najít skoro všechny ovládací prvky, které budeme k modulu připojovat a které stačí zahrnout do deskriptoru. Další výhodou této třídy je její podpora operačními systémy Windows a MacOS, jež mají již předinstalované ovládače pro zařízení typu HID a nevyžadují žádný doplňkový software pro provedení komunikace směrem ze zařízení do simulátoru. To znamená, že základní funkcionalita se zpracováním analogových a digitálních vstupů zachová u všech simulátorů.

5.2.3 Volba mezi UART a USB HID

Vzhledem k nativní podpoře zařízení typu HID hlavními počítačovými operačními systémy, rozšířenosti samotného portu USB a jednoduchosti použití pro uživatele se pro danou aplikaci zdá USB vhodnější možností pro realizaci komunikace mezi I/O-modulem a počítačem. Třída HID byla vytvořena přesně pro tento typ zařízení a na rozdíl od UART jsou téměř všechny třídy ovládacích prvků simulátoru již součástí komunikačního protokolu a nebude potřeba je realizovat od nuly, pouze přidat do report deskriptoru a posílat data v předem určeném formátu.

5.3 Platforma

Skoro všechny firmy zabývající se výrobou mikrokontrolérů nabízejí výrobky, které by odpovídaly požadavkům této aplikace. Proto volba výrobce nebude mít velký vliv na výsledný produkt, spíše jde o otázku preference. Jelikož jsem lépe seznámen s produkty firmy STMicroelectronics, budu volit z výrobků této značky. Má širokou nabídku mikrokontrolérů různých kategorií, ze kterých nebude problém vybrat mikrokontrolér vhodný pro řízení komunikačního modulu.

Při volbě konkrétního mikrokontroléru se bude vycházet z následujících parametrů:

- podpora rozhraní USB v režimu device
- pět a více digitálních vstupů
- přítomnost analogově-digitálního převodníku s minimálně 12 kanály a rozlišením převodu aspoň 12 bitů
- minimálně 23 digitálních výstupů, z nichž tři musí mít možnost generování PWM signálů

6 Komunikační cesta

Celá komunikační cesta se dělí na dva segmenty: na komunikaci mezi zařízením a operačním systémem počítače a komunikaci mezi operačním systémem a leteckým simulátorem. Celkově tři komunikační body, ve kterých se data přijímají a odesílají: zařízení, operační systém počítače, letecký simulátor. Část této cesty je již realizována, část bude potřeba ještě vytvořit, případně upravit pro danou aplikaci.

6.1 USB Device

Jako komunikační rozhraní pro propojení I/O-modulu s počítačem bylo zvoleno USB a konkrétně třída zařízení HID. Jak bylo uvedeno v kapitole 5.2.2, komunikace probíhá v režimu host–device. Pro samotné vysílání na fyzické vrstvě, definici jednotlivých paketů a směrování dat ze strany device využijeme USB-knihovny, kterou pro svoje výrobky poskytuje STMicroelectronics. Aby přenos fungoval, je třeba vytvořit správný popis zařízení.

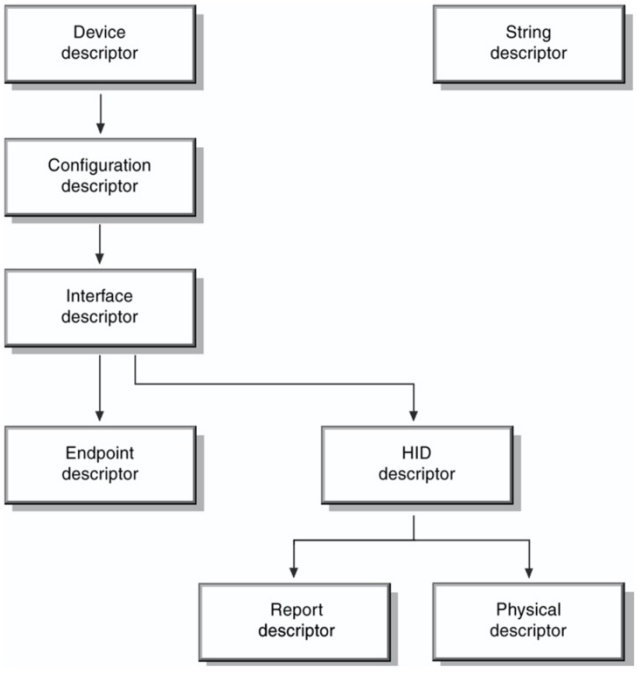
Informace o zařízení USB device je uložena v segmentech nazývaných deskriptory. Tato informace slouží ke směrování a zpracování posílaných dat. Deskriptory se dělí do několika úrovní a na každé se popisují jednotlivé vlastnosti zařízení. Jejich následující popis je převzat z oficiální dokumentace USB HID [38].

Na nejvyšší úrovni se nachází device deskriptor a string deskriptor. Device deskriptor určuje třídu a podtřídu zařízení, obsahuje identifikační číslo výrobce, identifikační číslo produktu, jeho verzi a název. V případě HID-zařízení se jeho třída neurčuje zde, ale až v interfacu deskriptoru. String deskriptor obsahuje veškeré textové řetězce, na něž se v jiných deskriptorech odkazuje jejich indexem ve string deskriptoru (například název zařízení v device deskriptoru se udává jeho indexem). Na další úrovni je configuration deskriptor. Ten určuje počet USB-rozhraní, které zařízení podporuje. Rozhraní mohou být jak v rámci jedné třídy (např. v rámci HID-třídy může mít jedno zařízení rozhraní myši i klávesnice), tak i z odlišných USB-tříd (např. HID, Audio a Hub). Další parametry, které daný deskriptor obsahuje, se týkají napájení zařízení. Udává, zda se zařízení napájí z VBUS-vodiče, nebo z vlastního zdroje, a také maximální proud, který by mohlo zařízení od VBUS potřebovat. Dále následuje interface deskriptor, ve kterém se nastavují rozhraní, jejichž počet se určil v configuration deskriptoru. V případě HID-zařízení se v tomto deskriptoru určí třída zařízení jako HID a uvede se, zda patří do podtřídy HID-zařízení pod názvem boot interface. Do této podtřídy náleží taková zařízení, která mohou být využívána BIOSem operačních systémů, k nimž jsou připojena během jejich bootu. Taktéž se zde určí počet endpointů, jež zařízení využívá pro komunikaci.

Endpoint je buffer umístěný na straně device. Host může přijímat nebo odesílat data z těchto bufferů i do nich. Endpointy se dělí do kategorií control a data. Každý device obsahuje minimálně jeden control endpoint, nazývaný Endpoint0. Tento endpoint je obousměrný, tedy host může během jednoho přenosu z něj data přijímat i do něj data zapisovat. Tento endpoint umožňuje hostu přijmout informaci o zařízení a případně provádět jeho konfiguraci. Data endpointy jsou volitelné, jednosměrné a používají se pro přenos dat.

Pro popis endpointů existuje samostatný deskriptor. Ten udává vlastnosti jednotlivých endpointů: adresu, směr a typ přenosu, maximální počet paketů, který zvládne endpoint přijmout/odeslat, a polling interval (časový interval, se kterým host data do zařízení posílá / data od zařízení požaduje). Endpoint0 žádný deskriptor nevyžaduje, jelikož jeho parametry jsou určeny předem. Určuje se pouze jeden jeho parametr v rámci device deskriptoru, a to maximální velikost paketu, jakou zvládne odeslat, nebo přijmout.

Třída HID definuje další tři deskriptory, typické pouze pro tuto třídu zařízení. Prvním takovým deskriptorem je HID-deskriptor. V tomto deskriptoru se zadává velikost následujících report a physical deskriptorů. Poslední physical deskriptor popisuje fyzické vlastnosti zařízení. Je pro zprovoznění komunikace nepovinný, proto se jím zabývat nebudeme. Report na rozdíl od všech předchozích deskriptorů není pevně definován a je plně vytvořen výrobcem zařízení v souladu s pravidly popsanými v dokumentaci HID-třídy. Obsahuje popis dat, která zařízení device vysílá a přijímá. Definuje typ dat, jejich délku a jejich význam. V tomto deskriptoru můžeme data přiřadit ke konkrétnímu typu tak, aby dále byla zpracována pomocí API (application programming interface) operačního systému. Například pokud device do počítače posílá data velikosti jeden byte a tato data označíme jako osu joysticku s hodnotami 0 – 255, aktivuje se na počítači API, které odpovídá za práci s joysticky a bude tento přijatý byte vnímat jako polohu osy.



Obrázek 8: Ilustrace struktury USB deskriptorů [38]

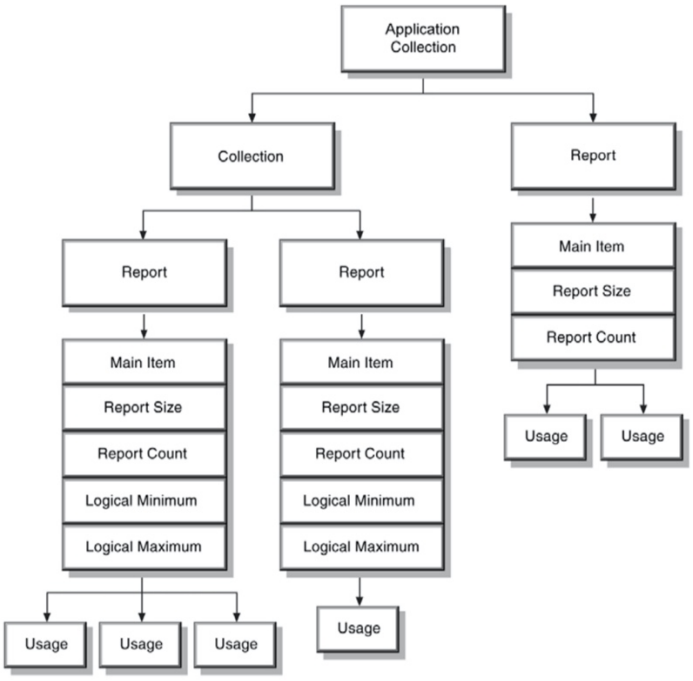
Report deskriptor se dělí na kusy informace zvané itemy. Každý item má jednobytový prefix, obsahující tag, typ a velikost, definující typ informace, kterou item v sobě nese. Existují tři typy itemů: main, global a local. Pro main itemy je definováno pět tagů: input, output, feature, collection a end collection. Tag input označuje data posílaná ze zařízení device do zařízení host. Tag output označuje data posílaná ze zařízení host do zařízení device, která mají význam pro uživatele (například data o stavu LED na zařízení). Tag feature označuje input i output data. Akorát se tentokrát nejedná o data, která by byla přímo určena pro uživatele, ale o data určená ke konfiguraci zařízení a ke změně jeho nastavení. Pomocí tagů collection a end

collection se vytváří skupiny itemů s tagem input, output a feature, které jsou nějakým způsobem spojeny (fyzicky, logicky, aplikačně apod.). Datová část main itemů udává některé vlastnosti přijímaných/odesílaných/sjednocených dat a význam jednotlivých bitů datové části je uveden v tabulce 2, převzaté z dokumentace USB HID.

Main itemům předcházejí local a global itemy, určující další parametry dat. Tyto parametry jsou uvedeny v tabulkách 3 pro local a 4 pro global itemy, taktéž převzatých z dokumentace USB HID.

Report deskriptor může mít několik main itemů a pro validní popis vysílaných dat musí pro každá data obsahovat následující itemy: input (output, feature), usage, usage page, logical minimum, logical maximum, report size a report count. Jejich popis lze nalézt v tabulkách 3 a 4. Všechny ostatní itemy jsou nepovinné.

Itemy usage a usage page popisují reálný význam vysílaných dat a třídí je do různých kategorií i přímo do typů zatížení. Tyto kategorie a jejich popisy lze dohledat v tabulkách HID Usage Tables.



Obrázek 9: Struktura zařízení typu HID [38]

Pro zprovoznění komunikace na straně zařízení bude třeba vytvořit všechny výše uvedené deskriptory a s využitím funkcí USB knihovny od STMicroelectronics zajistit vysílání a příjem IN a OUT zpráv.

6.2 USB Host

Tento segment komunikační cesty nebude třeba nijak modifikovat. Za komunikaci na straně host odpovídají USB-ovládače operačních systémů Windows a MacOS a s nimi spojené API. Příjem zpráv typu IN bude probíhat v intervalech zadaných v popisu data endpointu. Odesíláni dat se ve zprávách typu OUT taktéž bude provádět USB-ovládačem v souladu s parametry zadanými v endpoint deskriptoru.

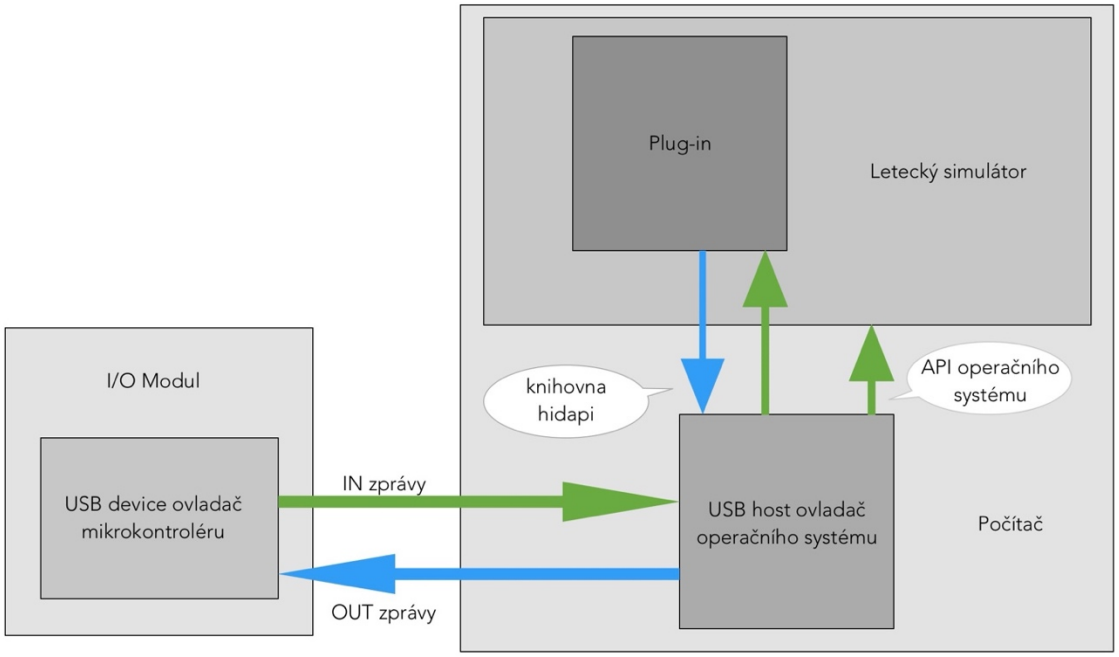
6.3 Simulátor

Komunikace mezi USB-ovládačem operačního systému a simulátory částečně probíhá automaticky. Jak bylo uvedeno v podkapitole 5.5, v případě, že jsou data ve zprávě typu IN označena určitým způsobem, budou zpracována API operačního systému určeného pro zpracování tohoto typu dat. Především se jedná o označení typu zařízení, se kterým jsou data spojena (myš, klávesnice, joystick atd.). V případě, že letecký simulátor má nativní podporu takového zařízení, automaticky probíhá propojení s API poskytnutým operačním systémem i výměna dat mezi API a simulátorem. To znamená, že veškeré osy a tlačítka budou libovolným simulátorem, podporujícím takové API rozhraní, rozpoznány automaticky a vstupní data není třeba nijak zvlášť zpracovávat.

Operační systémy taktéž poskytují přístup přímo ke komunikaci s USB-zařízeními. Pro zjednodušenou práci s USB-ovládači využijeme multiplatformní C++ knihovnu hidapi s licencí pro volné využití v libovolných projektech [39]. Poskytne jak přístup přímo k datům přijímaným ve zprávách IN (využíváno převážně pro debuggování), tak i k odesíláni zpráv typu OUT do I/O-modulu, což umožní ovládání aktuátorů. Přistupovat k datům budeme z plug-inu leteckého simulátoru X-Plane, pro nějž je zařízení primárně určeno.

Jak plyne z dokumentace simulátoru [40], plug-in je spustitelný kód, který běží uvnitř simulátoru a rozšiřuje jeho možnosti. Plug-iny pro X-Plane jsou dynamicky linkovanými knihovnami DLL (dynamically linked library) a představují sadu kompilovaných funkcí a s nimi spojených proměnných. Během toho, co je program spuštěn, volá funkce z této DLL-knihovny. K tomu využívá XPLM (X-Plane Plugin Manager). XPLM je taktéž DLL-knihovnou, která řídí všechny ostatní plugi-ny. Simulátor volá XPLM a XPLM volá funkce, uložené v ostatních knihovnách.

Do takových funkcí zahrneme spojení s USB-ovládačem operačního systému a komunikaci se zařízením, čímž dokončíme komunikační cestu.



Obrázek 10: Schematická ilustrace komunikační cesty

7 Prototyp

Pro vyzkoušení připojení všech ovládacích prvků, aktuátorů a zprovoznění USB-komunikace bylo rozhodnuto vytvořit prototyp zařízení, na jehož základě se bude dále stavět výsledný modul.

Jako základ pro postavení prototypu byla primárně z důvodu prodejní dostupnosti zvolena deska Nucleo-H723ZG s mikrokontrolérem STM32H723ZGT6, který splňuje všechny požadavky stanovené v kapitole 5.3. Jako vývojové prostředí bylo zvoleno CubeIDE pro vývoj softwaru zařízení a Visual Studio pro vývoj X-Plane plug-inu.

7.1 Vytvoření USB HID Device

Pro realizaci USB-komunikace jsem využil knihovnu USB device library, poskytnutou výrobcem mikrokontroléru. Je součástí vývojového prostředí CubeIDE, které kromě souborů s definicemi funkcí zařizujících komunikaci poskytuje i dva vzorové příklady HID-zařízení. Prvním příkladem je zařízení typu myš a druhý se nazývá custom HID device. Na jejich základě se dá postavit vlastní implementace. Z těchto dvou příkladů byl funkční pouze příklad s myší. Kód se zařízením custom HID device se po dlouhém debuggování zprovoznit nepovedlo. Proto – přestože by byl příklad s custom HID-zařízením vhodnější – bylo rozhodnuto vlastní zařízení budovat na základě příkladu s myší.

7.1.1 USB deskriptory

Nejprve je potřeba vytvořit vhodné deskriptory. (Pro zjednodušení následujícího popisu budeme předpokládat, že zápisem textového řetězce do deskriptoru se myslí zápis indexu tohoto řetězce ve string deskriptoru.)

Začneme device deskriptorem. Třídu a podtřídu zařízení neurčujeme a necháváme nulové hodnoty, jelikož se jedná o zařízení typu HID. Číslo výrobce (VID) a produktu (PID) je ve vzorovém příkladě již poskytnuto společností STMicroelectronics. Registrace vlastních VID a PID je možná u společnosti USB-IF za 6 000 amerických dolarů při platném členství cenou 5 000 dolarů ročně [41]. Jelikož jsou tyto ceny příliš vysoké a prototyp nebude komerčně využíván, v rámci projektu ponecháme VID a PID poskytnuté ve vzorovém deskriptoru. V případě masové výroby a komerčního využití se dá u společnosti STMicroelectronics zažádat o povolení k oficiálnímu použití jejich VID a PID [42]. Dále zadáme název zařízení „Cockpit IO module“ a určíme maximální délku paketu pro Endpoint0 64 bytů.

V následujícím configuration deskriptoru zadáme, že zařízení bude mít pouze jedno rozhraní. Pro toto rozhraní zvolíme vlastní napájení a určíme maximální proud. V případě, že se VBUS nepoužívá jako zdroj napájení, slouží pro vzájemnou detekci zařízení host a device. Pro tuto detekci může být spotřeba ze strany device až 200 uA [43]. Proto nastavujeme nejmenší možnou nenulovou hodnotu 2 mA.

Rozhraní máme pouze jedno, proto budeme potřebovat jen jeden interface deskriptor. V něm zvolíme třídu zařízení HID, boot interface volit nebudeme

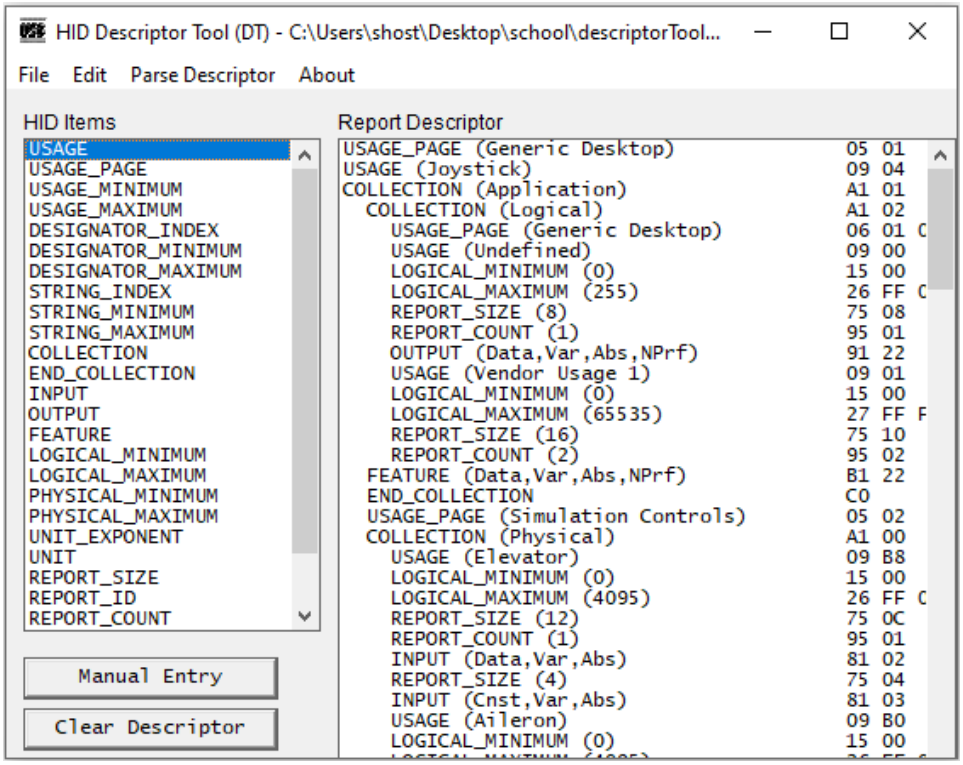
a zapíšeme dva endpointy – jeden pro posílání zpráv typu IN a druhý pro příjem zpráv typu OUT.

Pro tyto endpointy budeme potřebovat dva endpoint deskriptory. Ty se budou lišit typem (IN/OUT) a velikostí dat, jež jsou schopny přijímat/odesílat. Tyto velikosti budou odpovídat velikostem dat později popsaným v report deskriptoru. Oba endpointy používají typ přenosu interrupt, používaný pro paketový přenos dat v pravidelných intervalech. Tento interval nastavíme 1 ms, pro maximální odezvu.

7.1.2 HID a report deskriptory

V HID deskriptoru uvedeme, že používáme jeden report deskriptor, a později sem zkopírujeme velikost report deskriptoru, který vytvoříme pro danou aplikaci.

Pro vytvoření report deskriptoru použijeme aplikaci HID Deskriptor tool, dostupnou přímo ze stránek usb.org. Tento nástroj umožňuje pomocí grafického rozhraní vytvářet pole bytů, jež přímo představuje report deskriptor, který stačí vložit do kódu zařízení.



Obrázek 11: HID Descriptor Tool

Report deskriptor začneme volbou typu zařízení. Podle této volby bude host určitým způsobem vnímat přijímaná data. Vzhledem k tomu, že chceme, aby bylo zařízení rozpoznáváno co nejvíce simulátory, zvolíme joystick. To provedeme volbou usage – generic desktop a usage page – joystick. Dále pomocí collection itemu vytvoříme skupinu, spojenou stejnou aplikací (tou se myslí celé zařízení), a nyní všechna data popisovaná v rámci této skupiny budou hostem vnímána jako data generovaná zařízením typu joystick.

V rámci skupiny začneme daty, která nejsou určena přímo pro uživatele, ale pro nastavení zařízení. Vytvoříme další skupinu, tentokrát spojenou logicky, a v ní

popíšeme pět bytů. Všech pět bytů bude typu feature a využijeme je pro nastavení různých parametrů a kalibraci zařízení. První byte bude nabývat hodnot logical minimum 0 až logical maximum 255 a jeho usage nebude definováno z dostupných v tabulkách HID usage tables (neboli bude definováno výrobcem). Významem tohoto bytu bude mód nastavení, určující nastavované parametry. Další čtyři byty taktéž nebudou mít definován usage item a budou využívány pro přenos nastavovaných dat, příslušných danému módu. Jsou rozděleny na dvě šestnáctibitová čísla, nabývající hodnot 0 až 65 535. Tím logicky spojenou skupinu uzavřeme a dále se budeme zabývat pouze daty určenými pro uživatele. Tato data označíme pomocí usage page itemu jako ovládací prvky simulátoru (simulation controls) a rozdělíme je do skupin spojených fyzicky podle jednotlivých ovládacích prvků připojených k zařízení (bližší popis v kapitole 4).

První z nich budou berany/řídící páka. Tento prvek má dvě osy, reprezentující jeho polohu, které označíme pomocí itemů usage elevator a usage aileron, jež přímo popisují jejich funkci. Jelikož se osy vysílají ze zařízení device do zařízení host, budou označeny itemem input. Obě osy jsou dvanáctibitové s hodnotami 0 až 4 095. Jelikož se data při přenosu dělí na byty, doplníme každou osu o čtyři prázdné bity tak, aby jejich velikost vycházela přímo na dva byty. To způsobuje jak přehlednost při přenosu, tak jednodušší čtení a odesílání dat. Totéž se provede u všech dalších dvanáctibitových os.

Pro pedály je popis dat velmi podobný. Jedná se o tři dvanáctibitové osy, reprezentující polohu pedálů a brzd, které označíme itemy usage rudder, usage toe brake a input.

Totéž se týká plynového kvadrantu, u nějž pouze jinak označíme osy – usage throttle a usage slider. Slider použijeme pro osy směsi a nastavení vrtule, jelikož pro ně neexistuje dedikovaný usage. Osy reprezentují absolutní polohu příslušných pák.

Další dvanáctibitovou osu tvoří brzdová páčka, kterou označíme itemem usage brake. Osa reprezentuje polohu páčky.

Všechny spínače a přepínače zahrneme do jednoho logického prvku, který bude odpovídat za data typu boolean, reprezentující tlačítka. Každý bit bude reprezentovat stav jednotlivého tlačítka. Pro tlačítka existuje samostatný usage page – button. Logical minimum dat bude 0 a maximum 1, počet bude 32 (odpovídá počtu paralelních vstupů čtyř posuvných registrů) a budou typu input.

Data odpovídající za stav světelných diod jsou taktéž typu boolean. Stejně jako v předchozím případě pro ně existuje vlastní usage page – LEDs. Logické úrovně jsou opět 0 a 1, celkový počet tři a budou typu output, jelikož se data vysílají ze zařízení host. Stejně jako v případě os, data doplníme o prázdné bity (v tomto případě o pět bitů), abychom vyplnili celý byte.

Pro páčku na ovládání klapek neexistuje usage, proto opět použijeme slider. Opět se jedná o dvanáctibitovou osu typu input. Ta reprezentuje míru vychýlení klapek. Kromě páčky obsahuje tento prvek indikátor polohy. Pro vysílání aktuální polohy klapek využijeme osmibitové osy vysílané ze simulátoru do zařízení s následujícími itemy: výrobcem definovaný usage item, logical minimum 0 a logical maximum 255, output.

Zbývající trimovací kola budou mít obě stejný popis. Data o stavu kolečka budeme vysílat ve tvaru dvanáctibitové osy, reprezentující míru vychýlení trimovacích ploch. Ze simulátoru budeme přijímat data o stavu autopilota, který bude zapínat a vypínat

krokové motory a osu reprezentující míru vychýlení trimovacích ploch, která odpovídá současnému stavu simulovaného letadla. Ta bude určovat polohu krokového motoru a servomotoru. Informaci o stavu autopilota budeme reprezentovat jedním bitem s logickými úrovněmi 0 (vypnutý) a 1 (zapnutý), usagem auto-pilot enable a itemem output. Zbytek bytu vyplníme nulami. Osa reprezentující míru vychýlení bude šestnáctibitová s logickým úrovněmi 0 až 65 535 a taktéž označena itemem output.

Výsledný report deskriptor je možné nalézt v příloze B.

7.1.3 Rozšíření vzorového HID-příkladu

Vzorový příklad se zařízením typu myš má realizovanou velmi omezenou funkcionalitu oproti té, kterou komunikační modul vyžaduje. Jedná se o kód reprezentující mikrokontrolér jako myš, schopnou vysílat data typu input o velikosti dva byty. Reprezentace zařízení a velikost dat typu input byla změněna v předchozí podkapitole změnou deskriptorů. Nyní je třeba umožnit nejen odesílání dat typu input, ale i příjem dat označených itemy output a feature.

Vzorový příklad s myší nemá vůbec realizovány funkce pro příjem zpráv typu OUT. Obrys této funkcionality se ale dá nalézt ve vzorovém příkladu custom HID device, kterého využijeme pro její doplnění. Jelikož vzorový příklad se zařízením typu myš vůbec neobsahuje OUT endpoint, při inicializaci USB HID se s ním nijak nepracuje. Proto doplníme inicializaci OUT endpointu, a navíc zde spustíme přípravu na příjem prvního paketu. Dále z druhého vzorového příkladu sem přeneseme funkce, jež reagují na události příjmu zpráv typu OUT. Jedna bude odpovídat za zprávy, jež jsou doručeny do Endpoint0 (data označená itemem feature) a druhá bude odpovídat za zprávy doručené do OUT endpointu (data označená itemem output). V těchto funkcích data z endpointů zkopírujeme do bufferů (OUT_Buffer a FEATURE_Buffer), s nimiž budeme dále pracovat.

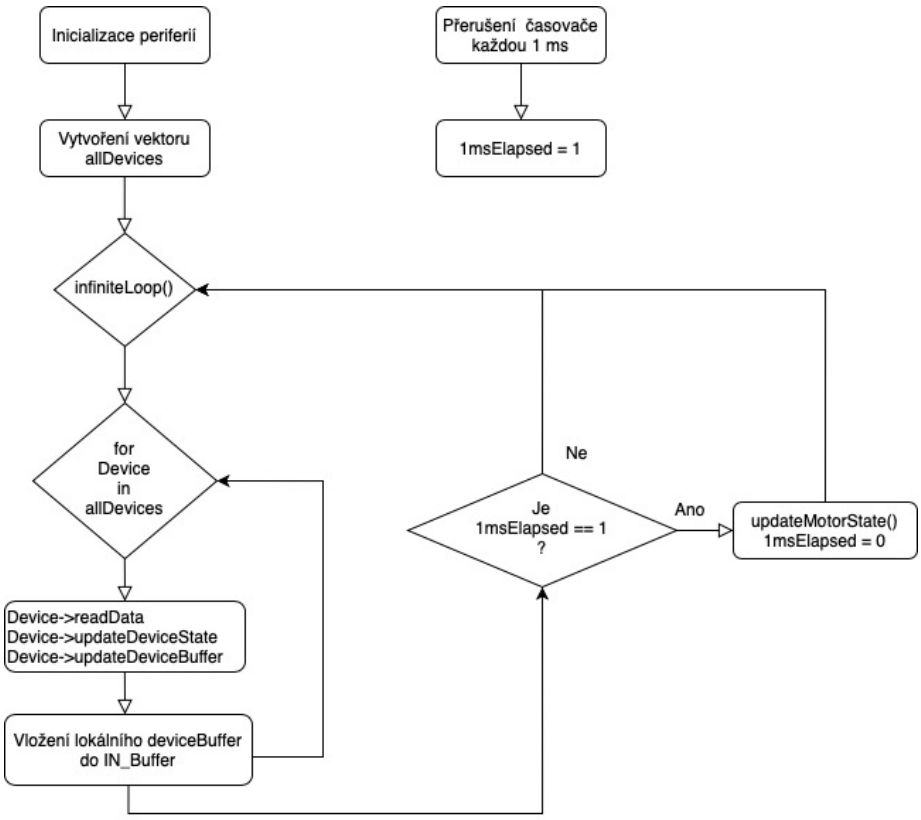
Nyní máme realizovánu plnou funkcionalitu pro komunikaci přes USB, a to jak funkce pro příjem dat, tak i pro jejich odesílání.

7.2 Hlavní program

Jelikož interval, se kterým se budou data vysílat, je poměrně dlouhý (1 ms), bylo rozhodnuto za účelem zjednodušení struktury a zmenšení počtu potenciálních chyb postavit program s co nejmenším počtem přerušení, založený na pollingu vstupních dat.

Pro každou skupinu dat definovaných v report deskriptoru byla vytvořena vlastní třída, reprezentující prvek seskupující tato data. Všechny tyto třídy jsou podtřídami jedné společné nadtřídy, přičemž všechny dědí několik stejných metod a jednu proměnnou. Děděnou proměnnou je deviceBuffer, obsahující data konkrétní třídy (např. třída pedály má ve svém bufferu uloženy tři dvanáctibitové osy doplněné nulami na šest bytů), která se z tohoto lokálního deviceBufferu později přidají do jednoho společného IN_Bufferu. Ten se následně odešle do simulátoru. Metody, jež mají všechny třídy, jsou returnDeviceBuffer, returnDeviceBufferSize, readData, updateDeviceState a updateDeviceBuffer. Jejich funkcionality se nejlépe přiblíží při popisu hlavní funkce programu.

V hlavní funkci se nejprve provádí inicializace všech potřebných periferií a vytváří se vektor z objektů výše popsaných tříd tak, aby se k nim dalo přistupovat podle indexů. V následující, nekonečné smyčce program prochází tímto vektorem a volá jednotlivé metody v něm uložených objektů. Nejprve se volá metoda readData, která načte do lokálních proměnných objektu údaje o stavu fyzických ovládacích prvků a bufferů OUT_Buffer a FEATURE_Buffer, do nichž jsou uložena data z OUT-zpráv. Dále následuje updateDeviceState, jež u objektů odpovídajících za ovládání aktuatorů na základě dat obdržených z OUT-zpráv provede změnu stavů těchto aktuátorů. Následně se volá metoda updateDeviceBuffer, ve které se do lokálního deviceBufferu uloží údaje o stavu ovládacích prvků. Jako poslední se volá returnDeviceBuffer, která na určité místo v IN_Bufferu, určené pomocí metody returnDeviceBufferSize předchozího objektu, uloží tato lokální data. Poté, co si program projde všechny objekty ve vektoru, odešle se IN_Buffer do počítače.



Obrázek 12: Schematická ilustrace hlavního programu

7.3 Připojení jednotlivých prvků

Po zprovoznění komunikace s počítačem a vytvoření struktury programu je třeba realizovat připojení jednotlivých součástek popsaných v kapitole 4, z nichž jsou tvořeny ovládací prvky letounu.

7.3.1 Potenciometry a Hallovy senzory

Všechny analogové signály se převádí na číslicové hodnoty pomocí dvou analogově-digitálních převodníků mikrokontroléru. Každý z nich disponuje několika kanály, mezi nimiž se dá přepínat a pomocí každého převodníku odečítat hodnoty z několika vstupů. Převod je prováděn s hloubkou 12 bitů v souladu s doporučeními sepsanými v aplikačním doporučení společnosti STMicroelectronics. Při zapnutí zařízení se provádí vnitřní kalibrace převodníků, je využit jejich plný rozsah 0–3,3 V, a navíc je použito číslicové filtrování pro zpřesnění výsledků.

Všechny převody se provádí v metodě readData-objektů obsahujících osy ovládacích prvků.

7.3.2 Spínače a přepínače

Pro připojení spínačů a přepínačů byly použity čtyři posuvné registry 74HC166, každý z nich disponuje osmi paralelními vstupy a jedním sériovým výstupem. Všechny vstupy registrů se uzemní přes odpory tak, aby jejich výchozí polohou byla logická 0. Spínací prvky se zapojí mezi napájecí napětí 3,3 V a takto uzemněné vstupy registrů a při jejich spojení dojde ke změně stavu vstupu na logickou 1. Komunikace s registry se provádí jednosměrně z registrů do mikrokontroléru pomocí třívodičové verze sběrnice SPI (bez vodiče MOSI) se společnými vodiči CLK a MISO pro všechny čtyři registry.

Při volání metody readData objektu odpovídajícího za spínače a přepínače se postupně zahájí komunikace s každým z registrů a načte se jejich aktuální stav.

7.3.3 Inkrementální rotační enkodéry

Zařízení bude podporovat rotační enkodéry s jedním napájecím a dvěma výstupními vodiči. Výstupní vodiče se přes odpory připojí k zemi a pomocí číslicových vstupů mikrokontroléru se z nich odebírají dva obdélníkové signály navzájem posunuté o 90 stupňů. Ze signálů se odečte jak míra pootočení osy, tak i směr. Důležitým parametrem enkodérů je jejich rozlišení, které je pro každý enkodér unikátní a uživatel bude mít možnost si ho nastavit v plug-inu.

Enkodéry se budou používat pro snímání polohy trimovacích kol, jež v leteckém simulátoru ovládají osu příslušného trimu. Signál z enkodéru se v metodě readData u objektů trimovacích kol pomocí rozlišení enkodéru přepočítá na míru posuvu trimovací osy. Rozsah trimovací osy je roven šesti plným otočením kola, což odpovídá rozsahu virtuálního kola použitého v modelech letadel v simulátoru X-Plane.

7.3.4 LED

Pro rozsvícení svítivých diod byl v prototypu jako v případě čislicových vstupů použit posuvný registr. Registr 74HC595 má jeden sériový vstup a osm paralelních výstupů s výstupním proudem až 25 mA. Na výstupy se přes odpory připojí diody. Komunikace s registrem se opět provádí pomocí SPI, ale tentokrát bez vodiče MISO.

Poloha podvozku se načte z OUT_Bufferu v metodě readData objektu obsahujícího data o podvozku. Simulátor poskytuje data o stavu jednotlivých kol. Pokud je kolo plně

vysunuto, v metodě updateDeviceState stejného objektu se na příslušné místo v registru nahraje 1 a LED reprezentující toto kolo se rozsvítí.

7.3.5 Servomotory

Ovládání servomotorů se provádí pomocí signálů s pulzně šířkovou modulací. K jejich generaci byl využit jeden z časovačů mikrokontroléru, který disponuje výstupními kanály schopnými takto modulovaný signál vytvářet. Nastavením dvou registrů časovače (compare a auto-reload) se určí střída a perioda signálu [44]. Periodu nastavíme na standardních pro servomotory 20 ms a rozmezí stříd pro nastavení polohy motoru na 5 až 7,5 %. Rozmezí stříd pro ovládání motorů bývá různé, proto bude mít uživatel možnost si tyto hodnoty pro jednotlivé servomotory nastavit v plug-inu.

Při volání metody readData se u objektů zařízení pro ovládání klapek a trimů načtou aktuální polohy virtuálních os, které mají ručičky servomotorů reprezentovat. Dále v metodě updateDeviceState příslušných objektů se poloha osy převede na odpovídající hodnotu compare registru kanálu, na nějž je servomotor připojen, čímž se změní střída generovaného signálu.

7.3.6 Unipolární krokové motory

Krokové motory, pro něž se používá unipolární zapojení, typicky mívají šest nebo osm výstupních vodičů, z nichž jsou dva v případě šestivodičového a čtyři u osmivodičového určeny pro připojení napájecího napětí [45]. Po spojení všech napájecích vodičů do jednoho získáme celkově jeden napájecí a čtyři řídicí vodiče pro zapojení jednoho unipolárního krokového motoru. Řídicí vodiče je třeba ve správném pořadí uzemňovat tak, aby se motor mohl otáčet. K tomu využijeme obvod ULN2003, složený ze sedmi zesilovacích prvků, jež poskytují výstupní proudy do 500 mA. Z těchto sedmi prvků použijeme čtyři, jejichž vstupy zapojíme do mikrokontroléru a výstupy spojíme s řídicími vodiči motoru. Při logické jedničce na vstupu obvodu bude odpovídající řídicí vodič motoru spojen se zemí a při logické nule bude připojen na napájecí napětí. Toto zapojení umožňuje pomocí čtyř číslicových výstupů mikrokontroléru ovládat pohyb motoru.

Pohyb trimovacího kola se určuje několika parametry: stavem autopilota v simulátoru, změnou virtuální osy trimu a volbou snímacího prvku trimovacího kola. První dva parametry se zjistí v metodě readData a snímací prvek si uživatel bude moct nastavit v plug-inu. Bude mít možnost si vybrat mezi rotačním enkodérem a potenciometrem. Tyto parametry se vyhodnotí v metodě updateDeviceState. Stav autopilota má vliv na to, zda je motor potřeba vůbec zapínat, jelikož trimovací kolo, které pohání, se pohybuje pouze při zapnutém autopilotu. Pokud je autopilot zapnutý, vyhodnotí se změna virtuální osy a přepočítá se na počet kroků, o které se musí motor pohnout. Výpočet se provádí pomocí počtu kroků potřebných pro jedno otočení, které je pro každý motor individuální a bude uživatelem zvoleno v nastaveních plug-inu. Poslední parametr má vliv na druh pohybu motoru. Hlavním rozdílem mezi rotačním enkodérem a víceotáčkovým potenciometrem je jejich rozsah. Zatímco rotační enkodér můžeme otáčet do nekonečna, víceotáčkový potenciometr je fyzicky omezen na určitý

počet otáček. Jelikož budou osy snímacího prvku a motoru spojeny, bude to mít přímý vliv na způsob, jakým budeme motor otáčet.

Pro pilota v případě trimovacího kola není tolik důležitá absolutní změna, o niž se trimovací kolo pohnulo, nýbrž směr jeho pohybu. Ve fyzickém letadle není důležité, která z těchto informací má vyšší prioritu, jelikož jsou u reálného prvku obě aktuální. V případě simulačního prvku se může stát, že krokový motor nemá dostatečnou rychlost, aby přesně kopíroval polohu virtuálního kola. To vede ke zpoždění, jež může mít za následek mylnou informaci pro pilota. Například autopilot bude provádět vyvážení na nos (kolečko se točí nahoru), ale v jednu chvíli se podmínky změní a začne vyvažovat na ocas (kolečko se točí dolů). Virtuální kolo se ihned začne točit na jinou stranu, zatímco krokový motor se v důsledku zpoždění bude nějakou chvíli stále točit nahoru. To může pilota zmást, jelikož mu simulační prvek poskytuje zastaralou informaci. Samozřejmě kdyby kolo kopírovalo pouze směr pohybu, nikoliv přesný úhel otočení, pak by stále nedávalo pravdivou informaci, přičemž její nepravdivost by ale nyní spočívala v míře vyvážení (reálné kolo by se otočilo o menší úhel než kolo virtuální). Jelikož každé trimovací kolo je unikátní a není dán univerzální úhel otočení, jenž by reprezentoval konkrétní míru vyvážení, není ani u reálného kola tato informace příliš přesná. Navíc je informace o míře vyvážení duplikována servomotorem, který udává míru vyvážení v rámci celé osy, a proto by tato nepřesnost úhlu otočení v pilotovi zmatek nezpůsobovala.

Kopírování směru otočení je možné realizovat pouze v případě rotačního enkodéru, u kterého díky jeho neomezenému rozsahu nebude záležet na rozdílu úhlu otočení reálné a virtuální osy. U víceotáčkového potenciometru budou muset osy splývat tak, aby nedošlo k překročení rozsahu fyzického prvku. Proto se v případě volby tohoto snímacího prvku bude motor vždy otáčet ve směru aktuální polohy virtuálního kola, dokud poloha virtuální a reálné osy nebude splývat. V obou případech je použit plný krok motoru, jelikož aplikace nevyžaduje jemné pohyby.

Program pohyb provádí pomocí časovače, který každou milisekundu vyvolává přerušení, aktivující funkci pohybu. Ve funkci se provede kontrola, zda je třeba motor pohnout, a v případě, že ano, pohne motor o jeden krok v požadovaném směru. To umožňuje dostatečnou rychlost 1 000 kroků za sekundu a zároveň zajišťuje, že u většiny motorů nedojde k přeskakování kroků.

7.3.7 Bipolární krokové motory

Bipolární krokové motory typicky mají čtyřvodičové zapojení [45], vyžadující pro jejich ovládání složitější obvody s H-můstky. Pro vyzkoušení ovládání motoru v prototypu byl použit budič A4988.

Tento obvod umožňuje vysíláním pulzů ovládat počet kroků, o který se má motor ve zvoleném směru pohnout. Poskytuje výstupní proud až 2 A, který se řídí velikostí měřícího odporu připojeného k budiči. Pro volbu výstupního proudu je tedy možné využít potenciometru.

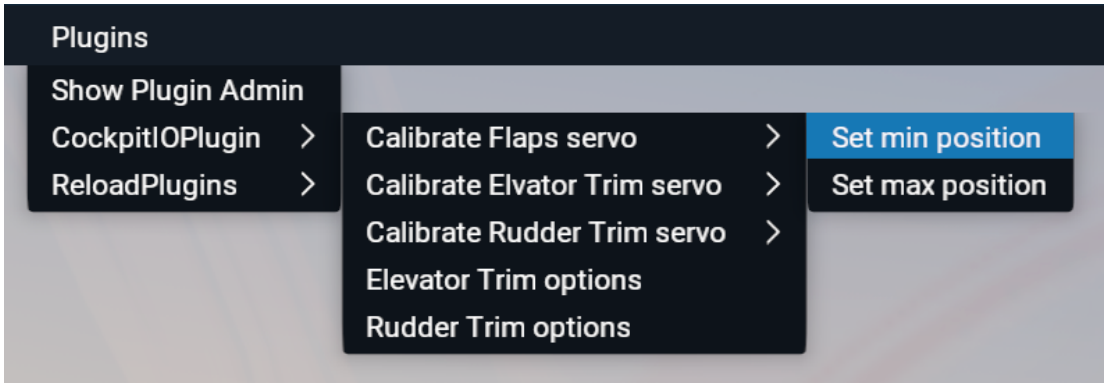
Logika pohonu motorů je stejná jako v případě unipolárních motorů. Typ motoru pro každé z vyvažovacích kol bude mít uživatel možnost zvolit v plug-inu.

7.4 Plug-in

Plug-in se skládá z několika částí. První z nich představuje periodicky volanou funkci, zjišťující současný stav letadla v simulátoru (stav autopilota a podvozku) a polohu virtuálních prvků, ovládaných simulátorem (osy polohy klapek a vyvažovacích kol). Druhá část představuje grafické rozhraní, v němž se dají nastavit jednotlivé parametry modulu.

V první části se využije API XPLMDataAccess poskytované X-Planem pro přístup k datům v simulátoru. Umožňuje vyčtení a změnu jak parametrů simulátoru, tak i přímo parametrů simulovaných letadel. Pomocí tohoto API se načte stav jednotlivých kol podvozku, autopilota i míra vysunutí klapek a vyvažovacích ploch. Tato data se převedou na formát zapsaný v report deskriptoru a odešlou v OUT-zprávách do I/O-modulu.

Druhá část je realizována pomocí rozhraní XPLMMenus, XPLMDisplay a XPLMGraphics. S využitím prvního se vytvoří lišta s menu, kde si uživatel může zvolit nastavení jednotlivých prvků připojených k desce. Pomocí posledních dvou se vytvoří okna s grafickým rozhraním, v nichž může uživatel nastavení provést.

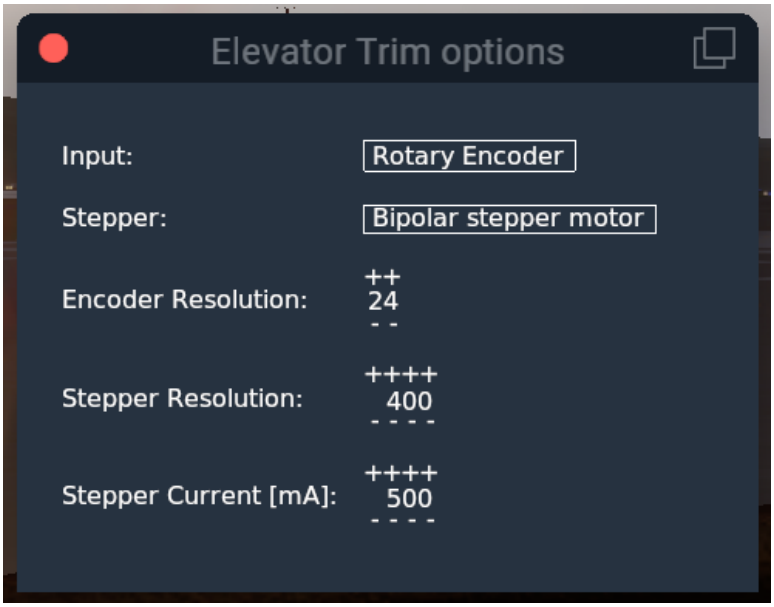


Obrázek 13: Lišta s menu nastavení plug-inu

Grafická rozhraní jsou velmi omezená. Dovolují pouze vytváření volně pohybujících se oken a v nich vykreslování jednoduchých přímek, vypisování čísel a textových řetězců. Žádné třídy pro typické grafické prvky, jako jsou tlačítka a textová pole, pro tato okna neexistují, a proto je třeba vytvořit vlastní. Tlačítka se vykreslí pomocí čtyř přímek, jejichž souřadnice se přepočítávají v závislosti na souřadnicích okna, v němž jsou umístěna. Kliknutí na tlačítko se vyhodnotí tak, že souřadnice kurzoru myši se neustále porovnávají se souřadnicemi všech tlačítek v okně. Pokud se kurzor umístí uvnitř obdélníku tlačítka a uživatel klikne na levé tlačítko myši, pro jehož vyhodnocení X-Plane poskytuje funkci, tlačítko se „stiskne“ a je možné provést akci. Složitější realizace vyžadují pole pro zadávání čísel (např. pro nastavení rozlišení motorů). Bylo využito několika tlačítek nad nastavovaným číslem i pod ním. Tlačítka nad číslem ho umožňují zvětšovat o tisíc, sto, deset a jedna. Tlačítka pod číslem ho naopak o dané hodnoty zmenšují. Díky tomu je možné nastavit libovolnou čtyřčíslicovou hodnotu.

V oknech se dají nastavit krajní polohy jednotlivých servomotorů. Uživatel nejdřív vybere, zda chce nastavit horní, nebo dolní polohu, a dále pomocí dvou tlačítek „nahoru“ a „dolů“ posouvá krajní polohu motoru. Takový způsob nastavení byl zvolen

proto, aby byl co nejintuitivnější a nevyžadoval žádné znalosti o tom, jak servomotory fungují. Také se dají nastavit parametry vyvažovacích kol. Všechny parametry lze zvolit v rámci jednoho okna. Zde si uživatel může vybrat snímací prvek (potenciometr, nebo rotační enkodér) a typ použitého krokového motoru (unipolární, bipolární nebo žádný). U enkodéru se uvádí jeho rozlišení a u motorů počet kroků za jedno otočení. Oba údaje jsou dostupné v dokumentacích ke kupovaným součástkám a pro uživatele by neměl být problém je dohledat.



Obrázek 14: Rozhraní pro nastavení parametrů trimovacího kola

Obrázek 15: Rozhraní pro nastavení dolní polohy servomotoru

K nastavení parametrů v modulu se využívá pět bytů označených itemem feature v report deskriptoru. První byte reprezentuje mód nastavení a uvádí informaci o tom, jaký parametr se momentálně nastavuje. Mód 1 reprezentuje nastavení krajních poloh servomotoru ukazatele klapek. Mód 2 reprezentuje nastavení servomotoru trimu kurzu. Dále podle stejného principu pokračují další módy pro všechny nastavované hodnoty. Do následujících čtyř feature bytů jsou uloženy samotné nastavované hodnoty. Aby se nastavování nemuselo po každém připojení provádět znovu, plug-in nastavené parametry ukládá do lokálního souboru. Jeho kopírováním se parametry dají přenášet na jiné počítače. Při zapnutí simulátoru se uložené hodnoty nastavení ze souboru načítají a jsou odeslány do modulu.

7.5 Vyhodnocení prototypu

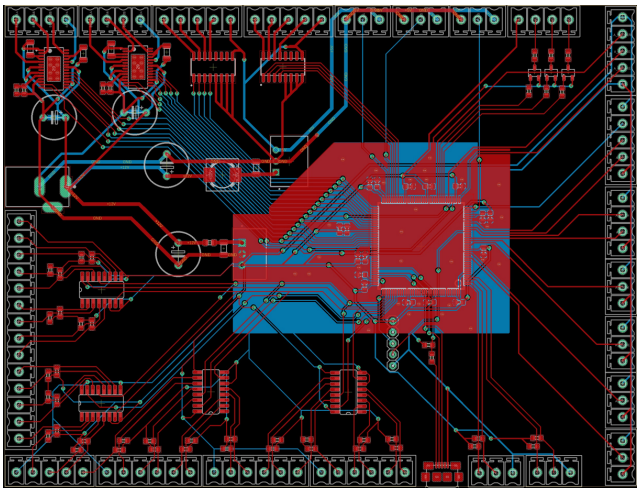
Prototyp posloužil jako testovací platforma pro vytvoření komunikační cesty mezi ovládacími prvky a simulátorem. Přenos zpráv přes USB-rozhraní se podařilo úspěšně zprovoznit na obou stranách a probíhá podle očekávání. Zařízení je počítačem rozpoznáno jako joystick, a tak ihned uživateli umožňuje ovládání většiny simulátorů. Dokonce je jako joystick rozpoznáno i launcherem aplikací Steam od společnosti Valve. Ten umožňuje kalibraci analogových os zařízení pro hry a simulátory, jež taková nastavení neobsahují.

Díky plug-inu se v simulátoru X-Plane navíc povedlo realizovat ovládání aktuátorů i jejich nastavení, umožňující velkou volnost při jejich výběru. Grafické rozhraní vytvořené v rámci plug-inu se ukázalo jako vhodné řešení. Nevyžaduje další vylepšení funkcionality, pouze hledání a opravu případných chyb.

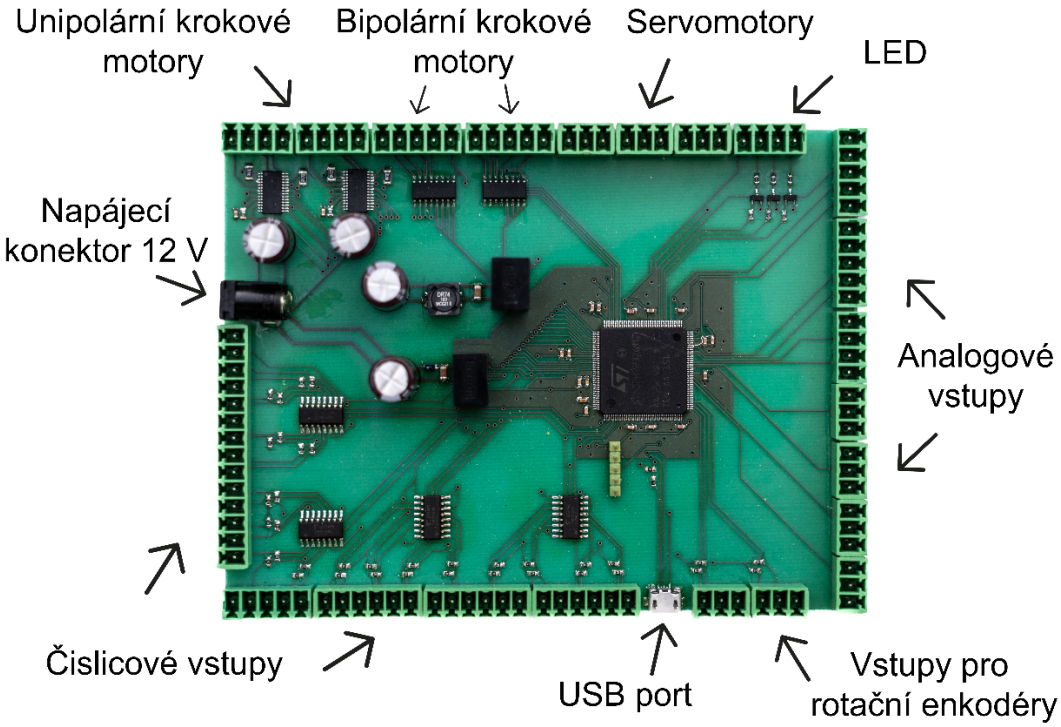
S připojením snímacích prvků a aktuátorů k prototypu žádné problémy nevznikly, nicméně bylo rozhodnuto o provedení několika změn ve výsledném zařízení.

8 Výsledné zařízení

Výsledný modul je navržen na desce plošných spojů, na jejíchž krajích jsou rozmístěny konektory pro připojení ovládacích prvků. Konektory jsou určeny jak pro připojení signálových vodičů, tak i pro napájení jednotlivých snímacích prvků a aktuátorů. Všechny prvky na desce byly zapojeny v souladu s aplikačními doporučeními udávanými výrobci těchto prvků. Vzhledem k současné čipové krizi nebylo možné si pro danou aplikaci pořídit mikrokontrolér, ten byl poskytnut přímo společností STMicroelectronics, která měla možnost zaslat několik vzorků STM32H723ZGT6. Jedná se o stejný model, jaký byl použit v prototypu. Schéma zapojení jednotlivých prvků je možné nalézt v příloze C.



Obrázek 14: Schéma DPS výsledného zařízení



Obrázek 15: Fotografie výsledného zařízení s popisem konektorů

8.1 Změny oproti prototypu

Ve výsledném zařízení bylo rozhodnuto vyměnit lineární regulátory, jež byly použity pro napájení jednotlivých prvků, za spínané regulátory R-78B5.0-2.0 (pro napájení servomotorů a unipolárních krokových motorů) a R-783.3-1.0 (pro napájení posuvných registrů a mikrokontroléru). Rozhodnutí bylo založeno převážně na mnohem lepší efektivitě spínaných regulátorů. Hlavním zdrojem napájení modulu bude zdroj na 12 V, který bude přiveden na ovládače bipolárních krokových motorů a spínané regulátory.

Druhou změnu představuje výměna posuvného registru pro ovládání světelných diod za tři tranzistory, ovládané třemi výstupy mikrokontroléru, a řídicí proud protékaný diodami. Posuvný registr s osmi výstupy je v tomto případě zbytečný a má komplikovanější zapojení.

Jako další byl vyměněn ovládač bipolárních krokových motorů. Nastavení výstupního proudu pomocí potenciometru se ukázalo jako nepraktické, navíc po uživateli vyžaduje zkušenosti s voltmetrem. Místo A4988 bylo proto rozhodnuto použít ovládače BD63511EFV-E2 s výstupním proudem až 1 A a s možností jeho nastavení vnějším referenčním napětím. To se dá realizovat pomocí digitálně-analogových převodníků mikrokontroléru a dalšího pole v grafickém rozhraní plug-inu, kde uživatel zadá požadovaný výstupní proud.

8.2 Testování funkčnosti

Při testování funkčnosti zařízení byly objeveny chyby v schematickém návrhu desky, které se projevily až po jejím osazení. První chybou byl jeden pin mikrokontroléru, který byl místo napájecího napětí připojen na zem a tak způsoboval zkrat. Tato chyba byla odstraněna pájením drátu, spojujícím daný pin s napájecím napětím. Další chybou bylo špatné zapojení posuvných registrů. Kvůli tomu došlo k jejich vzájemnému rušení na společném vodiči MISO a stavy digitálních vstupů nebylo možné načíst. Možnost připojení spínačů a přepínačů byla opět obnovena pájením propojovacích drátů. Opravené schéma zapojení pro budoucí výrobu zařízení je možné nalézt v příloze C na obrázku 29.

Zbylá funkcionalita byla zachována beze změn. Byla vyzkoušena postupným připojením jednotlivých elektronických prvků a zkoumáním, zda komunikace mezi nimi a simulátorem probíhá v pořádku. Dále bylo vyzkoušeno nastavení parametrů jednotlivých prvků v grafickém rozhraní plug-inu, zejména nově přidaného nastavení proudu bipolárních krokových motorů. Modul správně reagoval na změnu nastavení jednotlivých parametrů. Důkladněji byla vyzkoušena funkcionalita prvků sloužících pro osazení trimovacích kol. Jak bylo popsáno v podkapitole 7.3.6, chování trimovacích kol přímo závisí na zvolených elektronických prvcích. Bylo třeba vyzkoušet různé kombinace nastavení, aby se potvrdilo, že jejich chování odpovídá očekávanému. K vyzkoušení tohoto chování byla vytvořena improvizovaná trimovací kola spojením os enkodéru / potenciometru a krokového motoru, jak je to u reálných simulačních prvků. V obou případech se improvizované kolo chovalo dle očekávání. V případě použití rotačního enkodéru přesně kopírovalo směr pohybu virtuálního kola. V případě použití potenciometru se pohybovalo po směru aktuální polohy virtuálního kola, dokud polohy reálné a virtuální osy nesplývaly.

9 Srovnání s alternativami

V této kapitole porovnáme výsledné zařízení s alternativními výrobky, jejichž pomocí lze připojit vlastnoručně vyrobené ovládací prvky k leteckému simulátoru, popsanými v kapitole 3.

9.1 Jednoduchost použití a funkcionalita

V jednoduchosti použití je v rámci této práce navržený I/O-modul srovnatelný s profesionálními zařízeními společnosti FlightIllusion i jinými komerčně dostupnými I/O-moduly, které jsou typu plug'n'play. Nevyžaduje po uživateli žádné hlubší znalosti v oblasti elektroniky a zařízení se automaticky simulátory rozpoznává jako herní ovládač. Stačí pouze nakoupené prvky přímo zapojit do konektorů desky a o nic dalšího se už uživatel nemusí starat. Tím se výhodně liší od platforem typu RealSimControl, které po uživateli vyžadují základní chápání obvodů i toho, jak jednotlivé prvky fungují.

Samozřejmě pro platformy typu RealSimControl se uživatel může stále rozhodnout z důvodu jejich v podstatě neomezené funkcionality. Jelikož se jedná pouze o softwarové řešení, výsledný počet vstupních i výstupních bloků si uživatel určuje sám. Stejné platí i pro profesionální zařízení společnosti FlightIllusion, která podporují možnost podstatného navýšení počtu vstupů a výstupů, a uživatel tak není nijak omezen. Nicméně zařízení navržené v této práci bylo stavěno tak, aby poskytovalo univerzální možnosti pro připojení ovládacích prvků ultralehkých a lehkých vrtulových letounů, tím pádem by pro tyto účely mělo uživateli vystačit, větších možností by stejně nejspíše nevyužil.

Oproti jiným, neprofesionálním, komerčně dostupným I/O-modulům nabízí zařízení realizované v rámci této práce mnohem širší možnosti při práci se simulátorem X-Plane. Zároveň zachovává funkcionalitu podobnou alternativním modulům při práci s ostatními simulátory. Díky plug-inu je možné k desce připojovat krokové motory, servomotory a světelné diody. To umožňuje realizaci mnohem realističtějších prvků (např. trimovacích kol) než u jiných I/O-modulů. Zároveň se v jiných simulátorech zachovává základní funkcionalita s možností využití analogových a číslicových vstupů.

9.2 Cena

Cena součástek použitých pro výrobu jednoho zařízení činí 2 000 Kč a cena výroby desky plošných spojů, na jejímž základě je zařízení stavěno, je 1 000 Kč. Celkové náklady na výrobu jednoho zařízení tak činí 3 000 Kč, případně 120 eur. Při sériové výrobě by se v závislosti na počtu vyrobených kusů jak ceny součástek, tak i cena výroby o 10 až 20 % snížily.

To výsledné zařízení umísťuje mezi I/O-moduly s omezenou nebo žádnou podporou aktuátorů, jež jsou dvakrát až třikrát levnější, a profesionální zařízení společnosti FlightIllusion, jež si uživatel pořídí za trojnásobnou cenu. V případě platforem typu RealSimControl budeme předpokládat, že by cena byla velmi podobná, pokud by si uživatel chtěl postavit platformu s podobnou funkcionalitou, jakou nabízí

v práci navržené zařízení. Potřeboval by totiž nakoupit podobné nebo i stejné součástky a k tomu Arduino-desky.

10 Závěr

Cílem dané práce bylo navrhnout a realizovat modul rozhraní mezi leteckým simulátorem a ovládacím panelem, složeného z vlastnoručně vyrobených ovládacích prvků ultralehkých a lehkých vrtulových letounů.

V teoretické části práce byly popsány ovládací prvky takových letounů a elektronické prvky, pomocí kterých se dají vlastnoručně realizovat. Byly popsány současné možnosti připojení ovládacích prvků k simulátorům, jejich výhody i nedostatky. Z této rešerše byly odvozeny parametry navrhovaného modulu. Dále následoval popis komunikační cesty mezi modulem a simulátorem. Jako komunikační rozhraní pro připojení modulu k počítači bylo kvůli jeho rozšířenosti a jednoduchosti použití pro uživatele zvoleno rozhraní USB. Konkrétně bylo rozhodnuto modul naprogramovat jako zařízení kategorie USB HID. Komunikace se zařízeními této kategorie je nativně realizována v operačních systémech Windows a MacOs. Podrobný popis USB HID byl taktéž uveden v rámci teoretické části práce.

Praktická část se zabývala návrhem a realizací zařízení. Nejprve byl navržen a postaven prototyp na napájivém poli. Na prototypu bylo vyzkoušeno připojení jednotlivých ovládacích prvků a komunikace s počítačem. Pro komunikaci na straně simulátoru byl vytvořen plug-in pro simulátor X-Plane. Plug-in realizoval zpětnou komunikaci ze simulátoru do zařízení, což umožnilo ovládání aktuátorů. Zároveň bylo v rámci plug-inu vytvořeno grafické rozhraní, ve kterém se dají nastavovat jednotlivé parametry připojených ovládacích prvků. Celková funkcionalita prototypu odpovídala očekáváné a s malými změnami bylo realizováno výsledné zařízení na desce plošných spojů. Výsledné zařízení se nepovedlo navrhnout bez chyb, které se projevily až při jeho testování. Připojení prvků s analogovými výstupy, rotačních enkodérů, světelných diod i motorů proběhlo bez problémů. Zařízení fungovalo stejně dobře jako prototyp a plnilo svoji funkci. Chyba nastala u posuvných registrů, odpovídajících za připojení spínačů a přepínačů. Digitální vstupy se povedlo částečně zprovoznit a byl vytvořen nový návrh pro budoucí, opravenou verzi zařízení.

Na konci práce je navržené zařízení porovnáno s alternativami. Cenově i funkcionalitou se umístilo mezi moduly s omezenou podporou aktuátorů a profesionální zařízení od společnosti FlightIllusion. Vytváří tak novou, samostatnou kategorii zařízení pro připojení ovládacích prvků letounů.

Literatura

[1] GSA-010 I/O module – USB based [online]. [cit. 2022-01-07]. Dostupné z: https://www.flightillusion.com/interfacing/extender-cables-int/gsa-010-io-module-usb-based/

[2] GSA-055 Central Interface [online]. [cit. 2022-01-07]. Dostupné z: https://www.flightillusion.com/interfacing/central-interface-int/gsa-055-central-interface/

[3] RealSimControl [online]. [cit. 2022-02-07]. Dostupné z: https://realsimcontrol.com/index.html

[4] BU0836A 12-Bit Joystick Controller [online]. [cit. 2022-02-07]. Dostupné z: http://www.leobodnar.com/shop/index.php?main_page=product_info&cPath=66&products_id=204

[5] USB-AXES CARD [online]. [cit. 2022-02-07]. Dostupné z: https://www.opencockpits.com/index.php/en/iocards/available-cards/joystick

[6] USB CNC controller – PoKeys57U – flight simulators, automation [online]. [cit. 2022-02-07]. Dostupné z: https://www.poscope.com/product/pokeys57u/

[7] Plugin for low cost PoKeys USB and Ethernet universal I/O interfaces [online]. [cit. 2022-02-08]. Dostupné z: https://forums.x-plane.org/index.php?/forums/topic/62459-plugin-for-low-cost-pokeys-usb-and-ethernet-universal-io-interfaces/

[8] Chapter 6: Flight Controls. Pilot's Handbook of Aeronautical Knowledge [online]. Aviation Supplies & Academics, 2017, s. 12 [cit. 2022-02-08]. ISBN 9781619544741. Dostupné z: https://www.faa.gov/regulations_policies/handbooks_manuals/aviation/phak/media/08_phak_ch6.pdf

[9] Chapter 3: Basic Flight Maneuvers. Airplane Flying Handbook [online]. Aviation Supplies & Academics, 2021, s. 24 [cit. 2022-02-08]. ISBN 9781644250693. Dostupné z: https://www.faa.gov/regulations_policies/handbooks_manuals/aviation/airplane_handbook/media/05_afh_ch3.pdf

[10] NOVOTNÝ, Vojtěch. JAK SE ŘÍDÍ LETADLO [online]. In: . [cit. 2022-02-08]. Dostupné z: http://www.soukromypilot.cz/jak-se-ridi-letadlo.html

[11] 3D Printed DIY Flight Simulator Yoke Using Arduino [online]. [cit. 2022-02-08]. Dostupné z: https://www.thingiverse.com/thing:4855469

[12] DIY 3D Printed Flight Yoke for 2020 Microsoft Flight Simulator [online].

[cit. 2022-02-08]. Dostupné z: https://www.youtube.com/watch?v=nO-lanEc5vM

[13] Flight Simulator Joystick Info [online]. [cit. 2022-02-08]. Dostupné z: https://www.youtube.com/watch?app=desktop&v=XcKmBWGFUn8&t=0s

[14] 3D Printed DIY Flight Simulator Pedals (Rudder & Brake) Using Arduino [online]. [cit. 2022-02-08]. Dostupné z: https://www.thingiverse.com/thing:4874068

[15] CONRADIE, Danie. COMPLETE FLIGHT SIM CONTROLLER SET WITH 3D PRINTING AND HALL-EFFECT SENSORS [online]. [cit. 2022-02-08]. Dostupné z: https://hackaday.com/tag/rudder-pedals/

[16] BFF Motorised Trim Wheel [online]. [cit. 2022-02-08]. Dostupné z: http://bffsimulation.com/BFF_Motorised_Trim_Wheel.php

[17] Flight Simulator 06 -Trim Wheel DIY [online]. [cit. 2022-02-08]. Dostupné z: https://www.youtube.com/watch?v=V_fBZ1K7XSw

[18] Trim Wheel for Flight Simulators [online]. [cit. 2022-02-08]. Dostupné z: https://www.thingiverse.com/thing:4290141

[19] GTX MAX Cessna [online]. [cit. 2022-02-08]. Dostupné z: https://flypfc.com/shop/training-systems/fixed-wing-simulators/max-cockpit-systems/gtx-max-cessna/

[20] Flap Control Panel [online]. [cit. 2022-02-08]. Dostupné z: https://cessna172sim.allanglen.com/docs/flap-control-panel/

[21] Easily Install and Program your very Own "FLAPS Control Panel" [online]. [cit. 2022-02-08]. Dostupné z: https://www.desktopaviator.com/Products/Model_2530/index.htm

[22] JOHNSTON, Matthew. The Basics of the Cockpit [online]. [cit. 2022-02-07]. Dostupné z: https://calaero.edu/the-basics-of-the-cockpit/

[23] Throttle Quadrant and Trim Wheel: A modular 3D printed throttle quadrant and trim wheel for use with flight simulators [online]. [cit. 2022-05-08]. Dostupné z https://create.arduino.cc/projecthub/markbennettuk/throttle-quadrant-and-trim-wheel-d746e8

[24] 10 min builds: Flight sim throttle. Create your own Flight sim Throttle with Arduino [online]. [cit. 2022-02-08]. Dostupné z: https://www.youtube.com/watch?v=Vw-ns5Il6eQ

[25] Aircraft Landing Gear [online]. [cit. 2022-02-08]. Dostupné z: https://www.cfinotebook.net/notebook/operation-of-aircraft-systems/aircraft-landing-gear#retractable-landing-gear

[26] Landing Gear Lever DIY FSX [online]. [cit. 2022-02-08]. Dostupné z: https://www.youtube.com/watch?v=yK1iuslXfhA

[27] Flight Simulator Gear Switch [online]. [cit. 2022-02-08]. Dostupné z: https://www.thingiverse.com/thing:4073959

[28] Aviation Fuel Systems [online]. [cit. 2022-02-08]. Dostupné z: https://www.cfinotebook.net/notebook/operation-of-aircraft-systems/aviation-fuel

[29] Easily Install and Program your very Own Fuel Select Panel [online]. [cit. 2022-02-08]. Dostupné z: http://www.desktopaviator.com/Products/Model_2390/index.htm

[30] Ignition Systems [online]. [cit. 2022-02-08]. Dostupné z: https://www.cfinotebook.net/notebook/operation-of-aircraft-systems/ignition

[31] 5 Position Spring-Return Key Switch [online]. [cit. 2022-02-08]. Dostupné z: https://cessna172sim.allanglen.com/docs/key-switch/

[32] LETECKÝ KOMUNIKAČNÍ PANEL – PROFESIONÁLNÍ SIMULAČNÍ PANEL S PŘEPÍNAČI [online]. [cit. 2022-02-08]. Dostupné z: https://www.logitechg.com/cs-cz/products/flight/flight-simulator-switch-panel.html

[33] HW-GEN002 [online]. [cit. 2022-02-08]. Dostupné z: https://www.flightillusion.com/controls/switches-controls/hw-gen002/

[34] HW-GEN005 [online]. [cit. 2022-02-08]. Dostupné z: https://www.flightillusion.com/controls/switches-controls/hw-gen005/

[35] NANDA, Umakanta a Shushant PATTNAIK. Universal Asynchronous Receiver and Transmitter (UART). 3. vyd. Koimbator, Indie: IEEE, 2016. ISBN 978-1-4673-9207-5.

[36] DAWOUD, Shenouda a Peter DAWOUD. Serial Communication Protocols and Standards RS232/485, UART/USART, SPI, USB, INSTEON, Wi-Fi and WiMAX. Denmark: River Publishers, 2020. ISBN 9788770221542.

[37] Komunikace mikrokontroléru s PC po RS232, USART, USB a převodníky UART <-> USB, průmyslové sběrnice na dlouhou vzdálenost RS422 a RS485: B2B34MIK [online]. 41 [cit. 2022-02-08]. Dostupné z: https://moodle.fel.cvut.cz/pluginfile.php/324697/mod_resource/content/1/12_MIK_2021.pdf

[38] Universal Serial Bus (USB): Device Class Definition for Human Interface Devices (HID) [online]. In: . s. 97 [cit. 2021-12-01]. Dostupné z: https://www.usb.org/sites/default/files/hid1_11.pdf

[39] Libusb/hidapi: A Simple library for communicating with USB and Bluetooth HID devices on Linux, Mac and Windows. [online]. [cit. 2022-05-09]. Dostupné z: https://github.com/libusb/hidapi

[40] Developing Plugins [online]. 13. 2. 2019 [cit. 2022-05-09]. Dostupné z: https://developer.x-plane.com/article/developing-plugins/

[41] Getting a Vendor ID [online]. [cit. 2022-05-09]. Dostupné z: https://www.usb.org/getting-vendor-id

[42] ST Community: Has anyone managed to sub-license a USB PID from ST? [online]. [cit. 2022-05-09]. Dostupné z: Getting a Vendor ID [online]. [cit. 2022-05-09]. Dostupné z: https://www.usb.org/getting-vendor-id

[43] USB hardware and PCB guidelines using STM32 MCUs [online]. 2019 [cit. 2022-05-09]. Dostupné z: https://www.st.com/resource/en/application_note/dm00296349-usb-hardware-and-pcb-guidelines-using-stm32-mcus-stmicroelectronics.pdf

[44] STM32H7 - GPTIM: Advanced-control, general-purpose and basic timers [online]. [cit. 2022-05-09]. Dostupné z: https://www.st.com/content/ccc/resource/training/technical/product_training/group0/de/9a/c0/b2/74/7b/47/8c/STM32H7-WDG_TIMERS-General_Purpose_Timer_GPTIM/files/STM32H7-WDG_TIMERS-General_Purpose_Timer_GPTIM.pdf/_jcr_content/translations/en.STM32H7-WDG_TIMERS-General_Purpose_Timer_GPTIM.pdf

[45] Difference Between 4-Wire, 6-Wire and 8-Wire Stepper Motors [online]. 10. 1. 2019 [cit. 2022-05-09]. Dostupné z: https://knowledge.ni.com/KnowledgeArticleDetails?id=kA00Z000000PAkPSAW&l=cs-CZ

Příloha A

Tabulka 2: Datová část main itemů [38]

Main item tag	One-Byte Prefix (*nn* represents size value)	Valid Data	
Input	1000 00 *nn*	Bit 0	{Data (0) | Constant (1)}
		Bit 1	{Array (0) | Variable (1)}
		Bit 2	{Absolute (0) | Relative (1)}
		Bit 3	{No Wrap (0) | Wrap (1)}
		Bit 4	{Linear (0) | Non Linear (1)}
		Bit 5	{Preferred State (0) | No Preferred (1)}
		Bit 6	{No Null position (0) | Null state(1)}
		Bit 7	Reserved (0)
		Bit 8	{Bit Field (0) | Buffered Bytes (1)}
		Bit 31-9	Reserved (0)
Output	1001 00 *nn*	Bit 0	{Data (0) | Constant (1)}
		Bit 1	{Array (0) | Variable (1)}
		Bit 2	{Absolute (0) | Relative (1)}
		Bit 3	{No Wrap (0) | Wrap (1)}
		Bit 4	{Linear (0) | Non Linear (1)}
		Bit 5	{Preferred State (0) | No Preferred (1)}
		Bit 6	{No Null position (0) | Null state(1)}
		Bit 7	{Non Volatile (0) | Volatile (1)}
		Bit 8	{Bit Field (0) | Buffered Bytes (1)}
		Bit 31-9	Reserved (0)
Feature	1011 00 *nn*	Bit 0	{Data (0) | Constant (1)}
		Bit 1	{Array (0) | Variable (1)}
		Bit 2	{Absolute (0) | Relative (1)}
		Bit 3	{No Wrap (0) | Wrap (1)}
		Bit 4	{Linear (0) | Non Linear (1)}
		Bit 5	{Preferred State (0) | No Preferred (1)}
		Bit 6	{No Null position (0) | Null state(1)}
		Bit 7	{Non Volatile (0) | Volatile (1)}
		Bit 8	{Bit Field (0) | Buffered Bytes (1)}
		Bit 31-9	Reserved (0)
Collection	1010 00 *nn*	0x00	Physical (group of axes)
		0x01	Application (mouse, keyboard)
		0x02	Logical (interrelated data)
		0x03	Report
		0x04	Named Array
		0x05	Usage Switch
		0x06	Usage Modifier
		0x07-0x7F	Reserved

		0x80-0xFF	Vendor-defined
End Collection	1100 00 *nn*	Not applicable.	Closes an item collection.

Tabulka 3: Seznam local itemů [38]

Tag	One-Byte Prefix (*nn* represents size value)	Description
Usage	0000 10 *nn*	Usage index for an item usage; represents a suggested usage for the item or collection. In the case where an item represents multiple controls, a Usage tag may suggest a usage for every variable or element in an array.
Usage Minimum	0001 10 *nn*	Defines the starting usage associated with an array or bitmap.
Usage Maximum	0010 10 *nn*	Defines the ending usage associated with an array or bitmap.
Designator Index	0011 10 *nn*	Determines the body part used for a control. Index points to a designator in the Physical descriptor.
Designator Minimum	0100 10 *nn*	Defines the index of the starting designator associated with an array or bitmap.
Designator Maximum	0101 10 *nn*	Defines the index of the ending designator associated with an array or bitmap.
String Index	0111 10 *nn*	String index for a String descriptor; allows a string to be associated with a particular item or control.
String Minimum	1000 10 *nn*	Specifies the first string index when assigning a group of sequential strings to controls in an array or bitmap.
String Maximum	1001 10 *nn*	Specifies the last string index when assigning a group of sequential strings to controls in an array or bitmap.
Delimiter	1010 10 *nn*	Defines the beginning or end of a set of local items (1 = open set, 0 = close set).
Reserved	1010 10 *nn* to 1111 10 *nn*	Reserved.

Tabulka 4: Seznam global itemů [38]

Global item tag	One-Byte Prefix (*nn* represents size value)	Description
Usage Page	0000 01 *nn*	Unsigned integer specifying the current Usage Page. Since a usage are 32 bit values, Usage Page items can be used to conserve space in a report descriptor by setting the high order 16 bits of a subsequent usages. Any usage that follows which is defines 16 bits or less is interpreted as a Usage ID and concatenated with the Usage Page to form a 32 bit Usage.
Logical Minimum	0001 01 *nn*	Extent value in logical units. This is the minimum value that a variable or array item will report. For example, a mouse reporting x position values from 0 to 128 would have a Logical Minimum of 0 and a Logical Maximum of 128.
Logical Maximum	0010 01 *nn*	Extent value in logical units. This is the maximum value that a variable or array item will report.
Physical Minimum	0011 01 *nn*	Minimum value for the physical extent of a variable item. This represents the Logical Minimum with units applied to it.
Physical Maximum	0100 01 *nn*	Maximum value for the physical extent of a variable item.
Unit Exponent	0101 01 *nn*	Value of the unit exponent in base 10. See the table later in this section for more information.
Unit	0110 01 *nn*	Unit values.
Report Size	0111 01 *nn*	Unsigned integer specifying the size of the report fields in bits. This allows the parser to build an item map for the report handler to use. For more information, see Section 8: Report Protocol.
Report ID	1000 01 *nn*	Unsigned value that specifies the Report ID. If a Report ID tag is used anywhere in Report descriptor, all data reports for the device are preceded by a single byte ID field. All items succeeding the first Report ID tag but preceding a second Report ID tag are included in a report prefixed by a 1-byte ID. All items succeeding the second but preceding a third Report ID tag are included in a second report prefixed by a second ID, and so on. This Report ID value indicates the prefix added to a particular report. For example, a Report descriptor could define a 3-byte report with a Report ID of 01. This device would generate a 4-byte data report in which the first byte is 01. The device may also generate other reports, each with a unique ID. This allows the host to distinguish different types of reports arriving over a single interrupt in pipe. And allows the device to distinguish different types of reports arriving over a single interrupt out pipe. Report ID zero is reserved and should not be used.
Report Count	1001 01 *nn*	Unsigned integer specifying the number of data fields for the item; determines how many fields are included in the report for this particular item (and consequently how many bits are added to the report).
Push	1010 01 *nn*	Places a copy of the global item state table on the stack.
Pop	1011 01 *nn*	Replaces the item state table with the top structure from the stack.
Reserved	1100 01 *nn* to 1111 01 *nn*	Range reserved for future use.

Příloha B

Report descriptor I/O-modulu:

```
0x05, 0x01,                    // USAGE_PAGE (Generic Desktop)
0x09, 0x04,                    // USAGE (Joystick)
0xa1, 0x01,                    // COLLECTION (Application)
0xa1, 0x02,                    //   COLLECTION (Logical)
0x06, 0x00, 0xff,              //     USAGE_PAGE (Generic Desktop)
0x09, 0x00,                    //     USAGE (Undefined)
0x15, 0x00,                    //     LOGICAL_MINIMUM (0)
0x26, 0xff, 0x00,              //     LOGICAL_MAXIMUM (255)
0x75, 0x08,                    //     REPORT_SIZE (8)
0x95, 0x01,                    //     REPORT_COUNT (1)
0xb1, 0x22,                    //     FEATURE (Data,Var,Abs,NPrf)
0x09, 0x01,                    //     USAGE (Vendor Usage 1)
0x15, 0x00,                    //     LOGICAL_MINIMUM (0)
0x27, 0xff, 0xff, 0x00, 0x00,  //     LOGICAL_MAXIMUM (65535)
0x75, 0x10,                    //     REPORT_SIZE (16)
0x95, 0x02,                    //     REPORT_COUNT (2)
0xb1, 0x22,                    //     FEATURE (Data,Var,Abs,NPrf)
0xc0,                          //   END_COLLECTION
0x05, 0x02,                    //   USAGE_PAGE (Simulation Controls)
0xa1, 0x00,                    //   COLLECTION (Physical)
0x09, 0xb8,                    //     USAGE (Elevator)
0x15, 0x00,                    //     LOGICAL_MINIMUM (0)
0x26, 0xff, 0x0f,              //     LOGICAL_MAXIMUM (4095)
0x75, 0x0c,                    //     REPORT_SIZE (12)
0x95, 0x01,                    //     REPORT_COUNT (1)
0x81, 0x02,                    //     INPUT (Data,Var,Abs)
0x75, 0x04,                    //     REPORT_SIZE (4)
0x81, 0x03,                    //     INPUT (Cnst,Var,Abs)
0x09, 0xb0,                    //     USAGE (Aileron)
0x15, 0x00,                    //     LOGICAL_MINIMUM (0)
0x26, 0xff, 0x0f,              //     LOGICAL_MAXIMUM (4095)
0x75, 0x0c,                    //     REPORT_SIZE (12)
0x95, 0x01,                    //     REPORT_COUNT (1)
0x81, 0x02,                    //     INPUT (Data,Var,Abs)
0x75, 0x04,                    //     REPORT_SIZE (4)
0x81, 0x03,                    //     INPUT (Cnst,Var,Abs)
0xc0,                          //   END_COLLECTION
0xa1, 0x00,                    //   COLLECTION (Physical)
0x09, 0xba,                    //     USAGE (Rudder)
0x15, 0x00,                    //     LOGICAL_MINIMUM (0)
0x26, 0xff, 0x0f,              //     LOGICAL_MAXIMUM (4095)
0x75, 0x0c,                    //     REPORT_SIZE (12)
0x95, 0x01,                    //     REPORT_COUNT (1)
0x81, 0x02,                    //     INPUT (Data,Var,Abs)
0x75, 0x04,                    //     REPORT_SIZE (4)
0x81, 0x03,                    //     INPUT (Cnst,Var,Abs)
0x09, 0xbf,                    //     USAGE (Toe Brake)
0x15, 0x00,                    //     LOGICAL_MINIMUM (0)
0x26, 0xff, 0x0f,              //     LOGICAL_MAXIMUM (4095)
0x75, 0x0c,                    //     REPORT_SIZE (12)
0x95, 0x01,                    //     REPORT_COUNT (1)
```

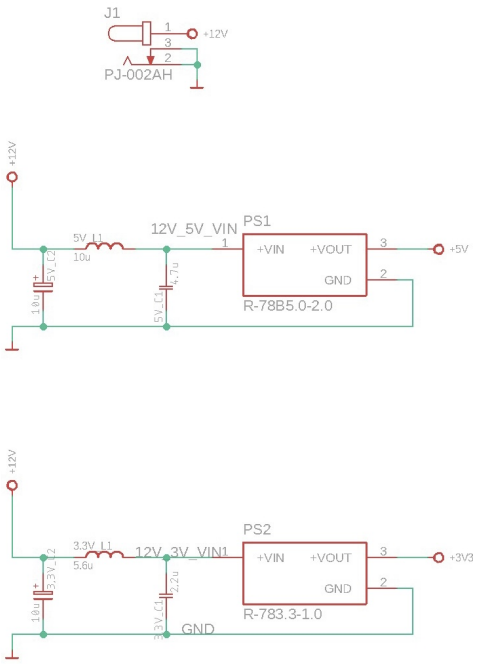
```
0x81, 0x02,                    //    INPUT (Data,Var,Abs)
0x75, 0x04,                    //    REPORT_SIZE (4)
0x81, 0x03,                    //    INPUT (Cnst,Var,Abs)
0x09, 0xbf,                   //    USAGE (Toe Brake)
0x15, 0x00,                    //    LOGICAL_MINIMUM (0)
0x26, 0xff, 0x0f,              //    LOGICAL_MAXIMUM (4095)
0x75, 0x0c,                    //    REPORT_SIZE (12)
0x95, 0x01,                    //    REPORT_COUNT (1)
0x81, 0x02,                    //    INPUT (Data,Var,Abs)
0x75, 0x04,                    //    REPORT_SIZE (4)
0x81, 0x03,                    //    INPUT (Cnst,Var,Abs)
0xc0,                        //  END_COLLECTION
0xa1, 0x00,                    //  COLLECTION (Physical)
0x09, 0xbb,                   //    USAGE (Throttle)
0x15, 0x00,                    //    LOGICAL_MINIMUM (0)
0x26, 0xff, 0x0f,              //    LOGICAL_MAXIMUM (4095)
0x75, 0x0c,                    //    REPORT_SIZE (12)
0x95, 0x01,                    //    REPORT_COUNT (1)
0x81, 0x22,                    //    INPUT (Data,Var,Abs,NPrf)
0x75, 0x04,                    //    REPORT_SIZE (4)
0x81, 0x03,                    //    INPUT (Cnst,Var,Abs)
0x0b, 0x36, 0x00, 0x01, 0x00,  //    USAGE (Generic Desktop:Slider)
0x15, 0x00,                    //    LOGICAL_MINIMUM (0)
0x26, 0xff, 0x0f,              //    LOGICAL_MAXIMUM (4095)
0x75, 0x0c,                    //    REPORT_SIZE (12)
0x95, 0x01,                    //    REPORT_COUNT (1)
0x81, 0x22,                    //    INPUT (Data,Var,Abs,NPrf)
0x75, 0x04,                    //    REPORT_SIZE (4)
0x81, 0x03,                    //    INPUT (Cnst,Var,Abs)
0x0b, 0x36, 0x00, 0x01, 0x00,  //    USAGE (Generic Desktop:Slider)
0x15, 0x00,                    //    LOGICAL_MINIMUM (0)
0x26, 0xff, 0x0f,              //    LOGICAL_MAXIMUM (4095)
0x75, 0x0c,                    //    REPORT_SIZE (12)
0x95, 0x01,                    //    REPORT_COUNT (1)
0x81, 0x22,                    //    INPUT (Data,Var,Abs,NPrf)
0x75, 0x04,                    //    REPORT_SIZE (4)
0x81, 0x03,                    //    INPUT (Cnst,Var,Abs)
0xc0,                        //  END_COLLECTION
0xa1, 0x00,                    //  COLLECTION (Physical)
0x09, 0xc5,                    //    USAGE (Brake)
0x15, 0x00,                    //    LOGICAL_MINIMUM (0)
0x26, 0xff, 0x0f,              //    LOGICAL_MAXIMUM (4095)
0x75, 0x0c,                    //    REPORT_SIZE (12)
0x95, 0x01,                    //    REPORT_COUNT (1)
0x81, 0x22,                    //    INPUT (Data,Var,Abs,NPrf)
0x75, 0x04,                    //    REPORT_SIZE (4)
0x81, 0x03,                    //    INPUT (Cnst,Var,Abs)
0xc0,                        //  END_COLLECTION
0xa1, 0x00,                    //  COLLECTION (Physical)
0x05, 0x09,                    //    USAGE_PAGE (Button)
0x19, 0x01,                    //    USAGE_MINIMUM (Button 1)
0x29, 0x20,                    //    USAGE_MAXIMUM (Button 32)
0x15, 0x00,                    //    LOGICAL_MINIMUM (0)
0x25, 0x01,                    //    LOGICAL_MAXIMUM (1)
0x95, 0x20,                    //    REPORT_COUNT (32)
0x75, 0x01,                    //    REPORT_SIZE (1)
```

```
0x81, 0x22,                    //    INPUT (Data,Var,Abs,NPrf)
0xc0,                          //  END_COLLECTION
0xa1, 0x01,                    //  COLLECTION (Application)
0x05, 0x08,                    //    USAGE_PAGE (LEDs)
0x19, 0x01,                    //    USAGE_MINIMUM (Num Lock)
0x29, 0x02,                    //    USAGE_MAXIMUM (Caps Lock)
0x95, 0x08,                    //    REPORT_COUNT (8)
0x75, 0x01,                    //    REPORT_SIZE (1)
0x91, 0x22,                    //    OUTPUT (Data,Var,Abs,NPrf) //first OUT byte LEDs
0xc0,                          //  END_COLLECTION
0xa1, 0x00,                    //  COLLECTION (Physical)
0x0b, 0x36, 0x00, 0x01, 0x00,  //    USAGE (Generic Desktop:Slider)
0x15, 0x00,                    //    LOGICAL_MINIMUM (0)
0x26, 0xff, 0x0f,              //    LOGICAL_MAXIMUM (4095)
0x75, 0x0c,                    //    REPORT_SIZE (12)
0x95, 0x01,                    //    REPORT_COUNT (1)
0x81, 0x02,                    //    INPUT (Data,Var,Abs)
0x75, 0x04,                    //    REPORT_SIZE (4)
0x81, 0x03,                    //    INPUT (Cnst,Var,Abs)
0x0b, 0x02, 0x00, 0x00, 0xff,  //    USAGE (Vendor Defined Page 1:Vendor Usage 2)
0x15, 0x00,                    //    LOGICAL_MINIMUM (0)
0x26, 0xff, 0x00,              //    LOGICAL_MAXIMUM (255)
0x75, 0x08,                    //    REPORT_SIZE (8)
0x95, 0x01,                    //    REPORT_COUNT (1)
0x91, 0x22,                    //    OUTPUT (Data,Var,Abs,NPrf) //second OUT byte FLAPS
0xc0,                          //  END_COLLECTION
0xa1, 0x00,                    //  COLLECTION (Physical)
0x0b, 0x36, 0x00, 0x01, 0x00,  //    USAGE (Generic Desktop:Slider)
0x15, 0x00,                    //    LOGICAL_MINIMUM (0)
0x26, 0xff, 0x0f,              //    LOGICAL_MAXIMUM (4095)
0x75, 0x0c,                    //    REPORT_SIZE (12)
0x95, 0x01,                    //    REPORT_COUNT (1)
0x81, 0x02,                    //    INPUT (Data,Var,Abs)
0x75, 0x04,                    //    REPORT_SIZE (4)
0x81, 0x03,                    //    INPUT (Cnst,Var,Abs)
0x0b, 0xb3, 0x00, 0x02, 0x00,  //  USAGE (Simulation Controls:Auto-pilot enable)
0x15, 0x00,                    //  LOGICAL_MINIMUM (0)
0x25, 0x01,                    //  LOGICAL_MAXIMUM (1)
0x75, 0x01,                    //    REPORT_SIZE (1)
0x95, 0x01,                    //    REPORT_COUNT (1)
0x91, 0x22,                    //  OUTPUT (Data,Var,Abs,NPrf)
0x75, 0x07,                    //  REPORT_SIZE (7)
0x91, 0x23,                    //    OUTPUT (Cnst,Var,Abs,NPrf)
0x0b, 0x02, 0x00, 0x00, 0xff,  //  USAGE (Vendor Defined Page 1:Vendor Usage 2)
0x15, 0x00,                    //    LOGICAL_MINIMUM (0)
0x27, 0xff, 0xff, 0x00, 0x00,  //    LOGICAL_MAXIMUM (65535)
0x75, 0x10,                    //    REPORT_SIZE (16)
0x95, 0x01,                    //    REPORT_COUNT (1)
0x91, 0x22,                    //    OUTPUT (Data,Var,Abs,NPrf)
0xc0,                          //  END_COLLECTION
0xa1, 0x00,                    //  COLLECTION (Physical)
0x0b, 0x36, 0x00, 0x01, 0x00,  //    USAGE (Generic Desktop:Slider)
0x15, 0x00,                    //    LOGICAL_MINIMUM (0)
0x26, 0xff, 0x0f,              //    LOGICAL_MAXIMUM (4095)
0x75, 0x0c,                    //    REPORT_SIZE (12)
0x95, 0x01,                    //    REPORT_COUNT (1)
```

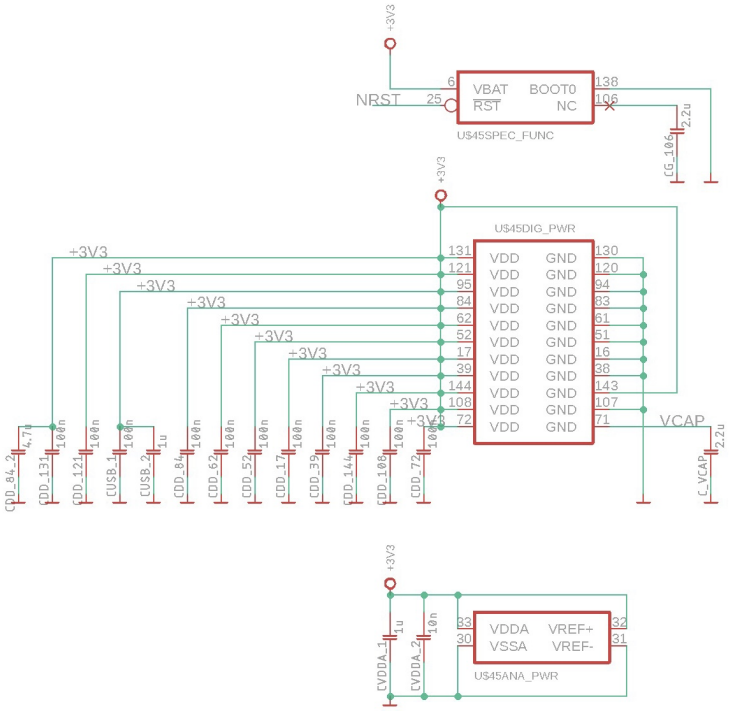
```
0x81, 0x02,                    //    INPUT (Data,Var,Abs)
0x75, 0x04,                    //    REPORT_SIZE (4)
0x81, 0x03,                    //    INPUT (Cnst,Var,Abs)
0x0b, 0xb3, 0x00, 0x02, 0x00,  //   USAGE (Simulation Controls:Auto-pilot enable)
0x15, 0x00,                    //   LOGICAL_MINIMUM (0)
0x25, 0x01,                    //   LOGICAL_MAXIMUM (1)
0x75, 0x01,                    //    REPORT_SIZE (1)
0x95, 0x01,                    //    REPORT_COUNT (1)
0x91, 0x22,                    //   OUTPUT (Data,Var,Abs,NPrf)
0x75, 0x07,                    //   REPORT_SIZE (7)
0x91, 0x23,                    //    OUTPUT (Cnst,Var,Abs,NPrf)
0x0b, 0x02, 0x00, 0x00, 0xff,  //   USAGE (Vendor Defined Page 1:Vendor Usage 2)
0x15, 0x00,                    //    LOGICAL_MINIMUM (0)
0x27, 0xff, 0xff, 0x00, 0x00,  //    LOGICAL_MAXIMUM (65535)
0x75, 0x10,                    //    REPORT_SIZE (16)
0x95, 0x01,                    //    REPORT_COUNT (1)
0x91, 0x22,                    //    OUTPUT (Data,Var,Abs,NPrf)
0xc0,                        //   END_COLLECTION
0xc0                         // END_COLLECTION
```

Příloha C

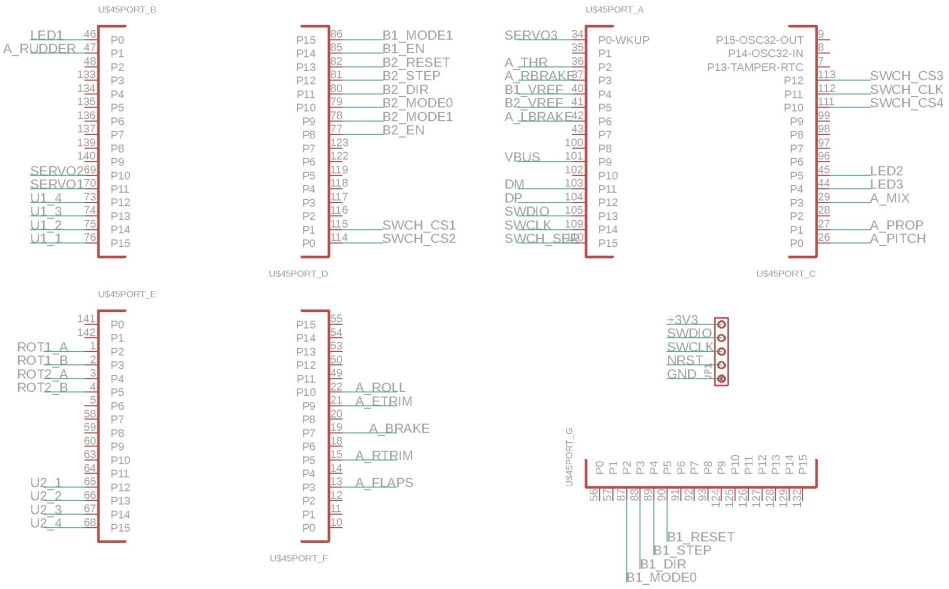
V této příloze se nachází schémata zapojení jednotlivých částí výsledného zařízení. Pro reprezentaci některých prvků byly požity modely se stejným očíslováním pinů a stejnými fyzickými rozměry, ale odlišnými názvy. Proto ne všechny názvy jednotlivých prvků ve schématu odpovídají názvům uvedeným v textu a ne všechny názvy pinů odpovídají reálným.



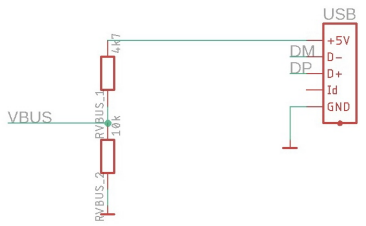
Obrázek 16: Schéma napájení desky



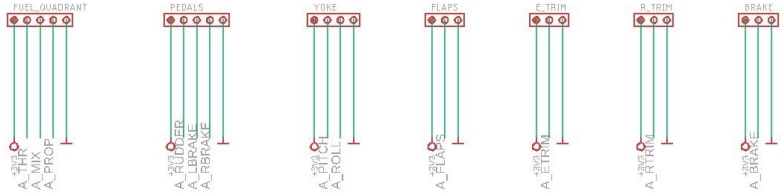
Obrázek 17: Schéma napájení mikrokontroléru



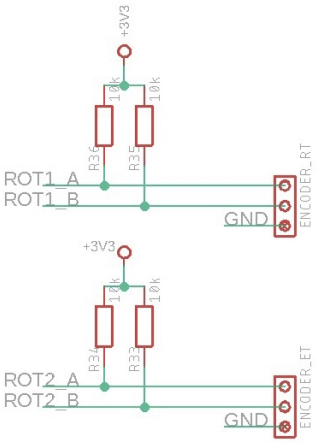
Obrázek 18: Schéma zapojení vstupních, výstupních a programovacích pinů mikrokontroléru



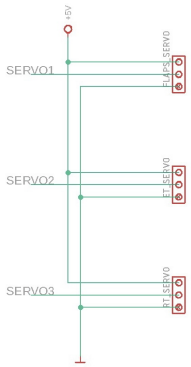
Obrázek 19: Schéma zapojení USB portu



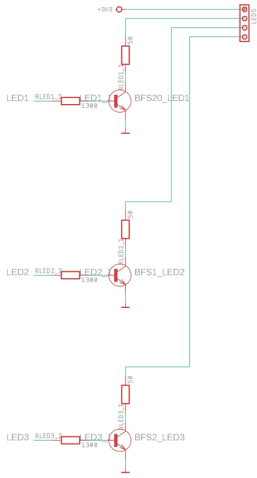
Obrázek 20: Schéma zapojení konektorů pro analogové vstupy



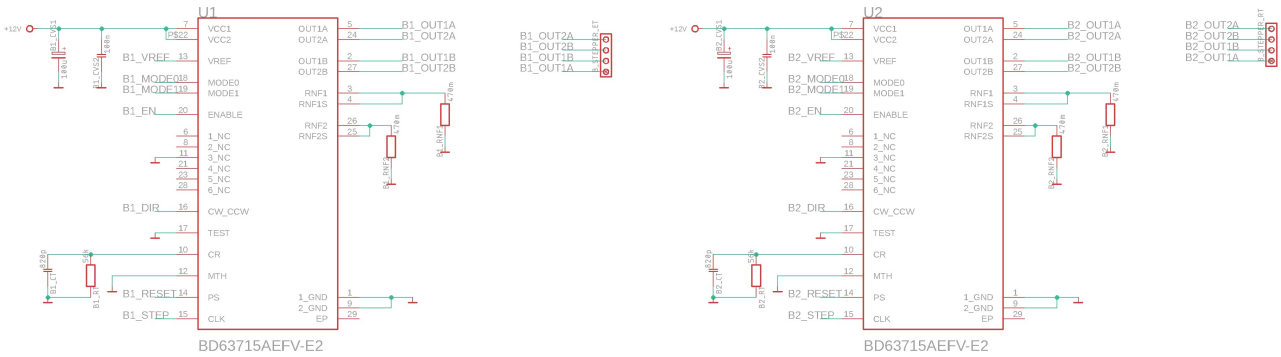
Obrázek 21: Schéma zapojení konektorů pro připojení rotačních enkodérů



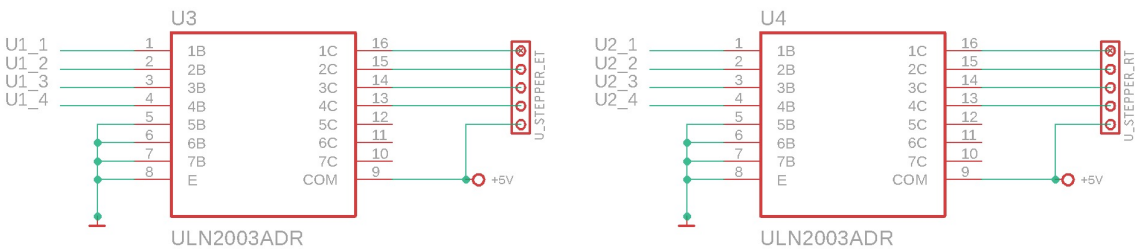
Obrázek 22: Schéma zapojení konektorů pro připojení servomotorů



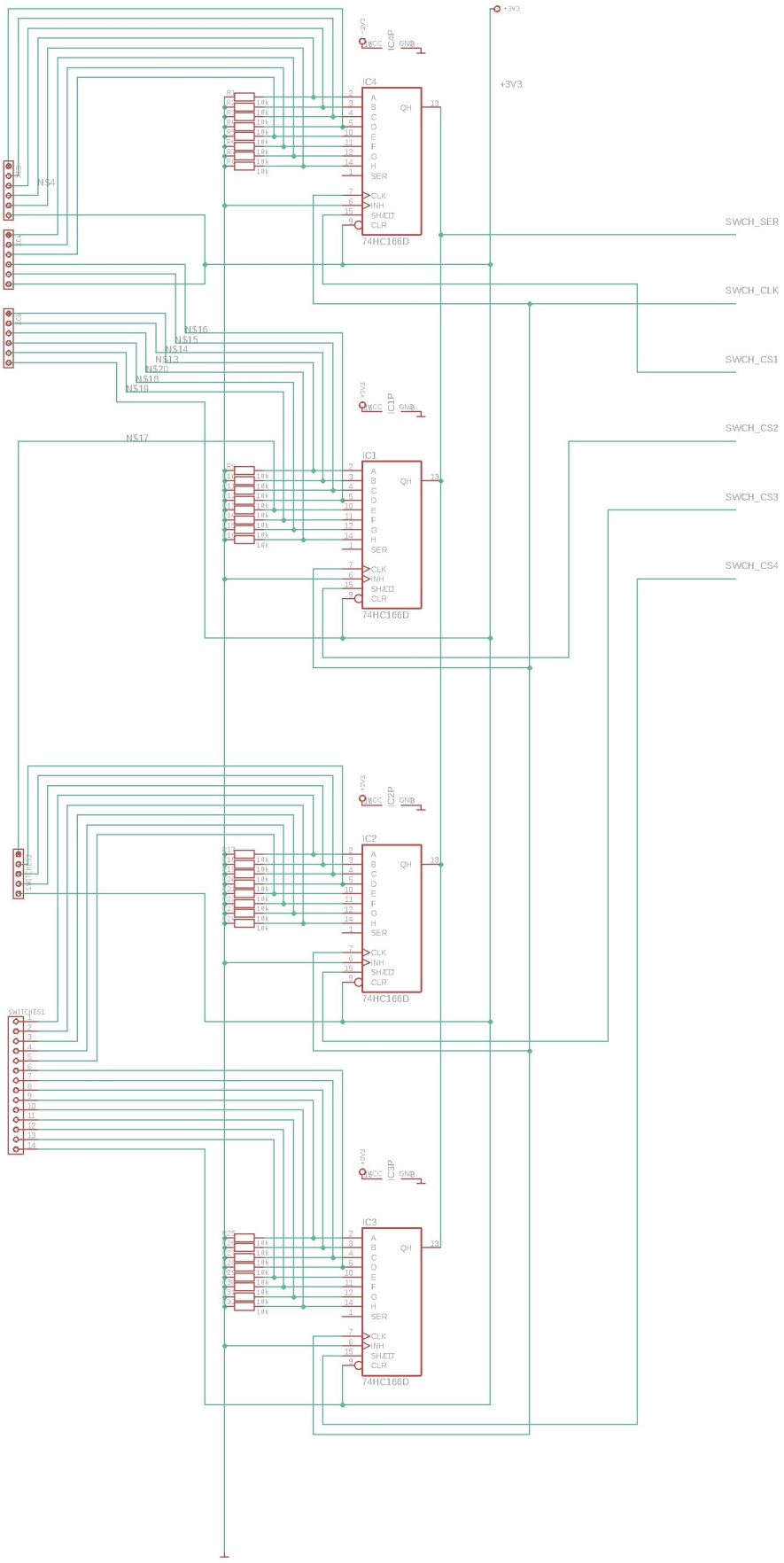
Obrázek 23: Schéma zapojení konektorů pro připojení LED



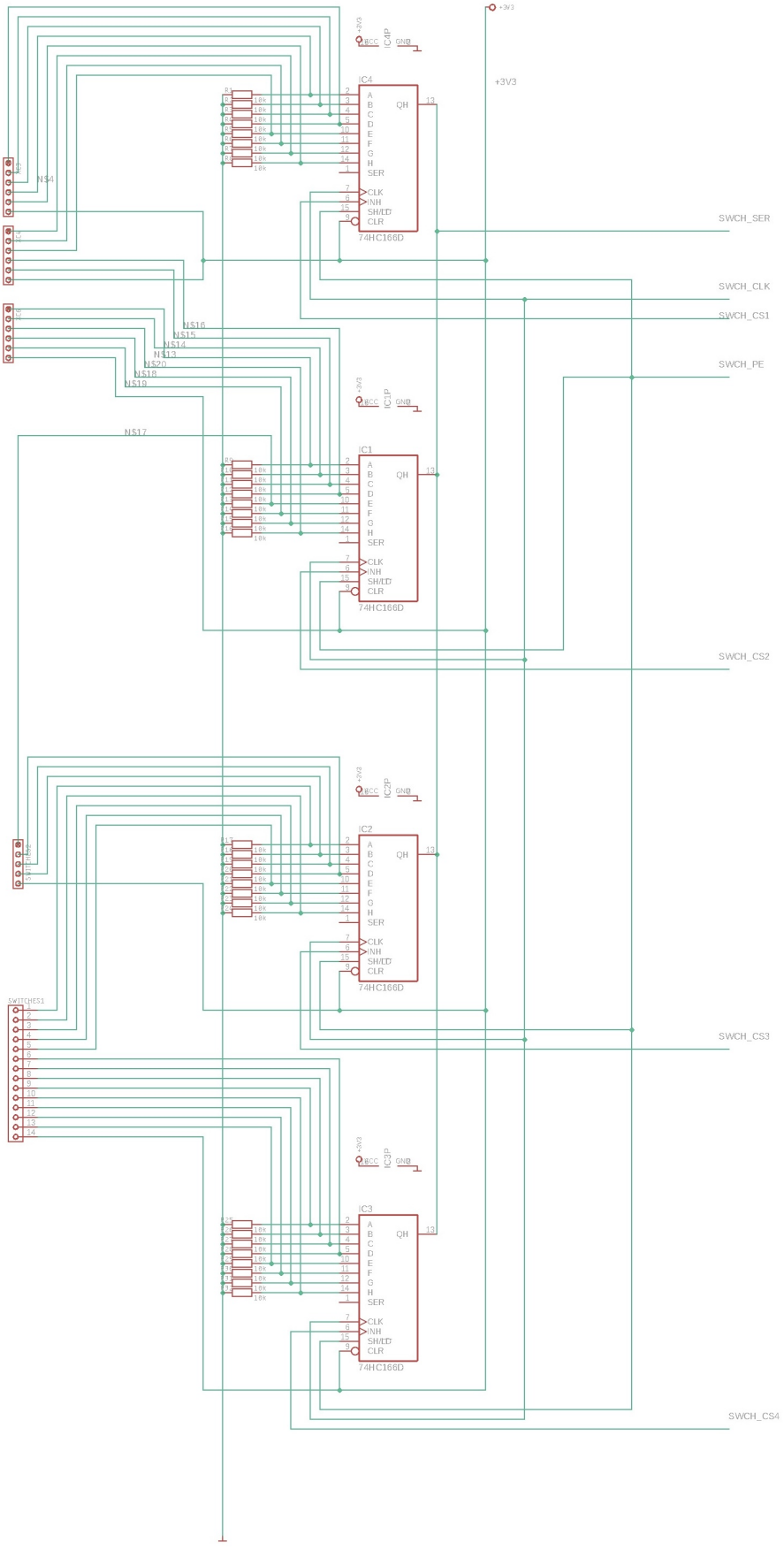
Obrázek 24: Schéma zapojení budičů bipolárních krokových motorů



Obrázek 25: Schéma zapojení budičů unipolárních krokových motorů



Obrázek 26: Původní schéma zapojení posuvných registrů



Obrázek 27: Opravené schéma zapojení posuvných registů