VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INTELIGENTNÍCH SYSTÉMŮ
DEPARTMENT OF INTELLIGENT SYSTEMS

VESTAVĚNÉ ZAŘÍZENÍ PRO ŘÍZENÍ ROBOTICKÉ RUKY

EMBEDDED DEVICE FOR ROBOTIC ARM CONTROL

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. JIŘÍ KYZLINK**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. FILIP ORSÁG, Ph.D.**
SUPERVISOR

BRNO 2020

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav inteligentních systémů (UITS) Akademický rok 2019/2020

Zadání diplomové práce

Student: **Kyzlink Jiří, Bc.**
Program: Informační technologie Obor: Počítačové a vestavěné systémy
Název: **Vestavěné zařízení pro řízení robotické ruky**
Embedded Device for Robotic Arm Control
Kategorie: Vestavěné systémy
Zadání:

1. Seznamte se s problematikou řízení pohybu robotického ramene (úloha inverzní kinematiky). Nastudujte příslušné algoritmy výpočtu ovládání servomotorů za účelem dosažení cílové pozice koncového efektoru.
2. Navrhněte desku plošných spojů s mikrokontrolérem (například STM32), která bude řídit pohyb robotického ramene se čtyřmi stupni volnosti.
3. Navrženou desku nechte vyrobit, osaďte součástkami, zprovozněte a vytvořte pro ni vhodný firmware pro řízení pohybu ramene a komunikaci se serverem.
4. Naprogramujte software v jazyce C# pro server, který bude s deskou komunikovat a řídit pohyb ramene na vyšší úrovni.
5. Otestujte funkčnost navrženého řešení (řízení jednotlivých kloubů, komunikace serveru a kontroléru, plynulost, přesnost a opakovatelnost pohybu).

Literatura:

- SICILIANO, Bruno. *Robotics: modelling, planning and control*. London: Springer, c2009. ISBN 978-1846286414.

Při obhajobě semestrální části projektu je požadováno:

- Body 1 a 2 zadání.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/
Vedoucí práce: **Orság Filip, Ing., Ph.D.**
Vedoucí ústavu: Hanáček Petr, doc. Dr. Ing.
Datum zadání: 1. listopadu 2019
Datum odevzdání: 3. června 2020
Datum schválení: 31. října 2019

Abstrakt

Tato diplomová práce se zabývá návrhem a realizací vestavěného zařízení (modulu), určeného k řízení robotické ruky. S nadřízeným systémem komunikuje prostřednictví USB a dle zadaných příkazů plynule pohybuje koncovým efektorem robotické ruky. Modul se skládá ze dvou oboustranných desek plošných spojů. První dovoluje připojení modulu na sběrnici CAN, zajišťuje spolehlivé napájení manipulátoru i obou částí modulu. Druhá, výpočetní deska, obsahuje výkonný mikropočítač umožňující komunikaci s inteligentními servomotory tvořící robotickou ruku a výpočet kinematických úloh dle požadavků nadřízeného systému. V rámci práce byla vyvinuta i aplikace umožňující ovládání manipulátoru pomocí grafického nebo webového rozhraní. Práce popisuje použité sběrnice, nástroje a postupy aplikované při návrhu zařízení i implementaci softwarového řešení. Na závěr byla provedena měření dokazující řádové zlepšení odezvy při komunikaci se servomotory i plynulosti pohybu manipulátoru oproti dříve používanému řešení.

Abstract

This diploma thesis deals with a design and implementation of an embedded device (module), used to control a robotic manipulator. The module instructs servomotors of the manipulator to move according to the commands received via USB interface. The module consists of two double-sided printed circuit boards. The first one allows connection to the CAN bus, redundant and thus reliable power supply for servomotors as well as the whole module. The second board, compute oriented one, embeds powerful microcontroller used to communicate with the servomotors and to solve the kinematics tasks. As a part of the thesis a graphical user interface as well as a web-oriented interface were developed. Both interfaces allow full control of the manipulator. All the communicating buses, tools and methods used during the design and implementation phases of the work are described in the thesis. Finally, measurements proved improvement of the motion smoothness and response times in several orders of magnitude in comparison to the previous system.

Klíčová slova

vestavěný systém, STM32, ARM, robotická ruka, návrh DPS, multiplexování napájení, inverzní kinematika

Keywords

embedded system, STM32, ARM, robotic manipulator, PCB layout, power multiplexing, inverse kinematics

Citace

KYZLINK, Jiří. *Vestavěné zařízení pro řízení robotické ruky.* Brno, 2020. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Filip Orság, Ph.D.

Vestavěné zařízení pro řízení robotické ruky

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením Ing. Filipa Orsága, Ph.D., uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....

Jiří Kyzlink

3.6.2020

Obsah

Seznam použitých zkratek

ADC	Analog-to-digital converter – analogově-digitální převodník
API	Application programming interface – rozhraní pro programování aplikací
ARM	Advanced RISC Machine – architektura procesorů
CAN	Controller Area Network – datová sběrnice
CRC	Cyclic redundancy check – cyklický redundantní součet
DMA	Direct memory access – přímý přistup do paměti
DPS	deska plošných spojů
DoF	Degrees of freedom – počet stupňů volnosti
ESD	Electrostatic discharge – elektrostatický výboj
ETL	Embedded Template Library – implementace knihovny vzorů určená pro použití na vestavěných systémech
HAL	Hardware abstraction layer – vrstva abstrakce hardwaru
IO	Integrovaný obvod
LED	Light-emitting diode – svítivá dioda
MCU	Microcontroller – mikropočítač
MFD	Multi-function device – multifunkční tiskárny
MUX	multiplexer
OMPL	Open Motion Planning Library – knihovna algoritmu pro plánování pohybu
OTG	USB On-The-Go – specifikace umožňující USB zařízení používat jako hostitele
PHY	Physical layer – fyzická vrstva, typicky přistupující k přenosovému médiu
PID	Proporcionálně integračně derivační regulátor
PWM	Pulse-width modulation – pulzně šířková modulace
RAM	Random-access memory – paměť s přímým přístupem
RGB	Red, green, blue – červená, zelená, modrá
RQA	Robotic Quality Assurance – robotická validace a verifikace kvality
RTOS	Real-time operating system – operační systém reálného času
SBC	Single-board computer – jednodeskový počítač (např. Raspberry PI)
SPI	Serial Peripheral Interface – sériové periferní rozhraní
STL	Standard Template Library – šablonová knihovna jazyka C++
TTL	Transistor–transistor logic – tranzistorově-tranzistorová logika
USART	Universal synchronous and asynchronous receiver-transmitter – univerzální synchronní a asynchronní vysílač a přijímač
USB	Universal Serial Bus – sériová sběrnice
UVLO	Undervoltage-lockout – detekce podpětí a zabránění běhu obvodu
WPF	Windows Presentation Foundation – knihovna pro vytváření grafických uživatelských rozhraní
XML	Extensible Markup Language – značkovací jazyk

Úvod

S rostoucí komplexitou softwarových produktů rostou i požadavky na jejich validaci a verifikaci, a tím náročnost testování. Při vývoji software pro vestavěné zařízení je automatické testování náročné, například kvůli interakci s displejem zařízení a mnohdy nemožné emulaci vestavěného zařízení. Provádění komplexních testovacích scénářů bylo donedávna výhradní výsadou lidských testerů. Testování pomocí testerů je náročné mj. na koordinaci a časovou dotaci. Také proto vznikl ve firmě Y Soft projekt určený k black-box automatické validaci a verifikaci vestavěných zařízení, v tomto konkrétním případě multifunkčních tiskáren. Projekt, interně pojmenovaný *RQA*[1], zahrnuje robotický manipulátor používaný k interakci s vestavěným zařízením, ať už klikáním na dotykovou obrazovku nebo přímo klikáním na hardwarová tlačítka. Oproti testerům je také schopný dodat další metriky, mezi typicky používané patří doba od kliknutí na tlačítko do překreslení obrazovky, do výtisku prvního listu papíru, celková doba potřebná pro přihlášení uživatele k tiskárně a další.

Manipulátor je sestaven ze čtyř inteligentních servomotorů *Dynamixel* komunikujících prostřednictvím sběrnice *RS-485* s nadřízeným systémem. Pro plynulý pohyb manipulátoru po definované křivce je nutné každému servomotoru v pravidelném intervalu zasílat novou cílovou pozici. Od počátku vývoje *RQA* byl nadřízeným systémem servomotorů počítač využívající běžný operační systém (Linux nebo Windows) a vzhledem k nutnosti pravidelné komunikace se servomotory, kterou nejsou běžné operační systémy bez vynaložení velkého úsilí schopné zařídit, docházelo ke krátkodobým záškubům v průběhu pohybu a obecně nedokonalému pohybu. Pro možnost využít manipulátor a tedy i celý systém *RQA* na validaci a verifikaci dalších scénářů vyvstal požadavek na provádění tahů po displeji (tzv. *swipe*). Při tomto druhu pohybu je již stávající řešení na hranici svých možností.

Cílem této diplomové práce je návrh, výroba a implementace vestavěného zařízení, které nahradí počítač s běžným operačním systémem v roli řídicího systému servomotorů. Díky plné kontrole nad sériovým portem a nepřítomností několika úrovní vyrovnávacích pamětí umožní řádově vyšší přesnost při odesílání dat servomotorům, čímž zvýší kvalitu pohybu a umožní provádění přesných a plynulých pohybů ramene robotického manipulátoru. Vyvíjený vestavěný systém bude řízen pomocí příkazů vyšší úrovně popisující cílovou polohu koncového efektoru manipulátoru a parametry cesty. S nadřízeným systémem bude komunikovat pomocí virtuální sériové linky zprostředkované rozhraním *USB* s možností budoucího rozšíření na komunikaci rozhraním *CAN*.

[1] RQA – *Robotic Quality Assurance* – robotické zajištění kvality

Kapitola 1

Teoretická část

Následující kapitola čtenáře seznámí s robotickým manipulátorem, popíše způsoby interakce s dotykovým displejem. Následně předloží stručný přehled použitých komunikačních sběrnic, rozhraní a protokolů a nastíní čtenáři možnosti řešení přímé a inverzní kinematické úlohy i plánování pohybu manipulátoru. V neposlední řadě představí ekosystém mikropočítačů rodiny *STM32*, možnosti vybraného mikropočítače a použité nástroje a knihovny usnadňující implementaci celého zařízení.

1.1 Inteligentní servomotory Dynamixel

Pohyblivé prvky robotické ruky tvoří inteligentní servomotory jihokorejské výroby od firmy *Robotis*, konkrétně typ *XH430-W210-R* z řady *Dynamixel X*.

Obrázek 1.1: Servomotory *Dynamixel XH430-W210-R* [6].

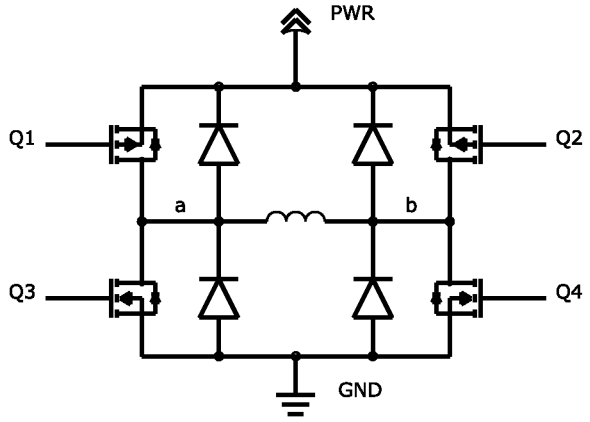
Servomotory jsou inteligentní díky vestavěnému mikropočítači řady STM32 s jádrem ARM Cortex-M3 běžícím na frekvenci 72 MHz, který mimo jiné zajišťuje komunikaci s nadřízeným systémem a řídícím integrovaný H-můstkem[1] (viz schématické znázornění na obrázku 1.2). Využitím PWM[2] a H-můstku lze ovládat moment a směr otáčení rotoru motoru a konečně, díky převodovce s převodovým poměrem $212,6:1$, i rotaci unašeče servomotoru. Díky integraci absolutního enkodéru servomotor disponuje znalostí aktuálního úhlu natočení unašeče. Enkodér ke své funkci nepotřebuje nulování, dle magnetického pole magnetu umístěnému na vnitřním konci hřídele unašeče je schopen rozpoznat absolutní natočení

[1] H-můstek – zapojení čtyř tranzistorů do tvaru písmene H takovým způsobem, že je možná ovládat směr toku proudu motorem.

[2] PWM – *Pulse-Width Modulation* – pulzně šířková modulace signálu

hřídele s rozlišením 12 bitů na 360°, tj. přibližně 0,088 ° bit^{-1}. Servomotor disponuje konfigurovatelným PID[3] regulátorem jehož vstupem je odchylka aktuální pozice od požadované a výstupem je střída PWM signálu řídícího H-můstek.

Servomotory jsou vybavené *coreless*[4] motorem od firmy *Maxon*, který z předchozích zkušeností zajišťuje vysokou spolehlivost i při náročném a dlouhodobém provozu, což se nedá tvrdit o servomotorech s konvenčním typem stejnosměrného motoru s komutátorem. Z posledního písmena v označení servomotoru (R) lze určit, že se jedná o typ s komunikační sběrnicí *RS-485*. V případě označení končícího písmenem T by servomotor komunikoval přes poloduplexní asynchronní komunikační sběrnici v TTL[5] úrovních. Sběrnice *RS-485* byla zvolena kvůli diferenciálnímu přenosu dat a tím pádem i vyšší odolnosti vůči elektromagnetickému rušení a jiným okolním vlivům.



Obrázek 1.2: Schématické znázornění H-můstku.

Servomotory se ovládají prostřednictvím zápisu případně i čtení hodnot vnitřních registrů. Tyto registry zahrnují mj. následující:

- statické informace o servomotoru určené pouze pro čtení (např. modelové číslo, verzi firmware, aj.),
- nastavení komunikace (např. identifikátor servomotoru, přenosovou rychlost sběrnice, verzi protokolu aj.),
- vymezení pracovního regionu (např. maximální dovolenou teplotu, napětí, proudové omezení, omezení střídy, rychlosti aj.),
- aktuální parametry (např. rychlost a pozice unašeče, zatížení servomotoru, proud tekoucí motorem, napájecí napětí aj.),
- cílové parametry (cílová pozice nebo rychlost unašeče),
- nastavení vnitřního PID regulátoru vzhledem k rychlosti i pozici,
- nastavení profilu pohybu servomotoru (konstantní zrychlení/rychlost),
- pole nepřímých data a nepřímých adres (pomocí zápisu do registru nepřímé adresy se vystaví registr na zapsané adrese na příslušném registru nepřímých dat).

Z registrů lze číst či do nich zapisovat prostřednictvím podporovaných protokolů nazývaných *DXL Protocol 1.0* a *DXL Protocol 2.0*. Oba protokoly nejsou zpětně kompatibilní,

[3]PID – Proporcionálně integračně derivační regulátor

[4]coreless – Uspořádání DC motoru takovým způsobem, že stator z permanentních magnetů je uvnitř rotoru tvořeného samostatně držícím vinutím [13]

[5]TTL – Transistor-transistor logic – specifikace napětí pro logickou '0' a '1'

i přesto je však možné použít oba na jedné sběrnici. Více informací o protokolech použitých pro komunikaci se servomotory je v kapitole 1.6.3.

Servomotory nabízejí několik způsobů řízení:

- řízení proudu – nezávisle na rychlosti a pozici unašeče je nastaven proud, na základě zpětné vazby ze senzoru proudu servomotor nastaví střídu PWM signálu a tím i proud motorem;
- řízení rychlosti – nastavuje se pouze rychlost a směr otáčení unašeče, na pozici není brát zřetel;
- řízení pozice – nastavuje se cílová pozice unašeče, na kterou se servomotor, snaží dostat, jedem způsob je využitelný pouze pro 360°, druhý umožňuje více otáček unašeče;
- řízení pozice a proudu – podobné jako předchozí mód, ale je možné specifikovat maximální dovolený proud a tím i moment servomotoru;
- řízení PWM – umožňuje přímé nastavení střídy PWM.

Pro použití v rámci robotického manipulátoru jsou relevantní pouze módy umožňující řídit pozici unašeče, nejvýhodnější je tedy mód s řízením pozice a proudu, který zamezí nadměrnému namáhání součástí servomotoru.

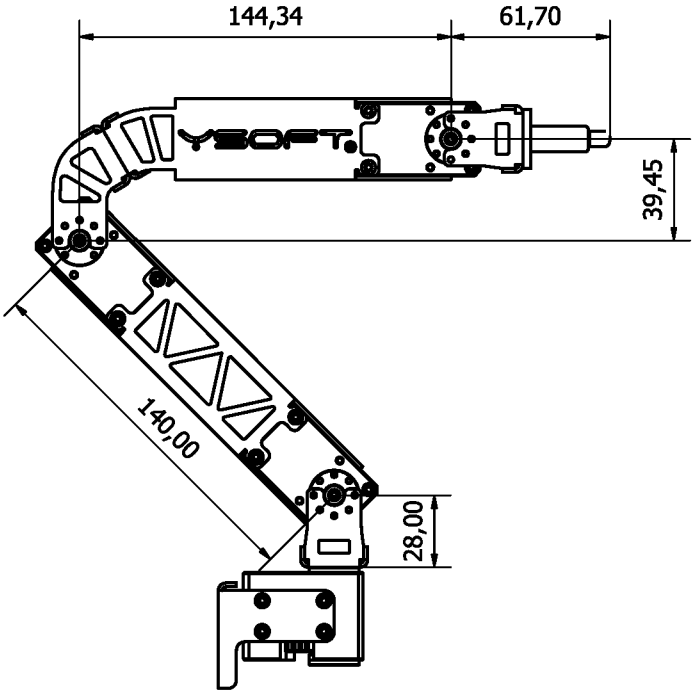
Servomotor dále disponuje množstvím bezpečnostních limitů – přetížení, nedostatek proudu, chyba enkodéru, přehřátí, napětí mimo rozsah. Překročení specifikovaných limitů, vyústí v převedení servomotoru do tzv. *shutdown* stavu. V tomto stavu je deaktivován H-můstek a je zamezeno jeho opětovné aktivaci, bliká červená LED indikující tento stav a je zakázána modifikace registrů řídicích pohyb. Jediná možnost, jak servomotor vrátit do provozuschopného stavu, je pomocí restartu a to buď hardwarového (odpojení a opětovným přivedení napájení) nebo softwarového (zasláním příkazu *reboot*).

1.2 Robotický manipulátor

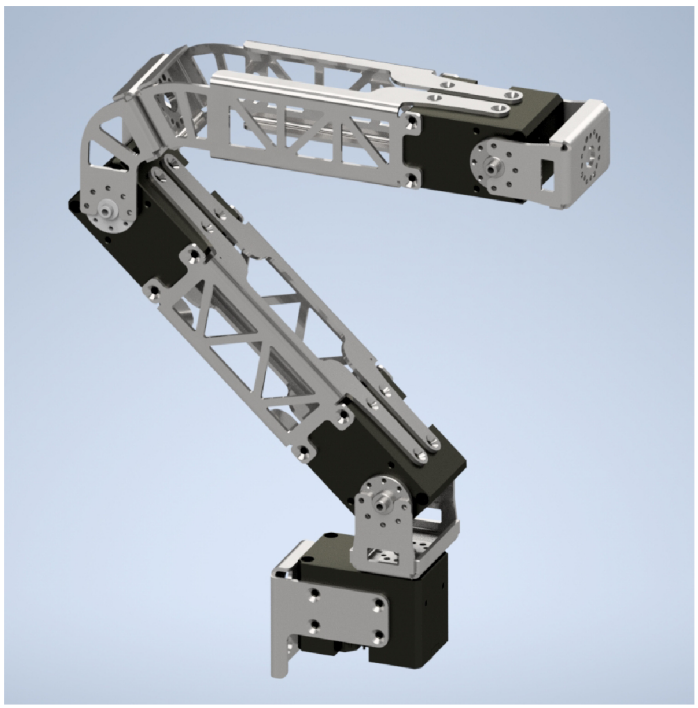
Aktuální verze robotického manipulátoru je sestavena ze 4 kusů inteligentních servomotorů Dynamixel *XH430-W210-R* popsaných v předchozí kapitole. Každý servomotor má pevně daný identifikátor používaný při komunikaci. První servomotor u základy manipulátoru je číslo 1, následně každý další má identifikátor o jedna větší. Poslední servomotor, naklánějící koncový efektor má číslo 4. Servomotory jsou vzájemně spojeny pomocí ohýbaných výpalků z 1,4 mm silné korozivzdorné oceli a šroubů, taktéž z korozivzdorné oceli. Celá sestava robotického manipulátoru je pomocí šroubu s plochou hlavou a výpalku uchycena k rychloupínací destičce standardního fotografickému stativu s kulovou hlavou a pevnou aretací nastavené pozice. Rychloupínací destička umožňuje jednoduše manipulátor odejmout ze stativu a přenést například před jiné zařízení určené k validaci a verifikaci. Modulární stavba robotické ruky dovoluje v případě potřeby nahradit stávající výpalky za jiné při zachování stejné koncepce. Na obrázku 1.3 je část výkresu s kótami důležitými pro implementaci úloh přímé a inverzní kinematiky. Při vhodné implementaci kinematických úloh bude možné pouze nadefinovat rozměry dle nového výkresu, bez vlivu na jiné součásti. Na obrázku 1.4 je render robotické ruky ve výchozí pozici z prostředí software pro CAD.

Konstrukce robotické ruky je unikátní a byla navržena ve firmě Y Soft primárně k ovládání multifunkčních (MFD - *Multi-Function Device*) tiskáren. Cílem bylo navrhnout ruku co nejjednodušší konstrukce s minimálním, avšak nelimitujícím počtem stupňů volnosti. Robotická ruka nemusí přenášet žádné břemena, pouze koncovým efektorem, s dostatečnou přesností, klikat na displej, případně hardwarová tlačítka poblíž displeje MFD. Některá MFD vyžadují pro správnou detekci kliku i určité zatlačení na displej, zpravidla se jedná o modely s rezistivním typem dotykové vrstvy, kde musí dojít k zatlačení a tím vytvoření

vodivého bodu a detekci kliknutí. Na konci robotické ruky je výtisk z 3D tiskárny jehož úkolem je držet koncový efektor, který je tvořen pružnou kopulí vodivé pryže.



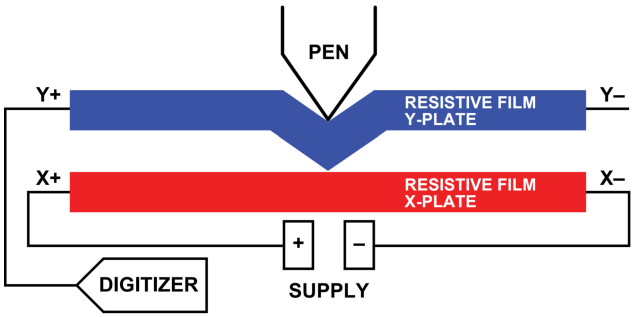
Obrázek 1.3: Část výkresu robotické ruky, včetně rozměrů důležitých pro kinematické úlohy.



Obrázek 1.4: Render robotické ruky bez koncového efektoru.

1.3 Ovládání dotykového displeje

Validační a verifikační systém *RQA* ovládá pomocí robotické ruky primárně dotykové obrazovky vestavěných zařízení. Pro zajištění maximální úspěšnosti detekce dotyku displeje koncovým efektorem je vhodné znát používané technologie. Na trhu jich existuje celá řada [35]. Historicky první technologií bylo snímání přerušení světelného paprsku prstem nad displejem. Avšak hlavní směr dnes používaných technologií lze rozdělit na rezistivní a kapacitní. U rezistivní technologie je snímací vrstva složená mj. ze dvou průhledných vodivých fólií, které jsou za normálních okolností od sebe izolované. Dotykem, respektive fyzickým promáčknutím vrchních fólií, dojde ke spojení dvou vodivých vrstev pod místem dotyku tak, jak je naznačeno na obrázku 1.5. Pomocí měření elektrického odporu mezi vrstvami je možné zjistit přesné místo spojení fólií a konečně i dotyku.



Obrázek 1.5: Zjednodušený nákres konstrukce rezistivní dotykové vrstvy [9].

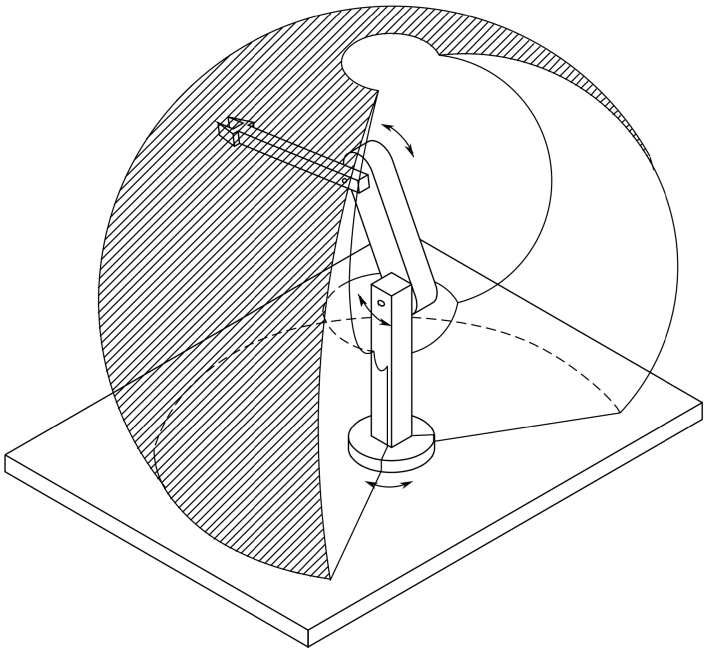
Druhá, kapacitní technologie, může být založena na dvou fenoménech. Prvním je narušení elektrostatického pole nad displejem vodivým elementem (prstem, stylusem, aj.). Druhým je měření kapacity mezi vodiči oddělenými dielektrikem. Dielektrikem je typicky krycí sklo displeje, prvním vodičem je prst, příp. stylus a druhým je vodivá mřížka. Řadič následně změří kapacitu jednotlivých sloupců a řádků mřížky a tím určí místo dotyku.

Hlavní rozdíl mezi technologiemi je, v případě snímání elektrického odporu, nutnost fyzického zmáčknutí snímací vrstvy, v případě snímání kapacity postačuje jen přiblížení, není třeba vyvinout fyzický tlak. Při ovládání dotykového displeje s rezistivní technologií robotem je výhodné využít kinetické energie manipulátoru a do displeje jemně *narazit*. Naopak, u kapacitní technologie je jemný dotyk vodivého koncového efektoru manipulátoru pro registaci doteku dostačující, a případný další tlak je kontraproduktivní, protože vyústí ve sklouznutí koncového efektoru a tím v nepřesný dotyk. Na základě předchozích zkušeností bylo ověřeno, že k úspěšné registraci dotyku u obou typů dotykových obrazovek stačí popsat trajektorii Bézierovou křivkou. Přesněji řečeno, trajektorie je v obou případech úsečka z bodu A do bodu B a pro popis pozice koncového efektoru v rámci úsečky v čase je použití Bézierovy křivky dostačující. Pro klikání na rezistivní dotykový displej se pomocí řídicích bodů nastaví malý překmit pozice a tím dojde k potřebnému krátkodobému zatlačení, u kapacitních je postačující přiblížení s plynulým dojezdem.

1.4 Přímá a inverzní úloha kinematiky

Při psaní následující kapitoly byly využity informace dostupné z publikací [22] a [34], a učební opora předmětu robotika [20]. Robotický manipulátor, se skládá z několika typů součástí, pevných spojovacích dílů (ramen) a pohyblivých kloubů (aktuátorů, servomotorů). Strukturu manipulátoru lze zpravidla rozdělit na ruku, která zajišťuje mobilitu, a zápěstí zajišťující obratnost koncového efektoru, který je poslední částí a interaguje s prostředím. Z pohledu kinematiky je podstatnou vlastností manipulátorů to, že se často jedná o otevřený kinematický řetězec. To znamená, že existuje právě jedna posloupnost ramen mezi uchycením manipulátoru (bází) a koncovým efektorem. Opakem je pak uzavřený kinematický řetězec, kde posloupnost ramen tvoří alespoň jednu smyčku.

U manipulátorů s otevřeným kinematickým řetězcem platí, že každý kloub přidává stupeň volnosti (DoF[6]). Pro dosažení volné polohy a orientace koncového efektoru v prostoru je tedy nezbytné mít manipulátor alespoň se šesti DoF – tři pro translační pohyb a tři pro rotační pohyb koncového efektoru kolem každé osy. Pokud manipulátor disponuje více než šesti DoF nazýváme jej redundantním. Manipulátory se dělí na několik skupin dle druhů a umístění kloubů, z nichž jediná důležitá je, v kontextu této práce, skupina *antropomorfních* manipulátorů. Jde o, v dnešní době, nejrozšířenější typ manipulátoru, který je charakteristický použitím rotačních kloubů a podobností s lidskou rukou. Díky podobnosti je možné části manipulátoru popisovat názvy původně popisující právě lidskou ruku – paže, loket, předloktí, zápěstí, aj. Koncový efektor je pro účely této práce realizován pomocí měkké vodivé pryže s textilním krytím. Tím je docíleno kompatibility s kapacitní i rezistivní technologií dotykových displejů zmíněných v kapitole 1.2.

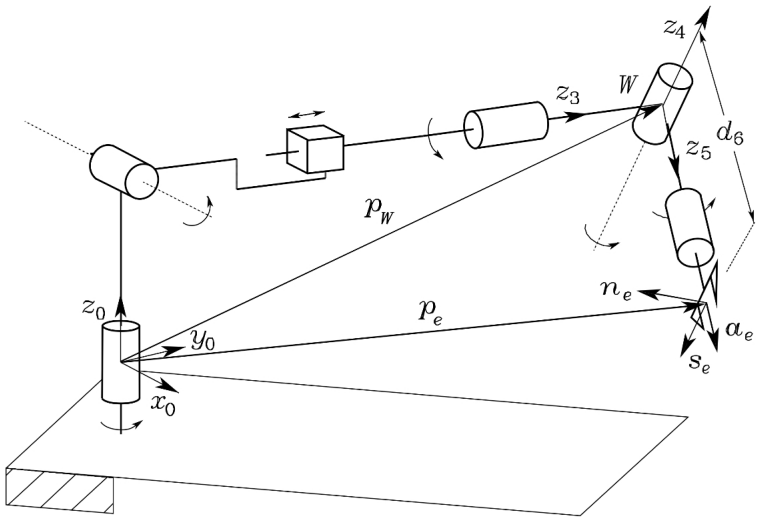


Obrázek 1.6: Antropomorfní manipulátor s vyznačeným pracovním prostorem [22].

[6] DoF – *Degrees of Freedom* – počet stupňů volnosti

Kinematika popisuje vztah mezi úhlem natočení případně pozicí aktuátorů a pozicí a orientací koncového efektoru. Zavedením zmíněného vztahu můžeme řešit úlohu přímé a inverzní kinematiky, kde úkolem přímé kinematiky je získat pozici a orientaci koncového efektoru jako funkci natočení případně pozice aktuátorů manipulátoru. Úkol inverzní kinematiky je přesně opačný, tedy nalézt metodu, jak pro danou pozici a natočení koncového efektoru manipulátoru nalézt odpovídající pozice aktuátorů. Pro popis pozice koncového efektoru manipulátoru využíváme kartézský souřadný systém, pro popis jeho orintace *eulerovy úhly*, avšak pro některé algoritmy je praktičtější orientaci popsat pomocí kvaternionu.

Důsledky pohybu jednotlivých kloubů na navazující ramena lze vyjádřit pomocí homogenních transformačních matic, které umožňují jednoduché skládání operací prostřednictvím násobení matic. Díky tomu lze úlohy přímé kinematiky pro manipulátory řešit analyticky a, což je pro vestavěný systém důležité, v konečném čase. Oproti tomu úloha inverzní kinematiky nemá vždy jednoduché analytické řešení, případně je velmi složité jej nalézt. Složité kinematické řetězy se řeší pomocí iteračních numerických metod, které jsou sice náročné na výpočetní prostředky a neposkytnou přesné výsledky v rozumném čase, ale lepší řešení neexistuje. Hledáním analytických řešení úlohy inverzní kinematiky se zabývá například projekt *OpenRave*, konkrétně modul *IKFast*, který je schopný pro standardně popsaný robotický manipulátor najít analytické řešení přímé i inverzní kinematické úlohy [7]. Výstupem algoritmu je zdrojový kód v jazyce *C++* obsahující třídu se dvěma metodami, jednou pro řešení přímé kinematiky a druhou pro inverzní kinematiku. Automatizované hledání řešení však není univerzální, například *IKFast* je limitován na řešení úloh kinematiky manipulátorů s maximálně sedmi DoF. Další omezení způsobuje použití vestavěného systému. Generovaná řešení nejsou přizpůsobena pro běh na vestavěném zařízení, jelikož jsou příliš veliká (až vyšší tisíce řádků zdrojového kódu) a obsahují dynamickou alokaci paměti. Ačkoliv na moderních procesorech výpočet zabere maximálně nízké jednotky milisekund v prostředí vestavěného zařízení by byla náročnost výpočtu příliš vysoká.



Obrázek 1.7: Manipulátor se sférickým zápěstím, vektor p_e označuje cílovou pozici koncového efektoru, vektor p_w pozici manipulátoru bez sférického zápěstí [22].

Náročnost řešení inverzní kinematické úlohy pro průmyslové robotické manipulátory dle [34] způsobila přizpůsobení architektury manipulátorů tak, aby byl výpočet inverzní kinematiky v maximální míře usnadněn. V praxi to znamená, že je možné manipulátor

rozdělit na několik celků a ty řešit samostatně. Zpravidla dojde k rozdělení na sférické zápěstí a zbytek manipulátoru, kdy zápěstí řeší primárně požadovanou orientaci koncového efektoru, čímž dojde k výpočtu požadované pozice osy aktuátoru zápěstí. Následně, se vypočítá jednoduchou přímou metodou inverzní kinematika pro zbytek manipulátoru a dosažení pozice prvního ramena zápěstí posledním ramenem manipulátoru bez nutnosti řešit natočení, to je opět řešeno na úrovni sférického zápěstí. Názorná ukázka rozdělení je na obrázku 1.7. Tím se inverzní kinematická úloha podstatně zjednoduší a je možné ji řešit pomocí přímých metod. Pro výpočet řešení přímých metod není třeba mnoho výpočetního výkonu a je tedy možné výpočty realizovat i přímo na vestavěném zařízení. Během řešení úlohy inverzní kinematiky je také vhodné brát v potaz možnou existenci více řešení a zohlednit ji při výpočtu, a to zpravidla takovým způsobem, aby se manipulátor pohyboval konzistentně. Případně využití dalších heuristik (např. zatížení jednotlivých aktuátorů, jejich teplotě, rovnoměrnému opotřebení aj.).

1.5 Plánování pohybu manipulátoru

Předchozí kapitola popisovala metody pro výpočet inverzní kinematiky, které jsou dostačující k nalezení konfigurace aktuátorů pro danou pozici a orientaci koncového efektoru v kartézském souřadném systému, avšak při řízení manipulátoru je nezbytné mezi pozicemi i plynule přejíždět. V případě této práce může být manipulátoru zadána cílová pozice koncového efektoru a vyvíjený vestavěný systém se postará o nalezení cesty a následný plynulý přejezd k této pozici. Proces, který popsanou funkcionalitu zajišťuje, se souhrnně nazývá plánování a exekuce trajektorie koncového efektoru.

Na trhu jsou dostupné knihovny, které hledání cesty v prostředí beze zbytku řeší, např. *OMPL* [27]. Některé při hledání dokáží zohlednit velké množství proměnných (zatížení koncového efektoru, zatížení jednotlivých aktuátoru). Knihovny zpravidla řeší i kolizní problémy, ať už manipulátoru samotného, tak kolize manipulátoru s prostředím, ve kterém se pohybuje. Dokáží trajektorii vyhledat tak, aby se manipulátor vyhnul kolizi a zároveň bylo dosaženo dané pozice koncového efektoru. Již z výčtu všech dostupných funkcionalit může být patrné, že hledání cesty není triviální problém, a jeho řešení může zabrat podstatné množství času, často i několik sekund na výkonných počítačích [16]. Průmyslové manipulátory používané v praxi využívají k výpočtům algoritmů specializovaných počítačů, které mají k dispozici řádově větší množství výkonu než vyvíjené vestavěné zařízení. Knihovny také závisí na dynamické správě paměti, a proto je jejich využití na vestavěném zařízení problematické.

Výhodou prostředí, ve kterém se pohybuje manipulátor řízený v rámci této práce, je nepřítomnost překážek. Přesněji řečeno, nepřítomnost překážek v běžné trajektorii manipulátoru. Tato skutečnost proces popsaný v předchozím odstavci řádově zjednoduší a na místo problému hledání cesty stačí řešit problém pohybu koncového efektoru po přímce, z bodu A do bodu B (tzv. *point-to-point* pohyb). Při znalosti výchozí i cílové pozice je možné vypočítat výchozí a cílové natočení jednotlivých aktuátorů, avšak vzhledem k nelineární závislosti pozice koncového efektoru a natočení jednotlivých aktuátorů není interpolace jejich pozice dostatečná. Koncový efektor by se sice dostal do správné cílové pozice, ale trajektorie efektoru by zpravidla neodpovídala požadovanému pohybu po přímce. Řešením této situace je interpolace pozice koncového efektoru a vypočítání nových pozic aktuátorů v pravidelných časových intervalech. Interpolaci lze provádět lineárně z bodu A do bodu B nebo je možné interpolovat s využitím křivek (např. polynomy n-tého řádu, spline, atd.) případně během jednoho pohybu projekt více bodů.

Při interpolaci je nutné držet se v rámci mechanických možností aktuátorů i manipulátoru. Mezi klíčové požadavky patří dodržení maximální dovolené rychlosti a zrychlení jednotlivých aktuátorů. Vzhledem k tomu, že nadřízený systém nemusí znát parametry manipulátoru a požadek na pohyb specifikuje rychlost, je nutné před započetím pohybu ověřit, že budou limity dodrženy a případně rychlost a akceleraci pohybu upravit tak, aby k jejich dodržení došlo.

1.6 Použité sběrnice a protokoly

V práci je využito mnoho protokolů pro komunikaci mezi různými komponentami a systémy. Následující kapitola každý protokol krátce popíše a nastíní jeho funkci v systému.

1.6.1 Sběrnice RS-485

Mikropočítač bude se servomotory komunikovat prostřednictvím asynchronní sběrnice RS-485, resp. dle standardu *EIA* pojmenovaného *EIA-485*. RS-485 je jedna z nejjednodušších sériových diferenciálních sběrnic. Využívá jednu či dvě kroucené dvojlinky pro přenos logických úrovní, díky tomu je sběrnice odolná vůči elektromagnetickému rušení, které se indukuje na obou vodičích ve stejné míře a tudíž nemění rozdíl napětí. Tato vlastnost je vzhledem k proudovým špičkám a využití PWM pro regulaci motoru servomotoru důležitá a zajišťujete přenos s minimem chyb a přeslechů[7]. Signály RS-485 jsou zpravidla označené *A* a *B*, někdy též + a - či *hot* a *cold*. RS-485, na rozdíl od *EIA-422* dovoluje na společnou sběrnici připojit více než 2 zařízení, dle standardu je maximum 32, avšak na trhu jsou dostupné řadiče, které zatíží sběrnice jen jednou osminou nominálního zatížení a tím umožní navýšení celkového počtu zařízení připojených ke sběrnici na více než 200 [32]. Sběrnice může být implementována buď jako plně duplexní, kdy se využívá dvou kroucených párů, nebo jako poloduplexní kdy se využije pouze jeden kroucený pár a je zodpovědností protokolu zajistit, kdy mají zařízení vysílat a kdy přijímat. U servomotorů je využita poloduplexní varianta. Sběrnice je založena na *master-slave* přístupu, to znamená, že veškeré transakce na sběrnici jsou iniciovány zařízením typu *master* a zařízení typu *slave* poslouchají, co *master* pošle a pouze pokud jsou vyzváni tak odpoví [3]. Popisované chování odpovídá protokolu popsanému v kapitole 1.6.3.

Pro komunikaci po sériových sběrnicích se u mikropočítačů obecně využívá periferie *U(S)ART*[8] umožňující (a)synchronní komunikaci s případným hardwarovým řízením toku. Mikropočítač řady *STM32* disponuje několika UART a USART periferiemi, liší se od sebe zdrojem hodinového signálu a tím maximální dosažitelnou přenosovou rychlostí, podporovanými funkcemi, případně i napojením na různé datové sběrnice mikropočítače, jak je patrné z blokového diagramu a tabulky 6 v katalogového listu mikropočítače [25]. Pro účely komunikace se servomotory jsou dostačující všechny periferie dostupné na vybraném mikrokontroléru. Sběrnice svými napěťovými úrovněmi neodpovídá úrovním používaným u mikropočítače a je tedy nutné využít integrovaný obvod schopný převést signály z mikropočítače na *RS-485*. Pro účely této práce byl vybrán integrovaný obvod *SN65HVD75* umožňující převod signálů periferie USART na sběrnici RS-485 při napájení a komunikaci

[7] přeslech – chybná interpretace log. 0/1 často z důvodu indukční vazby mezi komunikační a napájecí sběrnicí

[8] U(S)ART – *Universal (A)Synchronous Receiver/Transmitter* – univerzální (a)synchronní vysílač a přijímač

v napěťové hladině 3,3 V. IO má na straně RS-485 integrovanou ochranu proti ESD[9] způsobené člověkem i ochranu dle normy *IEC 61000-4-2* což pro návrh desky plošných spojů (DPS) znamená, že není třeba řešit další ESD ochranu proti silnému náboji typicky indukovanému na vodičích při spínání induktivní zátěže [32].

1.6.2 Sběrnice USB

Přestože je USB na velkém množství zařízení, jeho implementace ve vestavěných systémech nepatří k nejjednodušším. Sběrnice USB existuje v několika různých verzích, od *1.0* až po *4.0*, v rámci této práce bude popsaná výhradně verze *2.0* s přenosovou rychlostí *Full-Speed*, zkráceně *(FS)* 12 Mbit s^{-1} [17]. USB používá pro přenos dat a energie dva páry vodičů. První pár slouží k napájení (5 V a zem) a druhý, jehož vodiče jsou pojmenované *D+* a *D-*, ke komunikaci. Z pojmenování vodičů je patrné, že komunikace probíhá po diferenciálním páru, který díky závislosti na rozdílu napětí mezi vodiči, zvyšuje odolnost proti rušení. Sběrnice USB logicky tvoří stromovou topologii, avšak samotná fyzická spojení jsou vždy z body do bodu tzv. *point-to-point*. Kořenem logického stromu je tzv. *Root Hub*, disponující několika samostatnými porty pro připojení dalších zařízení nebo hubu.

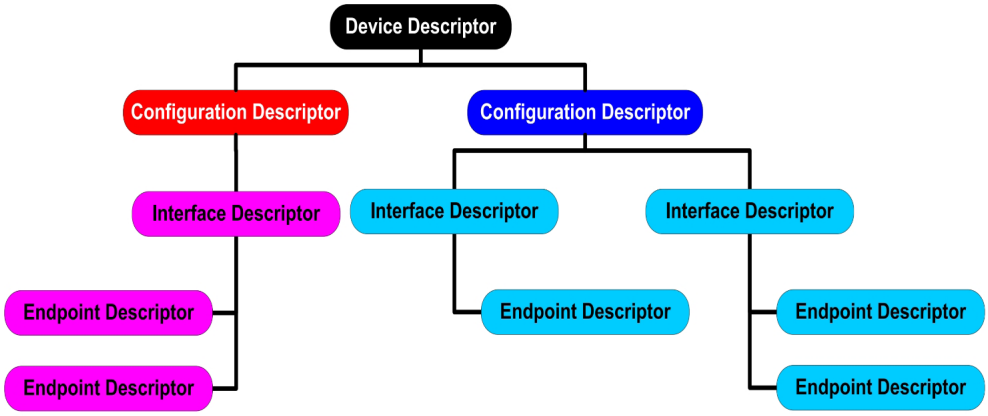
Komunikaci na sběrnici USB vždy začíná host, což může být root hub, případně i běžný hub, avšak nikdy nebude komunikaci začínat připojené zařízení. Komunikace na sběrnici probíhá v tzv. transakcích, které se skládají z po sobě jdoucích paketů. Paketů je několik druhů a jejich popis je mimo rámec této práce (viz například aplikační poznámka [17]). Při komunikaci se využívá tzv. endpointů, které buď přijímají nebo vysílají data. Endpointy jsou číslované a každé zařízení musí podporovat *EP0 OUT* a *EP0 IN* pro přenos řídicích dat z a do zařízení. Nad endpointy jsou tzv. *pipes* (roury), které slouží k předávání dat na úrovni klientského software. Roury mají definované použité endpointy, směr toku dat, požadovanou propustnost a typ používaného přenosu. Zařízení musí mít výchozí rouru pro řízení, která využívá *EP0 OUT* a *EP0 IN* a je obousměrná.

Samotné přenášení dat probíhá v přenosech (*transfer*), USB rozlišuje 4 typy přenosů: řídicí (control), přerušení (interrupt), izochronní (isochronous) a hromadný (bulk). Každý má své specifické využití. Řídicí jsou pro řízení sběrnice. Přerušení mohou navádět k tomu, že zařízení samovolně pošle data hostovi, avšak dle standardu i při přenosu přerušení musí zařízení vyčkat na vyžádání dat ze strany hosta a až následně může data poslat. Izochronní přenosy se využívají pro pravidelný přenos dat (audio, video). A poslední typ, hromadný přenos, je určený pro nárazový přenos většího množství dat, jako jediný nemá garantovanou přenosovou rychlost ani latence, protože při přenosu má nejmenší prioritu a využívá zbytkové přenosové pásmo sběrnice.

Každé USB zařízení musí být také schopné popsat svoji funkcionalitu pomocí standardem definovaných struktur, tzv. *deskriptorů*, kterých je opět více druhů pro popis různých částí protokolu. Deskriptory tvoří stromovou strukturu – od popisu nejobecnějších vlastností deskriptorem zařízení až po popis jednotlivých endpointů a rour. Deskriptory zařízení (*device descriptors*) popisují nejdůležitější vlastnosti zařízení, mezi které patří identifikační číslo výrobce tzv. *VID*, identifikace produktu *PID*(Product ID) používané pro správné přiřazení ovladače. Dále je pomocí deskriptorů popsaná konfigurace zařízení (*configuration descriptor*) zahrnující informace o počtu rozhraní, způsobu napájení zařízení a maximálním odebíraném proudu. Pro popis rozhraní je využit deskriptor rozhraní (*interface descriptor*), který obsahuje informace o počtu endpointů, dále pak informace o třídě, podtřídě a protokolu rozhraní. Při definici více rozhraní se zařízení stane tzv. kompozitním, to znamená,

[9] ESD – *Electrostatic discharge* – elektrostatický výboj

že jedno fyzické zařízení je složeno z více logických zařízení (lze si představit jako virtuální USB hub). Poslední deskriptor slouží pro popis endpointu (*endpoint descriptor*). Obsahuje vlastnosti každého nenulového (ten je implicitně řídící) endpointu, požadovaný typ přenosu dat, maximální velikost paketů a pro izochronní přenosy i požadovaný interval dotazování. Ukázka stromu deskriptorů na obrázku 1.8.



Obrázek 1.8: Strom deskriptorů popisujících USB zařízení [17].

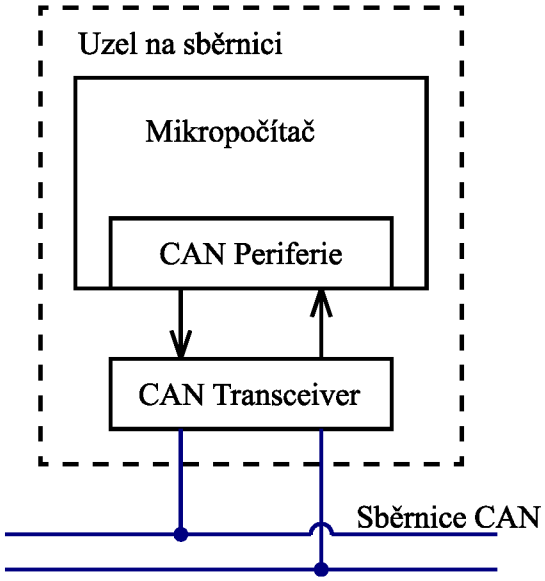
V předchozím odstavci byly zmíněny následující vlastnosti: třída, podtřída a protokol rozhraní. Ačkoliv si může každý pomocí deskriptorů definovat vlastní zařízení a následně k němu vyvíjet a dodávat vlastní ovladač, nebylo by to praktické. A proto skupina *USB Implementers Forum* zabývající se vývojem sběrnice definovala několik tříd popisující způsob komunikace u běžně používaných zařízení. Typicky používanou třídou je *Human Interface Device* (HID), což je třída, která slouží pro zařízení, pomocí kterých člověk interaguje s počítačem, podtřídou je následně možné zvolit zda jde o myš, klávesnici, atd. Výhodou využití definovaných tříd je implicitní použití systémového ovladače, zařízení zpravidla funguje hned po připojení bez dalšího nastavení. Systém vyvíjený v rámci této práce využívá třídu *Communications Device Class* (CDC), která umožňuje vytvoření virtuální sériové linky pro komunikaci s nadřízeným systémem. Využití virtuální sériové linky přináší oproti fyzické mnohá pozitiva: detekce a korekce chyb je řešena vyššími vrstvami, nedochází k přeslechům, velká přenosová rychlost, velice snadné připojení k počítači, případně jinému výpočetnímu zařízení. Mezi možné nevýhody virtuální sériové linky přes USB oproti fyzické patří možné vyšší latence a náročnost obsluhy rozhraní USB na vestavěném systému.

1.6.3 Sběrnice CAN

Systém vytvořený v rámci této práce bude umožňovat komunikaci s nadřízeným systémem pomocí sběrnice CAN (zkratka z Controlled Area Network). Sběrnice byla vyvinuta firmou *Robert Bosch* v roce 1986 pro využití v automobilovém průmyslu [2]. Během tohoto období docházelo ke zvyšování pohodlí v automobilech vlivem integrace nových senzorů, používání motorů s elektronicky řízeným vstřikováním a obecné digitalizaci automobilů. To způsobilo nárůst počtu řídicích jednotek a potřebu vyvinutí jednotné komunikační sběrnice a takovou sběrnicí se stala právě CAN. Po standardizaci jako *ISO 11898* se sběrnice i díky svým vlastnostem brzy rozšířila za hranice automobilového průmyslu, např. do námořního nebo zdravotnického sektoru [2]. Dnes již většina automobilového průmyslu používá nějakou variantu sběrnice CAN, typicky u řídící jednotky motoru se používá rychlejší varianta

a pro obsluhu jednotek pohodlí (elektrická stahovací okna, nastavení sedadla, klimatizace, ...) se využívá varianta s nižší přenosovou rychlostí. Sběrnice pro komunikaci využívá jen jeden kroucený pár, signalizace logických úrovní je diferenciální a komunikace pouze poloduplexní. Vodiče jsou pojmenované *CANH* a *CANL*. Na rozdíl od RS-485 je však sběrnice tzv. *multimaster* to znamená, že uzly na sběrnici můžou začít vysílat kdykoliv uznají za vhodné a na úrovni sběrnice je definováno chování v případě kolize. Řadič fyzické vrstvy se na sběrnici připojí zpravidla až ve chvíli, kdy je pod napětím a díky tomu může sběrnice bez problému fungovat i v případě připojených neaktivních uzlů.

Protokol CAN využívá standardní vrstvenou architekturu podobnou OSI modelu síťových protokolů. V případě, že má mikropočítač CAN periferii, jde zpravidla o řadič protokolu, nicméně převod logických úrovní a fyzická interakce se sběrnicí je zajišťována jiným IO, takzvaným CAN transceiverem. Transceiver detekuje stav sběrnice (dominantní a recesivní), a prostřednictvím výstupů přijímače (pinu) *Rxd* předává informaci CAN periferii. Druhý pin určený pro komunikaci s periférií je *Txd*, kterým se předává signál z periferie vysílači transceiveru. Na obrázku 1.9 je znázorněné rozdělení jednotlivých komponent. Transceiver zajišťuje správné elektrické signály na sběrnici, kompatibilitu s protokolem CAN (ISO 11898-1).



Obrázek 1.9: Architektura uzlu na sběrnici CAN.

Protokol který by mohl být používán při komunikaci s nadřízeným systémem a je postavený nad protokolem CAN byl navržen v rámci bakalářské práce [8], kde je i do hloubky popsán. V tomto odstavci budou uvedeny pouze informace nezbytné pro pochopení základní funkčnosti, v případě zájmu je další studium protokolu z citované práce ponecháno na čtenáři.

Protokol je postaven nad *Extended CAN* který se vyznačuje delším, 29 bitovým identifikátorem zprávy. Identifikátor je využitý k uložení informace o typu zprávy, adrese cílového uzlu, identifikaci typu vysílajícího uzlu a čísla rámce. Protokol podporuje několik typů zpráv lišících se prioritou a určením, mj. potvrzení přijetí zprávy, zápis, čtení, hledání zařízení na sběrnici atd. Jednotlivé uzly připojené ke stejné sběrnici musí mít unikátní identifikátor, který se v případě tohoto protokolu skládá z pevné části přiřazené jednotlivým typům uzlů a z uživatelsky nastavitelné části pro identifikaci více uzlů stejného typu. Maximální

množství přenesených užitečných dat v jednom rámci je 8 B, při přenášení delších zpráv je nutné zprávy fragmentovat do více rámců a následně poskládat zpět do původní podoby. Fragmentaci usnadňuje přítomnost čísla rámce již v identifikátoru rámce, pro číslo rámce je v hlavičce vyhrazeno 8 bitů, tudíž je možné zasílat zprávy s maximální délkou 2048 bajtů. Vzhledem k zamýšlenému použití protokolu pro přenos souřadnic by neměla být maximální velikost zprávy limitující.

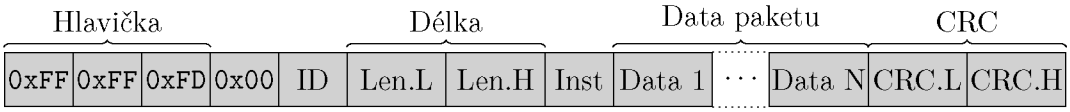
1.6.4 DXL Protokoly

DXL protokol pro komunikaci servomotorů s nadřízeným systémem existuje ve dvou verzích – *1.0* a *2.0*, protokol není zpětně kompatibilní a primárně využívaná bude jeho druhá verze [5]. Avšak pro poskytnutí kontextu bude krátce uvedena i první verze.

Protokol ve verzi *1.0* byl původně navržen pro starší servomotory s komunikací pomocí asynchronní sběrnice v TTL úrovních. Každý paket či rámec, zde jedno a to samé, začíná dvoubajtovou hlavičkou s hexadecimální hodnotou `0xFF`, bezprostředně následovanou bajtem identifikujícím přijímající servomotor, bajtem značícím délku paketu, několika (až 253) bajty přenášející užitečná data a celý paket končí bajtem vyhrazeným pro kontrolní součet. Protokol podporuje dva typy paketů – instrukční a stavový. Instrukční zasílá nadřízený systém servomotorům, stavový navrací servomotory řídícímu systému. Nevýhodou první verze protokolu a tudíž určitým impulsem pro návrh druhé verze byla zbytečně složitá komunikace s více motory najednou a vzhledem ke značné jednoduchosti a nepřítomnosti speciálních příkazů i náročné hledání servomotorů na sběrnici.

Protokol *2.0* byl dle *Git*[10] historie dokumentace [4] veřejně zdokumentován na začátku roku 2017 během doby, kdy výrobce začal dodávat servomotory s řízení pomocí sběrnice RS-485. Na rozdíl od verze *1.0* mj. podporuje více registrů, má delší hlavičku a standardní polynomiální CRC[11]. Protokol podporuje zápis dat do více servomotorů jedním paketem a tím umožňuje přesnější synchronizaci servomotorů, lepší poměr užitečných dat na komunikační sběrnici a díky tomu i její menší vytížení.

Komunikační protokol sestává ze dvou typů paketů – instrukční a stavový. Stejně jako v případě verze 1, tak i zde instrukční paket vysílá nadřazené řízení servomotoru a servomotor odpovídá stavovým paketem. Instrukční paket přenáší několik různých typů instrukcí mj. tzv. ping, zápis dat do registru, čtení z registru, restart servomotoru, čtení nebo zápis do více servomotorů najednou. Struktura instrukčního paketu je znázorněna obrázkem 1.10, kde *ID* je identifikátor servomotoru, bajt *Inst* specifikuje o jaký typ instrukce jde a postfixy *.L* a *.H* pořadí bajtů u hodnoty reprezentované více bajty.

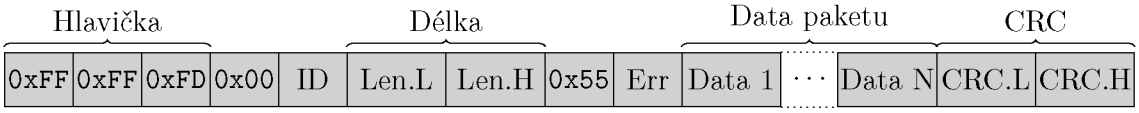


Obrázek 1.10: Struktura instrukčního paketu. Každé pole je jeden byte.

Na většinu instrukčních paketů servomotory odpovídají stavovým paketem. Jedinou výjimkou je tzv. *bulk write* a *sync write*, tedy zápis do více servomotorů najednou. Stavový paket, na rozdíl od instrukčního, obsahuje pole *Err* jehož prostřednictvím může servomotor předat nadřízenému systému informaci o selhání přenosu či provedení příkazu. Mezi definované chybové kódy patří například neshoda kontrolního součtu, zápis dat mimo rozsah,

[10] Git – verzovací systém
[11] CRC – *Cyclic Redundancy Check*, kontrolní součet pro detekci chyb při přenosu

zápis chybné délky, modifikace komunikační parametrů během pohybu aj. Stavový paket je jediný prostředek k získání dat ze servomotorů, struktura paketu je na obrázku 1.11.



Obrázek 1.11: Struktura stavového paketu. Každé pole je jeden byte.

Standardního způsobu ovládání servomotorů prostým zápisem cílové pozice do registrů a kontrolou dosažení cílového natočení nelze pro ovládání robotické ruky použít. Je nutné zapisovat do servomotorů novou pozici pravidelně, s frekvencí alespoň 50 Hz, a tím docílit pohybu koncového efektoru po vypočítané trajektorii. Avšak aby byl pohyb dostatečně plynulý je důležité správně nastavit vnitřní PID regulátor tak, aby byl pohyb dostatečně přesný, při přibližování k požadované cílové pozici se zpomaloval a zamezil příliš vysoké akceleraci a překmitům.

1.6.5 Protokol MAVLink

Vyvíjené vestavěné zařízení bude schopné prostřednictvím USB a virtuální sériové linky komunikovat s nadřízeným systémem – standardní počítač, případně tzv. SBC[12]. Obsahem komunikace budou řídicí zprávy (příkazy) podle kterých se bude manipulátor pohybovat a stavové zprávy (hlášení) informující o různých vlastnostech a stavu vestavěného zařízení a manipulátoru. Při komunikaci prostřednictvím sériové linky je vhodné vybrat existující protokol, jehož implementace zajistí

- detekci přijetí konce zprávy,
- rozpoznání typu zprávy,
- vložení přijatých dat do předem nadefinované datové struktury,
- implementaci kontrolního součtu.

Jedním z protokolů, který požadavky beze zbytku splňuje je protokol *MAVLink* [14].

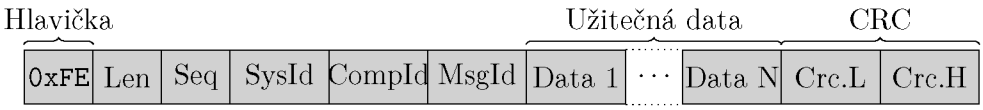
MAVLink (*Micro Air Vehicle Communication Protocol*) byl původně navržen pro komunikaci mezi pozemní stanicí a létajícím dronem, avšak je vhodný i pro výlučně pozemní využití. Protokol existuje ve dvou verzích: verze 1 a verze 2. Verze 2 byla od předchozí rozšířena o podporu podepisování zpráv a byly přidány příznaky signalizující vlastnosti, které *musí* příjemce implementovat pro správnou funkci, v případě, že je nepodporuje měl by rámec zahodit. Dále byly přidané příznaky, signalizující vlastnosti, které příjemce *může* implementovat avšak jejich podporování není pro korektní příjem rámce nutné. Jednou z klíčových vlastností protokolu je definice zpráv předem a následné generování zdrojových souborů, které umí pracovat s těmito dříve definovanými zprávami. Díky této vlastnosti je možné generovat efektivní kód pro více programovacích jazyků. V případě této práce jde o jazyk *C*, použitý ve vestavěném systému, a jazyk *C#* použitý v nadřízeném systému. Díky generování odpadá nutnost dynamické alokace paměti při zpracování zpráv vestavěným systémem a tím se zjednoduší implementace [15].

Pro definici zpráv se používá známý značkovací jazyk XML (eXtended Markup Language). Referenční příklady definice zpráv jsou dostupné na [14], k dispozici je i formální

[12]SBC – *single-board computer* – jednodeskový počítač, počítače na jedné DPS, např. Raspberry PI, či SafeQube.

schéma XML[13]. Definice zprávy nutně obsahuje číselný identifikátor `id`, dle kterého je rozpoznána při zpracování, název zprávy a pole parametrů pevné délky. Definice parametru zahrnuje jeho typ, název, volitelně jednotku a textový popis. Jednotka a popis jsou zde pro snadnější orientaci a případné grafické zobrazení. Dostupné typy vlastností zahrnují běžně používané typy v jazyce *C*, např. `uint8_t`, `int16_t[5]` a podobné. Protokol také dovoluje definovat vlastní výčtový typ (*enum* type) jehož definice se skládá z typu a množiny hodnot, názvu hodnot a případně i textového popisu. Takto definovaný výčtový typ je následně možné přidat k definici pole, čímž se dosáhne zvýšené čitelnosti. Díky tomu, že je protokol aktivně využíván množstvím projektů zabývajících se tvorbou řídicích systémů a autopilotů pro létající objekty (kvadrokoptéry, drony, aj.), je implementace generátorů zdrojových kódů a samotných generovaných kódů na vysoké úrovni. Také je k dispozici množství definovaných zpráv, kde je možné se inspirovat. *MAVLink* je velmi efektivní protokol, režie první verze protokolu je jen 8 B, u druhé je z důvodu podepisování zpráv a fragmentace režie o 6 B větší. V obou případech je množství režie dostatečně nízké a umožňuje použití protokolu i na linkách s nízkou propustností.

Vzhledem k charakteru vyvíjeného vestavěného zařízení není potřebné rámce podepisovat, ani řešit fragmentaci a proto bude použit *MAVLink* ve verzi 1, který se jeví jako dostatečný.



Obrázek 1.12: Struktura rámce MAVLink. Každé pole je jeden byte.

Komunikace protokolem *MAVLink* probíhá v rámcích, pro verzi 1 je struktura rámce na obrázku 1.12. Význam jednotlivých polí je následující:

- *Len* množství užitečných dat v bajtech tzn. maximálně 255 B na jeden rámec
- *Seq* sekvenční číslo rámce
- *SysId* číslo systému pro který je rámec určený
- *CompId* číslo komponenty pro kterou je rámec určený
- *MsgId* identifikátor typu zprávy
- *Crc.H, .L* vyšší a nižší bajt kontrolního součtu

V případě jazyka *C* je generovaný kód pouze do hlavičkových souborů, obsahuje funkce pro vytváření zpráv, jejich zabalení před odesláním a následně i jejich odeslání prostřednictvím dodané funkce, či předání pole bajtů pro odeslání programátorem definovanou funkcí. Pro přijímání bajtů a jejich zpracování v rámce je dostupná funkce `mavlink_parse_char`, které je předán jeden byte a výstupem je informace, zda je rámec přijatý celý. V případě, že je rámec přijatý celý lze dle `MsgId` zjistit, o jaký typ zprávy se jedná a následně přistupovat k jednotlivým, v *XML* dříve definovaným, přijatým polím. Funkcionalita generovaného kódu v jazyce *C#* je stejná, avšak implementace není optimalizovaná do stejné míry, protože to není potřeba.

[13]Formální schéma definice zpráv MAVLink, dostupné z https://github.com/ArduPilot/pymavlink/blob/master/generator/mavschema.xsd

1.7 Ekosystém a mikropočítače řady STM32H7

Když firma *STMicroelectronics* v roce 2006 oznámila licencování ARM jader, v té době *CORTEX-M3*, pro realizaci mikropočítačů (*MCU*) řady STM32 a nahrazení předchozích řad *STR7 STR9* a *STR710* očekávala zdvojnásobení trhu během následujících pěti let [21]. A nemýlila se, mikropočítače z řady *STM32* se staly velice oblíbeným, výkonný a cenově dosažitelným výpočetním prvkem ve velkém množství průmyslových řešení, ale i hobby projektů. Velmi oblíbený je u modelářů jako výkonný výpočetní prvek pro kvadrokoptéry a jiné typy létajících strojů. Dle mého názoru za tím mj. stojí otevřenost ekosystému, podpora v kompilátoru GCC, garance dlouhé životnosti a možnost vybrat si mikropočítač, který bude přesně odpovídat požadavkům.

Moji preferenci mikropočítačů řady *STM32* z velké části způsobila předchozí kladná zkušenost s ekosystémem. Výrobce pořádá přednášky, kde je možné si vyzkoušet práci s danými MCU. Pro zjednodušení prvotní fáze návrhu (výběr MCU, rozložení signálu na jednotlivé výstupy nebo generování kódu pro inicializaci periferií) nabízí bezplatně dostupný nástroj *STM32CubeMX*. Dalším preferenčním bodem byla podpora běžně používaného operačního systému reálného času (RTOS) nabízející základní synchronizační primitiva a datové kontejnery pro předávání proudových dat. V případě *STM32* je dostupná podpora operačního systému *FreeRTOS* [24] přímo od výrobce mikropočítače. Komunita si oblíbila také RTOS *NuttX* který disponuje standardizovaným API podobným a využívá se například u zmíněného řízení kvadrokoptér [18].

1.7.1 Mikropočítač STM32H753

Základem mikropočítače je vysoce výkonné jádro Cortex-M7, velké množství periférií, několik typů pamětí SRAM a 2048 kB velká paměť Flash pro uložení programu [26]. V rámci diplomové práce se budou s jistotou používat komunikační periferie: USB pro komunikaci s počítačem, USART společně s RS-485 transceiverem pro komunikaci se servomotory, případně další USART pro přenos ladících informací, I^2C pro komunikaci řadičem RGB LED.

Moderní mikropočítač obsahuje několik různých propojovacích sběrnic, v případě vybraného typu *STM32H753* jsou na čipu tři typy sběrnic. První je *AXI* (*Advanced eXtensible Interface*) – 64 bit široká sběrnice s vysokou propustností, která zajišťuje propojení procesoru s flash pamětmi, AXI SRAM, periferiemi s velkým datovým tokem – specifický typ řadiče přímého přístupu do paměti (*DMA*), grafické akcelerátory, paměťová rozhraní aj. Druhým typem je 32 bit široká *AHB* (*Advanced High-performance Bus*) určená primárně pro propojení periferií nevyžadujících maximální přenosovou rychlost, avšak, na poměry mikropočítačů, stále periferie náročné na datový tok. Mezi takové periferie patří běžné řadiče DMA, řadiče komunikačních rozhraní Ethernet, USB, řadič pro *SDMMC* (paměťové karty) a další. Klíčovou informací pro pozdější využití DMA je přítomnost pamětí SRAM1-3. Poslední, avšak neméně důležitým typem sběrnice je *APB* (*Advanced Peripheral Bus*), jedná se o nejpomalejší typ sběrnice v tomto mikropočítači a je využitá pro napojení všech ostatních periférií – časovačů, analogově digitálních převodníků, řadičů pomalejších komunikačních sběrnic (SPI, I^2C, U(S)ART, aj.), a také univerzálních vstupně výstupních portů. Celkem je v mikropočítači jedna propojovací matice pro sběrnici AXI a dvě propojovací matice pro sběrnici typu AHB. APB propojovací matici nepotřebuje, o propustnost se jednotlivé periferie dělí v časovém multiplexu, avšak aby nedocházelo k přílišnému blokování obsahuje mikropočítač tyto sběrnice celkem 3. Propojení mezi jednotlivými typy sběrnic je zajištěno pomocí převodových můstků schopných převádět signály AXI na AHB a nao-

pak, nebo AHB na APB (a naopak). Můstek mezi AXI a APB není z důvodu diametrálně rozdílných vlastností sběrnic praktický a tudíž ani dostupný.

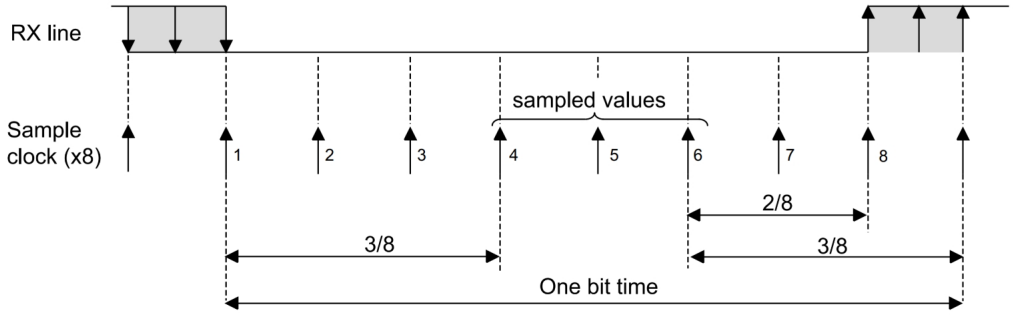
Při komunikaci se servomotory bude klíčové komunikovat bez chyb a s co možná nejvyšší mírou determinismu, to znamená vysílat jednotlivé rámce v pravidelných intervalech a bez problému přijímat odpovědi od servomotorů. Těchto cílů zpravidla nelze dosáhnout bez využití periferie DMA nebo zbytečně vysokého vytížení procesoru, kdy musí jádro kopírovat data mezi periferií a pamětí. Při využití DMA je nutné vědět, do které propojovací matice je sběrnice spojená s danou periferií připojena. Matice jsou v určité výkonové doméně, využití DMA v rámci jedné domény je bez časového postihu, avšak komunikace mezi více doménami zavádí časové prodlevy způsobené převodem. Celá sběrnice AXI je v doméně *D1*, sběrnice AHB má dvě propojovací matice. Jedna je v doméně *D2* a druhá, propojující obvody a periferie s nízkou spotřebou, je samostatné doméně *D3*. Oddělení obvodů s nízkým příkonem do samostatné domény je z důvodu dalšího snížení příkonu. V mikropočítači existuje propojení mezi doménami D1 a D2, ale neexistuje propojení mezi D3 a D1 nebo D2. Jak již bylo zmíněno, tak sběrnice APB nemá přepínače a tudíž nemá ani vlastní doménu, přenos dat při DMA je realizován přes můstky AHB/APB a řadiče DMA v D2. Znalost lokality řadiče DMA a periferie ve zmíněných doménách je pro správnou implementaci klíčová, v případě pokusu o přenos dat mezi D3 a D2 dojde k vyvolání přerušení a nastavení příznaku *TEIFx* informujícím o neplatném přenosu.

Periferie mikropočítače

Mikropočítač obsahuje 3 periferie, které umí realizovat DMA a přenášet data mezi *některými* pamětmi SRAM a *některými* periferiemi. První periferie umožňující DMA je MDMA (*Master DMA*), MDMA se nachází v doméně D1 (přímo na AXI) a je určena pro přenos většího množství dat z paměti do paměti, kde paměť může být připojena k periferii na AXI jak tomu například v případě SDMMC. Další periferií, umožňující DMA je dvojice nazvaná prostě *DMA1* a *DMA2*. Ty se nachází v doméně D2 a je možné je využít pro přenos dat z většiny běžných periferií do pamětí (a naopak), případně i jen mezi pamětmi SRAM. Každý z DMA řadičů v doméně D2 disponuje osmi kanály pro přenos dat, které je možné pomocí DMAMUX připojit na kteroukoliv periferii v blízkosti. Řadiče disponují množstvím přerušení a tak může mít programátor perfektní přehled o aktuálním stavu přenosu i bez zbytečného vytížení jádra mikropočítače. Řadiče disponují nastavení priority, velikosti rámce, počtu rámců k přenosu, volitelnou inkrementaci cílové adresy, módem kruhové paměti aj. Poslední typ periferie je *BDMA* (*Basic DMA*) nacházející se v doméně D3 a tudíž umožňuje přenos dat mezi periferiemi s nízkým příkonem a SRAM4, tato periferie nebude v práci využita.

Jednou z primárních funkcí mikropočítače bude komunikace s inteligentními servomotory Dynamixel (více v kapitole 1), komunikace bude probíhat prostřednictvím poloduplexní asynchronní sběrnice RS-485. V prostředí mikropočítačů se pro tyto účely využívá periferie UART (*Universal Asynchronous Receiver Transmitter*), případně USART lišící se přítomností výstupu pro synchronizaci hodinového signálu přijímače a vysílače, který však může být deaktivován a tím se z USARTu stane UART. Při využití asynchronní sběrnice se přijímač i vysílač synchronizují tzv. start bitem a jelikož je přenosová rychlost známá, není již problém zjišťovat stav signálu Rx (*Receiver* – přijímač) ve správný čas. Periferie U(S)ART dostupná v mikropočítači podporuje i automatické nastavení přenosové rychlosti, která se odvozuje z délky úvodního start bitu. Periferie disponuje módem určeným přímo pro poloduplexní komunikaci (např. zmíněná RS-485) v tomto módu je přímo vyvedený sig-

nál DE (*Driver Enable*) pro aktivaci vysílače. V konfiguračních registrech periferie je možné specifikovat dobu předstihu signálu DE před vysláním start bitu a následně dobu po kterou zůstane signál v log. 1 po odeslání posledního stop bitu. Doba je specifikovaná počtem vzorků. Pro zvýšení odolnosti vůči rušení a tím i spolehlivosti přenosu se využívá metody převzorkování (*oversampling*), které spočívá ve vytvoření více vzorků pro jeden přenesený bit, viz, obrázek 1.13. V případě použitého mikrokontroléru je to 8 nebo 16 vzorků. Konkrétní implementace převzorkování v mikrokontroléru STM32H7 určuje logickou hodnotu signálu dle majority u tří prostředních vzorků pro každý bit. V případě, že nejsou všechny tři vzorky stejné logické hodnoty je disparita indikována pomocí příznaku `NE` (*Noise Error*) v registru periferie, případně je možné povolit vyvolání přerušení.



Obrázek 1.13: Čtení příchozího bitu při převzorkování [26].

Periferie umožňuje příjem a vysílání dat třemi různými způsoby.První dvě možnosti jsou: kopírovat data do registrů periferie a opakovat, dokud nedojde k odeslání, nebo kopírovat data do registrů periferie s využitím přerušení. Oba tyto způsoby potřebují k přenesení více jak jednoho bajtu interakci procesoru. Třetí, a z hlediska potřebných cyklů procesoru nejefektivnější, možností je přenos dat s využitím DMA. V tomto módu je možné přenést libovolně velké, avšak předem (v době běhu) známé množství dat bez zásahu procesoru během vysílání, o průběhu vysílání je jádro notifikované přerušením od DMA, které je schopné signalizovat dokončení poloviny i celého přenosu. Přerušení při přenesení poloviny bloku dat je výhodné při nutnosti přenášet nepřetržitý tok dat, neboť umožňuje nastavení adresy dalšího bloku, který se automaticky přenese po tom aktuálním. Výhodou při vysílání dat je zpravidla předem známé množství dat určené k odeslání. Při příjmu taková možnost často není a je nutné detekovat konec bloku. Detekce může být v mikropočítači realizována pomocí dvou způsobů, prvním je tzv. `IDLE` přerušení, které je vyvoláno ve chvíli, kdy je signál vstupující do přijímače v klidovém stavu. Druhou možností je využít přerušení `RTO` (*Receiver TimeOut Flag* – příznak dosazení časového limitu přijímače) přerušení je vyvoláno ve chvíli, kdy není přijat start bit během nastavitelné doby od posledního stop bitu. Maximální klidová doba mezi stop a start bitem se nastavuje v registru `USART_RTOR` a je vyjádřena jako počet klidových bitů. Mikropočítač ještě nabízí možnost příjmu bloku dat, jejichž délka je známá až v průběhu přijímání, avšak tento mód nelze efektivně využít společně s DMA a proto nebude v rámci této práce dále uvažovaný.

1.8 Použité nástroje a knihovny pro vestavěné zařízení

Při vývoji firmware,[14] je stejně jako při vývoji běžného aplikačního software důležité myslet na přenositelnost a do určité míry platformní invarianci. U standardních aplikací běžících pod operačním systémem není vzájemná kompatibilita s různými počítači problém, operační systém poskytuje stejné funkce nezávisle na hardware. Avšak při vývoji pro vestavěné zařízení, kde je nutná přímá interakce s hardwarovými periferiemi mikropočítače, se kompatibilita zachovává poměrně složitě. Periferie, i když slouží ke stejnému účelu, nemají jednotné API, pomocí kterého se ovládají. Proto vestavěné systémy používají určité abstrakční vrstvy a mimo jiné právě o nich je následující kapitola.

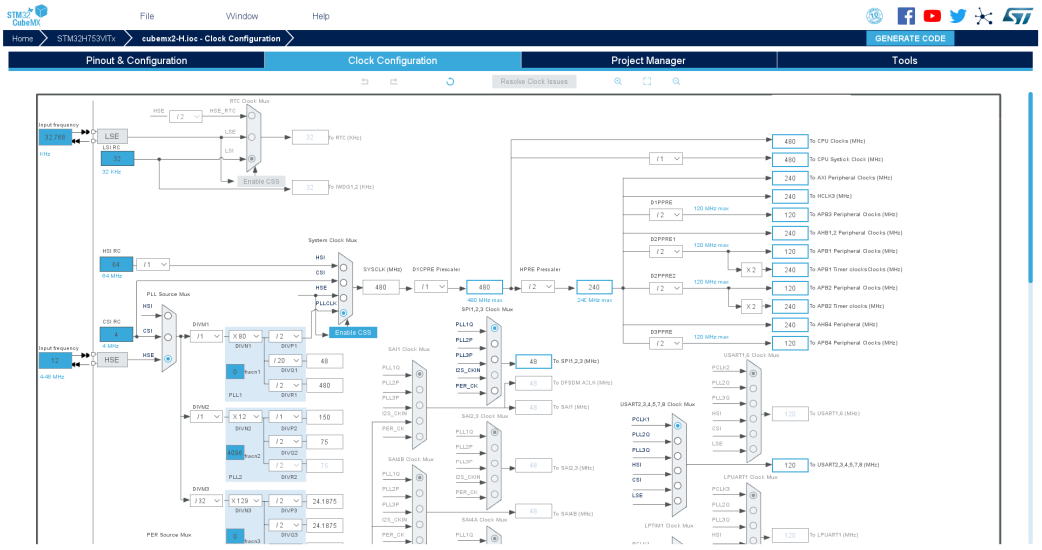
1.8.1 STM32CubeMX a STM32H7 HAL

Výrobce STMicroelectronics ke svým procesorů bezplatně poskytuje knihovny umožňující abstrahovat konkrétní hardwarovou implementaci a k periferiím přistupovat pomocí funkcí vyšší úrovně. Zkratku *STM32H7 HAL* lze dekódovat jako vrstvu abstrakce hardwaru[15] pro řadu mikropočítačů *STM32H7*. Při korektním využití *HAL* je možné poměrně jednoduše přecházet mezi různými typy mikropočítačů stejné řady. V některých případech, kdy se nevyužívá speciálních vlastností periferií obsažených pouze v některých řadách, je možné přejít i mezi řadami např. z řady *F7* na výkonnější a modernější *H7*. Cenou za poměrně jednoduchou přenositelnost je nižší dosažitelný výkon, který způsobuje režie abstrakční vrstvy. Pro aplikace, kde je výkon důležitější než přenositelnost je možné využít tzv. *LL*[16] ovladače. Knihovny umožňují volbu typu abstrakce pro každý typ periferie zvlášť, díky tomu je možné periferie s požadavkem na vysoký výkon řídit pomocí LL ovladačů a firmware důkladně optimalizovat, a pro ostatní periferií využít jednodušší přístup s HAL ovladači.

Jak již bylo zmíněno v předchozím odstavci, pro umožnění přenositelnosti kódu je nutné řídit periferie pouze pomocí funkcí dostupných v HAL. Při základní konfiguraci např. komunikačních periferií není složité držet se pouze možností, které HAL nabízí. Pro konfiguraci pokročilých parametrů periferie slouží rozšiřující funkce dostupné vždy pouze pro určitou řadu, případně jen vybrané typy mikropočítačů. Konfigurace pomocí přímého zápisu do registrů periferie je metoda, která není přenositelná a tudíž ani doporučená pro projekty, kde může dojít ke změně typu mikropočítače. Následně, při využívání periferie pro přenos dat mezi mikropočítačem a okolním systémem, je implementováno více možností jak s periferií komunikovat – dotazování (*polling*), přenos s přerušením, i přenos s využitím DMA. Výhody a nevýhody přenosu dat mezi pamětí a periferií byly blíže popsány v kapitole 1.7.1.

Moderní 32 bit mikropočítače jsou řádově složitější než jejich 8 bit předchůdci a pro svoji korektní funkci vyžadují komplexnější inicializaci. Typickým příkladem je nutnost inicializovat hodinové signály – vybrat správný zdroj signálu, pomocí soustavy děliček a násobiček upravit frekvenci na požadovanou hodnotu a přivést jej ke sběrnicím, jádru, periferiím nebo třeba vyvést ven. Další konfiguraci je nutné provést při využití každé periferie. Pro zjednodušení inicializace mikropočítače výrobce bezplatně dodává software nazvaný *STM32CubeMX*. Software umožňuje konfigurovat mikropočítač pomocí grafického rozhraní, pro každou periferii lze přehledně nastavit potřebné parametry. Nastavení hodinových signálů je doplněno schématem vnitřního propojení a algoritmem pro výpočet nastavení násobiček a děliček pro dosažení požadovaných frekvencí (viz obrázek 1.14).

[14] Firmware – označení software, který je součástí vestavěného systému.
[15] HAL – *Hardware Abstraction Layer* – vrstva poskytující abstrakci hardware
[16] LL – *Low-level* – nízkoúrovňový, avšak stále vyšší než přímý, přístup k periferiím



Obrázek 1.14: Snímek z programu STM32CubeMX, konfigurace hodinových signálů mikropočítače.

Software dále umožňuje konfiguraci tzv. *middleware* mezi které patří *FreeRTOS* a knihovna pro realizaci USB Host zařízení. Na základě nastavení je následně vygenerován projekt pro zvolené vývojové prostředí, případně jen Makefile. Vygenerovaný projekt obsahuje zdrojové soubory STM32H7 HAL, případně, pokud byla zvolena LL knihovna tak je přibalena také. Vygenerovaný projekt je bez problému přeložitelný a spustitelný na mikropočítači. Zdrojové soubory jsou opatřeny blokovými komentáři vymezující místo jehož obsah je zachován i po dalším vygenerování kódu, v případě změny konfigurace mikropočítače. Software také umožňuje výpočet spotřeby mikropočítače při různých režimech napájení, frekvencích jádra, periferií atd. Díky napájení modulu vyvíjeného v rámci této práce síťovým adaptérem nebude tato funkcionalita využita, avšak pro aplikace vyžadující bateriové napájení se může jednat o cennou pomůcku při stanovení výdrže.

1.8.2 Knihovna ETL

Využití *STL* knihoven moderního *C++* není v prostředí vestavěných systémů ideální volbou. Vestavěný systém sestávající z procesoru řady *STH32H7* je, na rozdíl od běžného počítače, omezen řádově nižším výpočetním výkonem a především velmi malou velikostí paměti RAM. *STL* obsahuje základní čtveřici komponent: algoritmy, kontejnery, funkční objekty a iterátory. Implementace komponent ve velké míře využívají dynamickou alokaci paměti i výjimky. Při programování vestavěného systému obecně není doporučeno používání dynamické alokace paměti, protože není možné zaručit dostupnost dostatečného množství souvislé paměti za běhu a následné hledaní chyb v kódu je velmi náročné. Ani při korektní alokaci a následném uvolnění paměti není kvůli možné fragmentaci, vycházející z podstaty hromady, zaručena dostupnost paměti pro další alokaci. Dalším potenciálním problémem při použití *STL* na vestavěném systému jsou výjimky, které, v případě vyvolání, můžou prodloužit dobu potřebnou pro vykonání funkce a tím znemožnit běh systému v reálném čase.

Knihovna ETL[17] popsané problémy řeší a nabízí i další funkce využitelné pro vestavěná zařízení [36]. Knihovna nepoužívá dynamickou alokaci paměti – vše je možné alokovat staticky. Pro kontejnery je využito šablon umožňujících specifikovat typ a maximální počet elementů. Při návrhu byla kladena snaha na co největší využití konstant a výpočtů při kompilaci. Definovatelný systém práce s chybami, který může používat výjimky, chyby předávat do specifikované funkce nebo je ignorovat. Knihovna také minimalizuje využití virtuálních metod, protože jejich volání je, kvůli VMT[18], pomalejší než volání běžných metod. Podstatnou součástí ETL je množství staticky alokovaných kontejnerů definovatelné velikosti, některé jsou podobné kontejnerům z STL například `stl::array`, `stl::list`. Často využívaným kontejnerem je `stl::pool`, který umožňuje statickou alokaci daného počtu objektů, metoda `allocate` vrátí ukazatel na rezervované místo v paměti a následně, metodou `release` je místo vráceno zpět k užívání kontejneru. Dalšími dostupnými kontejnery jsou fronta a zásobník. ETL bohužel nedisponuje integrací s žádným RTOS a tudíž lze kontejnery využít pouze v rámci jednoho vlákna, případně, po přidání zámků i z více vláken, avšak bez výhod, které přináší kontejnery dostupné z RTOS. ETL disponuje implementací dobře známých objektových vzorů, např. *observer* (pozorovatel) nebo *visitor*. Návrhové vzory jsou opět realizovány s důrazem na silnou typovou kontrolu a maximální využití optimalizací v průběhu kompilace. Knihovna taktéž nabízí vlastní, vskutku jednoduchou, implementaci plánovače a vláken, avšak této funkcionality nebude využito.

1.8.3 FreeRTOS

Vestavěné zařízení vyvíjené v rámci této diplomové práce bude vykonávat několik úloh najednou, např. obsluhu USB a komunikaci se servomotory. Mikropočítače operační systémy dříve nepoužívaly, protože omezovaly dostupný výkon a snižovaly množství využitelné paměti RAM. Postupem času se nároky na mikropočítače zvyšovaly a taktéž se zvyšoval jejich výkon i množství dostupné paměti, čímž bylo umožněno využití operačních systémů. V této práci bude využitý operační systém *FreeRTOS*, již dle názvu lze poznat že jde o operační systém reálného času (RTOS – *Real Time Operating System*), který je dostupný zdarma. *FreeRTOS* je od konce roku 2017 pod záštitou firmy *Amazon*, konkrétně dceřiné společnosti *Amazon Web Services*, díky čemuž získal mj. stabilní zdroj financování. *Amazon* také zdarma zpřístupnil dříve placenou knihovnu `FreeRTOS+TCP` umožňující síťovou komunikaci a celý projekt opatřil licencí *MIT*. Dále se podílí na vývoji knihoven umožňující komunikaci s *IoT* službami dostupné v rámci *AWS*.

Klíčovou funkcí RTOS je plánování procesů, *FreeRTOS* podporuje jak kooperativní plánování, kdy se procesy aktivně vzdávají jádra, tak preemptivní, kdy je díky přerušení procesům výpočetní jádro odebráno, případně se jej může i vlákno vzdát. Při plánování procesu je primární informací priorita procesu – nejdříve jsou vykonány odblokované procesy s vyšší prioritou, pokud je jich více, tak se o procesor dělí dle schématu *round-robin*. RTOS dovoluje nastavit tzv. `tickRate` což je maximální počet přepnutí kontextů za sekundu. Často je nastaven na 1000 a tedy kvantum času při round-robin rozdělení je 1 ms. Operační systém poskytuje programátorovi mj. synchronizační primitiva (semafory, mutexy aj.) využitelná pro řízení přístupu ke sdíleným zdrojům. RTOS také umožňuje využití kontejneru pojmenovaného *queue* (fronta), který, na rozdíl od běžně známé fronty, umožňuje vložit prvek jak na dno, tak na vrchol a tím efektivně supluje funkci zásobníku. Výhodou kon-

[17] ETL – *Embedded Template Library* – knihovna šablon pro vestavěné zařízení

[18] VMT – *Virtual Method Table* – tabulka virtuálních metod – mechanizmu umožňující použít virtuálních metod

tejnerů podporovaných v RTOS je možnost přístupu z jiného vlákna a blokování vlákna při čekání na prvek v kontejneru bez nadměrného dotazování a tudíž zbytečného využívání výpočetního času jádra. RTOS má informaci o tom, jaké vlákno je blokováno a v případě provedení operace, která vlákno odblokuje si tuto skutečnost poznačí a vezme v potaz při následujícím přepnutí kontextu. Při implementaci funkcionality využívající *FreeRTOS* bylo postupováno dle referenčního manuálu [1].

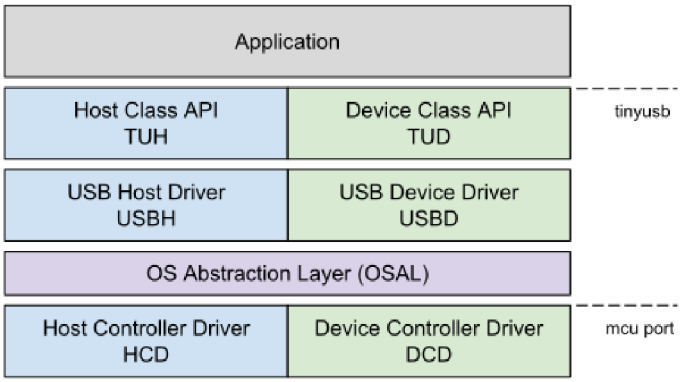
1.8.4 TinyUSB

Mikropočítač disponuje dvěma USB 2.0 periferiemi, obě umí pracovat v módu *device*, *host* nebo *OTG*[19]. Periferie se mj. liší maximální podporovanou rychlostí a způsobem fyzického návrhu zapojení dané periferie. První, označena *OTG-HS1* podporuje rychlosti až 480 Mbit s^{-1} (*USB high-speed*) avšak pouze při použití externího integrovaného obvodu jež zajišťuje přístup k fyzické vrstvě, tzv. *PHY* vrstva. Bez externí fyzické vrstvy se signály *D+* a *D-* připojí přímo na vývody mikropočítače. Maximální rychlost je poté limitována na *full-speed* což je 12 Mbit s^{-1}. Druhá, označena *OTG-HS2* navzdory označení podporuje jen režim bez externí *PHY* s maximálně *full-speed* rychlostí.

Rozhraní USB bude v rámci práce využito pro komunikaci mezi vyvíjeným vestavěným systémem a nadřízeným systémem, což může být běžný počítač nebo tzv. jednodeskový počítač. Při komunikaci bude nutné využít *USB Composite Device* pro implementaci více USB tříd jedním zařízením. Při nutnosti využít USB periferie mikropočítače řady STM32 se nabízí několik možností implementace softwarové části. Jedna z možností je využít oficiálně podporovanou knihovnu, která je součástí HALu. Výhodou je samozřejmě podpora přímo od výrobce, který přesně ví, jak s periferií pracovat. Knihovna implementuje několik standardních USB tříd mj. *CDC*. Bohužel knihovna neumožňuje jednoduše implementovat *USB Composite Device* jehož použití je pro zařízení vyvíjené v rámci této práce nutné. Funkcionalitu lze určitě implementovat pomocí nižších vrstev, ale následné udržování kompatibility s aktuální verzí knihovny by mohlo být problematické. Druhou, avšak nepříliš reálnou možností je využít *LL* rozhraní dostupné v *HAL*, implementace USB protokolu tímto způsobem by však byla dosti náročná, ať už časově tak potřebným množství znalostí.

Na trhu je dostupné množství komerčních implementací knihovny pro komunikaci prostřednictvím USB a při bližším průzkumu bylo zajištěno, že velká část nabízí i všechnu potřebnou funkcionalitu. Jelikož nejsem jediný, kdo by na mikropočítači rád implementoval *Composite Device* i USB protokol obecně, existuje knihovna s otevřeným zdrojovým kódem a permisivní licencí *TinyUSB*, ve zdrojových kódech název často zkracován na `tusb`. Knihovna *TinyUSB* implementuje *device* i *host* funkcionalitu USB, jak je naznačeno na obrázku 1.15. V módu *device* disponuje mj. implementací tříd *CDC* a *HID*. Z hlediska dostupné funkcionality knihovně nic nechybí a je možné ji bez problému využít pro komunikaci s nadřízeným systémem. Knihovna podporuje několik různých výrobců a typů mikropočítačů, mezi nimi i STM32H7. Podstatnou charakteristikou knihovny je provádění majoritní části algoritmu mimo přerušení, čehož je dosaženo ukládáním požadavků do fronty a v jejich následném postupném odbavování pomocí volání definované funkce. Díky integraci *TinyUSB* s operačním systémem *FreeRTOS* je možné využít jeho primitiva (frontu, semafor) právě pro zmíněnou frontu a tím ušetřit cykly, které by byly původně určené ke zjištění, zda

[19] USB OTG – zařízení může přepínat mezi módy *device* a *host*. Typické u mobilních telefonů, které po připojení k počítači fungují jako *device*, a po připojení např. flash disku prostřednictvím adaptéru se přepnou do módu *host* a načtou jej.



Obrázek 1.15: Zjednodušený pohled na vrstvy rozhraní USB s naznačením zodpovědnosti za funkcionalitu [10].

je nový požadavek ve frontě. Knihovna je aktivně vyvíjena nejen původním autorem a nyní zaměstnancem firmy *Adafruit*, která ji využívá ve svých produktech, ale i komunitou.

1.9 Shrnutí

Kapitola představila základní prvky, které budou využity při implementaci vestavěného systému. V první části jsou popsány mechanické vlastnosti manipulátoru a použitých servomotorů, je vysvětleno určení manipulátoru a poskytnutý náhled na technologie ovládaných dotykových displejů. Další část je zaměřena na teoretické základy úloh přímé a inverzní kinematiky i metody plánování pohybu manipulátoru. Vzhledem k tomu, že systém musí komunikovat se servomotory a nadřízeným systémem jsou představeny použité sběrnice a popsány protokoly, které tyto sběrnice využívají. Na závěr má čtenář možnost seznámit se s mikropočítači z rodiny *STM32*, i důvody, které vedly k výběru konkrétního typu mikropočítače *STM32H753*. Je popsána funkce použitých periferií mikropočítače a možnosti abstrakce jeho hardwaru.

Kapitola 2

Návrh a realizace systému

Při návrhu systému vyvíjeného v rámci této práce je nutné myslet více dopředu a nenavrhovat systém pouze na aktuální požadavky. Je velmi pravděpodobné, že v budoucnu se bude systém dále rozvíjet, a pokud by byl zbytečně omezený již při prvotním návrhu tak následné rozšiřování bude, minimálně z pohledu časové náročnosti, velmi drahé. V některých kapitolách může čtenáři přijít, že jsou určité části systému příliš předimenzované. Modul vyvinutý v rámci práce bude sloužit také jako platforma pro další rozvoj a vždy je jednodušší dostupné prostředky (výpočetní, komunikační, paměťové) systému snižovat než zvyšovat. Modul vyvíjený v rámci této práce je natolik specifický, že není možné využít již existující platformu, ale je nutné vyvinout a realizovat vlastní desku plošných spojů. Pro ověření komunikace se servomotory a obecně realizovatelnosti zadání této práce bude navržena zkušební DPS. Tím se omezí případně další iterace nutné pro realizaci finální DPS.

2.1 Požadavky

Výsledkem této diplomové práce, není vytvoření samostatně fungujícího celku, nýbrž vyvinutí komponenty, kterou bude možné v budoucnu integrovat do stávajícího subsystému pro interakci s testovaným zařízením, jenž je součástí systému *RQA* pro validaci a verifikaci. Subsystém pro interakci obsahuje senzory a aktory, ovládané prostřednictvím *REST API* volání na HTTP server, běžícím na jednodeskovém počítači [8]. Díky této skutečnosti je možné přesně specifikovat požadavky na dostupné vstupní a výstupní konektory společně s jejich funkcí.

Před započetím vývoje bylo nutné si ujasnit směr budoucího vývoje subsystému pro interakci s MFD, na jehož základě vzniky níže uvedené požadavky na navrhovaný vestavěný systém.

- Zachovat stávající modulární strukturu celého subsystému.
- Zachovat stávající propojení modulů pomocí plochého 6vodičového kabelu obsahujícího 12 V a 5 V napájení, kde 12 V nemusí být vždy dostupné.
- Zachovat komunikační vrstvu realizovanou robustní a v automobilovém průmyslu prověřenou sběrnicí CAN.
- Navrhovaný modul musí umožnit redundantní napájení sběrnice, tzn. při výpadku napětí na sběrnici a přítomnosti externího zdroje jej použít pro napájení sběrnice.
- Umožnit do budoucna zastat funkci některých, již realizovaných modulů a tím zjednodušit práci operátorovi při zapojování.

- Na DPS vyvést často používané komunikační sběrnice, vstupy ADC[1] a PWM[2] výstup pro ovládání modelářských servomotorů.
- Umožnit aktualizaci programu mikropočítače (firmware) bez nutnosti použít programátor (například pomocí *DFU*[3]).

Modulární struktura subsystému je časem ověřená a ukázala se jako dostatečně rozšiřitelná, díky čemuž nic nebrání budoucímu napojení vyvíjeného systému do subsystému v podobě dalšího modulu. Na rozdíl od modulů produkovaných do této doby však bude přikročeno ke koncentraci více funkcí do jednoho modulu, protože se ukázalo, že vyrábět samostatné moduly s jednou funkcí není efektivní ať už z hlediska času nutného pro návrh a realizaci, tak z pohledu fixních výrobních nákladů (masky pro výrobu DPS, nerezové šablony pro nanášení pájecí pasty, programování osazovacích automatů aj.).

Do dnešní doby mohla být sběrnice napájena pouze modulem sloužícím pro převod rozhraní USB na CAN a to pouze jedním. Sběrnice, ale i samotný modul, neumožňovaly současné připojení více zdrojů napětí, či napájení prostřednictvím jednoho modulu a komunikaci jiným. Popsaný návrh bez problému vyhovoval při využívání systému pro základní úlohy, ale po konzultaci se spolupracovníky bylo rozhodnuto, že možnost napájet sběrnici z více míst se do budoucna bude hodit. Navrhovaný modul bude schopný sběrnici napájet, pokud to bude potřeba, a zároveň více modulů umožní napájení z více míst. Navrhovaný modul bude také poskytovat napájecí napětí 5 V pro modelářské servomotory. I tato napájecí větev bude s možností napájení z více zdrojů proudu. Poslední, avšak neméně důležitou napájecí větví, bude 3,3 V pro mikropočítač a další integrované obvody na DPS. Jako zdroj proudu primární napájecí větve bude sloužit externí spínaný zdroj s napětím 12 V, napájení ze společné sběrnice a konektor USB, v případě, že bude modul připojený k počítači. Při volbě zdroje napájení bude uplatněný princip priority např. v případě, že je dostupný externí zdroj 12 V a zároveň je přítomné napětí ve společné sběrnici tak se pro napájení prvků s vyšším příkonem (modelářské servomotory a robotická ruka) použije externí zdroj napětí a nebude se zbytečně zatěžovat sběrnice. U primárních napájecích větví je nutné zajistit jejich ochranu před nadproudem v souladu s použitými externími zdroji. Také je nutné omezit proud dodávaný na společnou sběrnici a v případě překročení limitů sběrnici odpojit od napájení. A toto platí jak pro 12 V tak pro 5 V větev sběrnice.

rozhraní	**počet**
Inteligentní servomotory	2
SPI	2
I^2C	2
UART	2
Analogový vstup	4
PWM výstup	4

Tabulka 2.1: Předpokládaný počet potřebných komunikačních rozhraní.

V požadavcích na vestavěný systém je uvedeno, že se mají na DPS vyvést často používané komunikační sběrnice. Nyní sice nebudou mít předem dané určení avšak v případě, že by se k DPS připojoval nějaký další systém nebo jen další sběrnice, budou připravené pro použití. V rámci subsystému pro interakci s testovaným zařízením se mezi často používané

[1] *Analog to Digital Converter* – převodník analogového signálu na digitální

[2] *Pulse-Width Modulation* – pulzně šířková modulace

[3] DFU – device firmware upgrade – protokol umožňující přímé nahrání firmware do mikropočítače prostřednictvím USB

sběrnice řadí I²C, SPI a UART (RS-232, plně duplexní bez řízení toku) v logických úrovních 3,3 V. Předpokládané počty konektorů pro jednotlivá dostupná rozhraní jsou uvedené v tabulce 2.1.

2.2 Výběr mikropočítače

Při volbě vhodného mikropočítače byl kladen důraz na dlouhodobou dostupnost, přijatelnou cenu, vysoký výkon a v nejlepším případě i předchozí zkušenost s danou řadou ať už moji nebo spolupracovníků zadávající firmy Y Soft. Dle těchto kritérií byla vybrána řada vysoce výkonných mikropočítačů *STM32H7*. Při výběru konkrétního mikropočítače byly požadavky následující: vysoký výkon, dostatečný počet potřebných komunikačních periferií (2× CAN, 2× I²C, 2× SPI, 3× U(S)ART), hardwarová akcelerace hašovacích a kryptografických algoritmů a v neposlední řadě pouzdro pájitelné běžně dostupnou technologií, například *LQFP*[4] aj. Konkrétní typ vybraného procesoru je *STM32H753VI*, který bez problému splňuje veškeré požadavky na něj kladené a jako bonus obsahuje i matematický koprocesor s dvojitou přesností ulehčující práci hlavní ALU.

Mikropočítač obsahuje nejvýkonnější, aktuálně dostupné, licencované jádro *M7* z řady *ARM Cortex-M* což je řada procesorů určená pro použití v mikropočítačích [12]. Maximální dosažitelná frekvence jádra je 480 MHz, takové množství výkonu nebude v této aplikaci ani zdaleka využito. Avšak i při porovnání mikropočítače s jinými řadami, které mají méně výkonná jádra, menší množství periferií a nedisponují akcelerací výpočtu s dvojitou přesností vychází vybraný mikropočítač velice srovnatelně ať už z pohledu ceny nebo skutečné dostupnosti u distributorů.

Samozřejmostí u této třídy mikropočítačů je řadič přímého přístupu do paměti (DMA), umožňující periferiím přenášet data z nebo do SRAM mikropočítače bez plnění vyrovnávacích pamětí periférií jádrem. Procesor má k dispozici 1 MB paměti SRAM a na stejném křemíku je i 2 MB paměti typu Flash pro uložení programu. V případě, že by to nestačilo, je možné standardní sběrnicí *Quad-SPI* připojit další paměť Flash, případně i další paměť RAM. Nedostatek paměťového prostoru se v této práci neočekává. Každý vyrobený mikropočítač řady *STM32* má 96 bitový celosvětově unikátní identifikátor. Vybraný mikropočítač dále integruje velké množství periferií, mezi ty, které se budou v rámci práce potenciálně používat patří:

- 22 čítačů různých typů,
- 8× UART,
- 6× SPI,
- 4× I²C,
- 2× CAN-FD kontrolér[5],
- USB kontrolér,
- 3× ADC převodníky s multiplexorem a diferenciálním vstupem,
- kryptografický akcelerátor akcelerující výpočet AES, SHA.

[4] LQFP – Low-profile Quad Flat Package – čtvercové pouzdro s vývody po 4 stranách

[5] CAN-FD – *CAN flexible data rate* – rozšíření protokolu CAN umožňující flexibilní rychlost sběrnice

2.3 Výkonové spínací prvky

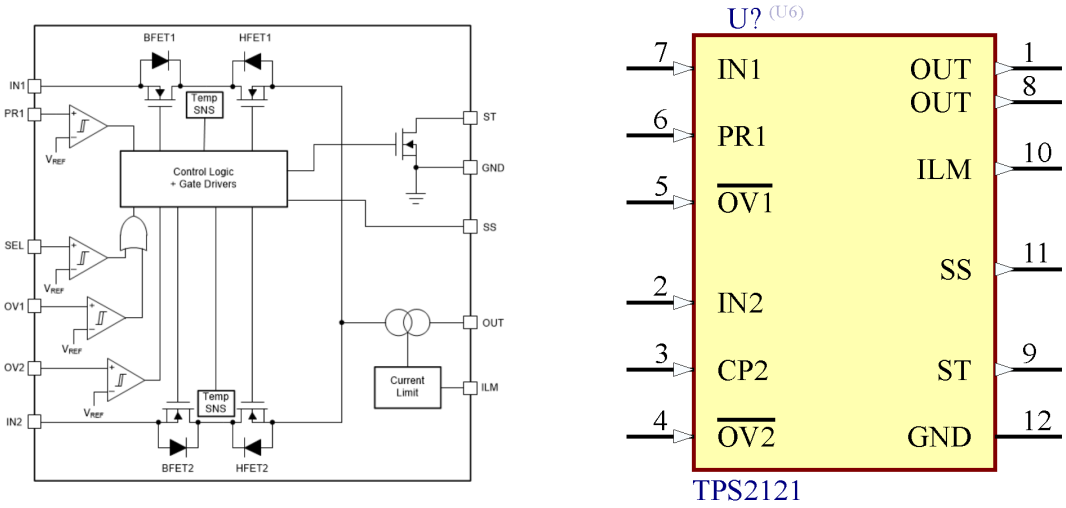
Od modulu se požaduje určitá míra správy napájecích větví, ať už vlastních, využitých pouze součástkami na DPS, tak externích, využitých modelářskými servomotory a robotickou rukou nebo větví ve společné sběrnici propojující subsystém pro interakci s testovaným zařízením. Realizaci obvodů pro spínání silových napětí[6] je vhodné nechat na odbornících a využít integrované obvody (IO), které navrhli. V této kapitole budou takové obvody popsány společně s IO pro snížení napětí na úrovně vhodné k napájení mikropočítače a ostatních logických prvků.

2.3.1 Multiplexování napájecích větví

Dle požadavků kladených na vyvíjený modul v sekci 2 bude nutné realizovat multiplexování napěťových větví. Modul bude mít dostupné 4 různé zdroje napětí, ze kterých bude napájet několik napěťových větví pro IO v modulu, případně ostatní prvky připojené k modulu nebo ke společné sběrnici. Přehled jednotlivých napěťových prvků je ve výčtu níže, na pravé straně jsou zdroje napětí, tzv. primární napěťové větve, na straně levé větve určené spotřebičům.

- 12 V z externího spínaného zdroje
- 12 V ze společné sběrnice
- 5 V z USB konektoru
- 5 V ze společné sběrnice
- 12 V, 2 A společná sběrnice
- 5 V, 2 A společná sběrnice
- 5 V, 3 A modelářské servomotory
- 3,3 V, 200 mA napájení mikropočítače
- 12 V, 4 A napájení servomotorů manipulátoru

Pro realizaci multiplexování napěťových větví byl vybrán integrovaný obvod *TPS2121* od firmy *Texas Instruments*. Obvod v sobě integruje výkonový přepínač, tzv. *power MUX*, možnost nastavení priority pro vstupní větve i nastavitelný omezovač proudu. Obvod je schopný pracovat se vstupním napětím až 22 V a maximální proudem 4,5 A, blokový diagram obvodu a použitá schématická značka jsou na obrázku 2.1.



Obrázek 2.1: Blokový diagram [33] (vlevo) a schématická značka (vpravo) integrovaného obvodu TPS2121.

[6] jedná se o silové napětí v kontextu vyvíjeného modulu, tj. napětí vyšší než běžné logické úrovně

Obvod se chová jako multiplexor, který má dva výkonové vstupy a jeden výkonový výstup. Vstup použitý pro napájení výstupu se vybírá pomocí signálů *PR1* a *CP2*. Obvod disponuje několika módy funkce, avšak pro tuto práci je relevantní pouze mód nazvaný *Automatic Switchover with Priority (XCOMP)* což lze přeložit jako *automatické přepojení s prioritou*. V tomto módu se porovnávají napětí na vstupech *PR1* a *CP2*. V případě poklesu napětí vstupu *PR1* pod úroveň *CP2* je přepnuto na druhý výkonový vstup a opačně. Při přepínání je chování obvodu blízké ideální diodě a díky tomu nedojde k citelnému poklesu napětí na výstupu. Případný malý pokles se vyrovná kondenzátory, a zařízení připojená na výstup bez problému pokračují v činnosti. Pomocí vstupů *OV1* a *OV2* je možné zvolit maximální dovolené napětí pro daný vstup. V případě, že napětí na vstupu překročí mez 1,06 V je daný výkonový vstup vyloučen z používání. Tím je možné zabránit poškození zařízení na výstupu obvodu vlivem příliš vysokého napětí. Vstupem I_{lim} je možné nastavit proudové omezení na výstupu. Rovnice

$$I_{lim} = \frac{65,2}{R_{lim}^{0.861}} \tag{2.1}$$

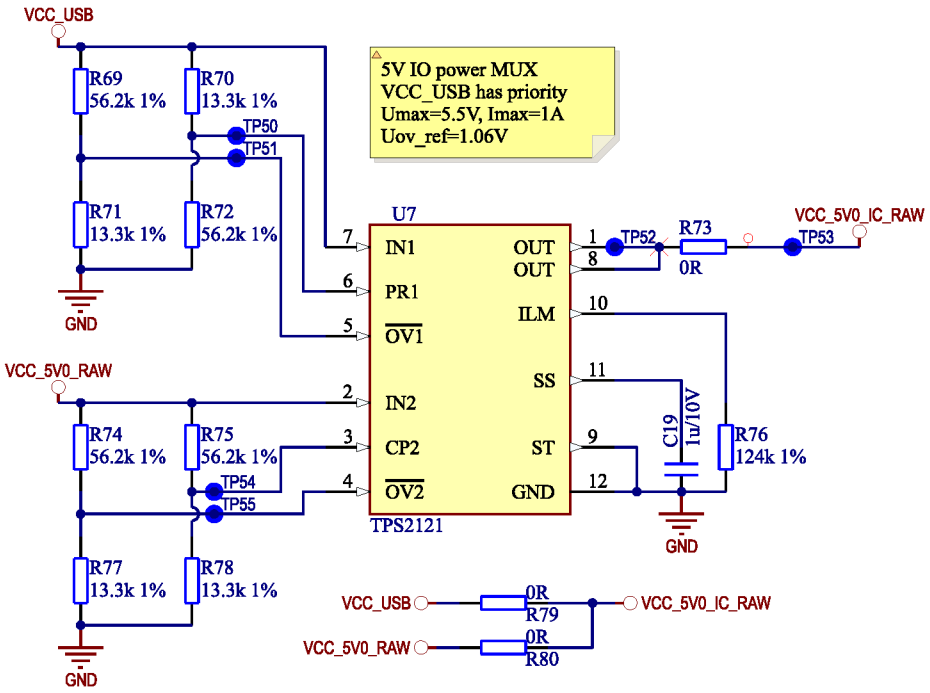
definuje výpočet hodnoty odporu rezistoru R_{lim}. V případě, že dojde k překročení nastaveného proudu spínací tranzistor uvnitř obvodu začne proud omezovat, tím dojde ke zvýšení ztrátového výkonu obvodu a zvýšení jeho teploty až k mezi teplotní deaktivace nastavené výrobcem na 160 °C, která způsobí odpojení výstupu od napětí. Díky odpojení výstupu klesne ztrátový výkon obvodu, ten se ochladí pod 150 °C a následně opět aktivuje. Poslední užitečný vstup je *SS* pomocí připojeného kondenzátoru se nastaví prvotní prodleva určená ke stabilizaci napěťových úrovní mj. na vstupech *PR1* a *CP2*, po jejímž vypršení dojde k prvnímu porovnání napětí na vstupech, které určí použitý výkonový vstup. Sekundární funkcí vstupu *SS* je nastavení funkce *soft start*. Funkce je aktivovaná pouze při prvotním sepnutí napětí, zajišťuje postupný náběh napětí na výstupu pro omezení napěťových špiček způsobených možnou indukční charakteristikou výstupního vedení. Integrovaný obvod bude využitý pro prioritní multiplexaci napájecích větví. Zapojení obvodu použité v modulu je na obrázku 2.2, rezistory *R79* a *R80* se běžně neosazují a slouží jen pro případné přemostění výkonového přepínače. Pomocí napěťového děliče sestaveného z rezistorů *R69* a *R71* je konfigurované maximální vstupní napětí. Pro stanovení vhodných hodnot rezistorů se vychází z následující rovnice:

$$U_{OV1} = U_{OV} \times \frac{R_{71}}{R_{71} + R_{69}}, \tag{2.2}$$

kde po dosazení dostaneme maximální dovolené vstupní napětí

$$U_{OV1} = 1,06\,\text{V} \times \frac{13,3\,\text{k}\Omega}{13,3\,\text{k}\Omega + 56,2\,\text{k}\Omega} \approx 5,54\,\text{V}. \tag{2.3}$$

Napěťový dělič se stejnou hodnotou rezistorů je použitý i pro druhý vstup. V rámci ušetření množství různých součástek jsou použity rezistory stejných hodnot i pro napěťové děliče konfigurující prioritu vstupů, jednou ve stejném pořadí, podruhé převráceně. Rezistor *R73* slouží pro oddělení obvodu od zbytku a používá se při vývoji a testování. Kruhové body s označením *TP* indikují využití měřicích bodů, které je opět možné použít při vývoji modulu nebo při testování pro ověření přítomnosti správného napětí.



Obrázek 2.2: Schéma jednoho zapojení obvodu TPS2121, rezistory *R79* a *R80* se neosazují.

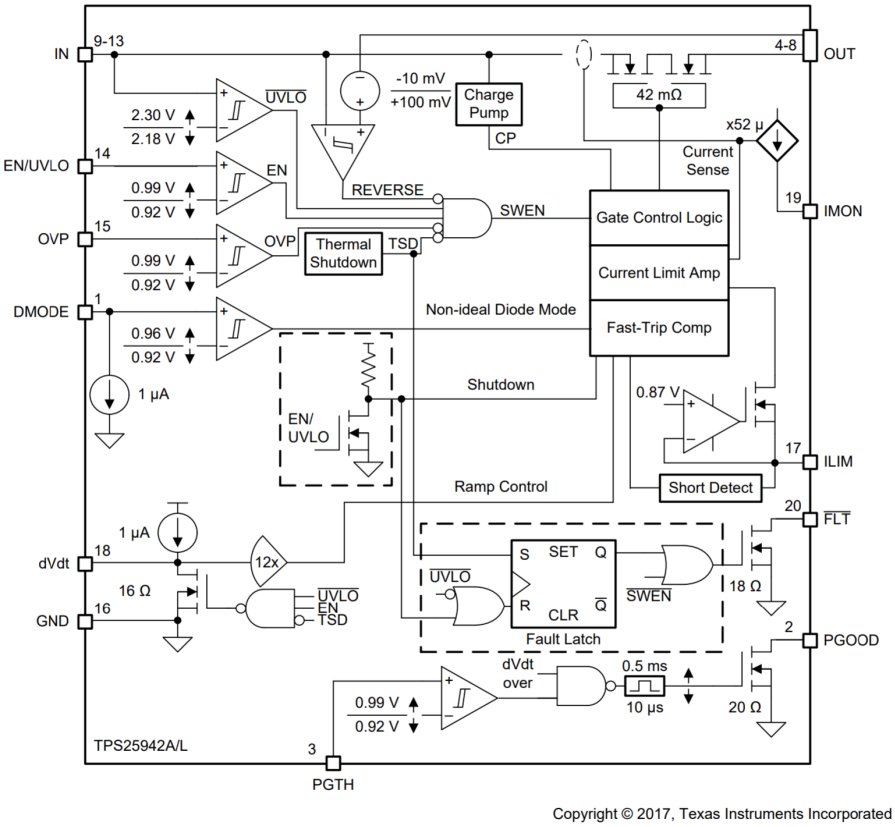
2.3.2 Ochrana zdroje napětí a multiplexování napájení na sběrnici

Pro napájení modulu je možné použít více zdrojů a jedním z nich je i externí spínaný zdroj stejnosměrného napětí tzv. SMPS[7]. Ačkoliv mají *SMPS* mechanismy, jak se chránit před odběrem většího než dovoleného množství proudu, tak je dobrým zvykem u spotřebiče použít nadproudovou ochranu také. Běžně se může jednat o jednorázovou trubičkovou pojistku, vratnou polymerovou pojistku (*PPTC*) nebo elektronickou pojistku s nastavitelným maximálním proudem. Napájené zařízení je také vhodné chránit před tzv. přepólováním tj. zapojením napájení naopak. Integrovaný obvod představený v sekci 2.3.1 disponuje množstvím funkcí avšak některé, např. možnost monitorování proudového odběru, stále schází.

Obvod disponující potřebnými funkcemi nese označení *TPS25942A* jehož výrobcem je, stejně jako v předchozím případě, společnost *Texas Instruments*. IO zajišťuje funkci elektronické pojistky, multiplexování napájení, monitorování výstupního proudu, ochranu proti přepólování napájecího napětí i proti zpětnému proudu z výstupní větve do vstupní a také indikaci stavu výstupu.

Napájení sběrnice z více míst je umožněno právě díky ochraně obvodu před zpětným proudem do vstupu, v takové situaci se výstup odpojí, avšak nadále monitoruje aby bylo v případ poklesu napětí na výstupu pod určitou mez zajištěno sepnutí integrovaného tranzistoru a tím udržení napětí na výstupu v tolerované hranici. Pro zajištění odolnosti proti přepólování výrobce doporučuje umístit diodu mezi zem IO a zem napájecího napětí a následně všechny napěťové děliče počítat ve vztahu k zemi IO, která bude, vzhledem k V-A charakteristice diody, odlišná od země celého modulu a to až o 0,7 V. Za účelem zajištění co nejmenšího rozdílu napětí mezi zemí IO a skutečnou zemí modulu byl namísto diody použitý malý signální unipolární tranzistor zapojený takovým způsobem, aby umožnil průtok

[7]SMPS – switched-mode power supply – spínaný napájecí zdroj, zpravidla vytváří ze střídavého proudu v zásuvce nižší, stejnosměrné, napětí.



Obrázek 2.3: Blokový diagram integrovaného obvodu TPS2594x [31].

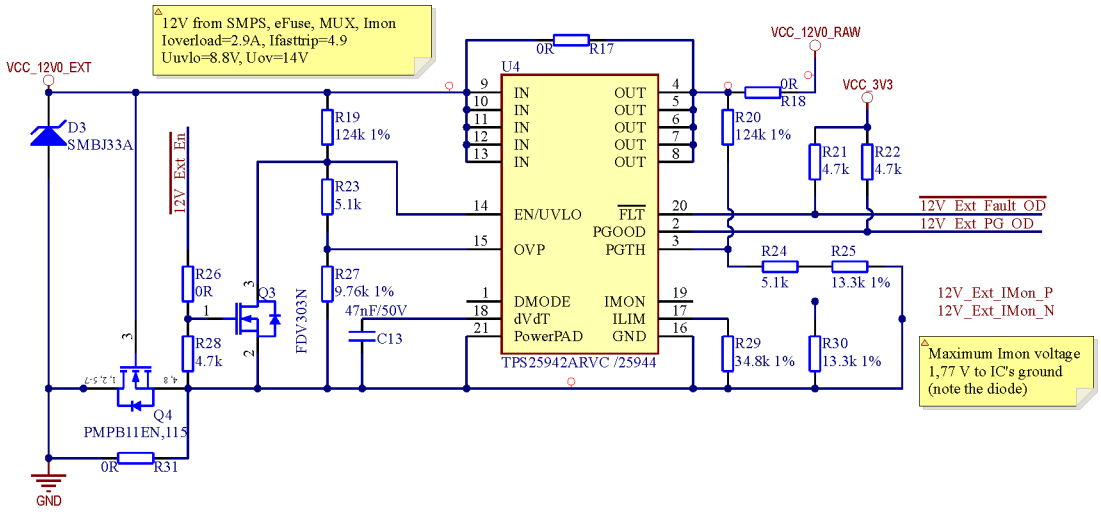
proudu pouze v jednom směru. Napěťové ztráty na tranzistoru budou při proudu v řádech jednotek mA zanedbatelné a tím zjednoduší návrh zapojení.

Obvod se konfiguruje podobným způsobem jako dříve popsaný *TPS2121* tj. množstvím napěťových děličů. Pro ulehčení návrhu děličů výrobce poskytuje *TPS2594x Design Calculation Tool* což je připravený dokument pro program Microsoft Excel, do kterého se vyplní požadované vlastnosti obvodu a následně spočítá vhodné kombinace rezistorů k jejich dosažení. V rámci optimalizace množství druhů rezistorů byla provedena manuální korekce napěťových děličů při zachování potřebných vlastností. Vstup *EN/UVLO*[8] uvádí obvod do provozu a umožní sepnutí integrovaného výkonového tranzistoru pouze při překročení hranice 0,92 V. Efektivně zabraňuje neočekávanému chování obvodů napájených prostřednictvím *TPS25942* v důsledku nízkého napětí. Stejně jako ochranu proti podpětí, IO nabízí i ochranu proti přepětí, opět nastavitelnou napěťovým děličem jehož výstup je připojený na vstup *OVP*, ochrana je aktivována při překročení napětí 0,99 V. Podstatným rozlišujícím prvkem, oproti *TPS2121*, je přítomnost proudového výstupu I_{mon}, kterým protéká proud přímo úměrný proudu protékajícím výkonovým tranzistorem IO. Po připojení měřícího rezistoru známé hodnoty je při znalosti převodní konstanty výstupu I_{mon} 52,3 µA A^{-1} měřením napětí na daném rezistoru jednoduché určit proud tekoucí obvodem. Dalším rozdílem oproti *TPS2121* je přítomnost výstupů indikující stav integrovaného obvodu. Dostupné stavové výstupy jsou *PG*[9] a $\overline{FLT}$ indikující nějaký problém. Oba výstupy jsou v provedení

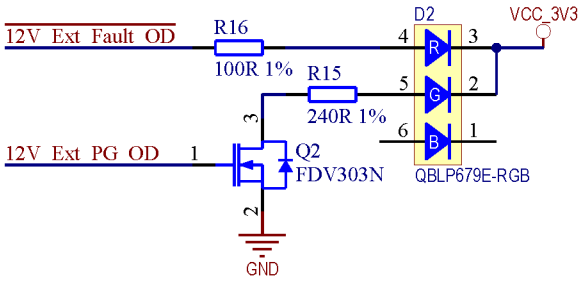
[8] UVLO – *Undervoltage-Lockout* – funkce deaktivace obvodu při detekci podpětí

[9] PG – *power good* – indikace toho, že je napájení v pořádku a není přítomen žádný problém

otevřený kolektor to znamená že logická *1* způsobí propojení výstupu emitorem tranzistoru se zemí. Proto je k výstupu připojen tzv. pull-up[10] rezistor s odporem 4,7 kΩ. Ke zmíněným výstupům je připojena i indikační LED v provedení RGB kde se využívá pouze R – červený a G – zelený kanál pro indikaci $\overline{FLT}$ resp. *PG*. Výsledné zapojení integrovaného obvodu je na obrázku 2.4, zapojení LED pro indikaci je na obrázku 2.5.



Obrázek 2.4: Schéma zapojení obvodu TPS25942, rezistory *R17* a *R31* se neosazují.



Obrázek 2.5: Schéma zapojení indikace k obvodu TPS25942.

2.3.3 Snižování napětí

Modul je napájen napětím 12 V, v případě, že je připojené USB do počítače je k dispozici i 5 V. Avšak pro korektní funkci modulu a případných dalších připojených prvků je nutné zajistit trvalou dostupnost 5 V a 3,3 V. Pro konverzi napětí na nižší úroveň je možné využít buď lineární regulátor nebo stejnosměrný spínaný zdroj tzv. *snižující DC-DC měnič*.

Výhodou lineárního regulátoru je nízké zvlnění výstupního napětí a jednoduchost integrovaného obvodu včetně veškerých pasivních podpůrných součástek, zpravidla pouze vstupních a výstupních filtračních kondenzátorů. Nevýhodou lineárního regulátoru napětí je množství produkovaného ztrátového tepla, které je vyjádřeno jako

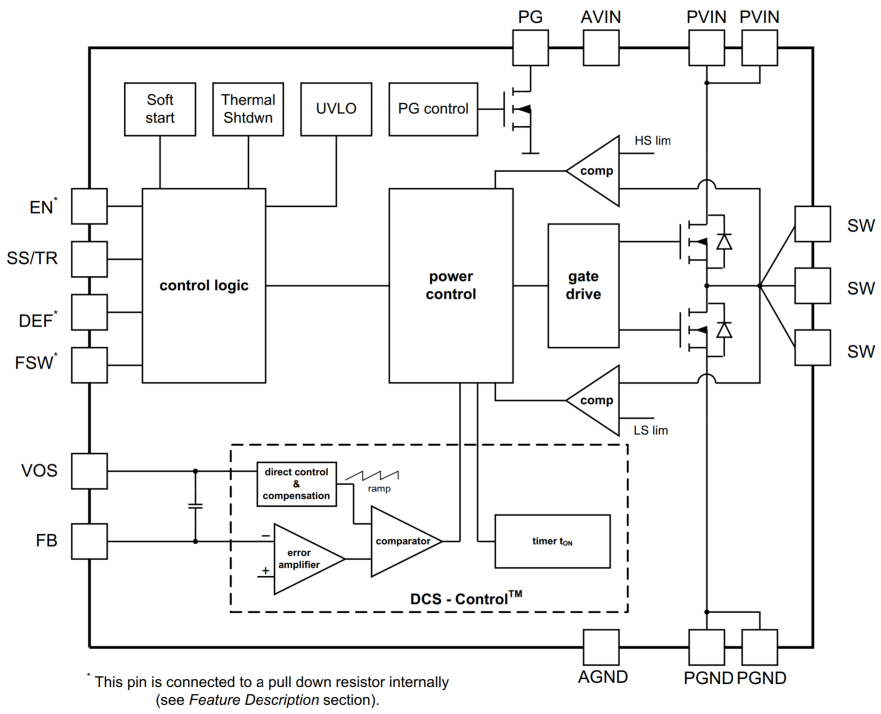
$$P = (U_{in} - U_{out}) \times I, \tag{2.4}$$

[10] pull-up – rezistor umístěný mezi kladným napájecím napětím a signálním vodičem.

kde U_{in} je vstupní a U_{out} výstupní napětí. Lineární regulátor se hodí do míst, kde je menší rozdíl vstupního a výstupního napětí a nízká proudová zátěž. Jedním z míst vhodných pro použití tohoto typu regulátoru je snížení napětí 5 V na 3,3 V pro potřeby mikropočítače.

Druhá možnost snížení napětí je pulzní stejnosměrná konverze, jenž zaručuje řádově nižší tepelné ztráty. Typická účinnost konverze se pohybuje nad 80 %. Nevýhodou takového řešení je větší použitá plocha na DPS, složitější návrh, možné rušení logických integrovaných obvodů i vyšší cena potřebných komponent. Proto se hodí například při snižování napětí z 12 V na 5 V pro napájení modelářských servomotorů, jejichž proudový odběr je v nízkých jednotkách A. Další větví, kde se použije spínaný regulátor je opět konverze 12 V na 5 V, avšak tentokrát pro potřebu napájení společné sběrnice. Maximální proudový odběr na této větvi je kolem 3 A.

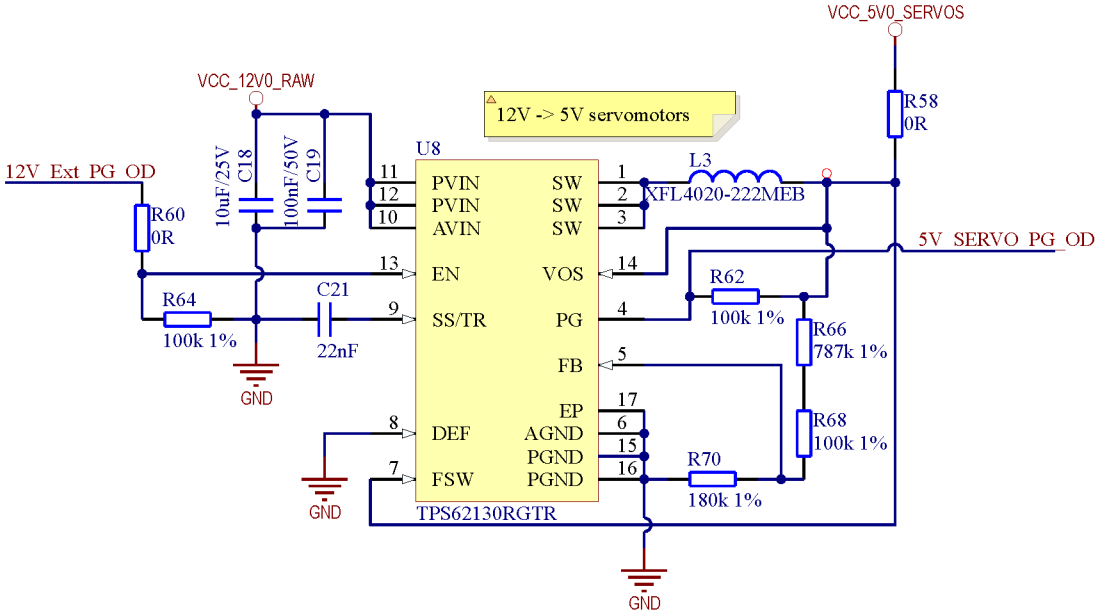
Pro výběr vhodného DC-DC měniče byl použit nástroj firmy *Texas Instruments* nazvaný *WEBENCH® Power Designer*. Vstupem do nástroje jsou požadované parametry měniče: rozsah vstupního napětí, požadované výstupní napětí a proud a preferovaný směr optimalizace návrhu (vyrovnaný, nízká cena, malá plocha na DPS, vysoká efektivita). Po zadání požadovaných parametrů byla prezentována řada řešení, výběr konkrétního typu měniče byl také ovlivněn již používanými obvody a pasivními součástkami ve firmě Y Soft, protože je vždy jednodušší využít součástky na skladě než vybírat jiné a ověřovat funkcionalitu nového návrhu.



Obrázek 2.6: Blokový diagram integrovaného snižujícího DC-DC měniče TPS62130 [30].

Konkrétním zvoleným měničem je synchronní step-down *TPS62130* opět od firmy *Texas Instruments*, jehož blokový diagram je na obrázku 2.6. Mezi hlavní vlastnosti měniče patří rozsah vstupního napětí od 3 V do 17 V, výstupní napětí nastavitelné pomocí napěťového děliče a maximální trvalý výstupní proud 3 A. Obvod obsahuje standardní ochranné mechanizmy schopné zachytit zkrat na výstupu nebo zamezit teplotnímu poškození. Další

vlastnosti, které již nejsou klíčové, avšak budou využity, jsou postupné zvyšování napětí na výstupu tzv. *soft-start* omezující proudové špičky nebo třeba signál *PG* informující o dosažení nastaveného napětí na výstupu. Jak již bylo zmíněno výše, tak spínaný měnič je, na rozdíl od lineárního regulátoru, nutné doplnit několika pasivními součástkami. Pro funkci měniče je klíčovou součástkou induktor. Dle doporučení výrobce byl vybrán stíněný induktor typu *XFL4020-222MEB* s indukčností 2,2 µH, proudovou zatížitelností až 3 A. Výstupní napětí je nastaveno napěťovým děličem s poměrem výstupního napětí 0, 169. Jedno z realizovaných zapojení měniče v modulu je na obrázku 2.7.



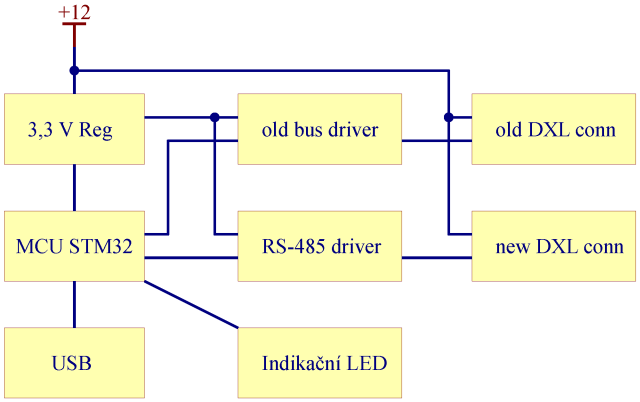
Obrázek 2.7: Schéma zapojení DC-DC měniče TPS62130.

2.4 Zkušební DPS

Jak již bylo zmíněno v úvodu této kapitoly pro test realizovatelnosti a vyzkoušení komunikace se servomotory byla navržena jednodušší DPS. Zjednodušené blokové schéma zkušební DPS je na obrázku 2.8. Jejím úkolem bylo sloužit jako prostředek pro ověření:

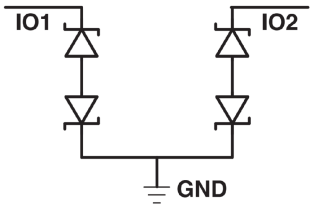
- návrhu napájení mikropočítače,
- stability mikropočítače při zatížení servomotorů,
- komunikace se servomotory prostřednictvím dvou typů sběrnic,
- USB komunikace s počítačem,
- dostatku výpočetního výkonu (velmi orientačně).

Výše jsou zmíněné 2 typy sběrnic, v teoretické části práce byla popsána pouze novější sběrnice která přesně odpovídá RS-485. Starší typ sběrnice se využíval pro předchozí řady motorů a na rozdíl od RS-485 není plně standardizovaná. Namísto novějšího typu využívá pro komunikaci pouze jeden vodič, není tedy diferenciální a tím je více náchylná k rušení i přeslechům, které jsou způsobené servomotory. Sběrnice využívá logické úrovně standardního sériového portu RS-232, díky čemuž je možné využít integrované řadiče této sběrnice, což ulehčuje implementaci. Konektor staršího typu sběrnice je pouze 3vodičový (zem, 12 V, data), na rozdíl od 4vodičového u nové sběrnice.



Obrázek 2.8: Blokové schéma zkušební DPS.

Během návrhu bylo využito informací čerpaných z knihy [11], která popisuje mj. správný způsob návrhu DPS pro vestavěné aplikace, vodítka, jak postupovat při kreslení schématu i kladení cest na DPS. Při návrhu napájení mikropočítače jsem vycházel z aplikační poznámky výrobce [23], která popisuje, jak správně navrhnout napájení pro spolehlivý chod. Pro návrh zapojení komunikačních sběrnic bylo využito katalogových listů daných integrovaných obvodů. Pro novější typ sběrnice to byl obvod *SN65HVD11D*, který nemá integrované ochranné prvky proti ESD. Proto byly mezi integrovaný obvod a konektor přidány tlustovrstvé rezistory 10 Ω a co nejblíže ke konektoru dvoukanálový ochranný integrovaný obvod *TPD2E007*, vnitřní schéma zapojení integrovaného obvodu je na obrázku 2.9. IO v sobě, pro každý chráněný kanál, integruje dvojici transilů[11] zapojených do série proti sobě propojujících chráněný kanál se zemí. Díky tomu se vysokonapěťové špičky svedou do země a řadič sběrnice se před nimi ochrání. Celkové schéma zapojení řadiče sběrnice spolu s ochranným IO je na obrázku 2.10.

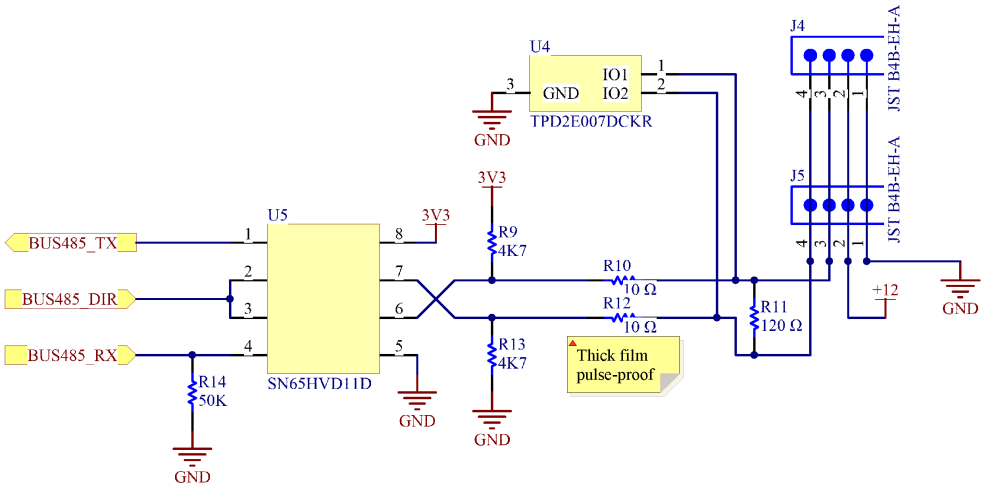


Obrázek 2.9: Vnitřní schéma zapojení integrovaného obvodu TPD2E007[29].

Dále bylo na testovací DPS realizováno zapojení řadiče staršího typu sběrnice pro komunikaci se servomotory. Zapojení je podobné sběrnici RS-485, proto zde nebude detailně popsáno. Dalším úkolem bylo vyzkoušet komunikaci s počítačem prostřednictvím sběrnice *USB 2.0* v rychlosti *Full Speed* (dále jen *USB FS*). Mikropočítač obsahuje *USB FS* periferii, postačí tedy jen připojení konektoru a analogicky s RS-485 vložit do cesty ochranné prvky pro zamezení zničení mikropočítače vlivem ESD. Pro ochranu USB byl vybrán ochranný prvek *STM-USBLC6-2-6*, který byl, jak již označení napovídá, vyvinut speciálně pro USB sběrnice. Avšak podobně jako *TPD2E007* obsahuje transil a několik diod pro svedení špičkového proudu do země. V rámci zjednodušení zkušební DPS bylo pro napájení mikropočítače

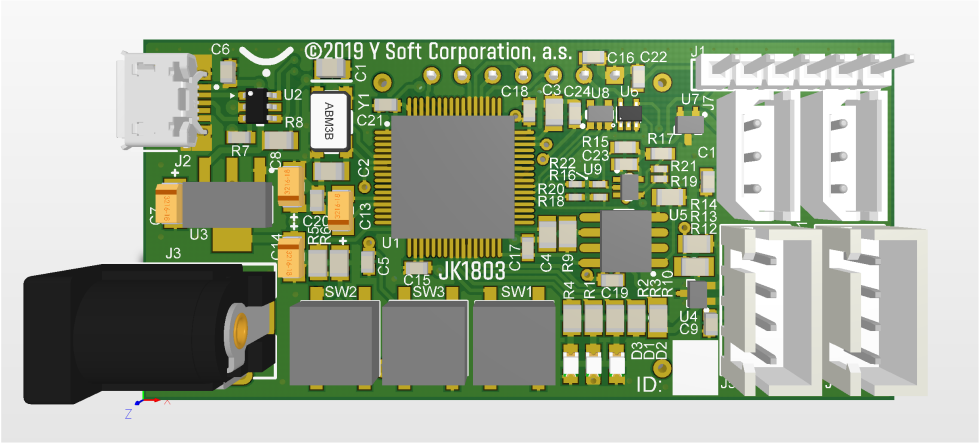
[11] Transil – polovodičový prvek podobný diodě, navržen na vysoké impulzní proudy, prvek se při přetížení zkratuje.

a dalších komponent vyžadující napěťovou větev 3,3 V využito lineárního regulátoru napětí *LM1117MP-3.3* ve standardním zapojení dle katalogového listu jednoho z výrobců (10 µF kondenzátor na vstupní a výstupní napěťovou větev pro zamezení kmitání a napěťové nestability).



Obrázek 2.10: Schéma zapojení řadiče sběrnice RS-485 na testovací DPS.

Testovací DPS byla úspěšně navržena, zadána do výroby a po obdržení osazena. Při návrhu bylo záměrně omezeno překrytí prokovených otvorů nepájivou maskou pro usnadnění ladění a snazší opravu případných chyb. Chyby na desce skutečně byly, avšak pouze na místech kde byla možnost jednoduché opravy. Po opravě chyb byla otestována komunikace se servomotory a následně bylo přikročeno k návrhu finálního modulu. Pro účely otestování komunikace se servomotory byl mikropočítač naprogramovaný testovacím programem, který měl za úkol motory inicializovat a následně s nimi pohnout na předem dané pozice. Dále byl testovací program rozšířen o řešení úlohy inverzní kinematiky řízení robotické ruky takovým způsobem, aby se hýbala v ose Z (osa kolmá k pomyslné rovině tvořené unašečem prvního servomotoru).



Obrázek 2.11: Pohled na navrženou DPS v prostředí návrhového systému *Altium Designer*.

2.5 Návrh elektroniky modulu

Po dokončení návrhu zkušební desky plošných spojů a ověření komunikace se servomotory bylo přistoupeno k návrhu elektroniky v podobě, která bude použita při konstrukci finálního modulu. Návrh schematického zapojení, obou desek plošných spojů a následně i sestavy celého modulu byl vytvořen v návrhovém prostředí *Altium Designer*.

Schématický návrh modulu je rozdělen do logických celků dle účelu (např. konektory, DC-DC měniče, mikropočítač a podpůrné obvody, řadiče sběrnic aj.). Původně bylo v plánu modul navrhnout jako jednu desku plošných spojů obsahující veškeré potřebné součástky. Avšak již v průběhu navrhování schématu vznikla myšlenka rozdělit modulu na dvě samostatné DPS - silovou a digitální, resp. logickou část. Rozhodnutí rozdělit modul na dvě samostatné desky bylo finálně učiněno po navržení motivu majoritní části silových funkčních bloků, konkrétně multiplexu napájení a DC-DC měniče pro napájení servomotorů. Dalším důvodem, který ovlivnil rozhodnutí byla snaha o udržení kompaktnosti modulu. Rozdělení mělo také pozitivní vliv na zjednodušení návrhu silové části, kde je možné pohodlně využít obě strany DPS pro výkonové účely. Avšak má i své negativní důsledky, například nutnost řešení propojení obou desek, které, jak se později ukázalo, nebylo úplně přímočaré.

Po prvotním rozložení komponent, především konektorů po stranách obou DPS, a přibližném sesazení desek součástkami k sobě byla zjištěna minimální vzdálenost mezi deskami ≈5,6 mm. S touto informací bylo přikročeno k řešení mechanického spojení a elektrického propojení. První omezujícím prvkem byly běžně dostupné délky distančních sloupků velikosti *M3*, druhým, podstatně více omezujícím prvkem byly cenově dostupné konektory a jejich protikusy. První verze návrhu počítala s propojením DPS pomocí 18pinového konektoru sloužícím pro přenos napájení a logických signálů mezi oběma DPS. Nalézt konektor splňující danou výšku, počet pinů a možnost povrchové montáže nebyl velký problém, jako vhodný zástupce byl vybrán povrchově osaditelný konektor s roztečí 2 mm řady *TW-SM* protikus řady *SMM* od firmy *Samtec*. Avšak následně, při zjišťování dostupnosti a nákladů na propojení pomocí zmíněných 18pinových konektorů byla zjištěna neočekávaně vysoká cena (při odběru nízkých desítek kusů se cena řešení blížila 100 Kč) a nízká dostupnost řešení.

Jako alternativa bylo navrženo omezení počtu propojení na takové množství, aby bylo možné použít propojovací konektor, který je běžně skladem u větších distributorů elektronických součástek a díky tomu je řádově levnější. Jelikož je velká část signálů na propojovacím konektoru digitální, lze redukce počtu dosáhnout například použitím integrovaného obvodu rozšiřujícím počet vstupů a výstupů pomocí sériové sběrnice. Na trhu jsou dostupné i modely s vnitřní pamětí a tak lze případně zajistit jednoznačnou identifikaci desky či persistentní uložení případných dat. Avšak, stejně jako v předchozím případě, tak i toto řešení není cenově optimální.

Do portfolia společnosti Y Soft patří i čtečky identifikačních karet, které se skládají z několika DPS kde se také muselo řešit propojení jednotlivých desek a díky množství vyrobených kusů jsou pořizovací náklady na použité konektory minimální. Proto bylo rozhodnuto, že se použije úplně stejný typ 14pinového konektoru i protikusu. Pro redukci počtu propojených vodičů bylo zvoleno řešení s mikropočítačem, který bude řídit celou výkonovou část modulu a komunikovat s hlavním mikrokontrolérem. Oproti hlavnímu mikropočítači jsou zde minimální nároky na výkon a je možné zvolit cenově dostupnou řadu mikropočítačů *STM32F0*. Konkrétní typ vybraného mikrokontroléru je *STM32F031G6U*, který v kompaktním 28pinovém pouzdru *QFN*[12] disponuje jádrem Cortex-M0 na frekvenci

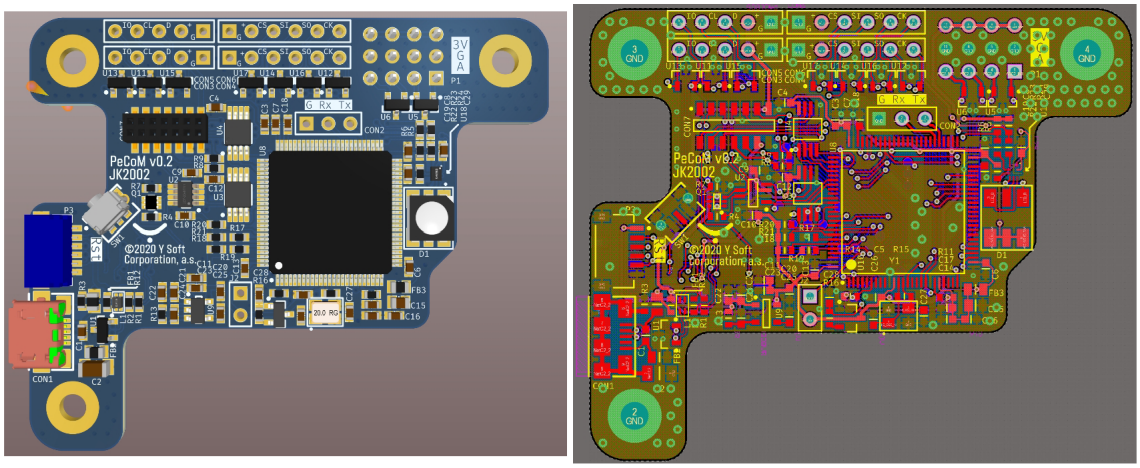
[12] QFN – *Quad Flat No-leads* – čtvercové pouzdro bez vývodů

48 MHz, což je zcela dostačující. Mikropočítač, stejně jako ostatní z rodiny *STM32*, obsahuje unikátní identifikátor díky němuž bude možné zjistit, jaká výkonová deska je kde použitá a případným dlouhodobým sbíráním metrik zajistit optimální provozní podmínky jak pro desku samotnou, tak pro ostatní moduly do ní připojené.

2.5.1 Návrh finálních desek plošných spojů

Samotné elektrické schéma není pro realizaci DPS dostačující. Na základě schématu je nutné navrhnout, jak budou součástky na desce plošných spojů rozmístěné a následně je propojit při splnění všech návrhových pravidel. Pro správnou funkci je také důležité při návrhu zohlednit např. tok proudu vodiči a tím jejich potřebnou šířku, resp. plochu, omezit dlouhé proudové smyčky, zajistit kvalitní napájení komponent aj.

Obě DPS byly navrženy oboustranně s rozměrem opsaného obdélníku 60 mm × 50 mm, tloušťkou desek 1,5 mm a tloušťkou měděné fólie 18 µm. Desky byly navrženy v 6. konstrukční třídě, to znamená minimální šířku vodičů a izolačních mezer 150 µm, nejmenší průměr vrtané díry 0,2 mm a okruží kolem prokoveného otvoru 125 µm. Použitá konstrukční třída se může zdát příliš vysoká (čím vyšší, tím přesnější kresbu je možno vyrobit), avšak jde o důsledek použitých součástek které nastavené limity plně využívají. Zpravidla limitující jsou malé výkonové součástky pro jejichž chlazení se používá tzv. *thermal via*[13], kterých je pro efektivní funkci lepší použít více malých než málo velkých. Řídicí deska také obsahuje obvod *LP5562* v pouzdru *BGA*[14], kde průměr kuličky je ≈0,26 mm a na ploše o velikosti 1,65 mm × 1,25 mm jich je 12. Obě desky jsou, pro zjednodušení výroby osazené výhradně z jedné strany, na druhé straně se nachází pouze konektor používaný při vývoji pro účely ladění, který není pro funkci ani programování potřebný.

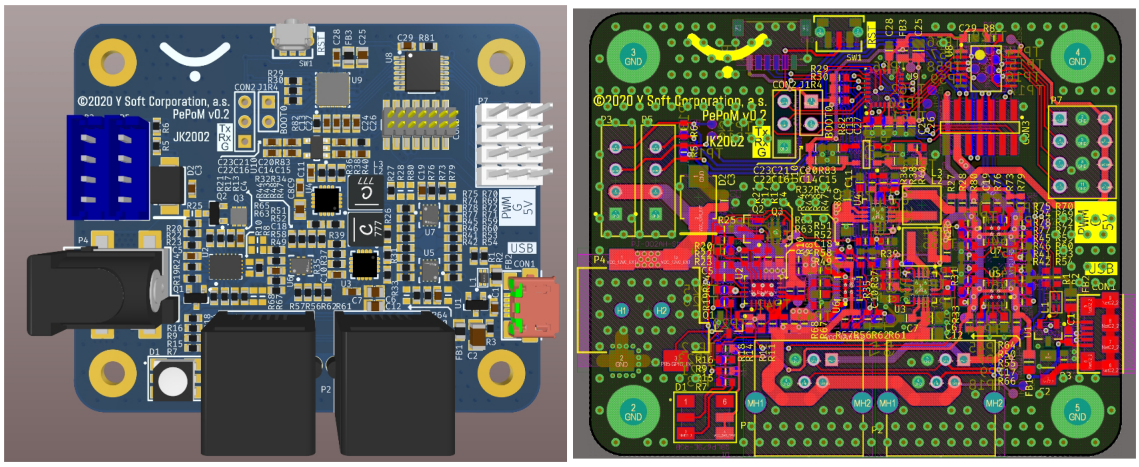


Obrázek 2.12: Návrh DPS pro výpočetní část modulu ve 3d (vlevo), a s viditelnými spoji v prostředí návrhového software (vpravo).

Návrh desek obsahuje testovací body, což jsou malé kruhové plošky nekryté nepájivou maskou umožňující snadné připojení měřicích přístrojů. V prvních verzích návrhu DPS jsou testovací body také často využívané pro realizaci úprav zapojení například v důsledku chyby návrhu. A ani desky navržené v rámci této diplomové práce nebyly ve své první revizi zcela bezchybné. Naštěstí díky testovacím bodům a využití rezistorů s nulovým odporem

[13] thermal via – prokovené otvory sloužící k přenosu tepla od jeho zdroje
[14] BGA – *Ball Grid Array* – pouzdro na které jsou kontakty realizovány polem cínových kuliček

(propojek) bylo možné, i přes chyby v návrhu, desky oživit a úspěšně naprogramovat. V návrhu je také patrné větší množství prokovených otvorů spojující polygony s nulovým potenciálem na horní a spodní straně DPS, ty slouží pro zlepšení vlastností země, omezení parazitních kapacit samotné DPS a tím přispívá ke zlepšení vlastností napájecích obvodů.



Obrázek 2.13: Návrh DPS pro výkonovou část modulu ve 3d (vlevo), a s viditelnými spoji v prostředí návrhového software (vpravo).

Jak může být z návrhu patrné, DPS byly navrhovány v co nejmenším provedení. Z pohledu návrhu bylo limitující rozmístění konektorů, které musí být logicky uspořádané a dostupné po stranách. Většina konektorů – napájecí, připojení inteligentních i modelářských servomotorů a rozšiřující konektory se sběrnicí CAN značně omezují dostupnou plochu využitelnou pro návrh logické DPS, jejíž tvar a velikost jsou přizpůsobeny vystupujícím konektorům.

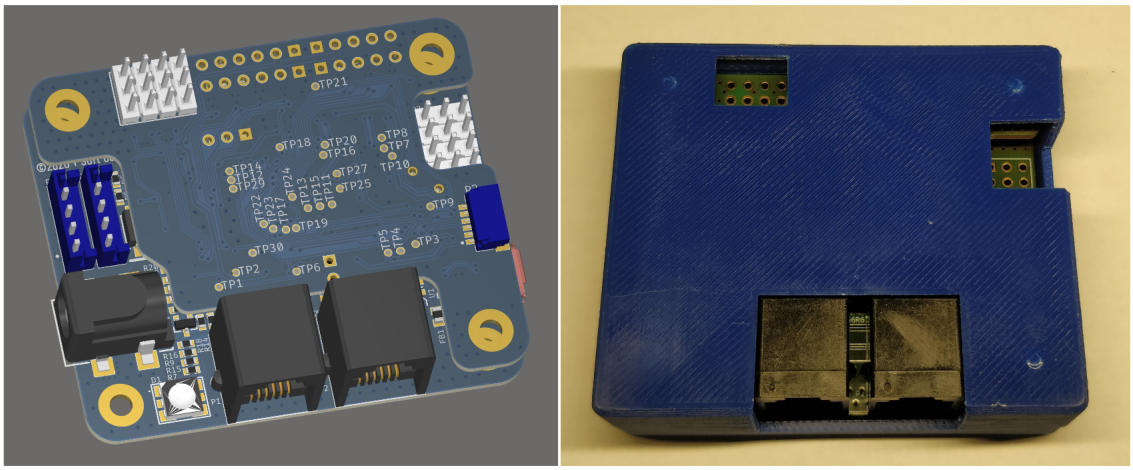
2.5.2 Zapouzdření modulu

Samotné desky plošných spojů sice zajišťují veškerou funkcionalitu avšak pro praktické použití takové zařízení není vhodné, protože je náchylné na vnější vlivy – zkrat o kovovou součást, elektrostatický výboj do nechráněného spojení atd. Řešením popsaných problémů je prostá krabička. Návrhový systém *Altium Designer* umožňuje složení desek plošných spojů, verifikaci korektního zapojení propojovacích konektorů, detekci kolize objektů na deskách a v neposlední řadě i export sestavy do formátu *step*. Sestava bylo poté importována do programu pro návrh mechanických součástí a byla pro ni vytvořena plastová dvoudílná krabička na míru. Horní část krabičky obsahuje závitové vložky (prosté plastové sloupky) velikosti $M3$. Ty jsou spolu se třemi kovovými distančními sloupky využity pro uchycení výpočetní DPS. Druhá část krabičky obsahuje čtyři zahloubené otvory pro šrouby $M3 \times 12$ se zahloubenou hlavou a vnitřním šestihranem. Pro výrobu obou částí krabičky bylo využito tiskáren *YSoft be3D eDee* s FFF[15] technologií tisku.

Složení krabičky je prosté. Výpočetní deska se vloží to horní části krabičky, kde díky použití pouze tří distančních sloupků je právě jedna možná varianta vložení. Připevní se lehce dotaženými kovovými distančními sloupky. Na výpočetní desku se zapojí výkonová a to takovým způsobem, že díry v desce jsou soustředné se závity ve sloupcích, krabička

[15] FFF – *Fused Filament Fabrication* – technologie 3D tisku jejímž principem je postupné nanášení tenkých vrstev roztaveného plastového materiálu

se přiklopí druhou částí a zajistí čtyřmi šrouby. Šrouby se vkládají do kovového závitu distančního sloupku a tudíž nehrozí stržení, ke kterému by mohlo dojít v plastovém závitu. Podrobnější fotografie uložení modulu v krabičce se nachází v příloze a datovém souboru.

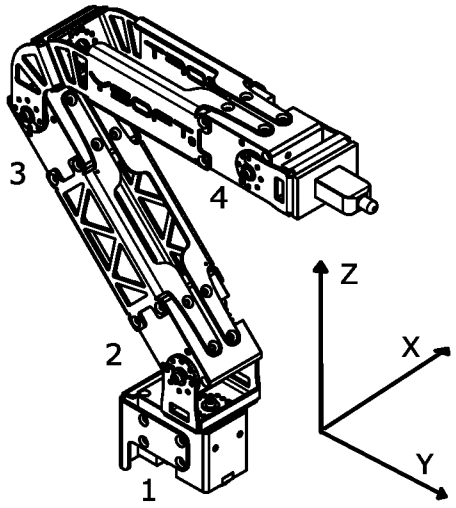


Obrázek 2.14: Návrh kompletního modulu v prostředí programu *Altium Designer* (vlevo) a celý modul v plastové krabičce z 3D tiskárny (vpravo).

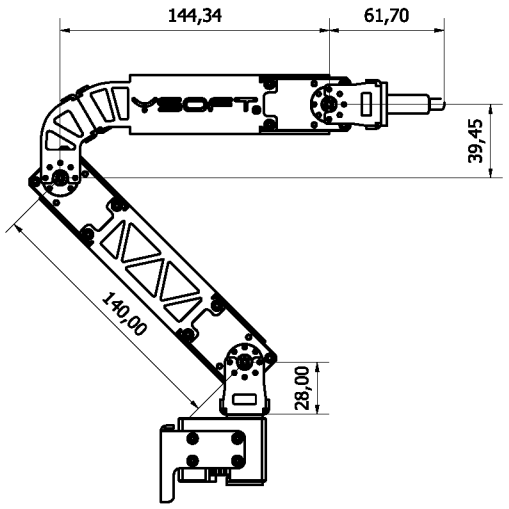
2.6 Praktické řešení kinematických úloh

V kapitole 1.3 má čtenář možnost seznámit se s teoretickými možnostmi řešení přímé a inverzní kinematické úlohy, avšak pro řízení je nutné převést teoretické předpoklady do praktické realizace.

Souřadnice v manipulačním prostoru robota budou popisovány pomocí kartézského souřadného systému orientovaného tak, že rovina $Y - Z$, při nulovém natočení prvního servomotoru, protíná osy všech ostatních servomotorů a osa X míří při pohledu ze předu vpravo (viz obrázek 2.15). Počátek souřadného systému je v průsečíku osy otáčení prvního a druhého servomotoru. Manipulátor se skládá ze tří ramen, mezi kterými se nachází čtyři servomotory umožňující rotační pohyb. Na posledním servomotoru je rameno s koncovým efektorem. Rozměry manipulátoru důležité pro řešení kinematických úloh jsou zaznamenány na obrázku 2.16. Označení jednotlivých servomotorů, ramen a vektorů použitých při výpočtu je na obrázku 2.17.



Obrázek 2.15: Nákres robotického manipulátoru s čísly servomotorů a osami souřadného systému.



Obrázek 2.16: Část výkresu robotické ruky, včetně rozměrů důležitých pro kinematické úlohy.

U manipulátoru se typicky snažíme převést prostorové výpočty na plošné, které lze řešit intuitivně, často pomocí primitivních goniometrických funkcí. Vstupní informací pro inverzní kinematickou úlohu jsou souřadnice cílového bodu e a rotace koncového efektoru vyjádřena pomocí e_a. Řešení inverzní kinematické úlohy pro dosažení daného bodu a rotace spočívá v rozdělení na jednodušší problémy jejich postupném řešení.

1. Vektor p_e je roven cílové pozici koncového efektoru,
2. se znalostí složek x a y určit úhel motoru 1 pomocí funkce atan2 2.6,
3. vypočítat vektor p_w z požadovaného natočení koncového efektoru e_a a vektoru p_e pomocí rovnic 2.7,
4. výpočet délky vektoru p_w, který udává vzdálenost mezi osou druhého a čtvrtého servomotoru, úhel, který svírá vektor s rovinou $X - Y$ je tzv. elevace φ
5. se znalostí $|p_w|$ a využitím kosinové věty určit úhel servomotoru a_3 pomocí rovnic 2.8 a následně i a_2 a a_4. Tím známe všechny potřebné úhly.

Obrázek 2.17: Schematický nákres manipulátoru.

6. Poslední nutný výpočet je převedení úhlů v modelu na reálné úhly natočení servomotorů, provedeno pomocí rovnic 2.10. V praxi se následně úhel přepočítá na celé číslo udávající pozici servomotoru.

$$|p_e| = \sqrt{e_x^2 + e_y^2 + e_z^2} \tag{2.5}$$

$$a_1 = \operatorname{atan2}(e_x, e_y) \tag{2.6}$$

$$\begin{aligned} p_{w_x} &= p_{e_x} + \sin\left(e_\alpha - \frac{\pi}{2}\right) * \sin(a_1) * a_3 \\ p_{w_y} &= p_{e_y} + \sin\left(e_\alpha - \frac{\pi}{2}\right) * \cos(a_1) * a_3 \\ p_{w_z} &= p_{e_z} + \cos\left(e_\alpha - \frac{\pi}{2}\right) * a_3 \\ \varphi &= \arcsin\left(\frac{p_{w_x}}{|p_w|}\right) \end{aligned} \tag{2.7}$$

$$\begin{aligned} \alpha &= \arccos\left(\frac{l_1^2 + |p_w|^2 - l_2^2}{2 * l_1 * |p_w|}\right) \\ \beta &= \arccos\left(\frac{l_2^2 + l_1^2 - |p_w|^2}{2 * l_1 * l_2}\right) \\ \gamma &= \arccos\left(\frac{l_1^2 + |p_w|^2 - l_2^2}{2 * l_2 * |p_w|}\right) \end{aligned} \tag{2.8}$$

$$
\begin{aligned}
a_2 &= \alpha + \varphi \\
a_3 &= \gamma \\
a_4 &= \beta - \varphi - e_a
\end{aligned}
\tag{2.9}
$$

$$
\begin{aligned}
m_1 &= \pi - a_1 \\
m_2 &= 2\pi - a_2 \\
m_3 &= a_3 + \pi/2 - 15{,}29°(\text{mechanický offset}) \\
m_4 &= a_4\pi + 15{,}29°
\end{aligned}
\tag{2.10}
$$

Popsaný způsob výpočtu inverzní kinematické úlohy byl prakticky ověřený a shledán perfektně funkčním pro potřebné využití. A to při výrazně nižší časové i paměťové náročnosti výpočtu v porovnání s generováním analytického výpočtu inverzní kinematiky, např. pomocí *IKFast* jak bylo popsáno v kapitole 1.3.

Při řešení přímé kinematické úlohy je využito podobného přístupu – převod na plošné řešení trojúhelníku s tím rozdílem, že úhly a některé strany jsou známé. Následně, se znalostí úhlu natočení servomotoru č. 1, je proveden převod z polárního souřadného systému (znalost φ a $|P_e|$) na kartézský, čímž je přímá kinematická úloha vyřešena. Výpočetní náročnost přímé kinematické úlohy je opět, díky použití jednoduchých trigonometrických operací, nízká.

2.6.1 Navigace pod po bodu

Vyřešením inverzní kinematické úlohy je možné vypočítat pozice pro servomotory tak, aby bylo dosaženo kýžené pozice a orientace koncového efektoru. Následující kapitola se zabývá problémem přesunu koncového efektoru manipulátoru z výchozí pozice a orientace do cílové a to po předem naplánované trajektorii.

Vzhledem k určení manipulátoru není, alespoň prozatím, nutné plánovat komplexní trajektorie zahrnující detekci kolize, či další pokročilé vlastnosti. Avšak, pro spolehlivou simulaci lidského klikání na dotykový displej je klíčové implementovat plánování trajektorie s konfigurovatelným způsoben dosažení cílového bodu. U předchozího řídicího systému se v praxi ověřilo, že je dostatečné to tyto účely využít kubické Bézierovy křivky a pomocí dvou řídících bodů popsat tvar křivky a tím mj. způsob dosažení koncového bodu. Oproti běžně plánovaným trajektoriím je pro účely klikání na displej, primárně u rezistivní technologie, vhodné cílovou pozici mírně překmitnout a koncovým efektorem na vrstvu detekující dotyk více zatlačit, více informací je v sekci 1.2.

Při přesunu koncového efektoru manipulátoru ve volném prostoru do cílové pozice není využití Bézierovy křivky vhodné. Při využití Bézierovy křivky je zrychlení koncového efektoru zpravidla buď kladné nebo záporné, koncový efektor tedy stráví minimum času pohybu maximální rychlostí čímž dochází k neefektivnímu využití manipulátoru. Řešením je použití trajektorie s lichoběžníkovým rychlostním profilem. Pro trajektorie se definuje zrychlení a maximální rychlost koncového efektoru, při pohybu budou, díky lineární akceleraci, méně namáhány servomotory a zároveň bude výsledný pohyb zpravidla rychlejší než při použití béziérovy křivky se stejnou maximální rychlostí.

Předchozí odstavce řešily pouze problém pohybu koncového efektoru z výchozího bodu do cílového. Manipulátor je však sestaven ze servomotorů, jejichž úhel natočení není vzhledem k pozici koncového efektoru lineární. Při pohybu koncového efektoru, například v ose X, konstantní rychlostí bude pozice servomotoru číslo 1 v čase tvořit graf funkce *arctan(x)*. Tato skutečnost problém plánování trajektorie mírně komplikuje a před započetím pohybu je nutné ověřit, že budou dodrženy dynamické limity servomotorů, v opačném případě by došlo k pohybu po neočekávané trajektorii a možné kolizi se zařízením před manipulátorem.

V praxi dostává řídicí systém požadavky na pohyb manipulátoru ve formě příkazu, které specifikují typ trajektorie - kubická Bézierova křivka, nebo lichoběžník, a parametry dané trajektorie popsané v předchozích odstavcích. Systém po příjmu příkazu ověří dosažitelnost cílové pozice a v případě úspěchu pohyb simuluje. Simulace zahrnuje zjištění celkového času pohybu a následné vzorkování (virtuální) pozice koncového efektoru, řešení inverzní kinematické úlohy a ověření dodržení statických a dynamických vlastností servomotorů. Kde dynamické vlastnosti zahrnují maximální zrychlení, maximální rychlost a statické zahrnují minimální a maximální úhel natočení. Limity jsou specifikovány pro každý servomotor samostatně. V případě, že došlo k překročení statických limitů je příkaz k pohybu odmítnut, nadřízený systém je skutečnosti informován a pohyb není dále zpracováván. V případě, že došlo k překročení dynamických limitů je zjištěna míra překročení a parametry trajektorie jsou následně upraveny takovým způsobem, aby při reálném pohybu k překročení limitů nedošlo. Poté je příkaz předán do fronty k reálnému vykonání a nadřízený systém je informován o akceptaci příkazu.

Pro zkrácení času nutného k ověření akceptovatelnosti příkazu na pohyb manipulátoru je možné vzorkovat pozice servomotorů a řešit inverzní kinematickou úlohy s nižší frekvencí, než při reálném vykonávání pohybu. V praxi, při vzorkovací frekvenci 100 Hz a pohybu trvajícím 5 sec, trvá ověření příkazu přibližně 5 ms. V praxi převážná většina pohybů trvá do 1 s a čas potřebný k ověření je přibližně lineární ve vztahu k trvání pohybu, tj. ≈1 ms, rychlost ověření je tedy dostačující není třeba dalších optimalizací. Pro extrémní případy disponují příkazy pohybu příznakem který umožňuje přeskočit akceptační kontrolu a příkaz ihned zařadit do fronty k vykonání. Také je dostupný příkaz pro přímé nastavení polohy koncového efektoru manipulátoru bez interpolace z výchozí do cílové pozice. Vykonání pohybů je delegováno na samostatné vlákno s vysokou prioritou. Tím, je zajištěn plynulý pohyb robotické ruky a minimalizovaný jitter při přenosu nových pozic servomotorů.

2.7 Řídicí příkazy

Jak již bylo nastíněno v kapitole 1.6.4 a 2.6 vyvíjený řídicí systém s nadřízeným systémem komunikuje pomocí příkazů přenášených protokolem MAVLink. Příkazy je možné rozdělit do několika skupin – pohyb manipulátoru, pohyb samostatného servomotoru, čtení pozice manipulátoru nebo samostatného servomotoru a speciální příkazy, mezi které patří inicializace manipulátoru a uspání manipulátoru.

Po připojení řídicí jednotky ke zdroji napájení a sběrnici USB dojde inicializaci řídicí jednotky, enumeraci USB zařízení a připojení servomotorů ke zdroji napájení. Servomotory jsou pod napětím, avšak H-můstky jsou deaktivované a s unašečem lze bez námahy hýbat. Aktivace servomotorů musí být provedena kontrolovaně a to takovým způsobem, aby nedošlo k poškození zařízení před manipulátorem. Servomotory umožňují aktivaci H-můstku a zachování aktuální pozice, v praxi je vhodné koncový efektor manipulátoru dostat do známé počáteční pozice. Díky přímé kinematice není problém zjistit pozici koncového efektoru po aktivaci H-můstků, v praxi se však ukázalo že pro osobu zapínající manipulátor

je praktičtější, když se koncový efektor manipulátoru přesune do známé polohy. Mimojiné se tím ověří správná orientace manipulátoru před testovaným zařízením a korektní funkce všech servomotorů. Příkaz pro inicializaci manipulátoru zaslaný řídicí jednotce způsobí výrazné snížení maximální rychlosti servomotorů následnou aktivaci H-můstků a přesun koncového efektoru na definovanou výchozí pozici, aktuálně $[0; 130; 120]$ v osách x, y, z se sklonem koncového efektoru 45° vůči rovině X-Y.

Druhý speciální příkaz zajistí tzv. uspání manipulátoru, což v praxi znamená deaktivaci H-můstků servomotorů. Je jistě patrné, že není ideální H-můstky jen deaktivovat protože by došlo k nepředvídatelnému pohybu manipulátoru a případnému poškození testovaného zařízení. Před deaktivací je nutné provést sekvenci pohybů vedoucích k přesunu manipulátoru do pozice, ve které je deaktivace H-můstků servomotorů bezpečná a nedojde k ohrožení testovaného zařízení ani případného operátora. Jako vhodná sekvence pohybů pro tyto účely se jeví přesun manipulátoru do výchozí pozice, která je určitě bezpečná. Následně se manipulátor přesune do takové pozice, že koncový efektor míří vzhůru a ramena mezi motory 2, 3 a 3, 4 jsou vodorovné. Poté je možné H-můstky deaktivovat bez rizika poškození manipulátoru či testovaného zařízení. Uspávání manipulátoru může na první pohled působit, oproti průmyslovým manipulátorů, které něco podobného nenabízejí, nadbytečně avšak je nutné si uvědomit rozdíl mezi servomotory na průmyslových manipulátorech a inteligentními servomotory použitými v rámci této práce. Průmyslové servomotory na manipulátorech zpravidla buď obsahují brzdu, která je normálně (bez přísunu proudu) aktivní nebo mají samosvornou převodovku. To umožňuje průmyslovým manipulátorům držet pozici bez zbytečného maření elektrické energie a zahřívání. Oproti tomu servomotory použité v manipulátoru nemají ani brzdu, ani samosvornou převodovku a pro držení pozice je nutné použití elektrického proudu.

Třída příkazů umožňující pohyb manipulátoru byla již naznačena v kapitole 2.6. Řídicí jednotka umožňuje použití 3 typů příkazů pro pohyb manipulátoru. První, umožňující pohyb koncového efektoru s lichoběžníkovým profilem rychlosti, je určený pro přesun manipulátoru do cílové pozice rychle a efektivně. Druhý, využívající Bézierovy křivky, je určený primárně pro klikání na obrazovku testovaného zařízení a umožňuje nadřízenému systému nastavení *profilu kliknutí*. Třetí, umožňuje zadání pozice koncového efektoru, na kterou se efektor bezodkladně přesune, nevyužívajíc žádné interpolace. Příkazy pro pohyb manipulátoru mají, oproti ostatním příkazům, jednu společnou vlastnost: jejich provedení závisí na pozici koncového efektoru před započetím pohybu. Proto je pro nadřízený systém klíčová znalost stavu jednotlivých příkazů pohybu. Stavem je v tomto kontextu myšleno, zda byl příkaz přijatý a bude proveden jakmile na něj přijde řada, případně důvod nepřijetí. Aby bylo možné stav přijetí příkazu předat nadřízenému systému je každý pohyb označen 16 bit identifikátorem, který musí být od přijetí do dokončení pohybu unikátní. Příkazy taktéž označují příznak, zda má řídicí systém zaručit dodržení limitů manipulátoru. Příkazy pro pohyb manipulátoru jsou řazeny do fronty a jsou provedeny jakmile na ně přijde řada.

Při testování vestavěných zařízení systémem obsahující manipulátor, pro který je vyvíjena řídicí elektronika, občas vyvstane potřeba ovládat nějaký mechanický prvek pomocí samostatného servomotoru. Typicky se může jednat o mechanický spínač či přepínač. Pro takové případy jsou dostupné příkazy umožňující ovládání a zjištění pozice samostatného servomotoru. Na rozdíl od dříve zmíněných příkazů pro pohyb koncového efektoru, u pohybu samostatného serva není nutné složitě ověřovat dosažitelnost pozice ani upravovat parametry pohybu, protože pro samostatné servo nejsou definovány limity. Jakmile je příkaz přijat na sběrnici RS-485 se vyšle pokyn pro daný servomotor k přesunu do dané pozice. V pří-

padě žádosti o pozici je pozice přečtena ze servomotoru, následně přepočítána na radiány a odeslána nadřízenému systému.

Poslední, doplňkové příkazy slouží ke zjištění aktuálního stavu řídicí jednotky a zjištění počtu pohybových příkazů určených k vykonání ve frontě.

2.8 Nadřízený řídicí systém

Veškerá dosavadní práce byla odvedena na vestavěném systému, který je nyní schopný dle zadaných pokynů pohybovat robotickým manipulátorem. Avšak aby bylo možné manipulátor ovládat ze systému *RQA* je nutné implementovat aplikaci běžící na počítači, ke kterému bude řídicí jednotka připojena. Aplikace bude s manipulátorem komunikovat prostřednictvím sériové linky a bude umožňovat řízení manipulátoru grafickým rozhraním a vzdáleně pomocí webového *API*. Grafické rozhraní bude primárně určeno pro otestování funkcionality vestavěného systému během vývoje, sekundárně pro předvedení funkcionality v rámci této diplomové práce. Webové rozhraní může být použito při integraci systému do celku *RQA*.

V relativně blízké budoucnosti je plánovaný přechod z komunikace po virtuální sériové lince a protokolu MAVLink na standard CANOpen. Takový přechod by z velké části vyloučil z použití implementovaný nadřízený systém. Na první pohled se může zdát že přímá implementace CANOpen by byla snadnější, avšak opak je pravdou. CANOpen je implementačně náročnější a také se musí podřídit ostatním komponentám v systému *RQA*, které v době vývoje ještě nebyly finalizované.

Jednou z nejdůležitějších součástí aplikace, která slouží jako základ pro obě dostupná rozhraní je komunikace s vestavěným systémem prostřednictvím virtuálního sériového portu. Práce s virtuálním sériovým portem se, z pohledu implementace, neliší od práce s reálným, použije se metod dostupných ve jmenném prostoru `System.IO.Ports`. jak již bylo zmíněno v předchozích kapitolách této práce, byla implementována komunikace vestavěného systému protokolem MAVLink. Pro dekódování protokolu je možné z popisu tříd vygenerovat třídy zajišťující dekódování a kódování zpráv. Současně dojde i k vygenerování tzv. *POCO*[16] tříd pro uchování silně typované zprávy. Tyto silně definované zprávy je možné pomocí enkodéru převést na pole bajtů a to odeslat vestavěnému systému. V případě opačného směru toku dat jsou přijaté bajty předány metodě dekodéru, která, podobně jako v případě vestavěného systému, informuje volajícího o konci rámce, tentokrát vyvoláním události. Zachycením události je příjem zprávy z vestavěného systému dokončen. Nad tímto jednoduchým rozhraním byla implementována vrstva abstrakce skrývající implementační detaily a umožňující programátorovi jednoduché ovládání robotického manipulátoru pomocí asynchronních metod (řešeno pomocí nativní abstrakce ze jmenného prostoru `System.Threading.Tasks`).

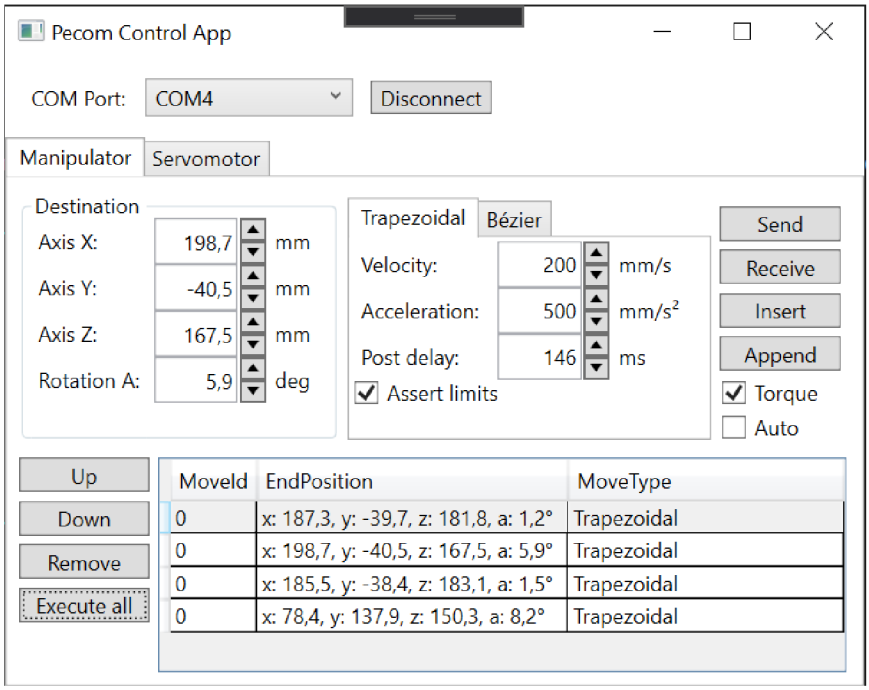
2.8.1 Grafické rozhraní

Aplikace je naprogramována na platformě *.NET Core* v jazyce C#. Na platformě *.NET Core*, která oproti dříve populární platformě *.NET Framework* nabízí otevřený zdrojový kód dostupný na serveru github, možnost běhu na všech běžně dostupných operačních systémech (Windows, Linux, MacOS). Mj. díky nativnímu běhu na Linuxu je možné aplikaci v budoucnu snadno kontejnerizovat, např. pomocí nástroje *docker*. Grafické rozhraní je implementováno v knihovně Windows Presentation Foundation (*WPF*), která v aktuální

[16] POCO – *Plain Old Clr Object* – jednoduché třídy obsahující proměnné jednoduchého typu

verzi (.NET Core 3.1.4) umožňuje běh na platformě *.NET Core*, avšak vzhledem k silné vazbě na rozhraní pro vykreslování *DirectX*, které je silně svázáno s operačním systémem Microsoft Windows, není jednoduše možné, aby bylo grafické rozhraní multiplatformní. To však v případě této aplikace není problém, protože systém *RQA* bude využívat výhradně webové rozhraní. Pozitivním efektem logického rozdělení aplikace na několik knihoven je možné spustit webové rozhraní bez grafického a umožnit tedy běh i na operačním systému Linux.

Grafické rozhraní je, jak již bylo zmíněno, implementováno pomocí *WPF* a dovoluje odeslat řídicímu systému téměř jakoukoliv zprávu. U zpráv pro pohyb umožňuje definovat veškeré potřebné parametry pohybu. Pro pohyb po lichoběžníkové křivce maximální rychlost a zrychlení, pro Bézierovu křivku řídicí body a maximální rychlost. Dále je možné jednotlivé pohyby přidávat do fronty, která může být posléze celá vykonána. Grafické rozhraní je možné použít i k ovládání samostatného servomotoru zápisem cílové pozice a rychlosti pohybu. Aplikace pomocí tlačítka *Torque* umožňuje deaktivovat H-můstky servomotorů manipulátoru, jeho manuální přemístění do jiné polohy a opětovnou aktivaci H-můstků. Pomocí tlačítka *Auto* je možné ihned odesílat úpravy pozice z aplikace do řídicí elektroniky bez explicitního příkazu. Další dostupnou funkcí je možnost vyčtení pozice manipulátoru. K dispozici jsou dva režimy, první vrací pozici manipulátoru po dokončení všech naplánovaných pohybů. A druhý využívá možností servomotorů vyčíst aktuální pozici a následně s pomocí přímé kinematické úlohy přenést skutečnou pozici na které se manipulátor v okamžiku požadavku nacházel. Podoba grafického rozhraní je na obrázku 2.18, v horní části programu je pole pro výběr sériové linky a tlačítko na připojení/odpojení.

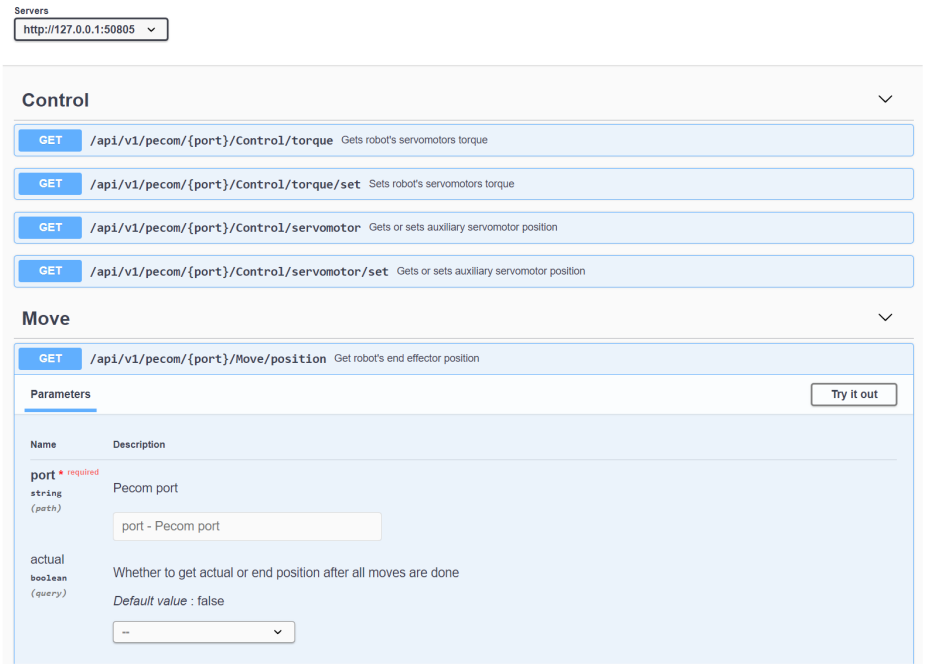


Obrázek 2.18: Grafické rozhraní pro ovládání manipulátoru.

2.8.2 Webové rozhraní

Druhé rozhraní umožňujíc ovládání robotického manipulátoru je webové. Prostřednictvím standardních HTTP požadavků umožňuje ovládání manipulátoru z téměř jakéhokoliv prostředí, ať už se jedná o libovolný programovací nebo skriptovací jazyk. Tím je umožněno použití manipulátor i mimo systém *RQA* pro úlohy se kterými se v době implementace systému nepočítalo, případně je pro ně použití *RQA* nevhodné. Z pohledu dostupné funkcionality jsou v současné době rozhraní srovnatelná, avšak dlouhodobě bude převažovat využití webového a proto i další vývoj bude směrovat výhradně touto cestou. Pro implementaci webového rozhraní je využito knihoven ASP.NET Core.

Dokumentace a popis webového rozhraní byla vytvořena s pomocí nástroje *NSwag*, který popisuje rozhraní standardem *OpenAPI* [19]. Projekt lze pomocí balíčku `NSwag.AspNetCore` [28] integrovat do webového rozhraní a tím na předem dané adrese zveřejnit popis rozhraní ve formátu *JSON*[17]. Pomocí dalšího balíčku je možné dokument popisující rozhraní zobrazit v grafické, lidsky čitelné podobě. Taková podoba umožňuje komukoliv zavolání HTTP metod přímo z webového prohlížeče. Aby byl popis rozhraní kompletní je vhodné nad metody zajišťující odbavení daných *HTTP* požadavků přidat atributy. Pomocí atributů je možné popsat návratové kódy, možné návratové typy a další relevantní parametry pro webové rozhraní.



Obrázek 2.19: Webové rozhraní knihovny *NSwag* umožňující grafické zobrazení *OpenAPI* popisu webového rozhraní aplikace.

[17] JSON – *JavaScript Object Notation* – způsob reprezentace dat pomocí JavaScriptového objektového zápisu

Kapitola 3

Testování

Testování slouží k odhalení chyb před vydáním produktu a je integrální součástí každého běžného softwarového produktu. Při testování dělíme testy do skupin dle podrobnosti. Nejnižší úrovní jsou jednotkové testy, které v případě objektově orientovaného návrhu kontrolují korektní funkci jednotek, tedy tříd a metod. Testy se zapisují často přímo pomocí zdrojového kódu a zpravidla jsou zodpovědností programátora. Na vyšší úrovni jsou integrační testy kontrolující vzájemné propojení jednotek. Kde jednotkami nejsou jen třídy a metody, ale i integrace ostatními rozhraními – operačním systémem, hardwarem a případně dalšími komponentami. Nejvyšší úrovní jsou systémové testy, které testují systém jako funkční celek. Jednotkových testů je z principu nejvíce a se zvyšující se úrovní počet testů klesá.

3.1 Testování firmware

Skutečnosti uvedené v předchozím odstavci lze poměrně snadno aplikovat při vývoji běžné aplikace, bohužel u vestavěného systému je situace týkající se testování řádově složitější. Jednotkové testy lze často i v případě vývoje na vestavěný systém spouštět na systému běžném, vypovídající hodnota takových testů nebude 100 %. Odlišnosti mezi architekturami jsou příliš velké a implementace standardní knihovny je taktéž značně rozdílná. Jednou z cest, jak jednotkovými testy funkčnost opravdu ověřit je vytvoření speciální verze firmware určené výhradně k tomuto účelu. I při korektním vytvoření vlastní verze firmware můžou nastat komplikace způsobené interakci s hardwarem, přerušeními nebo operačním systémem reálného času. Další možností jak testovat vestavěné systému pomocí jednotkových testů je použití virtualizace resp. emulace cílové architektury na běžném počítači. Na rozdíl od předchozího řešení testy nepotřebují ke svému běhu hardware a můžou tedy běžet jako součást procesu sestavení na sestavovacím serveru. I toto řešení má, v případě této práce, výrazné limitace, jednou z nich je náročná integrace do již zavedeného systému v rámci firmy Y Soft. Vzhledem k náročnosti vytvoření systému pro běh jednotkových testů a možnosti dosažení pouze limitovaného pokrytí nebyly jednotkové testy v rámci vestavěného systému v této práci vyvinuty.

Testy vyšších úrovní (integrační a systémové) již zpravidla potřebují přístup k hardware, simulaci přerušení a interakci s operačním systémem reálného času. Pro otestování interakce s hw by bylo nutné vyvinout testovací přípravek, do kterého by se vestavěný systém založil a umožnil nadřízenému systému získat znalost o stavu vestavěného systému. V praxi se takový přístup běžně používá. Vestavěný systém se založí do přípravku, naprogramuje, podstoupí sérii testů, které, díky přípravku znají stav vestavěného systému, a po vyhod-

nocení pokračuje dále. Vývoj a výroba takového přípravku společně s vývojem software pro otestování je netriviální záležitostí a přesahuje rozsah této diplomové práce. Použitý operační systém reálného času (FreeRTOS) nemá vestavěnou podporu pro testování a je tedy opět velice náročné provést testy zahrnující práci s operačním systémem.

Dílčí části firmware vestavěného zařízení byly důkladně testovány při vývoji, kdy je snadnější navození potřebného stavu, např. pomocí změny obsahu paměti za běhu zařízení. Podrobně ověřenou částí jsou úlohy přímé a inverzní kinematiky, kde je ověření správnosti poměrně jednoduché. Pomocí posuvného měřítka bylo ověřeno dosažení kýženého bodu v prostoru. Kombinací přímé a inverzní kinematiky, tedy navigací manipulátoru do známého bodu pomocí inverzní kinematiky, vyčtení pozic servomotorů, získání dosaženého bodu přímou kinematikou a porovnáním se známou pozicí lze ověřit správnost obou kinematických úloh. Dále byly provedeny testy stability spočívající v trvalém pohybu manipulátoru po delší časovou dobu, v tomto případě přibližně 48 hodin. Pro dlouhodobé testy byl program nakonfigurován tak, aby po vyprázdnění fronty příkazů přidal do fronty další příkazy a následně periodicky zjišťoval aktuální stav naplnění fronty.

U systému, který zajišťuje řízení pohybu robotického manipulátoru může být důležitým aspektem absolutní přesnost a opakovatelnost pohybu. Avšak tyto parametry jsou z majoritní části ovlivněny použitými servomotory a výrobní tolerancí výpalků. Minoritní vliv může mít výpočet kinematických úloh, avšak vzhledem k použití analytické metody výpočtu a datového typu s dvojitou přesností je chyba zanedbatelná. Navrhovaný systém tedy přímo neovlivňuje absolutní přesnost ani opakovatelnost pohybů manipulátoru a proto na tyto parametry nebude brán zřetel.

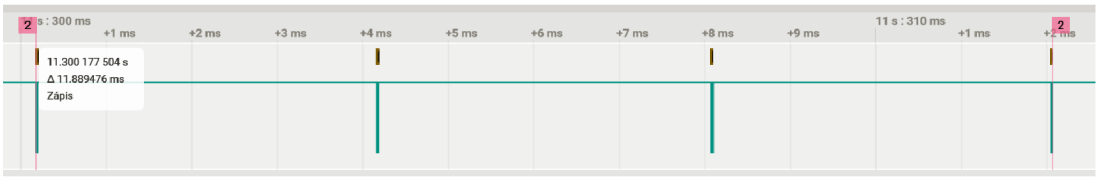
3.2 Porovnání s předcházejícím řešením

Jedním z klíčových parametrů úspěšnosti řešení této práce je zhodnocení výsledné plynulosti pohybu manipulátoru. Plynulost manipulátoru je přímo závislá na důsledném dodržování požadovaných časových intervalů při komunikaci se servomotory. Následující sekce se zabývá hodnocením a porovnání vybraných metrik komunikace se servomotory.

3.2.1 Doba trvání zápisu nové pozice

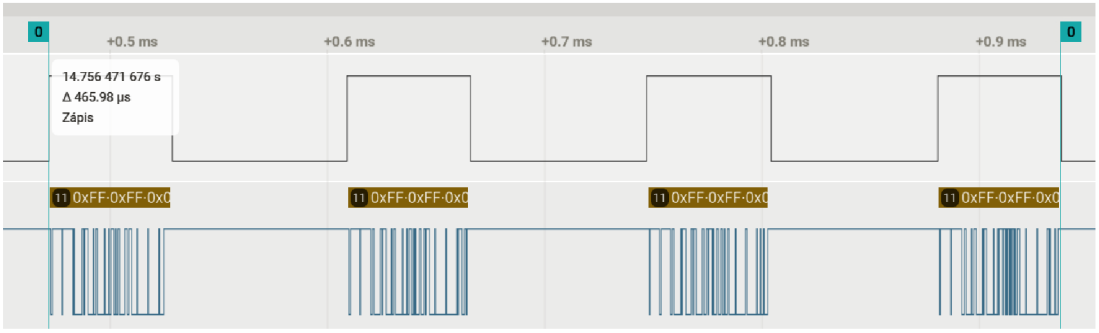
Při komunikaci se servomotory je nutné po každém odeslaném příkazu vyčkat na jeho potvrzení a další příkaz zaslat až po vypršení časového intervalu nebo obdržení potvrzení. Pro odeslání nové pozice pro celý manipulátor je nutný zápis pozic do 4 servomotorů. Pro měření byla použita situace, kdy se manipulátor plynule pohybuje mezi několika body. Pomocí logického analyzátoru *Saleae Logic Pro 8* byl zaznamenán signál na datovém vodiči po dobu přibližně 10 s. Software logického analyzátoru byl poté použitý pro analýzu komunikace a vytvoření csv souborů obsahující časové značky a přenesená data. Pro následnou analýzu získaných dat a výpočet statistických parametrů byl využitý Microsoft Excel.

První zvolenou metrikou je doba potřebná pro zápis nových pozic do 4 servomotorů. Tato metrika není pro plynulý pohyb manipulátoru klíčová, avšak slouží pro snadnější odhad minimální periody zápisu nové pozice manipulátoru. Na obrázku 3.1 je reprezentativní příklad zápisu nové pozice starým řešením, které pracuje s periodou zápisu 20 ms. Doba od první sestupné hrany na datovém vodiči po poslední vzestupnou je ≈12 ms. Při letmém pohledu na nasnímaný průběh je patrná značná, avšak spíše záporná, odchylka od reprezentativního příkladu.



Obrázek 3.1: Průběh zápisu pozic do servomotorů na původním systému.

V případě vyvinutého řešení je doba zápisu pozice do 4 servomotorů značně kratší, ≈470 µs, což je oproti původnímu řešení přibližně 25× zlepšení. Na obrázku 3.2 patrný průběh reprezentativního zápisu, horní průběh se z vodiče určující směr toku dat, spodní jsou vysílaná data. Nový systém pracuje s periodou zápisu nové pozice 10 ms a pro řešení případných problémů s doručením dat tedy zbývá ještě 9,5 ms, což je, z pohledu vestavěného systému dostatek času.



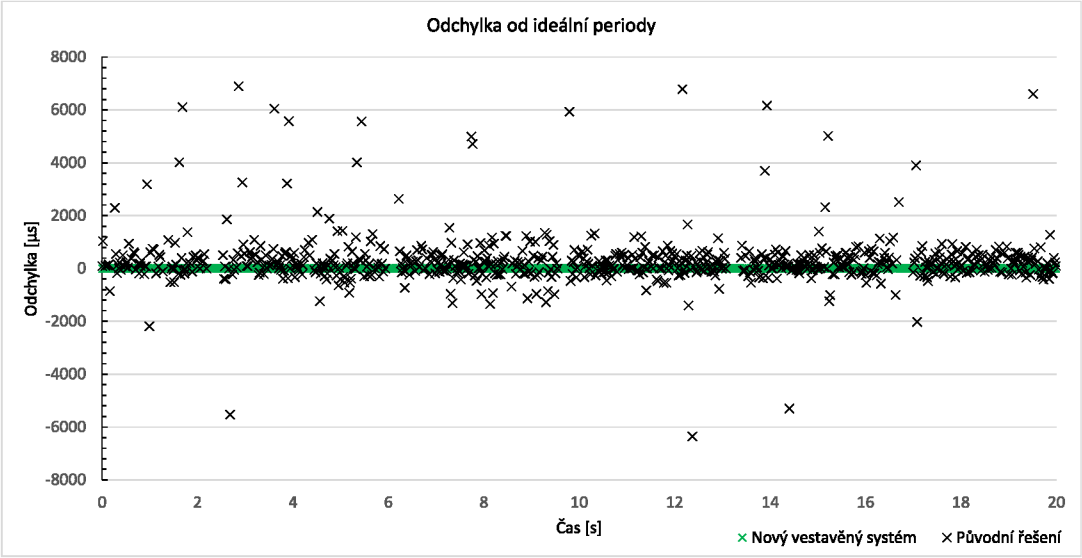
Obrázek 3.2: Průběh zápisu pozic do servomotorů na systému vyvinutém v rámci této práce.

3.2.2 Odchylka od ideálního času odeslání pozice

Další zvolenou metrikou je odchylka od ideálního času odeslání prvního rámce. Ideálním časem se rozumí čas odeslání prvního rámce předchozí komunikace zvýšen o periodu zápisu pozice. Tato metrika je, na rozdíl od předchozí, pro plynulost manipulátoru zcela zásadní a čtenáři dává dobrý přehled o plynulosti, kterou je jinak složité změřit či popsat. V ideálním případě bude výsledkem odchylka blízká nule, avšak pro plynulý pohyb je dostatečné, aby byla odchylka stabilní, bez jakýchkoliv extrémů.

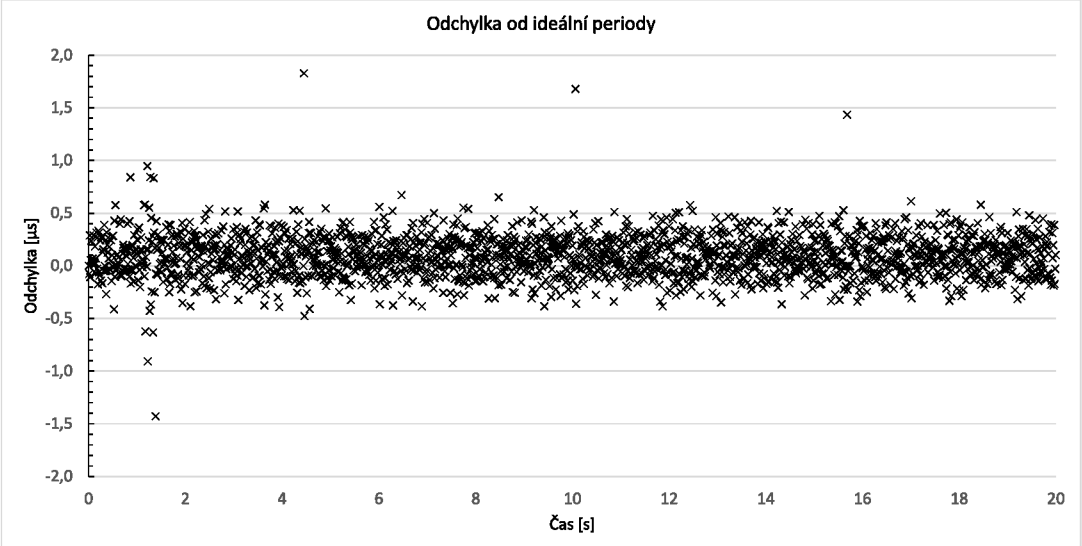
Pro měření času byl opět použitý logický analyzátor *Saleae Logic Pro 8*. U tohoto způsobu měření existují 2 možné zdroje chyby. Prvním je nepřesnost krystalu použitého v rámci logického analyzátoru a při měření krátkých časových úseků, v tomto případě kolem 20 ms a 10 ms, je zanedbatelný. Druhý je dán diskrétním vzorkováním signálu. Signál byl vzorkován s frekvencí 250 MHz tj. periodou 4 ns, maximální možná chyba z tohoto zdroje je 8 ns což je opět v rámci této práce zanedbatelné.

Při měření původního systému byla zaznamenána komunikace během 20 s pohybu manipulátoru. Následně byly odfiltrovány jen první rámce při zápisu pozice, celkový počet vzorků byl 1300. Ideální perioda zápisu dat je 20 ms, reálný průběh odchylky je znázorněn černými křížky na obrázku 3.3. Z obrázku jsou patrné výrazné odchylky od požadované periody především při dokončení jednoho pohybu a počátku následujícího. Z dat byly také odfiltrovány extrémní výchylky nad 7000 µs avšak i tak je z průběhu patrné poměrně velké množství kladných i záporných špiček, které způsobují trhy při pohybu.



Obrázek 3.3: Průběh odchylky od ideálního času odeslání prvního rámce v čase na původním (černý) a novém (zelený) systému v jednom grafu.

Stejné měření bylo zopakováno na systému vyvinutém v rámci této práce. Tedy opět zaznamenána komunikace za přibližně 20 s pohybu manipulátoru. V tomto případě je ideální perioda zápisu nové pozice poloviční, tedy 10 ms. Zelené křížky na předchozím obrázku znázorňují průběh pro nový systém. Vzhledem k dramatickému snížení odchylky není na průběhu nic patrné. Proto je u průběhu na obrázku 3.4 značně snížen rozsah osy Y a znázorněn průběh odchylky pouze pro nový systém. Za povšimnutí určitě stojí rozsah osy Y, který je pouze 4 µs. Oproti původnímu řešení došlo k velmi výraznému snížení rozptylu a tím ke zvýšení plynulosti a obecné kvality pohybu manipulátoru.



Obrázek 3.4: Průběh odchylky od ideálního času odeslání prvního rámce v čase na systému vyvinutém v rámci této práce. Oproti předchozímu průběhu je rozsah osy Y o více než tři řády menší.

3.2.3 Shrnutí dosažených měření

Porovnání klíčových statistických parametrů popsané metriky je v tabulce 3.1. Z hodnot je patrné, že v rámci metriky došlo ke zlepšení o 3 až 4 řády. V případě nového systému jsou dosažené hodnoty výrazně lepší a plynulejší pohled je znatelný na první pohled.

Dříve používaný způsob řízení robotického manipulátoru a komunikace se servomotory využíval virtuální sériovou linku a převodník na sběrnici RS-485. Veškeré výpočty i komunikace se servomotory byly realizovány programem v jazyce C#, který běžel na vestavěném zařízení SafeQube. *SafeQube* patří mezi relativně výkonná (2× ARM Cortex A7 na 800 MHz, 2 GB DDR3 RAM) vestavěná zařízení využívající standardní kernel Linux a operační systém upravený na míru tomuto zařízení. Jako běhové prostřední pro aplikaci byl zvolen *.NET Core 2.1* přibalený k aplikaci. Při komunikaci se servomotory proudí všechna data přes USB, jejíž maximální dotazovací frekvence je 1000 Hz, to znamená, že zpoždění příjmu datového rámce ze servomotorů může být jen na převodníku až 1 ms. Při potřebě odeslat, přijmout a zpracovat komunikaci pro 4 servomotory dojde k sečtení dílčích prodlev a výsledkem je teoretická nejkratší doba trvání zápisu (první metrika, 3.2) 4 ms, v praxi je tato doba dvou až tří násobná. u druhé metriku, odchylky od ideální periody zápisu, budou relativně vysoké hodnoty způsobeny kumulativně. Na nejnižší vrstvě zpožděním USB při čekání na volnou sběrnici – zařízení není na sběrnici samo a o datový tok se dělí. Svůj díl bude zřejmě mít i operační systém, který musí obsluhovat i jiné procesy a nebyl optimalizován pro práci v reálném čase. A konečně bude mít svůj podíl na dosaženém výsledku i použitý programovací jazyk, resp. běhové prostředí.

Nové řešení, vyvinuté v rámci této práce, odstraňuje majoritní část zdrojů problému s plynulostí pohybu. Komunikace se servomotory je řešena na nejnižší úrovni a s pomocí přerušení je dosaženo řádově lepších reakčních časů. Díky využití mikropočítače a operačního systému reálného času si programátor může jednoduše určit co v daném čase poběží a s jakou prioritou, což značně usnadňuje dodržení dostatečně nízké odchylky při komunikaci.

Dosažené výsledky byly prezentovány spolupracovníkům pracujícím na systému *RQA* ve firmě Y Soft a při přímém srovnání byl rozdíl v plynulosti patrný na první pohled.

	původní systém	nový systém
Standardní odchylka (µs)	907,541	0,143
95. percentil (µs)	1344,657	0,411
99. percentil (µs)	6013,925	0,566

Tabulka 3.1: Porovnání klíčových statistických parametrů odchylky od ideálního času odeslání pozice pro původní a nový systém.

Závěr

Cílem této diplomové práce byl návrh a realizace vestavěného systému, modulu, určeného pro řízení robotické ruky se čtyřmi stupni volnosti sestavené z inteligentních servomotorů. Robotická ruka je používána v rámci systému pro automatické testování multifunkčních zařízení firmou Y Soft, a jejím úkolem je klikání koncovým efektorem na dotykové displeje a mechanická tlačítka testovaných zařízení. Klíčovým cílem navrhovaného systému bylo zvýšení plynulosti pohybu manipulátoru oproti předchozímu systému.

Modul tvoří dvě, mechanicky a elektricky spojené, oboustranné desky plošných spojů vlastního návrhu. Vlastní návrh byl zvolen z důvodu potřeby vyvedení komunikačních sběrnic CAN, SPI a I^2C a možnosti budoucího rozšíření. První deska plošných spojů zodpovídá za stabilní napájení 12 V pro servomotory manipulátoru a také konverzi napětí na nižší použité pro napájení modulu. Deska umožňuje redundantní napájení s přednastavenou prioritou jednotlivých zdrojů napětí, kterými můžou být síťový zdroj nebo společná sběrnice. Deska také modul chrání před přepólování nebo příliš vysokým vstupním napětím a nadproudem. Druhá deska plošných spojů obsahuje výkonný mikropočítač řady *STM32H7*. Mikropočítač přijímá z nadřízeného systému, prostřednictvím rozhraní USB a protokolu MAVLink, příkazy popisující kýženou rychlostní křivku, cílový bod a orientaci koncového efektoru. Modul podporuje pohyb koncového efektoru dle Bézierovy křivky nebo lichoběžníkové rychlostní křivky. Mikropočítač zajišťuje validaci přijatých příkazů a jejich kompletní vykonání. Vykonání zahrnuje výpočet pozice a rychlosti pohybu koncového efektoru v časových okamžicích dle zvolené křivky, řešení kinematických úloh pro získání pozic servomotorů a zápis požadovaných úhlů natočení do servomotorů prostřednictvím sběrnice *RS-485*. Pro možnost řízení manipulátoru ze vzdáleného místa byla implementovaná aplikace komunikující s řídicím modulem prostřednictvím USB a umožňující řízení manipulátoru lokálně pomocí grafického nebo vzdáleně pomocí webového rozhraní.

Následně byly provedeny měření porovnávající systém realizovaný v rámci této práce a předchozí řešení, ve kterém počítač pomocí převodníku z USB na *RS-485* komunikoval se servomotory přímo. Hlavní měřenou metrikou byla odchylka skutečného času odeslání příkazu servomotorům od ideálního. Tato metrika je úměrná výsledné plynulosti pohybu a lze ji snadno změřit. S realizovaným systémem bylo dosaženo výsledků lepších až o čtyři řády. Například 99. percentil odchylky u předchozího systému byl v řádech nízkých jednotek milisekund se zřetelně viditelnými špičkami, u nového systému byly odchylky mírně nad polovinou mikrosekundy bez špiček přesahujících $\pm 2\,\mu s$. Další měřené metriky zahrnují dobu potřebnou k zápisu nové pozice do servomotorů, která klesla z $\approx 12\,ms$ na $\approx 470\,\mu s$. Přesnost a opakovatelnost pohybu zůstala, kvůli své závislosti na servomotorech, zachována.

Práce byla vytvořena ve spolupráci s firmou Y Soft, která mi umožňuje systém dále rozvíjet. Do budoucna plánuji mj. implementovat podporu komunikace protokolem *CANopen* a rychlostní křivku tvaru písmene *S* s možností definice ryvu.

Literatura

[1] Amazon Web Services. *The FreeRTOS^{TM} Reference Manual: API Functions and Configuration Options* [online]. 2017 [cit. 2019-10-12]. Dostupné z: https://www.freertos.org/wp-content/uploads/2018/07/FreeRTOS_Reference_Manual_V10.0.0.pdf.

[2] CAN in Automation. *History of CAN technology* [online]. [cit. 2019-11-23]. Dostupné z: https://www.can-cia.org/can-knowledge/can/can-history/.

[3] Catsoulis, J. *Designing embedded hardware.* 2. vyd. Sebastopol: O'Reilly, 2005. ISBN 0-596-00755-8.

[4] David a Will, S. *Robotis e-Manual Protocol 2.0* [online]. Robotis, 2017 [cit. 2019-11-20]. Dostupné z: http://emanual.robotis.com/docs/en/dxl/protocol2.

[5] David a Will, S. *Robotis e-Manual: Protocol Compatibility Table* [online]. Robotis, 2017 [cit. 2020-02-20]. Dostupné z: http://emanual.robotis.com/docs/en/popup/faq_protocol_compatibility_table/.

[6] David, Will, S. a Sookyung, S. *Robotis e-Manual XH430-W210-T/R* [online]. Robotis, 16. srpna 2019 [cit. 2019-11-20]. Dostupné z: http://emanual.robotis.com/docs/en/dxl/x/xh430-w210/.

[7] Diankov, R. *Automated Construction of Robotic Manipulation Programs.* 2010. [cit. 2020-01-16]. Disertační práce. Carnegie Mellon University, Robotics Institute. Dostupné z: http://www.programmingvision.com/rosen_diankov_thesis.pdf.

[8] Dušek, O. *Platforma pro automatické testování vestavěných zařízení.* Brno, 2018. Bakalářská práce. Vysoké učení technické v Brně. Vedoucí práce doc. Ing. Zdeněk Vašíček Ph.D.

[9] Finn, G. New Touch-Screen Controllers Offer Robust Sensing for Portable Displays. *Analog Dialogue.* 2010, č. 44, s. 2.

[10] Ha, T. *An open source cross-platform USB stack for embedded system* [online]. GitHub, 2020 [cit. 2019-11-02]. Dostupné z: https://github.com/hathach/tinyusb.

[11] Jack G., G. *Embedded hardware.* Amsterdam; Boston: Elsevier/Newnes, 2008. Newnes know it all series. ISBN 978-0-7506-8584-9.

[12] Martin, T. *The Designer's Guide to the Cortex-M Processor Family.* 2. vyd. Elsevier Science, 2016. ISBN 978-0-0810-0634-4.

[13] maxon motor ag. *Permanent magnet DC motor with coreless winding* [online]. 2012 [cit. 2019-10-02]. Dostupné z: https://www.maxongroup.com/medias/sys_master/8803450421278.pdf.

[14] Meier, L. a přispěvatelé do projektu MAVLink. *MAVLink Developer Guide* [online]. GitHub, 2020 [cit. 2019-11-02]. Dostupné z: https://mavlink.io/en/.

[15] Meier, L. a přispěvatelé do projektu MAVLink. *MAVLink Micro Air Vehicle Protocol – The standard communication protocol for drones* [online]. GitHub, 2020 [cit. 2019-11-02]. Dostupné z: https://github.com/mavlink.

[16] Moll, M., Şucan, I. A. a Kavraki, L. E. Benchmarking Motion Planning Algorithms: An Extensible Infrastructure for Analysis and Visualization. *IEEE Robotics & Automation Magazine*. September 2015, sv. 22, č. 3, s. 96–102. DOI: 10.1109/MRA.2015.2448276.

[17] Murphy, R. a Family, A. P. *AN57294; USB 101: An Introduction to Universal Serial Bus 2.0* [online]. Cypress Semiconductor Corp., leden 2020 [cit. 2020-03-05]. Dostupné z: https://www.cypress.com/documentation/application-notes/an57294-usb-101-introduction-universal-serial-bus-20.

[18] Nutt, G. E. *NuttX – Supported platforms* [online]. NuttX, 2019 [cit. 2019-10-02]. Dostupné z: http://nuttx.org/#supported_platforms.

[19] OpenAPI Initiative. *The OpenAPI Specification: a broadly adopted industry standard for describing modern APIs.* [online]. OpenAPI Initiative, únor 2020 [cit. 2020-03-20]. Dostupné z: https://www.openapis.org/.

[20] Orság, F. *Studijní opora předmětu ROB* [online]. ÚITS, FIT, VUT v Brně, listopad 2006 [cit. 2019-11-23]. Dostupné z: https://www.fit.vutbr.cz/study/courses/ROB/private/ROB.pdf.

[21] Ramachandran, H., Spencer, M. a Natalia, C. *STMicroelectronics Licenses ARM CORTEX-M3 Processor for use in next-Generation 32-BIT Microcontrollers* [online]. STMicroelectronics, říjen 2006 [cit. 2019-11-20]. Dostupné z: https://web.archive.org/web/20140215043846/http://www.st.com/web/en/press/t2077.

[22] Siciliano, B. *Robotics : modelling, planning and control.* London: Springer, 2010. Advanced textbooks in control and signal processing. ISBN 978-1-84628-641-4.

[23] STMicroelectronics. *AN4488 – Getting started with STM32F4xxxx MCU hardware development* [online]. říjen 2018, Rev 7 [cit. 2019-10-02]. Dostupné z: https://www.st.com/content/ccc/resource/technical/document/application_note/76/f9/c8/10/8a/33/4b/f0/DM00115714.pdf/files/DM00115714.pdf/jcr:content/translations/en.DM00115714.pdf.

[24] STMicroelectronics. *STM32 3rd-party Embedded Software – FreeRTOS Kernel* [online]. STMicroelectronics, 2019 [cit. 2019-10-02]. Dostupné z: https://www.st.com/en/embedded-software/freertos-kernel.html.

[25] STMicroelectronics. *STM32H753xI Datasheet - production data* [online]. 2019, Rev 7 [cit. 2019-10-02]. Dostupné z: https://www.st.com/resource/en/datasheet/stm32h753zi.pdf.

[26] STMicroelectronics. *RM0433 Reference manual - STM32H742, STM32H743/753 and STM32H750 Value line advanced Arm®-based 32-bit MCUs* [online]. 2020, rev. 7 [cit. 2020-02-02]. Dostupné z: https://www.st.com/resource/en/reference_manual/dm00314099-stm32h742-stm32h743-753-and-stm32h750-value-line-advanced-arm-based-32-bit-mcus-stmicroelectronics.pdf.

[27] Sucan, I., Moll, M. a Kavraki, E. The Open Motion Planning Library. *Robotics & Automation Magazine, IEEE.* Prosinec 2012, sv. 19, s. 72–82. DOI: 10.1109/MRA.2012.2205651.

[28] Suter, R. *The Swagger/OpenAPI toolchain for .NET, ASP.NET Core and TypeScript.* [online]. GitHub, 2020 [cit. 2020-02-12]. Dostupné z: https://github.com/RicoSuter/NSwag.

[29] Texas Instruments. *TPD2E007 2-Channel ESD Protection Array for AC-Coupled/Negative-Rail Data Interfaces datasheet* [online]. Leden 2016 [cit. 2019-10-02]. Dostupné z: http://www.ti.com/lit/ds/symlink/tpd2e007.pdf.

[30] Texas Instruments. *TPS6213x 3-V to17-V, 3-A Step-Down Converter In 3x3*

QFN Package datasheet [online]. Srpen 2016 [cit. 2019-10-02]. Dostupné z: http://www.ti.com/lit/ds/symlink/tps62130.pdf.

[31] Texas Instruments. *TPS25942x/44x 2.7 V-18 V, 5-A eFuse Power MUX With Multiple Protection Modes datasheet* [online]. Září 2017 [cit. 2019-10-02]. Dostupné z: http://www.ti.com/lit/ds/symlink/tps25942a.pdf.

[32] Texas Instruments. *SN65HVD7x 3.3-V Supply RS-485 With IEC ESD protection datasheet* [online]. 2019 [cit. 2019-10-02]. Dostupné z: http://www.ti.com/lit/ds/symlink/sn65hvd75.pdf.

[33] Texas Instruments. *TPS212x 2.8-V to 22-V Priority Power MUX with Seamless Switchover datasheet* [online]. Září 2019 [cit. 2019-10-02]. Dostupné z: http://www.ti.com/lit/ds/symlink/tps2120.pdf.

[34] Švejda, M. *Kinematika robotických architektur*. Plzeň, CZ, 2011. Práce ke státní doktorské zkoušce. Západočeská univerzita v Plzni. Dostupné z: http://home.zcu.cz/~msvejda/_publications/2011/rigo.pdf.

[35] Walker, G. A review of technologies for sensing contact location on the surface of a display. *Journal of The Society for Information Display*. 2012, sv. 20, s. 413–440.

[36] Wellbelove, J. *Embedded Template Library: A C++ template library for embedded applications* [online]. Aster Consulting Ltd, 2020 [cit. 2020-02-15]. Dostupné z: https://www.etlcpp.com/.

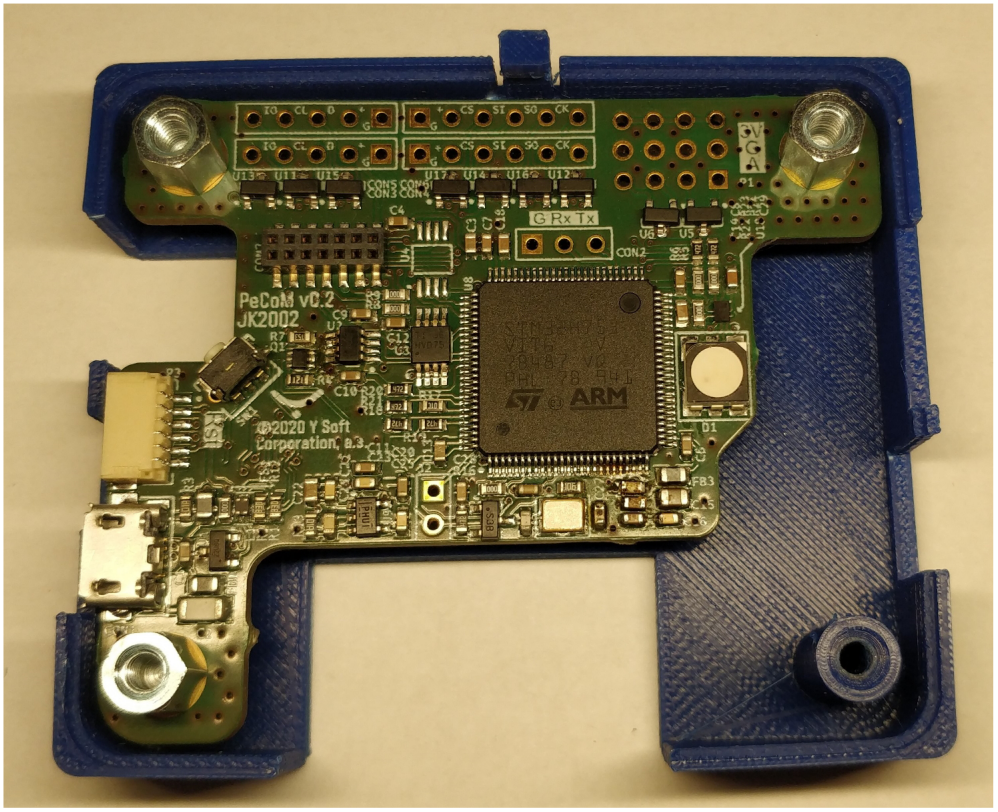
Příloha A

Data na paměťovém médiu

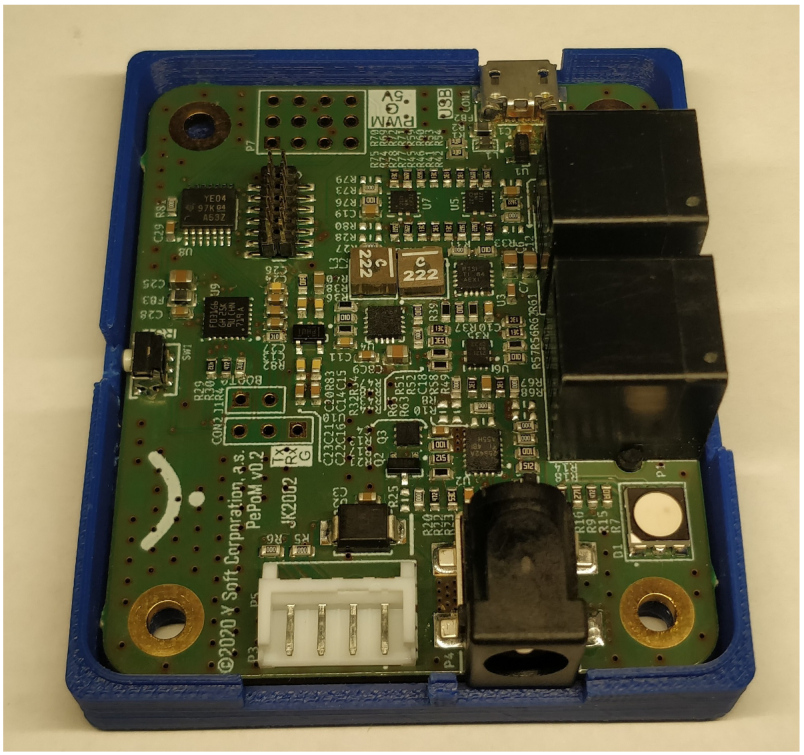
- `/`
  - `dokumentace` ..... zdrojové soubory textové části práce
  - `measurements` ..... nameřená data a grafy
  - `media` ..... obrázky modulu a video manipulátoru v pohybu
  - `pcb` ..... data týkající se desek plošných spojů
    - `doc/PeCoM` ..... dokumentace výpočetní DPS ve formátu *PDF*
    - `doc/PePoM` ..... dokumentace výkonové DPS ve formátu *PDF*
    - `src` ..... zdrojové soubory DPS ve formátu programu *Altium Designer*
  - `software` ..... data týkající se programového vybavení
    - `bin` ..... sestavený software: firmware mikrokontroléru a nadřízený systém
    - `src` ..... zdrojové soubory programového vybavení
      - `firmware` ..... zdrojové soubory firmware mikropočítače
      - `mavlink` ..... formální definice zpráv protokolu *MAVLink*
        - `generate-mavlink.ps1` skript pro generování enkodéru a dekodéru zpráv *MAVLink*
      - `pcm-test-app` ..... zdrojové soubory nadřízené aplikace
        - `test.ps1` ..... *PowerShell* skript využívající webové rozhraní aplikace

Příloha B

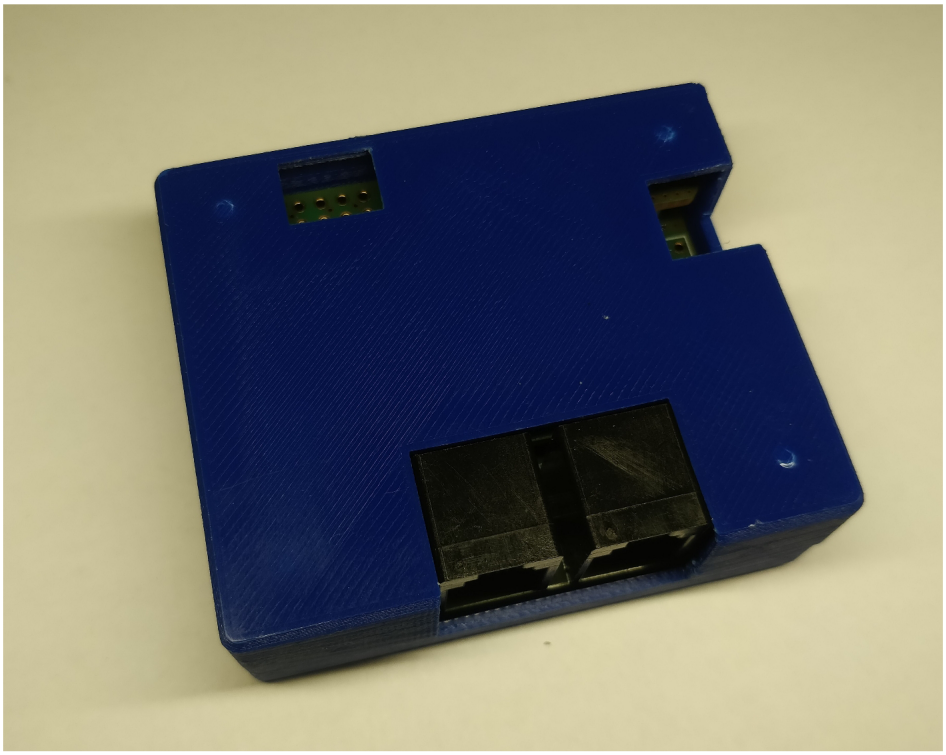
Obrázky vestavěného systému



Obrázek B.1: Výpočetní deska uchycená v horní polovině krabičky.



Obrázek B.2: Výkonová deska uchycená ve spodní polovině krabičky.



Obrázek B.3: Celý vestavěný systém v krabičce z 3d tiskárny.