VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

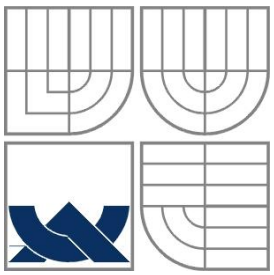
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

OVLÁDÁNÍ TŘÍOSÉHO TISKOVÉHO STROJE

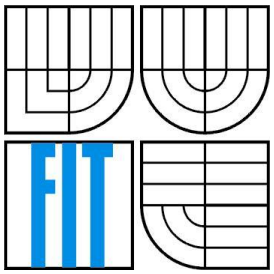
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE Bc. Darius Kočár
AUTHOR

BRNO 2010



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

OVLÁDÁNÍ TŘÍOSÉHO TISKOVÉHO STROJE

3D PRINTING MACHINE CONTROL

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Bc. Darius Kočár
AUTHOR

VEDOUCÍ PRÁCE — Ing. Michal Španěl
SUPERVISOR

BRNO 2010

Abstrakt

Tato práce se zabývá vytvořením knihovny pro řízení tříosého tiskového stroje na UPGM FIT. Osy stroje jsou realizovány krokovými motory firmy Microcon v součinnosti s řídícími jednotkami CD30x od stejné firmy. Komunikace mezi aplikací a řídícími jednotkami probíhá po sériové lince RS232 ve znakovém komunikačním protokolu. Součástí této práce je také aplikace demonstrující použití této knihovny.

Abstract

This thesis is about creating library for control of the 3D printing machine. Printer active axes are realized by stepper motors from Microcon Corporation connected with CD30x control unit. Communication between application and control units is via RS232 serial link. Part of this work is dedicated to application showing typical usage of this library.

Klíčová slova

3D tiskárna, Microcon, C++, řídící jednotka CD30x, RS232, krokový motor SX23-2727

Keywords

3D printer, Microcon, C++, CD30x control unit, RS232, SX23-2727 stepper motor

Citace

Kočár Darius: Ovládání tříosého tiskového stroje, Diplomová práce, Brno, FIT VUT v Brně, 2010

Ovládání tříosého tiskového stroje

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením Ing. Michala Španěla. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……

Darius Kočár
květen 2010

Poděkování

Chtěl bych poděkovat vedoucímu práce Ing. Michalu Španělovi za jeho zájem o práci a mnoho podnětných připomínek a návrhů k řešení, bez kterých by tato práce nebyla takovou jaká je. Dále bych poděkoval paní One Dzilbute za korekturu litevského překladu aplikace.

Obsah

1 Úvod

3D tiskárny slouží k výrobě třírozměrných objektů skládáním z vrstev materiálu. Na rozdíl od tradiční výroby, kdy dochází k opracovávání surového materiálu do výsledné podoby, tato technologie je založena na součtové výrobě. Mezi její výhody patří jednoduchost, ekonomičnost a hlavně rychlost výroby. Tato technologie dovoluje výrobu prototypů v kancelářských podmínkách, aniž by kladla vysoké požadavky na uživatele. V současné době lze takto vytvářet objekty z mnoha různých druhů materiálů. Jedním ze směrů, kterým se 3D tiskárny ubírají je jejich použití ve výrobě plošných spojů, především ve výrobě vícevrstvých plošných spojů. V budoucnu tento směr nabízí rozšíření až k tisku samotných součástek a tak k extrémně rychlé výrobě prototypů elektrických zařízení.

Tato práce se zabývá tvorbou software částí ovládání 3D tiskového stroje umístěného na Ústavu počítačové grafiky a multimédií. Tato 3D tiskárna používá pohony od firmy Microcon. Řídící a napájecí část je od stejného výrobce. K tiskárně je pořízena také tisková hlava umožňující tisk pomocí různých materiálů. Konstrukce ani řízení této hlavy ovšem není cílem této práce. Komunikace aplikace (počítače) s tiskárnou probíhá po sériové lince, která je emulována přes sběrnici USB. Komunikace je znaková s protokolem daným výrobcem.

Výsledná knihovna i aplikace je napsána v C++. Pro platformě nezávislý přístup k sériové lince je použita knihovna Boost. Konkrétně její část zvaná asio. Pro GUI je použito knihovny wxWidgets, kdy vlastní definice GUI byla provedena pomocí aplikace wxFormBuilder, ze které byly následně vygenerovány zdrojové soubory v C++. Výsledná knihovna potom poskytuje univerzální rozhraní pro řízení tříosého tiskového stroje, které je již programově nezávislé na řídících jednotkách a jejich motorech.

Pro tuto knihovnu bude poté vytvořena demonstrační aplikace, kterou bude možno i prakticky použít pro vlastní tisk třírozměrných předmětů. Pro účely této aplikace bude i dále vyvinuta knihovna pro práci s třírozměrnými modely uloženými ve standardních typech souborů jako VRML, OBJ a další.

Tato práce je rozdělena do několika kapitol, které provádějí přes celý proces řešení. Po úvodu následuje kapitola 2 Typy pohonů stručně pojednávající o řešení tiskového stroje z pohledu hardware. Ve třetí kapitole je popisován Tiskový stroj na FIT. Je zde rozebráno současné řešení tiskového stroje. Zaměřeno je hlavně na software stránku, tedy na komunikační protokol řídících jednotek motorů a nástrahy a omezení, které toto řešení přináší v kombinaci s použitým hardware. Kapitola 4 Návrh již začíná řešit podstatu práce. Stylem shora dolů je zde rozebrán návrh software části. Její rozdělení do menších celků a definice funkčnosti těchto celků. Kapitola 5 Implementace pak rozebírá tyto funkční celky již na úrovni programovacího jazyka. Výsledkem této kapitoly jsou funkční verze aplikace a knihoven. V závěru je zhodnoceno celkové řešení jak z hlediska splnění požadovaných cílů, tak i z hlediska funkčnosti a další použitelnosti v projektu 3D tiskárny jako celku.

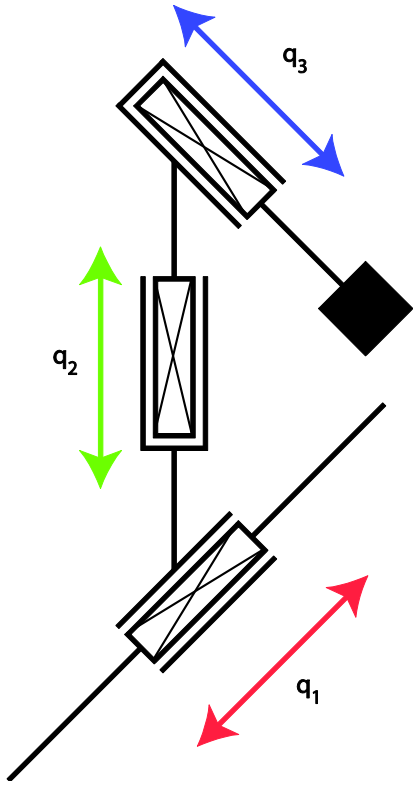
2 Typy pohonů

K zajištění volného pohybu tělesem, v našem případě tiskovou hlavou v třírozměrném prostoru, jsou třeba efektory. Pod pojmem efektory jsou chápany různé typy koncových silových zařízení. Jsou to zpravidla různé typy motorů. Počínaje obyčejnými lineárními motory, přes různé variace krokových motorů až po velmi speciální případy hydraulických ramen. V této kapitole jsou všechny tyto základní typy pohonů rozebrány z hlediska principů fungování a z nich plynoucích parametrů. Následně je diskutována vhodnost jejich použití v aplikaci tříosého tiskového stroje.

Spolu s problémem optimalizace výběru pohonu je zde také diskutována problematika celkové konstrukce tříosého stroje. Pohyb objektu - v našem případě tiskové hlavy (dále jen hlavy) lze řešit mnoha způsoby. Při vlastní realizaci efektorů lze použít buď rotační manipulátory, či translační manipulátory. Rotační manipulátor má například jiné požadavky na pracovní prostor oproti translačnímu manipulátoru. Výběr manipulátoru také nepřímo ovlivní přesnost pohybu hlavy. Při výběru řešení se postupuje pomocí aplikace omezení a požadavků. Výsledkem by měla být tabulka konstrukcí seřazených podle vhodnosti. Toto ale není cílem práce, takže následuje pouze krátká diskuse nad typy pohonů, jejímž výsledkem je nejlepší řešení a jeho srovnání s již existujícím strojem.

První omezení vyplývá ze samotné konstrukce použité tiskové hlavy. Tato hlava poskytuje nejlepší výsledky pří uvolňování tiskového materiálu směrem shodným s gravitační silou – tedy dolů. Tímto můžeme konstrukci stroje omezit na 3 stupně volnosti.

Uvolňování tiskového materiálu probíhá hromadně v jednorozměrných řadách. Je třeba brát rovněž v úvahu, že hustota tisknutých bodů je daná a to jak z hlediska materiálu použitého výrobcem, tak i odstupem tiskových bodů při hromadném tisknutí v hlavě. Výsledný „vytištěný“ produkt by měl mít stejnou kvalitu materiálu v celém svém objemu. Při použití rotačních manipulátorů by tohoto kritéria nebylo dosaženo, protože s rotací ramene by docházelo i k rotaci hlavy oproti již vytisknutým řadám. Mimo to mají standardní rotační manipulátory nevýhodu v tom, že jejich přesnost je závislá na vzdálenosti od osy rotace. Ale v případě tiskárny potřebujeme hlavu pozicovat v prostoru. V tomto případě by tedy výsledná přesnost byla vysoce závislá na vzdálenosti hlavy od osy otáčení ramene. Použití rotačních manipulátorů je tedy z tohoto důvodu



Obrázek 1: Koncepce manipulátoru ve schematickém zobrazení.

nevhodné a to na kterékoliv ose.

Výsledné řešení tedy bude moci pohybovat hlavou ve 3 stupních volnosti. Translační pohyb je pro tento účel mnohem vhodnější, protože přesnost jeho regulace a nastavení je vysoká v celém rozsahu použití. Z tohoto důvodu budou k pohybu na všech stupních použity translační manipulátory. Výsledek je zatím shodný s dodaným řešením. Nyní přejdeme k výběru technologie vlastních manipulátorů. Následující podkapitoly se nebudou zabývat konkrétními výrobky, či řešeními jednotlivých firem. Řešení probíhá na úrovni technologií.

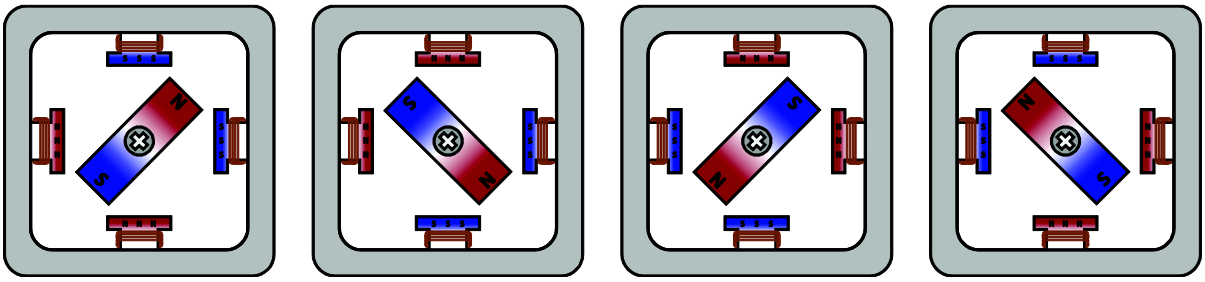
2.1 Rotační motory s převodem na translační pohyb

Rotační motor je nejrozšířenějším typem pohonu. K převodu na translační pohyb se v parametrech tiskového stroje používá převodovka se šnekem. Rotační motory jsou mnoha typů. Není zde rozebíráno, proč se pro pohon nehodí motor benzínový, či turbína a zaměřeno je přímo na malé elektrické motory. Takové motory můžeme vhodně rozdělit podle napájecího napětí:

- Stejnosměrné motory
- Střídavé motory
- Krokové motory

Motorem je potřeba pohybovat o určitý, přesně stanovený úsek. Při použití motorů stejnosměrných či střídavých tak vyvstává potřeba je dovybavit enkódérem, který by zpětnou vazbou poskytoval informace o aktuální poloze motoru (osy). Enkódér může být buď absolutní či inkrementální. Rozdíl je pouze v tom, že při použití inkrementálního enkódéru je třeba po resetu řídícího software znovu kalibrovat stroj. Minimálně tedy ve smyslu nalezení počátku. [5]

Jiným typem jsou krokové motory. Tento typ motoru se vyznačuje svoji přesností a možností řízení po jednotlivých krocích otáčení hřídele. Je zde omezení v podobě maximálního momentu, po jehož překročení se realita začne rozcházet s řídícími signály. To je dáno tím, že zde chybí zpětná vazba o vykonané práci motoru. Maximální moment je ale obecně dostačující pro potřeby tiskového stroje. Krokové motory mají negativní vlastnost v podobě rezonance při nízkých frekvencí. Tu je možné odstranit pomocí mikrokrokování. [5]



Obrázek 2: Pracovní stavy dvoufázového krokového motoru.

Oba typy motorů jsou srovnatelné a vhodné. Z hlediska jednoduchosti (a tím i ceny) je vhodnější krokový motor. Pro obecné motory zase hovoří varianta s absolutním enkódérem, která by byla odolná proti nečekaným výpadkům napájení v průběhu tisku. Při kalibraci po obnovení napájení by si tiskárna model v lepším případě rozbila, v horším případě by poškodila i sebe. Tomuto stavu je možné předejít zálohovaným napájením a postupy pro přerušení tisku s jeho následnou obnovou. Výsledkem této krátké diskuse je preference krokového motoru. [5]

2.2 Lineární motory

Lineární motory se vyznačují tím, že již ze své podstaty poskytují translační pohyb. Konstrukce vychází z rotačních motorů, kdy je plášť jakoby rozbalen po celé dráze pohybu. Teoreticky jsou možné opět všechny typy (střídavé, stejnosměrné a krokové). V praxi se používají střídavé a krokové. Absence převodovky je jejich výhodou spočívající v jednoduché konstrukci a nižším opotřebení. Jejich cena se totiž pohybuje v násobcích cen rotačních motorů s převodovkami. [2] [5]

2.3 Hydraulické pohony

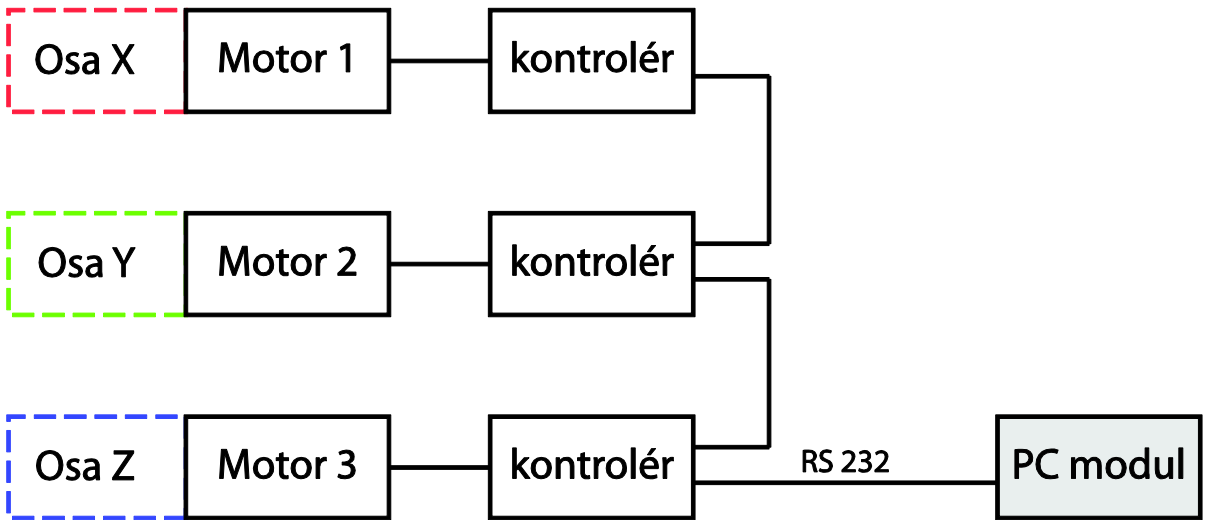
Hydraulické pohony se vyznačují schopností přenosu velké síly a relativně dostatečnou přesností. Jejich nevýhodou na druhé straně jsou pomalá rychlost a vyšší cena ve srovnání s elektrickými pohony. Vzhledem k tomu, že tisková hlava uvažovaná pro naše řešení není nijak těžká, není z tohoto důvodu nutné použití této technologie. Mimo to je zde opět problém s určením aktuální polohy.

3 3D tiskový stroj na FIT

Z porovnání hlavních vlastností a charakteristik možných pohonů a omezujících kritérií vyplývajících pro zajištění pohybu tiskové hlavy vyplynulo, že potřebné požadavky a podmínky nejlépe splňují rotační motory s převodem na translační pohyb. V případě použití stejnosměrných nebo střídavých motorů je nutné je dovybavit enkódérem.

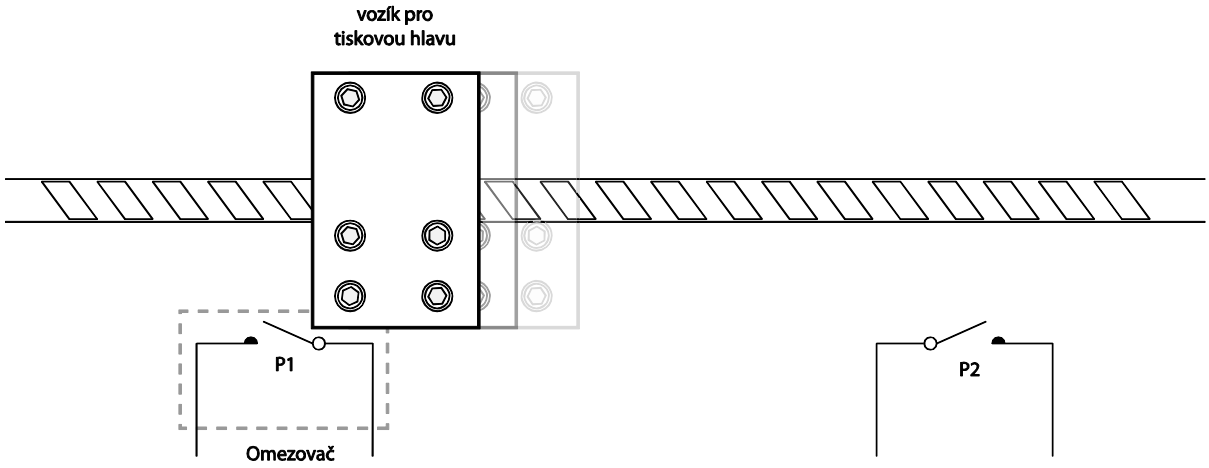
Výsledkem výběru typu pohonu je použití tří krokových motorů s převodem na translační pohyb. Každý z nich zajišťuje posuvný pohyb v jedné ose, které jsou na sebe vzájemně kolmé. Tímto je zajištěna funkce ve všech třech stupních volnosti. Toto řešení určuje pracovní prostor tiskové hlavy ve tvaru krychle.

Konstrukce 3D tiskového stroje na UPGM FIT odpovídá této specifikaci. Celý tiskový stroj je umístěn v rámové hliníkové konstrukci o vnějších rozměrech přibližně 1 x 1 x 1,5 metru (výška x délka x šířka). Pro vlastní pohon os jsou použity tři krokové motory výrobce Microcon SX23-2727 dovybavené spojkou XY25 a vedením Kuroda SG3310A-400P. Každý motor je připojen na kontrolér CD30x. Sériová komunikační rozhraní těchto kontrolérů jsou zapojena paralelně s výstupní řídící sériovou linkou. Napájení celého stroje je řešeno ze sítě 230V/50Hz. K převodu napětí na úrovně potřebné u kontrolérů a motorů jsou použity transformátory PS 35. Blokové schéma zapojení jednotlivých součástí tiskového stroje je znázorněno na obrázku 3 níže. [6]



Obrázek 3: Blokové schéma hardware zapojení tiskového stroje FIT.

Každá osa, resp. motor, je doplněn o dva omezovače vymezující pracovní prostor tiskové hlavy. Omezovač je realizován spínačem, k jehož sepnutí dochází při průjezdu tiskové hlavy. Názorně je funkce omezovačů zobrazena na obrázku 4 níže.



Obrázek 4: Umístění omezovačů na tiskovém stroji.

Použity jsou dva druhy omezovačů

- Mechanické
- Indukční

Pro každou osu je použit jeden indukční omezovač a jeden mechanický. Indukční omezovač poskytuje možnost více sepnutí po dobu své životnosti. Tato vlastnost je využita při kalibraci stroje – podrobněji rozebírané v kapitole Asynchroní volání. Jelikož je celá konstrukce tiskového stroje až na výjimky z hliníku, tak je u indukčního omezovače průjezd vozíku detekován pomocí malého magnetu, který je umístěn na vozíku. V případě mechanického je průjezd detekován mechanickým vychýlením plíšku spínače.

3.1 Krokový motor SX23-2727

Technické parametry krokového motoru SX23-2727 jsou uvedeny níže v tabulce 1. V kombinaci se spojkou XY25 a vedením Kuroda SG3310A-400P jsou vhodné pro účely tiskového stroje.

Krokový motor SX23-2727	
Statický moment [Nm] (bipol. napájení, celokrok, jmen. proud v obou fázích)	2,7
Jmenovitý proud [A] (sériové / paralelní zapojení)	2,7 / 5,4
Indukčnost [mH] (sériové / paralelní zapojení)	6,4 / 1,6
Odpor [Ω]	1,4 / 0,35

(sériové / paralelní zapojení)	
Moment setrvačnosti rotoru [kgm^2x10^{-3}]	0,053
Hmotnost [kg]	1,13

Tabulka 1: Parametry krokového motoru SX23-2727.

3.2 Řídicí jednotka CD30x

Řídicí jednotka CD30x je velmi malých rozměrů 105 x 57 x 47 mm. Obsahuje 10 galvanicky oddělených uživatelských vstupů a 4 uživatelské výstupy. Tyto V/V jsou dostupné na svorkovnici. Výrobce dále uvádí jako nadstandardní přednost zvýšenou ochranou proti rušení v podobě rozsáhlé zemnící plochy a zcela galvanicky odděleným kontrolérem M1486. Pro komunikaci s řídícím software je použita sériová linka, kterou lze volitelně připojit přes konektor Canon či přímo přes svorkovnici. Pro signalizaci stavů jsou použity barevné LED diody. Parametry výkonové části lze nalézt v tabulce 2 níže. [6]

Řídící jednotka CD30x	
Napájecí napětí	12 - 48 VDC
Amplituda proudu	0,4 - 3,3 A
Nastavení proudu	v šestnácti stupních
Dělení celokrok	nastavení pomocí spínače SIP
Doporučený počet mikrokroků na celokrok	4, 8, 16
Automatické snížení proudu po zastavení motoru	ANO
Možnost programového vypnutí koncového stupně	ANO
Momenty vhodných krokových motorů	1,2 - 8,5 Nm
Rozměry	105 x 57 x 47 mm
Doporučený napájecí zdroj	PS 35

Tabulka 2: Parametry řídící jednotky CD30x.

3.3 Mikrokontrolér M1486

Mikrokontrolér, kterým je vybavena řídicí jednotka je také od výrobce Microcon. S velkou pravděpodobností se však jedná o přeznačený standardní mikrokontrolér. Všechny jeho parametry

jsou k nalezení v brožurce „stepper motor controller Microcon M1486“.[10] Zde jsou zmíněny pouze podstatné technické a provozní informace oproštěny od obchodní a marketingové terminologie.

Mikrokontrolér poskytuje možnost řídit motor až do rychlosti 40000 kroků/s, délka dráhy přitom může být až 16 000 000 kroků při nepoužívání funkce mikrokrokování. Dále obsahuje 21 V/V portů, z nichž 10 pro vstup a 4 pro výstup jsou přístupné uživateli v rámci začlenění mikrokontroléru do jednotky CD30x. V realizaci tiskového stroje jsou 2 vstupy využity pro zapojení omezovačů. Označení omezovač je použito pro indikátor dorazu.

Programovatelné jsou veškeré parametry provozu motoru jako rychlost, zrychlení atd. až po tvar proudu řídicího motoru. Nevýhodou se může jevit nemožnost plynule ovlivnit parametry pohybu před jeho dokončením. Komunikace s nadřazenou jednotkou probíhá po sériové lince. Přenosovou rychlost lze měnit zkratovacími propojkami. Komunikační protokol umožňuje připojení až 16 mikrokontrolérů, resp. řídicích jednotek na jednu linku. [6][10]

Komunikační protokol je definován povelovým souborem. Všechny povely jsou ASCII znaky, kdy nezáleží na velikosti znaku (a/A). Celkově je v protokolu obsaženo více než 50 povelů. Povely jsou tvořeny s cílem jejich snadného zapamatování. Například povel pro nastavení rychlosti je v. Odvozen je od anglického slova velocity, které v češtině znamená rychlost. Stručný seznam všech povelů lze nalézt níže v tabulce 3. Jejich detailní rozbor lze poté nalézt v manuálu „stepper motor controller Microcon M1486“, který je součástí příloh této zprávy. [10]

Povel	Zkratka	Popis
\	Reset	uvedení kontroléru do výchozího stavu
@ (Num)	Address	zadané číslo návěští se přiřadí této programové řádce
A (Num)	Acceleration	zrychlení, rozsah = 1 až 65 000 kroků/s^2
B (Num)	Backward	zpět, zadání dráhy v negativním směru, rozsah = 1 až 16e6
C (Num 1 to 21)	Clear	nastav zadaný výstup do hodnoty logická nula, rozsah 1 až 21
C (Num 40 to 63)	Clear	potlačení zadané přídavné funkce
C75	Clear Kill	obnovení vykonávání programu
D	Direction	směr, změna směru příštího pohybu
E	End of look	konec smyčky
F (Num)	Forward	dopředu, zadání dráhy v pozitivním směru, rozsah= 1 až 16e6
G (Num)	Go absolute	dráha zadána absolutní polohou
G+	Go positive	trvalý pohyb v pozitivním směru až do externího přerušení
G-	Go negative	trvalý pohyb v negativním směru až do externího přerušení
H	Home	vykonej pohyb do výchozí polohy
I (Num)(Value)(Num)	If	jesliže na zadaném vstupu je zadaná hodnota skoč na zadané
	náv□ští	("H" - High logická jedna, "L" Low - logická nula)
J (Num)	Jump	skok na zadané návěští
K	Kill	okamžitý přechod do brzdného režimu, přerušení vykonávání programu
L (Num)	Loop	smyčka, opakuj provádění následujících instrukcí
M (Num)	Microstepping	počet mikrokroků na celokrok v dolním pásmu rychlostí (až do 64)

N (Num)	Number	výběr tvaru průběhu proudu při mikrokrokování
O (Num)	One	čekej dokud zadaný vstup nebude mít hodnotu logická jedna
P (Num)	Profile	rychlost, při které se lineární rozběhová charakteristika mění na parabolickou
Q (Num)	Qualification	počet mikrokroků na celokrok v horním pásmu rychlostí (1, 2, 4)
R	Run	vykonej pohyb s aktuálními hodnotami parametrů
S (Num)	Start/stop	rychlost start/stop; rozsah = 16 až 1950 kroků/s
T (Num 1 to 21)	Turn on	nastav zadaný výstup do hodnoty logická jedna, rozsah 1 až 21
T (Num 40 to 63)	Turn on	zapnutí zadané přídavné funkce
U (Num)	Upload	vyšli hodnotu čítače absolutní polohy či hodnotu interní proměnné
V (Num)	Velocity	maximální rychlost
W (Num)	Wait	čekej zadaný počet milisekund; rozsah = 1 až 16 e6
X (Num)	indeX	volba kontroléru
Z (Num)	Zero	čekej dokud zadaný vstup nebude mít hodnotu logická nula
[	Disable	odklad provedení následujících povelů
]	Enable	provedení předchozích povelů
((Num)	Seek negative	jdi na limit v negativním směru
) (Num)	Seek positive	jdi na limit v pozitivním směru
= (Num)	Equal	přiřazení zadané hodnoty čítači absolutní polohy
: (Num)	Load	ulož zadanou hodnotu do interní proměnné
? (Num)	Query	načti data na specifikovaných vstupech a ulož do interní proměnné
! (Num)	Order	zapiš hodnotu interní proměnné na specifikované výstupy
+ (Num)	Add	přičti zadanou hodnotu k interní proměnné
- (Num)	Substract	odečti zadanou hodnotu od interní proměnné
/ (Num)	Divide	dělení interní proměnné zadanou hodnotou
* (Num)	Multiply	násobení interní proměnné zadanou hodnotou
> (Num)	Move to register	přesun dat z interní proměnné do zadaného registru
< (Num)	Move from register	přesun dat ze zadaného registru do interní proměnné
' (Num)	Subroutine	podprogram
. (Num)	End of subroutine	konec podprogramu

Tabulka 3: Přehled povelů kontroléru M1486. [10]

Jednotlivé příkazy povelového souboru lze potom skládat do povelových souborů. To jsou uzavřené celky, které provádí ucelenou operaci. Následují dva příklady, jak vypadá povelový soubor:
Příklad 1 povelového souboru [10]:

- \ :“Reset“ - Do registrů parametrů jednotlivých povelů jsou zapsány počáteční hodnoty. U kontroléru M1486E nesmí být povel "Reset" součástí povelového souboru, další povely mohou být vyslány až po uplynutí času potřebného na vymazání paměti EEPROM (viz popis povelu "Reset").

- **f12800** ;“Forward“ - Zadání dráhy v pozitivním směru 200 celých kroků (12800 nejmenších mikrokroků).
- **r** ;“Run“ - Vykonej pohyb o délce 200 celých kroků s počátečními hodnotami rychlosti start/stop, maximální rychlosti, zrychlení a dělení kroku.

Spuštěním povelového souboru z příkladu 1 dojde k posunu motoru o 12800 mikrokroků vpřed. Jedná se o ilustrační povelový soubor. V praxi spíše nepoužitelný, protože často je třeba udržovat kontext mezi jednotlivými pohyby. Následující příklad 2 povelového souboru je již komplexnější a ukazuje, jak může vypadat typický povelový soubor.

Příklad 2 povelového souboru[10]:

- **[**;"Enable" začátek povelového souboru
- **S200** ;"Start/stop" počáteční rychlost 200 kroků/s
- **V1000** ;"Velocity" maximální rychlost 1000 kroků/s
- **A5000** ;"Acceleration" zrychlení 5000 kroků/s2
- **F39616** ;"Forward" zadaná dráha 39616 mikrokroků, směr dopředu
- **L3** ;"Loop" následující povely budou provedeny třikrát
- **R** ;"Run" vykonej pohyb
- **W250** ;"Wait" čekej 250 milisekund
- **E** ;"End of loop" konec smyčky
- **]** ;"Disable" konec povelového souboru

Provedením tohoto povelového souboru dojde v úvodu k nastavení parametrů pohybu motoru. Konkrétně počáteční rychlosti, která nepřímo definuje sílu, s jakou se bude motor rozjíždět/brzdit. Dále je nastavena rychlost pohybu a akcelerace na tuto rychlost z počáteční start/stop rychlosti. Akcelerací je i analogicky definována prudkost brzdění. Jako poslední je nastavena relativní velikost pohybu ve směru vpřed.

Pozn.: U akcelerace a maximální rychlosti je jako délková jednotka použit celý krok. Při definování vzdálenosti pohybu se však používají nejmenší mikrokroky – tedy ne aktuálně nastavená jemnost pohybu (počet mikrokroků), ale maximální možný počet mikrokroků. Toto číslo pak musí být dělitelné aktuálním počtem mikrokroků.

Následuje smyčka, která bude třikrát zopakována. V ní je již pouze uspání trvající 250ms následované vykonáním výše definovaného pohybu.

Často používaným příkazem je ještě **u** – upload. Tímto příkazem je řídící jednotka vyzvána k poslání dat. Na tomto příkazu je zajímavé, že k jeho provedení dojde až po vykonání všech

předchozích příkazů. Jeho použití je tedy v synchronizaci pohybu s jednotkou PC. Modifikací předchozího příkladu 2 povelového souboru vznikne blokující verze:

- **…**
- **R** ;"Run" vykonej pohyb
- **W250** ;"Wait" čekej 250 milisekund
- **E** ;"End of loop" konec smyčky
- **U20** ;“upload“ pošli verzi řídící jednotky
- **]** konec povelového souboru

Před dokončením tohoto povelového souboru je proveden příkaz U20, který pošle verzi řídící jednotky. Informace je to naprosto nepodstatná. Důležité je, že když zablokujeme program čekáním na tuto informaci, tak v době jejího přijetí již bude proveden celý povelový soubor. Toto je důležité zvláště při programovém řízení jednotky, kdy je možné příkazy generovat a posílat násobně rychleji, než by je stihla řídící jednotka vykonávat.

Na základě experimentování s ovládáním tiskárny byla stanovena základní struktura knihovny. Tato struktura je rozebírána dále v kapitole 4.3 Knihovna řízení tiskového stroje.

4 Návrh

4.1 Hardware

Konstrukce Tiskového stroje na UPGM FIT je daná a vhodná. Pro potřeby řízení obsahuje koncové spínače (omezovače). Tyto spínače vymezují pracovní prostor tiskové hlavy.

Ve výsledku tvoří celá tiskárna jeden kompaktní celek, ke kterému vede pouze napájení a řídicí sériový kabel, který by při dalším případném zdokonalení tiskového stroje bylo možno odstranit a nahradit bezdrátovým přenosem.

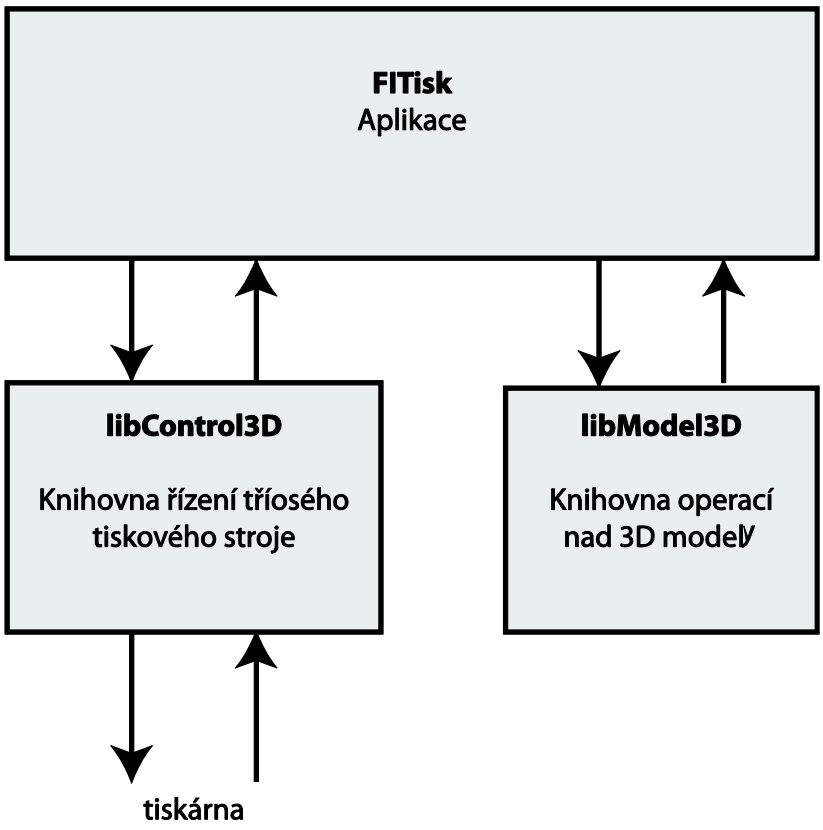
Pomocí demonstrační aplikace vytvořené Ing. Michalem Španělem byla otestována funkčnost celého řešení. Tato aplikace poté výtečně posloužila ve fázi zprovozňování knihovny řízení.

4.2 Software

Celá aplikace je rozdělena do kompaktních, na sobě nezávislých celků. Teprve jejich složením vznikne použitelná aplikace. Důvodem k tomuto kroku je minimalizace závislostí v kódu a také závislostí na programovacích jazycích. Jako hlavní blok se přímo nabízí knihovna pro řízení tiskového stroje. Nad ní bude postavena aplikace s uživatelským rozhraním. Aplikace bude dále využívat knihovnu pro manipulaci s třírozměrnými objekty. Tato knihovna bude dodávat data potřebná pro tisk modelu. V budoucnu se počítá s přidáním knihovny řízení tiskové hlavy.

Pro použití v programu je vhodné zavést zkrácené názvy knihoven. Knihovna pro řízení tříosého tiskového stroje bude nazývána **libControl3D**. Knihovna pro operace nad 3 rozměrnými modely pak libModel3D. Výsledná aplikace se bude nazývat „FITisk“. Název by měl vystihovat, že se jedná o projekt FIT a zároveň spojitost s tiskem. Schéma komunikace mezi jednotlivými bloky je na obrázku 5.

Programovací jazyk by měl být ze skupiny C/C++, C# nebo Python. Dále by měl být výsledný kód pokud možno přenositelný minimálně mezi Windows a Linux. Ze skupiny lze vyřadit jazyk C#, který pod Linuxem nemá oficiální podporu. Na výběr zbývá C/C++ nebo Python. Se silou jazyka není problém u žádného z nich. V knihovně libModel3D může a velmi pravděpodobně i bude docházet k časově intenzivním výpočtům a proto se preference přiklání k použití C/C++. Jako programovací jazyk byl zvolen C++. Hlavními důvody pro jeho volbu je také existence mnoha kvalitních open source knihoven jako například Boost, či wxWidgets. Použití těchto knihoven umožní implementovat pokročilé funkce, či moderní GUI a to vše na platformě nezávislém kódu. Knihovna pro operaci s 3D modely bude psána v jazyce C++. V rámci konzistence zvolíme C++ i pro knihovnu řízení tříosého tiskového stroje. Programovací jazyk GUI je vybrán v kapitole 4.3 Grafické uživatelské rozhraní [1] [3] [4]



Obrázek 5: Schematický model návrhu software.

4.3 Knihovna řízení tiskového stroje

V následující kapitole jsou rozebírány dílčí problémy, které bylo třeba při návrhu a vývoji aplikace řešit. Jedná se o detekci a obsluhu chyb v kapitole timeout. Řešení simultánního pohybu motorů ve stejnojmenné kapitole. Nebo asynchronní volání funkcí třídy. Správa nastavení celé knihovny je poté rozebrána v poslední kapitole.

Timeout

Komunikace v protokolu Microcon probíhá ve stylu dotaz-odpověď. Je zde tedy jednotka PC, která posílá příkazy a dotazové příkazy. Pak je zde jednotka řídící (slave), která zpracovává tyto příkazy. Pokud se jedná o dotazový příkaz, tak komunikační protokol přesně definuje formát a délku odpovědi. Co není definováno, je časový rámec do kdy musí odpověď přijít. Chybu, že by řídící jednotce odpověď nedošla či došla poškozená, ale nelze vyloučit. Ke ztrátě či poškození dat může dojít na mnoha místech.

- Zpracování ve firmware řídící jednotky

- Přenos po sériové lince, respektive i konverze na sběrnici USB a její přenos.
- Zpracování na straně jednotky PC.

Pravděpodobnost chyby při přenosu je velmi nízká. Její hodnota velmi záleží na podmínkách, ve kterých jsou tyto sběrnice (RS232 a následně USB) provozovány. Obecně lze ale pokládat přenos za spolehlivý. Častý je však případ, kdy v konektoru není vůbec zapojen žádný kabel. V takovém případě dochází ke ztrátě všech vysílaných dat. [11]

Správné zpracování dotazu ve firmware řídící jednotky již bude mít řádově nižší pravděpodobnost chyby. Výrobce deklaruje, že tato řada výrobků není vhodná pro aplikace v medicíně mající přímí vliv na život a zdraví pacienta. Chyba zde sice nebude znamenat smrt člověka, ale i přes to by měl být její výskyt ošetřen.

Posledním článkem je zde jednotka PC, tedy vlastní knihovna řízení tiskového stroje a aplikace na ní postavená. Tato část má velký potenciál, aby obsahovala nejednu chybu. Zároveň je i posledním článkem před tím, než by se chyba dostala k uživateli. Z těchto důvodů bylo přistoupeno k ošetření dotazových příkazů pomocí časovače. Nepřijde-li v časovém limitu korektní odpověď, tak je tato situace indikována vyšším vrstvám řízení pomocí výjimky. Zpracování této výjimky může být na nižší úrovni ve formě zotavení se funkce. Například by mohlo dojít k opětovnému pokusu o přijetí korektní odpovědi. Tato možnost je obecně nedoporučována, protože pokud by uspěla, tak je velmi pravděpodobné, že k této chybě nedošlo z důvodu selhání na straně řídící jednotky či přenosu dat. Takový stav ukazuje spíše na špatně zvolený timeout vzhledem ke konstrukci a nastavení stroje. Doporučené zpracování je oznámení chyby uživateli vhodně zkombinované s možnostmi jejího řešení.

Jako další je zde záležitost řízení přístupu k sériové lince. Sériová linka je navržena pro propojení dvou zařízení. V této konfiguraci poskytuje plně duplexní přenos. Ten je zajištěn díky vyčlenění datových vodičů pro každý ze dvou směrů komunikace. V konfiguraci tiskového stroje jsou ale na jednu sériovou linku připojeny 4 zařízení. Tři řídící jednotky a jednotka PC. Tímto se ze sériové linky stala de facto sběrnice. Sběrnice, která ale neposkytuje jakékoliv řízení přístupu k médiu. [11] Vzhledem k nemožnosti zasahovat do řídící jednotky a také z důvodu obecnosti celého řešení byl přístup k médiu vyřešen vztahem master/slave. Master jednotkou je zde PC a slave jednotkami jsou řídící jednotky. Právo komunikovat na sběrnici má pouze master jednotka. Master jednotka může delegovat toto právo na řídící jednotku. To se děje typicky v případě dotazového příkazu. Toto právo je časově omezeno. Vypršení tohoto časového limitu je bráno jako závažná chyba. Po jejím výskytu je třeba resetovat všechny řídící jednotky, aby při další komunikaci byl zajištěn konzistentní stav. Indikace a ošetření této chyby je rozebrána v předchozím odstavci.

Při testování a ladění tohoto řešení bylo přistoupeno k doplnění funkčnosti timeoutu o třídu `ScopedTimeout_t`. Lze říci, že timeout v řádu sekund je třeba pouze u příkazů manipulující s motory. Při použití příkazu inicializace osy, který nastavuje parametry zrychlení, rychlosti atd. je

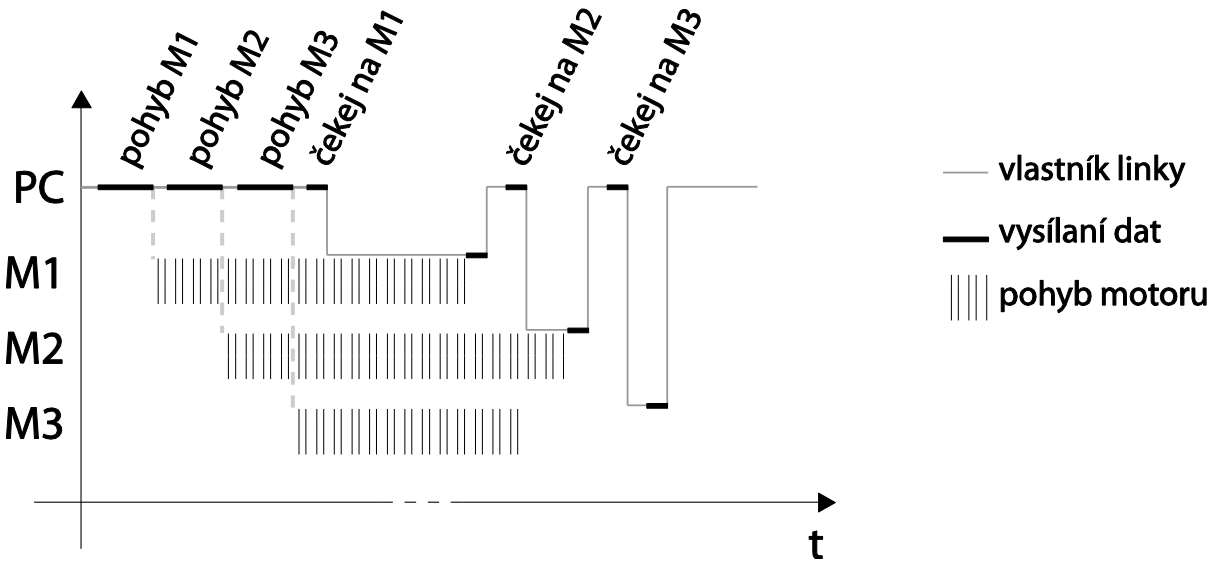
zbytečné čekat několik sekund, protože pokud řídící jednotka neodpoví obratem, tak pravděpodobně není připojena na sériové lince. Tato třída tento problém řeší tím způsobem, že po dobu existence její instance je timeout změněn na novou hodnotu. Využito je například u již zmiňované inicializace, kdy je timeout snížen na 500ms.

Simultánní pohyb motorů

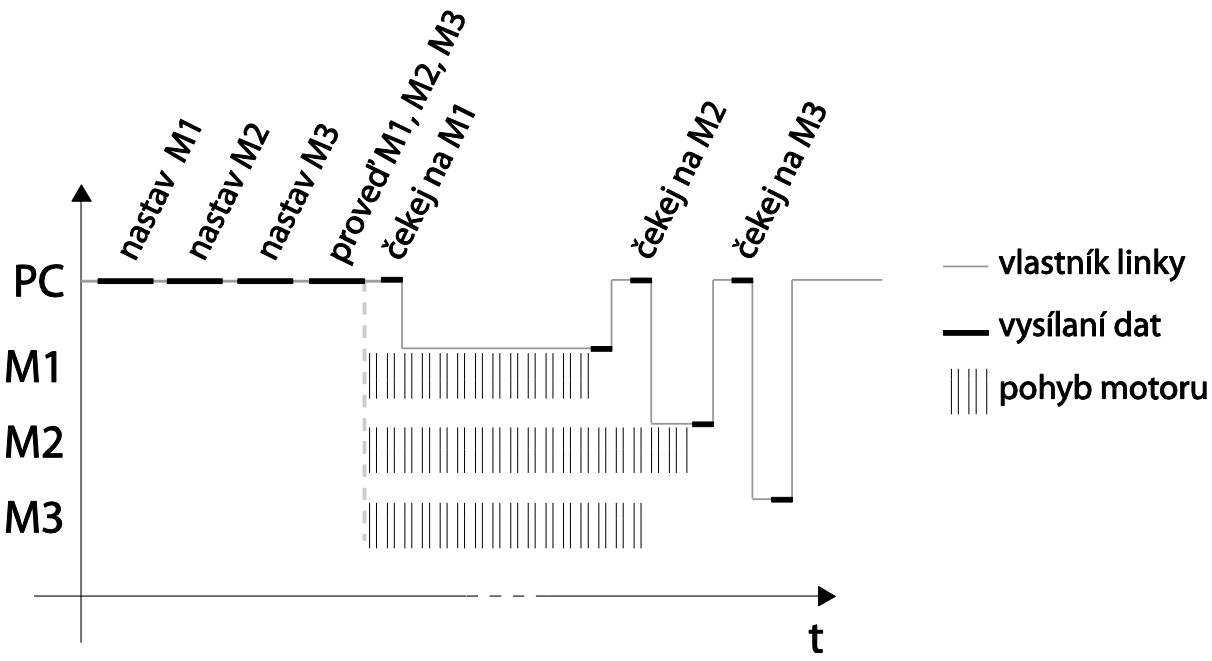
Při jednotlivém přístupu k řídícím jednotkám vzniká problém jak zajistit simultánní pohyb více motorů zároveň. K jeho zajištění bude třeba rozdělit fázi vlastního pohybu a fázi synchronizace do dvou povelových souborů. Povelový soubor pohybu lze přitom vytvořit různými přístupy.

Jako první je možnost nadefinovat pohyb postupně jednotlivým řídícím jednotkám pomocí individuálních povelových souborů. Poté vyslat další povelový soubor všem jednotkám obsahující pouze příkaz run, který spustí provádění pohybu na všech osách zároveň. Schéma přístupu k lince a vysílání zpráv pro tento typ řízení je zobrazen v obrázku 7. Výhodou tohoto řešení je, že se motory opravdu rozjedou současně. Na druhou stranu je třeba vždy nastavovat všechny řídící jednotky. Když se některý motor nemá pohybovat, tak je třeba mu to explicitně nadefinovat. V opačném případě by totiž při zavolání příkazu run provedl poslední známý pohyb.

Druhou možností je sloučit nastavení pohybu s příkazem run. Takto bude třeba komunikovat jen opravdu s těmi řídícími jednotkami, které mají konat pohyb. Začátek pohybu již ale nebude současný. Tato vlastnost je dobře vidět na digramu přístupu k lince na obrázku 6. Je zde znázorněn nejhorší možný případ pro tuto variantu – pohyb všech motorů zároveň. Na časové ose lze vidět, že pohyb motorů nezačíná v jednom čase. V praktickém provoze je však toto zpoždění nepostřehnutelné.



Obrázek 6: Schéma přístupu k lince při pseudosimultánním pohybu více motorů zároveň.



Obrázek 7: Schéma přístupu k lince při simultánním pohybu více motorů zároveň.

V návrhu potom bylo přistoupeno k použití druhé možnosti řízení. Důvodem k tomuto kroku je, že tiskárna se drtivou většinu času bude pohybovat pouze v jedné ose – tedy pomocí jednoho motoru. Použití více motorů zároveň je omezeno na případy inicializace hlavy či její umístění mimo model na konci tisku. V těchto případech je možné tolerovat, že ke spuštění motorů nedochází v jednom okamžiku. Vedlejším efektem tohoto jevu jsou menší nároky na zdroj proudu a s tím spojené nižší hodnoty vyzařovaného elektromagnetického šumu, protože proudové špičky motorů jsou rozloženy v čase.

Asynchronní volání

Z komplikovaného přístupu k sériové lince nepřímo vyplývá i další vlastnost celé knihovny. Tou je blokační volání všech funkcí, neboli funkce nevrátí zpět řízení, dokud není celá provedena. U funkcí typu inicializace, nastavení zrychlení a podobných to je vlastnost nepodstatná. Jsou zde však funkce nastavení polohy, kalibrace, jejichž efektem je pohyb motorů a následné čekání na dokončení tohoto pohybu. Doba provádění takovýchto funkcí je pak řádově delší oproti funkcím čistě virtuálním, které pouze odešlou několik málo bajtů přes sériovou linku. Při použití knihovny v aplikaci ovládané příkazovým řádkem je to chování vhodné. Problém nastává s aplikací s Grafickým Uživatelským Rozhraním (dále GUI). V takové aplikaci dochází ke zpracování podmětů od uživatele formou událostí, přičemž zpracovávání zablokuje běh celého GUI. Toto blokování je pak rychle rozpoznáno uživatelem, kterému nereagují prvky GUI na jeho podměty. S časovým zpožděním (v řádu sekund) je toto zablokování detekováno i operačním systémem, který, například v případě Windows, začne hledat řešení tohoto problému. Záhy je nabídnuto vyřešení pomocí ukončení aplikace. Uživatel pak

má oprávněný dojem, že aplikace se nekorektně sama ukončuje – lidově řečeno „padá“. Řešení tohoto problému bylo nalezeno v použití vláken. Aplikace má několik instancí provádění programu. Z toho je vždy aspoň jedna instance připravena obsluhovat události.

Pozn.: V případě wxWidgets je zde použito pouze jedno vlákno obsluhy GUI, protože při použití více vláken je třeba zajistit vzájemné vyloučení při manipulaci s GUI. To je však i samotnými tvůrci wxWidgets vřele nedoporučováno.

Stále však nesmí být prováděno více operací zároveň, protože by mohlo dojít k současnému použití sériové linky více jednotkami. Nejlepším řešením by bylo vytvoření vlákna, která by zajišťovalo komunikaci s tiskárnou. Tomuto vláknu by pak byly předávány příkazy z GUI. Bylo by zde samozřejmě třeba řešit asynchronní chybové stavy a také řídit velikost fronty příkazů.

Pozn.: Při použiti „slider“ a tažením jezdce uživatelem, je počet generovaných událostí v řádu desítek.

Dále by docházelo ke stavům, které nejsou až tak zřejmé. Například při vygenerování 10 událostí typu „posun o 1mm vpřed“, by poté došlo k 10 přískokům motoru o 1 mm. To vše místo posunu o 10mm které požadoval uživatel. Takových posloupností je velmi mnoho a stroj by se tak často choval sice korektně, ale z hlediska uživatele nepochopitelně. Tento problém byl vyřešen kompromisem. Knihovna jako taková neprovádí žádné operace asynchronně. Použití vláken je ponecháno na vyšších vrstvách.

Zbývá zde však operace kalibrace, která může být časově velmi náročná. Při kalibraci celého stroje dochází k postupné kalibraci jednotlivých os. Kalibrace jedné osy sestává z několika kroků:

- Aktivace omezovačů
- Nastavení maximální rychlosti na start/stop rychlost
- Posun touto rychlostí neomezeně daleko směrem dozadu.
- Registrace aktivace omezovače.
- Deaktivace omezovače
- Nastavení původní maximální rychlosti.
- Posun o hodnotu „odstup od omezovače“ vpřed.
- Vynulování čítače pozice v řídící jednotce a vynulování interního čítače pozice knihovny.

Časově nejnáročnější operací je zde posun start/stop rychlostí směrem vzad k omezovači. Experimentálně bylo zjištěno, že to může trvat až 10 sekund. Tento čas se může navíc výrazně měnit, neboť i rychlost start/stop je předmětem uživatelského nastavení. Celkově může jít i o desítky sekund.

Je zde třeba dát vyšším vrstvám řízení/uživateli možnost přerušit kalibraci. Toto řešení zároveň nesmí být nijak omezující pro použití knihovny. V knihovně vznikla pro tento účel bázová třída `ProgressUpdater_t`.

```
class ProgressUpdater_t {
public:
    ProgressUpdater_t():c(false) {};

    virtual void update(int , int = 100) {};
    virtual void done(int = 0, std::wstring message = L"") {};
    virtual void interrupionPoint() {};

protected:
    bool c;
};
```

Tato třída definuje kontrolní rutinu, zda se má kalibrace přerušit (`interrupionPoint`). Tato funkce by měla být volána vždy, když je možno korektně přerušit provádění. Pokud vznikne požadavek na toto přerušení, tak bude vytvořena výjimka, která bude zpracována ve volané funkci. Jako příjemný bonus může tento objekt obsahovat i rutiny pro aktualizaci postupu v kalibraci (`update`). Použití této třídy je však stále volitelné. Pokud funkci nebude předán ukazatel na tento typ třídy, tak se bude chovat jako obyčejná funkce. Použití této funkce a její zapojení do aplikace je detailněji rozebráno v kapitole 4.4 - Návrhu GUI.

Správa nastavení

Celá knihovna bude obsahovat velké množství nastavení, které ovlivní chod tiskového stroje. Pro komplexní správu všech těchto nastavení je použito třídy funkcí serialize knihovny Boost. Tyto funkce se vyznačují tím, že dokáží uložit stav objektu do sekvence bajtů. Objekty definující rozhraní knihovny mají tyto funkce nadefinované. Rekurzivně je potom volají na objekty jimi spravované. Vzniká tak stromová struktura, kterou lze vidět například při exportu do XML. Bezpečnost použití těchto funkcí je na vysoké úrovni. Jsou totiž definovány jako private s tím, že friend třídou je pouze třída archivace Boost. Tím je zaručeno, že je nelze zavolat z libovolného místa v kódu. [3]

Ne vždy je však třeba uložit/načíst celý stav objektu. Často je vhodné pouze změnit či získat rychlost pohybu na jedné z os. Pro tyto účely třída obsahuje veřejné (public) funkce getSpeed, setSpeed a analogicky pojmenované funkce pro práci s ostatními parametry. Jejich kompletní výčet se specifikací lze nalézt v programové dokumentaci, která je součástí příloh na DVD.

4.4 Grafické uživatelské rozhraní

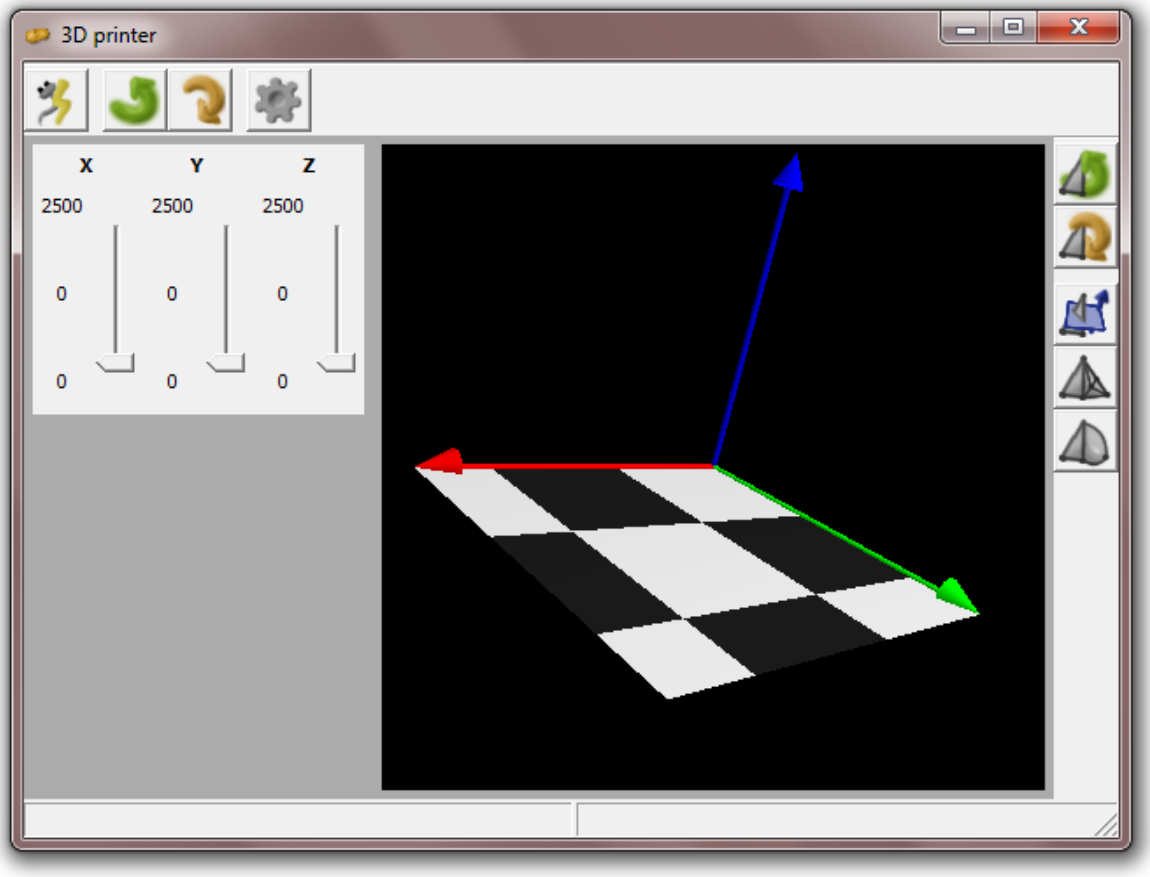
V úvodu této kapitoly jsou formulovány požadavky na GUI aplikace a funkčnost, kterou by měla aplikace poskytovat. V následujících podkapitolách je rozebírána realizace této funkčnosti. Zvláště pak asynchronního provádění kalibrace.

Grafické rozhraní by mělo být intuitivní, přehledné a přitom poskytovat veškeré požadované funkce. Aplikace by měla poskytovat informace o stavu tiskárny, tj. o aktuální poloze ve všech třech jejích osách, připojení, atd. Dále by zde mělo být okno sloužící k nastavování ne zcela běžných parametrů jako jazyka aplikace, parametrů sériové linky a dalších. Jedním z požadavků také bylo, aby aplikace dovolovala uložit všechna tato nastavení do souboru. Požadované funkce byly získány od uživatelů pomocí iteračního navrhování GUI.

Jako manažer vývoje a programátor v jedné osobě jsem návrh doplnil a obohatil o věci nevyslovené, ale očekávané. Ukládání stavu aplikace by mělo být generické a ukládaný soubor by měl být ve standardním formátu. Toho je možno dosáhnout použitím funkcí serialize knihovny Boost, která nabízí uložení a obnovení stavů objektů do několika typů souborů – binární, XML, atd. Mimo toto explicitní ukládání nastavení bude přítomna i implicitní práce s konfigurací. Myšlenka je taková, že při spuštění programu se aplikace pokusí načíst defaultní konfigurační soubor. Pokud se to nepovede, tak se spustí s defaultním nastavením. Při ukončení se analogicky pokusí uložit aktuální nastavení. Přitom je důležité, aby se dokázala zotavit z jakékoliv chyby, která přitom nastane bez interakce s uživatelem. Uživatel zde má stále možnost pracovat s konfiguračním nastavením manuálně.

Jelikož výsledné požadavky neobsahovaly velké množství snadno dosažitelných funkcí, tak je možno aplikaci postavit s výhledem na dotykové ovládání, tedy nejčastěji používané funkce ve formě velkých tlačítek. Tato tlačítka budou vyvedena v barevných piktogramech. Jejich návrhy byly testovány na uživatelích. Pokud uživatel alespoň rámcově nepoznal, co daný piktogram představuje, tak byl daný návrh piktogramu následně přepracován. Ve výsledné aplikaci se při najetí myší na tlačítko s takovým piktogramem navíc zobrazen textový popis. Jeden z raných návrhů hlavního okna lze vidět na obrázku 8. Výsledný vzhled lze potom vidět na obrázku 25 v závěru práce.

Dalším z požadavků je možnost lokalizovat aplikaci. Aplikace by nejen měla být schopna zobrazovat veškeré textové popisy v různých jazycích, ale měla by dodržovat odpovídající pravidla pravopisu. Jedná se například o zarovnání textu doprava při použití farsi (Perština). S Farsi souvisí i další požadavek na schopnost korektně zobrazovat znaky, které nepatří do nám známé abecedy. Překlady by měly být odděleny od vlastní aplikace. Jejich struktura by měla být standardní s ohledem k možnému použití aplikace pod operačním systémem Linux. V návrhu se potom počítá s tím, že i u spustitelné aplikace lze přidávat jazyky bez zásahu do kódu. Aplikace by také měla obsahovat okno, kde bude zobrazen načtený model tak, aby si ho mohl uživatel před tiskem vizuálně zkontrolovat.



Obrázek 8: Raný návrh GUI aplikace pro tisk na Tiskovém stroji FIT.

Po naprogramování GUI probíhalo dále testování aplikace na uživatelích, kdy se mimo ověření navržených řešení testovala i praktická očekávání uživatelů. Jedním z těchto očekávání byl například intuitivní pohyb 3D modelu pomocí myši/prstu. Toho bylo docíleno použitím knihovny trackball, která scénu simuluje virtuální koulí, kterou lze otáčet ve všech osách.

Z výsledného návrhu je možno zformulovat požadavky na GUI toolkit. Kromě již obligátní přenositelnosti to je podpora C++ knihoven. Dále pak možnost tvorby akcelerovaných OpenGL oken. Posledním požadavkem je licence, která musí být minimálně pro akademické použití zdarma. Těmto požadavkům vyhovuje velké množství nástrojů. Z důvodu konzistence programovacího jazyka a spolehlivosti bylo zvoleno použití wxWidgets.

Vytváření komplexního GUI textovou definicí vzhledu přímo v kódu je velmi neintuitivní a náročné na představivost a znalosti. Proto bylo k vytváření vzhledu použito programu wxFormBuilder. Tento open source projekt usnadňuje návrh formulářů s následným exportem do kódu v jazyce C++. [8] Jedná se o komplexní nástroj, který generuje kód kompatibilní s mnoha překladači. Dovoluje též definovat bázové třídy pro specifické třídy GUI, které je možné tímto způsobem obohatit o nadstandardní funkčnost. K vygenerovaným definicím vzhledu GUI lze potom navíc vygenerovat deklaraci zděděných tříd, do kterých poté stačí doplnit funkčnost. Celkově se jedná o velmi užitečný nástroj, který ušetřil mnoho času při vývoji. [8]

Internacionalizace – vícejazyčnost

K vícejazyčnosti bylo využito vestavěné podpory Internationalization (i18n) ve wxWidgets. Její realizace spočívá v tom, že celé GUI aplikace a všechny texty, které mají být lokalizované, jsou označeny makrem _(„text“). Toto makro je při překladu rozbaleno na funkci GetText, která se pokusí přeložit vnitřní text a vrací tento překlad. Základní texty jsou typicky psány v angličtině. Obecně však může být použit jakýkoliv jazyk, ale potom i z tohoto jazyka bude probíhat překlad do všech ostatních jazyků. Toto řešení je navíc velice robustní. Pokud nejsou provedeny žádné další kroky, tak aplikace bude korektně fungovat v tomto jediném jazyce.

Dalším krokem je extrakce makrem označených řetězců do souboru s příponou *po*. Tyto soubory je již možno předat překladatelům. K překládání slouží například velmi kvalitní a přesto opět open source program poEdit. V tomto programu se načte *po* soubor a provede se překlad. Rozhraní je intuitivní a celá práce s programem se podobá vytváření titulků k filmu/seriálu. Výsledek se uloží opět do *po* souboru a zároveň se vygeneruje *mo* soubor obsahující již pouze označené přeložené texty. Tento soubor je pak distribuován s aplikací.

V aplikaci je potom přidáno nastavení jazyka do konfiguračních parametrů, které jsou udržovány mezi jednotlivými běhy aplikace. Podle tohoto nastavení je potom při startu aplikace načten odpovídající *mo* soubor s překladem.

Lokalizace navíc nepracuje pouze na úrovni překladu textů. Aplikovány jsou také místní zvyklosti týkající se zarovnání textu či psaní desetinných čárek v číslech. Všechna tato opatření potom mají velký vliv na uživatele, kterému nepřipadá vše tak nové a neznámé.

Obrázek 9: Okno nastavení aplikace v Japonštině.

Toto řešení má jedinou nepříjemnou vlastnost. Při změně jazyka nedojde k aktualizaci již vytvořených oken a popisů. Bohužel ani programově nelze tuto situaci vyvolat. Známé řešení, které bylo pokusně implementováno, spočívá ve zrušení všech oken a jejich následné vytvoření. To však v uživatelích působí větší zmatek než neaplikování zvoleného jazyka. Vzhledem k těmto faktům a zároveň s přihlédnutím k faktu, že jazyk aplikace se bude měnit spíše výjimečně, bylo celé toto řešení

shledáno za použitelné. Výsledná aplikace byla kromě angličtiny přeložena do češtiny, litevštiny a částečné také do japonštiny. Zde se velmi hodí vlastnost, že nepřeložené texty zůstávají v angličtině. Japonština totiž přejímá mnoho anglických výrazů. Okno nastavení v japonštině je možno vidět na obrázku 9 výše.

Připojení a kalibrace

Připojením k tiskárně se v aplikaci míní otevření sériového portu a následná inicializace všech os. Inicializace zahrnuje nastavení parametrů rychlosti, akcelerace a dalších. Na závěr je přivedeno napětí na motory. Při této operaci je použito lokálního snížení timeoutu na 500ms. Nejvíce chyb totiž nastává právě zde, kdy je zvolen špatný port, baud rate, id řídící jednotky apod. Nebo se prostě zapomene připojit kabel tiskárny k počítači. Tato inicializace probíhá v obsluze události připojení. Tímto snížením je tak zajištěna rychlá odezva aplikace.

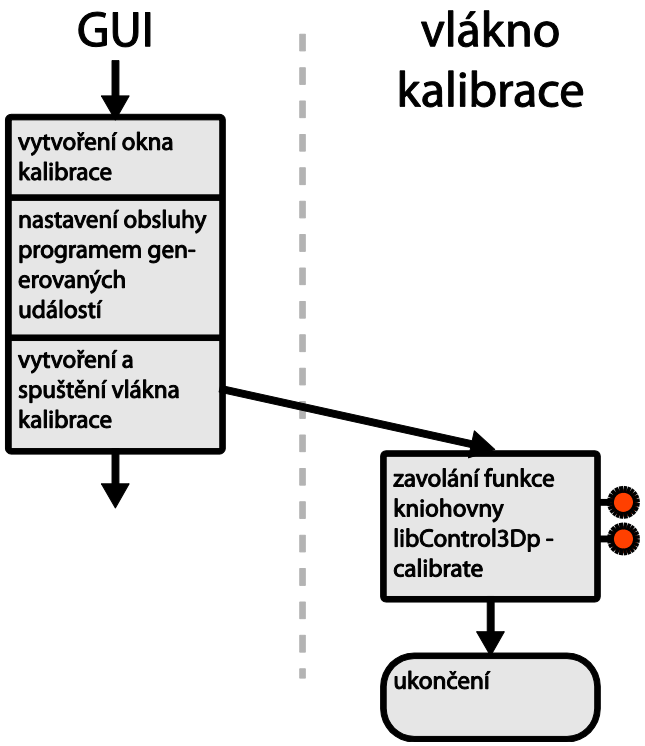
Kalibrace je již složitější. Funkčnost kalibrace byla rozebírána v kapitole 4.3 Návrh knihovny tiskového stroje – Asynchronní volání. V krátkosti se jedná o blokující funkci, která však definuje body v provádění programu, kdy ji lze přerušit.

Je zde tedy žádoucí, aby bylo vytvořeno zvláštní vlákno, ve kterém bude probíhat kalibrace. Původní vlákno mezitím může stále obsluhovat události. Tato funkčnost je však velmi základní a v dnešních aplikacích neomluvitelná. U výkonů, které trvají déle jak 3 sekundy, je třeba, aby uživatel měl rámcové informace o tom, v jaké fázi se provádění právě nachází a aby ho mohl kdykoliv přerušit. Tato funkčnost vyžaduje obousměrnou komunikaci mezi oběma vlákny. Jedním směrem bude vlákno kalibrace informovat vlákno GUI, v jaké fázi se proces kalibrace aktuálně nachází. Druhým směrem pak vlákno GUI informuje, že je požadováno ukončení. Tento druhý směr by za jistých podmínek bylo možné nahradit vynuceným ukončením vlákna kalibrace. Tento přístup by však mohl znamenat nekonzistenci ve stavu řídících jednotek vzhledem k jednotce PC (aplikaci). Využito je třídy `ProgressUpdater_t` , která definuje body přerušení, kde je zaručena konzistence. Použití této třídy spočívá v její specializaci, kdy jsou přepsány potřebné virtuální funkce. Pokud potom přijde požadavek na ukončení kalibrace (uživatel klepnul na tlačítko storno či křížek), tak je tomuto objektu nastaven příznak požadavku ukončení. Tímto zpracování celé události končí. Nyní musí dojít vlákno kalibrace do kontrolního bodu, kdy zjistí, že je požadováno ukončení. V tomto momentě je vytvořena výjimka, která ukončí provádění kódu.

Pozn.: Místo výjimek lze použít návratovou hodnotu této kontrolní funkce. Tato hodnota je zkontrolována a ukončí provádění zavoláním return pro návrat z funkce. V tomto případě je však po návratu třeba kontrolovat z jakého důvodu funkce skončila.

Komunikace od vlákna kalibrace ke GUI je zajištěna další funkcí `ProgressUpdater_t::update`. Tato funkce při zavolání vytvoří speciální uživatelskou událost, která je odeslána vláknu GUI. Zpracováním této události je potom aktualizace progress baru.

Graf běhu programu při vytváření tohoto vlákna lze vidět na obrázku 10. Nestandardní oranžové tečky znázorňují body přerušení provádění. Všimnout si lze i zaregistrování obsluhy uživatelských událostí, které předchází vlastnímu spuštění vlákna.



Obrázek 10: Graf běhu programu při zpracování události kalibrace.

Poslední záležitostí je indikace ukončení činnosti vlákna. Vlákna mohou být obecně dvou druhů:

- Oddělená (detached)
- Připojitelná (joinable)

U připojitelných vláken je třeba v jiném vláknu procesu vyzvednout návratový kód ukončeného vlákna. V případě oddělených vláken to není třeba, vlákno zanikne s ukončením. Zvolena byla varianta připojitelných vláken. Jak bude dále ukázáno, možné jsou obě varianty.

Přímo před vlastním ukončením provádění vlákna je vytvořena a odeslána událost informující o ukončení. Tato událost zároveň nese i kód ukončení podobně jako návratová hodnota funkcí. Zpracování této události je vyčkání na vlastní ukončení a následné zrušení modálního okna kalibrace, čímž je odblokováno ovládání aplikace a lze opět pracovat s tiskárnou. V případě vynuceného ukončení je postup analogický. Zvažována byla i implementace hlídání zda nedošlo k uváznutí vlákna.

Vzhledem k plné funkčnosti, která nejevila žádné známky chybného chování, bylo od tohoto v závěru upuštěno.

Tisk

Při stisku tlačítka Tisk dojde k zablokování funkcí spojených s ovládáním stroje a práce s modelem. Exkluzivní přístup k těmto funkcím získá aplikace. Ta bude provádět tři základní činnosti:

- Řezání modelu
- Řízení tiského stroje
- Řízení tiskové hlavy

Řezáním modelu se rozumí získání rastrového obrázku popisujícího vrstvu, která se vytiskne tiskovou hlavou. K získání tohoto rastrového obrázku dojde proložením roviny modelem. Průnikem těchto dvou těles je množina úseček. Rasterizací těchto úseček s následným vyplněním uzavřených oblastní se získá požadovaný řez modelem.

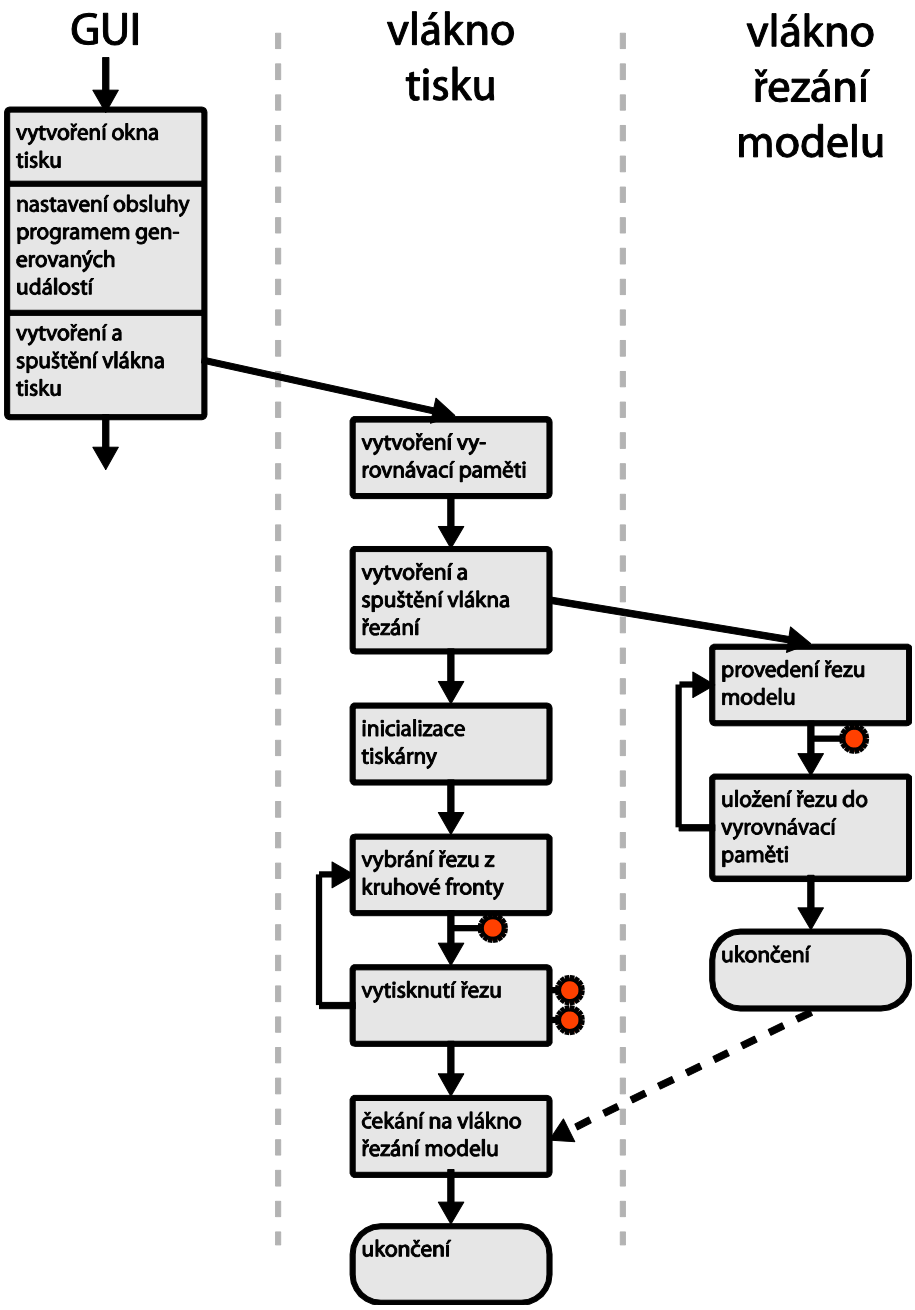
Řízení tiskového stroje probíhá v závislosti na mnoha parametrech. Vždy jde ale o pásový posun po dvourozměrné ploše tisknutého objektu (vrstvě). Tento pohyb je synchronizován s tiskovou hlavou, která v jeho průběhu pokládá tiskový materiál. Po každém vytisknutí celé vrstvy dojde ke zvýšení polohy tiskové hlavy nad modelem o tloušťku právě vytisknuté vrstvy.

Tisková hlava je v průběhu tisku plněna daty z řezů modelu, které indikují, kde má být vytištěn materiál. Její parametry jako maximální tloušťka tištěného řádku, množství pokládaného materiálu, rychlost pokládání a další ovlivňující pohyb musí být brány v potaz při řízení pohybu stroje. Jelikož práce na tiskové hlavě byly v experimentálním stádiu, tak v aplikaci je vytvořena pouze bázová třída reprezentující tiskovou hlavu. Tato třída má své metody značené identifikátorem virtual. Při specializaci této třídy na konkrétní tiskovou hlavu se předpokládá předefinování těchto metod. Díky označení virtual dojde při volání těchto metod k aplikaci pozdní vazby a budou zavolány metody specializované třídy (tedy ne bázové). Celá tato třída je umístěna do zvláštního projektu libCartridge. V této třídě jsou nadefinovány předpokládané funkce ve formě callbacků (zpětných volání):

- `onStart`
- `onEnd`
- `onLayerStart`
- `onLayerEnd`
- `onStripStart`
- `onStripEnd`

Tento seznam si neklade za cíl být úplný. Při vlastní implementaci tiskové hlavy ho bude třeba dále rozšiřovat. Cílem této třídy je spíše ukázka jejího zakomponování do celé aplikace. Tato ukázka by měla usnadnit následnou implementaci člověku neznalého detailního chodu aplikace.

Úkolem aplikace je na základě všech parametrů řídit všechny výše zmíněné činnosti a zajistit jejich synchronizované provádění. Tisk bude časově náročný. Doba tisku jednoho modelu sice velmi záleží na jeho velikosti a požadované kvalitě, ale obecně se jedná minimálně o minuty, častěji však o hodiny. Z tohoto důvodu je zde třeba zpětná vazba k uživateli, aby věděl jak daleko je proces tisku. V průběhu tisku nebude moci uživatel nijak zasahovat. Jediné co bude moci provést je zastavení tisku. Jelikož nejsou známy přesné parametry tiskové hlavy, zvláště pak její kapacita tiskového materiálu, tak byla ještě přidána funkce pozastavení tisku. Idea je, že v případě nízkého stavu zásob tiskového materiálu uživatel (či sama aplikace) pozastaví tisk, čímž se zastaví tisková hlava (pravděpodobně i se zásobníkem materiálu). Následně uživatel doplní materiál a opět pustí tisk, který naváže přesně v místě přerušení.



Obrázek 11: Diagram běhu programu při obsluze události tisku.

Z důvodu zajištění promptních reakcí GUI zde bylo přistoupeno k použití vláken stejně jako v případě kalibrace. Tím však veškerá podobnost končí. Kalibrace byla zajišťována knihovní funkcí Calibrate, která již měla přednastavené body přerušení a aktualizace stavu. V případě tisku je řízení ponecháno na aplikaci. Řízení pohybu je totiž velmi provázané s ostatními parametry tisku a jeho integrování do knihovny by nebylo vhodné. Celé vlákno tisku je v programu realizováno jednou třídou, která svými veřejnými (public) metodami poskytuje rozhraní pro komunikaci s vlákny. Kromě metod pro spuštění běhu vlákna a vyzvednutí návratového kódu po jeho ukončení se jedná o `interrupt`, `pause` a `isPaused`. Komunikace směrem od vlákna tisku a bufferu je opět zajištěna zasíláním zpráv. Pro operaci řezání modulu bylo vytvořeno speciální vlákno. Komunikace mezi

vláknem tisku a vláknem řezání modelu je zajištěna pomocí kruhového bufferu. Celkový diagram běhu programu při tisku lze vidět na obrázku 11. Tak jako u diagramu běhu kalibrace jsou i zde použity nestandardní oranžové tečky značí pozice bodů přerušení v programu. V těchto bodech je možné přerušit/pozastavit provádění programu. Jejich rozmístění v kódu je definováno snahou o minimalizaci maximálního času mezi jejich voláním.

Vlákno GUI

Schéma začíná příchodem události na tisk modelu ve vlákně GUI. Ve zpracování této události je vytvořeno okno tisku. Dále jsou zaregistrovány obsluhy speciálních událostí od vlákna tisku a bufferu. Těmito událostmi je potom GUI informováno o průběhu tisku, respektive o stavu bufferu. Následně je vytvořeno a spuštěno vlákno tisku. Tímto končí obsluha události tisku a mohou být zpracovávány další události. Vytvořené okno informující o stavu tisku modelu je modální. Tím je zabráněno aktivaci jakékoliv funkce z hlavního menu, která by kolidovala s probíhajícím tiskem.

Vlákno tisku

Vlákno tisku začíná vytvořením vyrovnávací paměti – bufferu. Jeho velikost je definována v nastavení (konkrétně se jedná o nabídku Nastavení – Tisk - Obecné). Tento buffer slouží k přenosu řezů z vlákna řezání modelu do vlákna tisku. Ve třídě `Buffer_t` byl využit objekt `circular_buffer` z knihovny Boost. Pro správu přístupu k objektu mezi vlákny bylo opět použito řešení z knihovny Boost – `boost::mutex` a `boost::condition`. Použití těchto objektů je analogické s knihovními funkcemi pthread. Navíc poskytují různé užitečné objekty jako například `scoped_lock(mutex)`, pro uzamčení mutexu po dobu platnosti dané instance objektu. Funkčnost je shodná s objektem `ScopedTimeout_t` ze třídy sériového portu probírané výše v kapitole 4.3 Řízení tiskového stroje – timeout. [3] Tento objekt má také vnitřní (private) funkce pro zasílání událostí GUI o jeho stavu. Zpracování těchto událostí je aktualizace progressbaru simulujícího plnost vyrovnávací paměti. Při vytváření této třídy byla využita technika šablonového programování. Specifikovat je třeba třídu, která bude položkou bufferu.

Následuje vytvoření a spuštění vlákna řezání modelu, které ihned po svém spuštění začne plnit buffer. Mezitím se ve vlákně tisku inicializuje pracovní prostor tiskárny, šířka tisknutých pruhů, tloušťka tisknuté vrstvy apod.

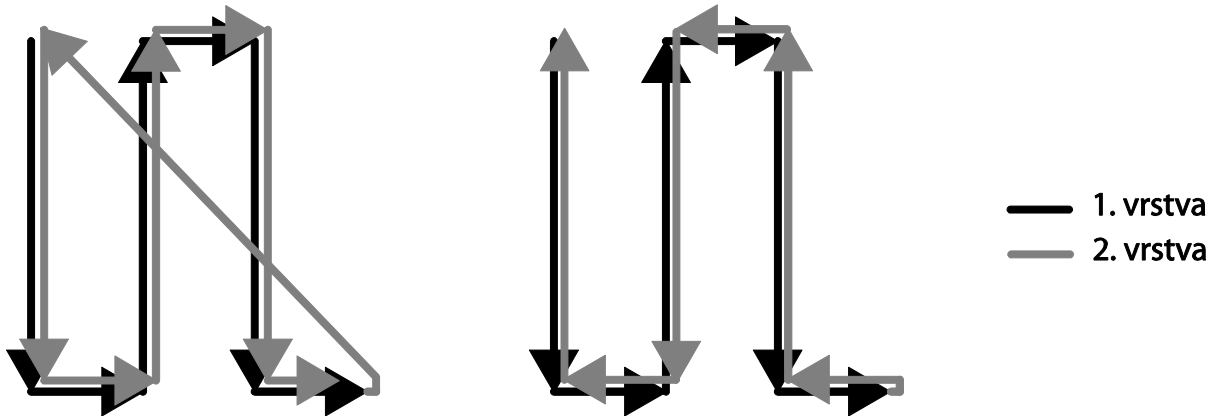
Po spuštění vlákna řezání modelů dojde k inicializaci tiskárny, tiskové hlavy a dále začne vlastní tisk. Ten obecně probíhá vytištěním jedné vrstvy, posunutím modelu o tloušťku právě vytištěné vrstvy níže a tak stále dokola dokud není vytištěna poslední vrstva. Poté dojde k odjetí tiskové hlavy do počátku souřadnicového systému tak, aby co nejméně zavazela při vyjímání vytištěného modelu. V průběhu tisku jsou volány jednotlivé CB na tiskovou hlavu.

Dráha, po které se bude tisková hlava pojíždět je komplexní záležitostí, na kterou lze pohlížet z mnoha úhlů pohledu. Může se jednat o:

- minimalizaci uražené vzdálenosti
- maximalizaci doby schnutí materiálu
- minimalizaci počtu tisknutých pásů
- minimalizace počtu tisknutých vrstev
- a další

Minimalizace počtu tisknutých pásů/vrstev je spíše software záležitostí. Bylo by třeba zjistit, zda je technicky vůbec možné model vytisknout otočený. Tento druh optimalizace by výrazně zrychlil tisk plochých modelů postavených na výšku. Jeho aplikace je nezávislá od dalších optimalizací.

Maximalizace doby schnutí materiálu je z programovacího hlediska velmi jednoduchá. Pohyb by byl prováděn z počátku souřadnicového systému po celé ploše vrstvy. Následná vrstva by se tiskla stejnou dráhou, tedy opět z počátku. Dráhu hlavy lze vidět na obrázku 12 níže. Minimalizace uražené vzdálenosti předpokládá, že nejvíce času je ztraceno při vlastním pohybu. Doba schnutí materiálu zde není vůbec brána v úvahu. V takovém případě tisk z počátku probíhá stejně jako v předchozím případě. Při posunu na další vrstvu se však hlava vrací úplně stejnou cestou zpátky do počátku. Tento cyklus se opakuje s periodou dvou vrstev. Srovnání posledních dvou přístupů lze vidět na obrázku 12. Implementována byla varianta minimalizace uražené dráhy.



Obrázek 12: Srovnání dráhy tiskové hlavy při maximalizaci doby schnutí materiálu (vlevo) a minimalizaci uražené dráhy (vpravo).

Po vytisknutí celého modelu je vyzvednuta návratová hodnota vlákna řezání modelu. Následně je zaslána tato informace ve formě události vláknu GUI a provádění vlákna je ukončeno. GUI v reakci na událost ukončení tisku vyzvedne návratový kód vlákna tisku a oznámí tuto radostnou událost uživateli zablikáním aplikace v panelu úloh.

Pokud dojde k požadavku na předčasné ukončení tisku, tak vlákno tisku ukončí tisk. Vyšle požadavek na ukončení vlákna řezání modelu. Pokud je buffer plný, tak z něho vyjme i jednu položku, protože je možné, že vlákno řezání je zrovna zablokované ve funkci vkládání do bufferu a

nemuselo by se o tomto požadavku dozvědět. Následně je vyzvednut návratový kód vlákna tisku. Dále je zaslána zpráva o ukončení tisku, jejíž součástí je i kód identifikující důvod ukončení, případně i slovní popis, který je připojován, pokud se jedná o chybu.

Vlákno řezání modelu

Činnost tohoto vlákna velmi jednoduchá. Ve smyčce provádí řezání modelu a výsledek vkládá do vyrovnávacího bufferu. Toto vlákno nijak neinformuje vlákno GUI o svém průběhu. Tato činnost logicky přísluší až třídě bufferu. Pokud je buffer plný, tak dojde k zablokování běhu vlákna na funkci vkládání do bufferu. Po provedení posledního řezu je vlákno ukončeno. V systému však existuje, dokud není ukončen i tisk zbývajících řezů v bufferu a vyzvednut návratový kód. V případě požadavku na předčasné ukončení je přerušena smyčka řezání a vlákno se ukončí.

5 Implementace

Navržené části byly implementovány v jazyce C++. Jelikož se s programováním začínalo takříkajíc na zelené louce, tak bylo třeba definovat jasná pravidla psaní kódu. Místo vymýšlení vyzkoušeného bylo přistoupeno k použití „Google C++ Style Guide“. Revidována byla pouze doporučení ohledně používání výjimek, které se z historických důvodů v Google nepoužívají. Dále byla revidována pravidla, jejichž účel lze efektivněji splnit použitím knihovních funkcí Boost. [1]

Při vývoji bylo použito také nemalé množství knihoven třetích stran. Knihovny použité při vývoji musí splňovat několik základních požadavků. Musí mít licenci, která dovoluje jejich použití pro akademické použití. Jejich použití nesmí snižovat funkčnost celku. Knihovna opět musí být přenositelná minimálně mezi platformami Windows a Linux. A pokud možno by součástí knihovny měla být kvalitní dokumentace v češtině, angličtině či alespoň v litevštině.

5.1 Použité knihovny

Boost

První knihovnou je Boost. Jedná se o velmi komplexní knihovnu, která v jednotlivých částech aspiruje na zařazení do formovaného standartu C++0x. Tak jako standart C++ je i knihovna multiplatformní. Celá knihovna masivně využívá šablon. Z toho vyplývá i její struktura, kdy většina funkcionality je obsažena v hlavičkových souborech. [3]

Z této knihovny jsou využity definice celočíselných typů (boost/integer.hpp). Tyto typy jsou využity hlavně v knihovně libControl3D. [3]

Dále je využito serializace (boost/serialization). Serializace v tomto podání znamená dekompozici stavu objektu do sekvence bytů. Zpětně lze z této sekvence obnovit původní stav objektu. Tato funkce je využita napříč celým programem. [3]

Pro definici výjimek je použito boost/exceptions. Výjimky v podání knihovny Boost poskytují bezpečné objekty pro tvorbu těchto výjimek s programově specifikovatelnými datovými typy, které se k nim vážou. Tyto vázané proměnné lze navíc při zpracování aktualizovat. Toho je využito k tvorbě strukturovaného popisu chyb, které způsobily výjimku. Použity jsou v libControl3D a libModel3D.

Poslední funkčností převzatou z knihovny Boost je část zvaná asio, která poskytuje asynchronní síťovou komunikaci. V případě libControl3D se konkrétně jedná o přístup k sériovému rozhraní nezávislý na platformě. [3]

VCG

Knihovna VCG je C++ knihovna pro manipulaci a zpracování trojúhelníkových a čtvercových sítí. Je poskytována pod licencí GPL. Tato knihovna je použita v libModel3D. Opět se jedná o C++ knihovnu využívající šablon. Její nevýhodou, která se ukázala až v pozdějších fázích projektu je, že část příslušné dokumentace je dostupná pouze v italštině. Tato nectnost nebyla zpočátku pozorována, protože základní části jsou v angličtině. Až při hlubším zkoumání bylo naraženo na italštinu. [7]

Z této knihovny je vyžito třídy pro popis modelu pomocí trojúhelníkové sítě. Pro import a export těchto tříd potom statických funkcí, které poskytují požadované procesy. Rozhodující je však použití několika funkcí pro manipulaci s vlastními modely. Konkrétně se jedná o funkce pro průnik objektu s rovinou, dělení trojúhelníků a vyhlazování trojúhelníků. [7]

wxWidgets

Mezi knihovny se řadí také toolkit wxWidgets pro tvorbu GUI univerzálního mezi platformami. Opět se jedná o jazyk C++. Licence je sice „wxWidgets licence“, ale požadovanou volnost poskytuje. [4]

Při implementaci navrženého GUI bylo využito open source aplikace wxFormBuilder. Jedná se volnou aplikaci, kdy se pomocí GUI specifikuje vzhled jednotlivých oken a dialogů. Následně jsou automaticky vygenerovány C++ třídy, které je popisují pomocí tříd wxWidgets. Z těchto tříd jsou následně odvozeny vlastní třídy, které jsou doplněny o funkčnost. [4]

trackball

V tomto případě se nejedná o ucelenou knihovnu. Je to jeden zdrojový soubor s odpovídajícím hlavičkovým souborem. Vytvořen byl společností Silicon Graphics, Inc. Šíření je povoleno pod podmínkou, že přiložený copyright bude šířen spolu s kódem.

Tato knihovna implementuje virtuální trackball. Použití je v navigaci ve scéně s virtuální tiskárnou a modelem.

5.2 Knihovna řízení tiskového stroje

Tato knihovna (libControl3D) je jádrem této práce. Knihovna poskytuje rozhraní pro řízení tiskového stroje pomocí intuitivních příkazů. Tyto příkazy jsou realizovány jako volání metod nad objektem realizujícím pracovní prostor tiskárny. Ze zadání vyplývá, že tříosá tiskárna se pohybuje ve třírozměrném prostoru. Toto omezení je implementováno obecně a knihovnu lze použít obecně v n-rozměrném prostoru.

Obecný návrh nijak výrazně nezkomplikuje použití knihovny, ale jako bonus umožní nativní ovládání tiskárny například pouze ve dvou osách. Což je de-facto degradace na obyčejnou tiskárnu. V budoucnu pak lze ovládání rozšířit na čtyři a více os.

Zpracování chybových stavů probíhá pomocí výjimek. Výjimky mají strukturovaný popis. V praxi to znamená, že pokud je volána funkce setAbs pro nastavení absolutní polohy hlavy, tak tato volá funkci setAbs na jednotlivých osách. Tyto funkce zapíšou odpovídající příkaz na sériovou linku. Pokud nastane výjimka (např.: „Sériový port uzavřen“), tak je zachycena v každé funkci a jsou k ní přidány další textové informace popisující podmínky, které k ní vedly. Pokud výjimka nebude zachycena, tak její textový popis by pak mohl vypadat následovně:

```
Nastala vážná chyba.
Nastavení hlavy na souřadnice [1,1,1] se nezdařilo.
Nastavení motoru na souřadnici 1256 kroků se nezdařilo
Nezdařilo se vyslání příkazu „f1256“ na port „COM1“.
```

Tento přístup umožňuje kromě pochopitelného popisu chyby také možnosti pro zotavení z takovýchto chyb. Pokud například do aplikace přijde takováto výjimka tak je možné otevřít okno s nastavením sériového portu a informovat uživatele, že je něco v nepořádku se sériovou linkou. Samozřejmě tento přístup nebrání tradičnímu zpracování v podobě chybového okna a ukončení aplikace.

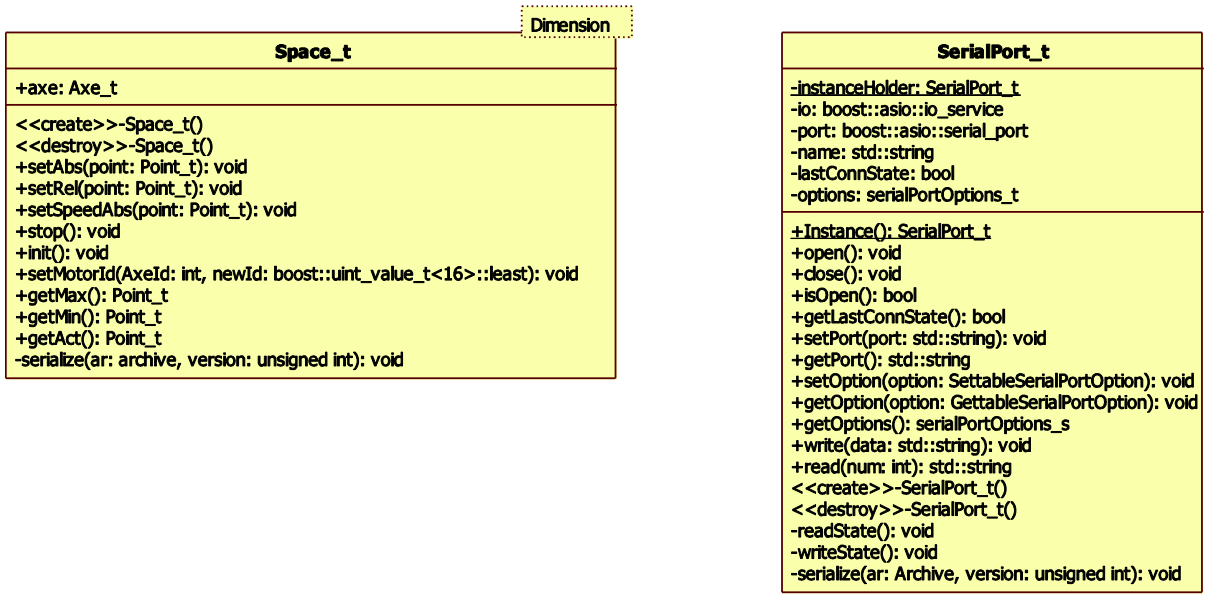
Další funkci, kterou knihovna zabezpečuje je přepočet jednotek. Ovládání motoru totiž probíhá v krocích či mikrokrocích. Při posunu polohy o hodnotu 1 je ale očekávané posunutí motoru o 1mm. Jeden krok to rozhodně není. K tomuto přepočtu je ale nutné znát poměr mezi milimetry a mikrokroky. Tento poměr je třeba ručně zadat pro každou osu pohybu. Pokud není zadán, tak je přepočet nastaven na defaultní hodnotu, která je vhodná pro tiskový stroj na FIT.

Rozhraní

Návrh rozhraní je definován logickou vrstvou pod knihovnou a nad knihovnou. Pod knihovnou se jedná o komunikační protokol, kterým se komunikuje s řídicími jednotkami jednotlivých motorů. Komunikace s řídicí jednotkou probíhá po sériové lince ve znakovém komunikačním protokolu definovaném výrobcem Microcon.[10] Jeho kompletní dokumentaci je možné nalézt v dokumentu „manual_kontroleru.pdf“, který je součástí příloh. Specialitou tohoto řešení je, že všechny kontroléry jsou připojeny na společnou sériovou sběrnici RS232. Tato sběrnice neposkytuje žádné řízení přístupu k médiu. Při komunikaci je tedy třeba dodržovat pravidlo exkluzivního přístupu, aby nedošlo ke kolizi. K tomu je využito modelu master-slave. Kdy *master* jednotka řídí celou komunikaci a řízením komunikace zajišťuje bez kolizní přístup k médiu. *Master* jednotkou je tato knihovna, respektive počítač. Řízenými *slave* zařízeními jsou řídicí jednotky motorů. Podrobněji je tento přístup rozebírán v kapitole 4.3 – Knihovna řízení tiskového stroje.

Nad knihovnou je definice rozhraní provedena s ohledem na intuitivnost, snadnost a očekávanost použití z vyšší vrstvy. Je zde definován objekt sériového portu, který slouží k nastavení parametrů sériové komunikace jako identifikátor portu, komunikační rychlost, počet stop bitů a další. Zde je třeba zaručit, že tento objekt bude v aplikaci maximálně v jedné instanci. Více instancí by totiž vedlo k chybám při přístupu k sériové lince. Tohoto požadavku je docíleno aplikací návrhového vzoru singleton.[12] Další částí rozhraní je objekt prostoru. Tento objekt umožňuje vytvoření obecně *n* rozměrného prostoru, v němž je pohyb v každém rozměru definován jedním motorem. Objektů prostoru může být obecně neomezené množství. Je však třeba myslet na to, že knihovna není thread-safe a tedy přístup k více takovýmto objektům z více vláken zároveň může vést k chybě. Modely tříd definujících rozhraní vyšších vrstev jsou uvedeny na obrázku 13 níže.

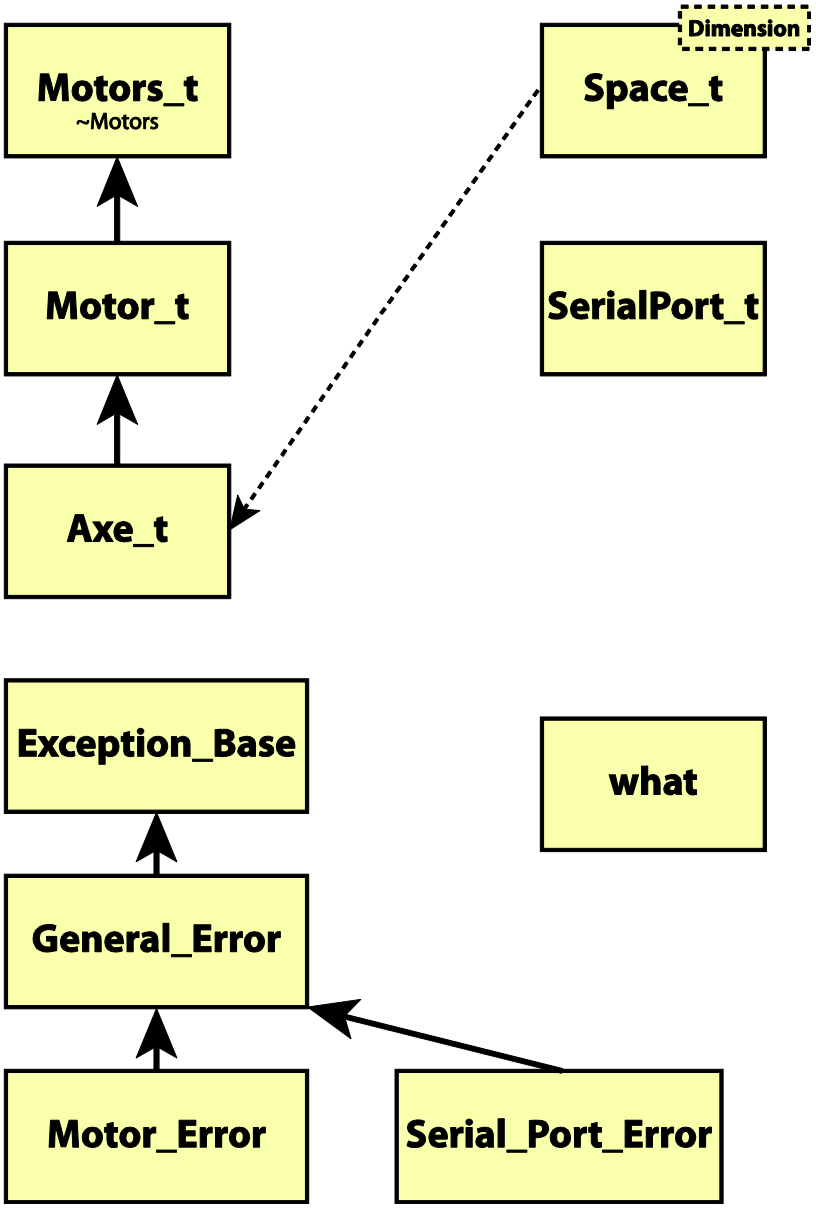
Účelem proměnné Dimension specifikující třídu `Space_t`, je definice počtu ovladatelných os (tedy motorů či stupňů volnosti) tiskového stroje. V případě použití knihovny pro tisk třírozměrných objektů se jako nejlogičtější jeví vytvoření jednoho třírozměrného prostoru, ve kterém se pohybuje tisková hlava. Moderní tiskové hlavy však mají tendenci zrychlovat tisk za simultánního použití mnoha tiskových trysek najednou. Tímto dochází při tisku malých předmětů k nadbytečnosti jednoho stupně volnosti pohybu hlavy, pokud tisková hlava pokryje svoji šířkou tisku celý jeden rozměr tisknutého objektu. Přístup s uživatelskou definicí počtu stupňů volnosti tiskového stroje umožňuje tento problém řešit na vyšší úrovni – úrovni, kde jsou již k dispozici informace o tiskové hlavě. Zároveň však neomezuje tradiční použití s pohybem ve všech osách.



Obrázek 13: Rozhraní knihovny libControl3D.

Model

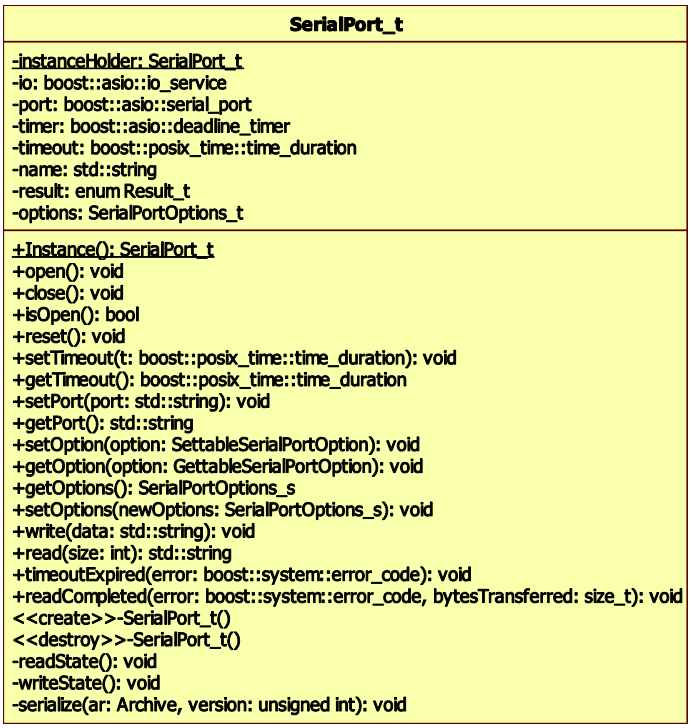
Celkový objektový model lze vidět na obrázku 14. Pro přehlednost se jedná pouze o zjednodušený model, který neobsahuje popis tříd, jejich synonyma a další ne až tak podstatné informace. Skládá se ze skupiny tříd definující výjimky (spodní polovina diagramu). Dále potom z třídy definující jednu osu volnosti a obecnějších tříd, ze kterých je odvozena. Nechybí zde ani výše zmíněné třídy rozhraní s vyšší vrstvou řízení. Co vidět není, jsou třídy vnořené, jako například třída `ScopedTimeout_t`, která je součástí třídy SerialPort_t. Jejich forvard deklarace jsou však vidět v detailnějších obrázcích, které jsou součástmi jednotlivých kapitol. Následující kapitoly se zabývají jednotlivými třídami a stručným popisem jejich funkcionality. U každé z těchto podkapitol je kompletní diagram třídy navržený v programu StarUML.



Obrázek 14: Zjednodušený objektový model knihovny libControl3D.

SerialPort_t

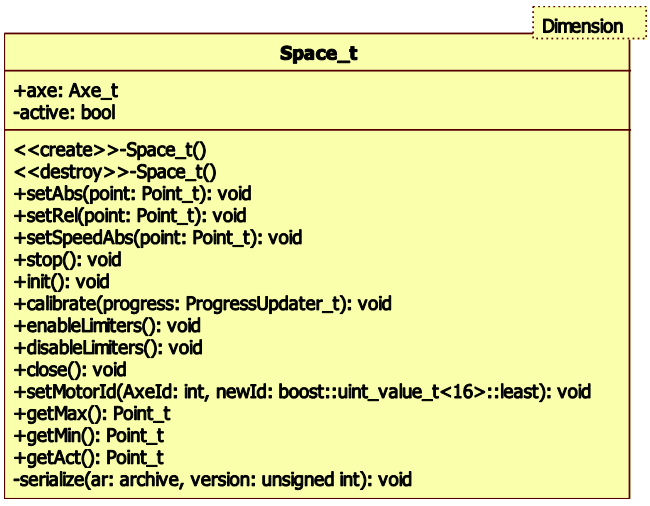
Třída `SerialPort_t` slouží k práci se sériovým portem a jeho nastavováním. Celá tato třída rozšiřuje třídu `serial_port` knihovny Boost pro práci se sériovým portem. Doplněny jsou zde metody pro serializaci a deserializaci objektu. Co nelze vidět v objektovém modelu jsou statické funkce přetěžující operátory << a >> pro intuitivní práci K použití s těmito metodami je zde vytvořen i objekt definující konec řádku. K tomuto řešení bylo nutné přistoupit z důvodu požadavků komunikačního protokolu řídicí jednotky. Tento protokol vyžaduje, aby jednotlivé příkazy byly ukončeny speciálním znakem 0x0D. Při příchodu objektu označujícího konec příkazu je tento objekt nahrazen tímto znakem. Díky tomuto přístupu není nutné tento znak kopírovat za každý příkaz a tím se vyhnout použití makra ho definující (Makra obecně jsou v jazyce C++ nedoporučována).



Obrázek 15: Objektový model třídy SerialPort_t.

Dále tato třída obsahuje vnořenou třídu ScopedTimeout_t jež slouží k lokálnímu předefinování velikosti timeout. Její účel je podrobněji rozebrán v kapitole 4.3 Knihovna řízení tiskového stroje – timeout.

Tato třída dále obsahuje funkce save a load potřebné pro funkčnost serializace knihovny boost. Použití této funkčnosti je čistě volitelné. V plné míře ji lze nahradit pomocí ostatních veřejných funkcí.



Obrázek 16: Model třídy Space_t.

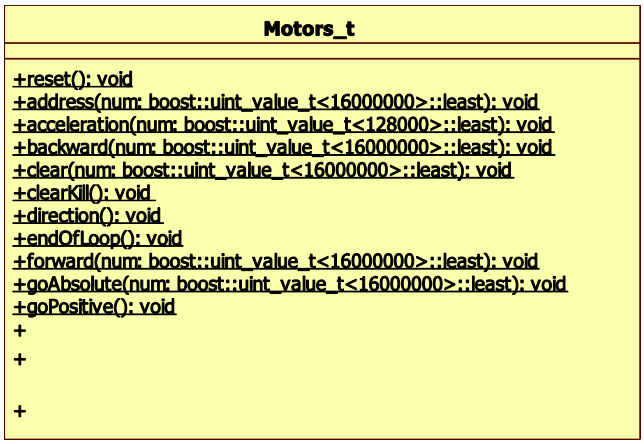
Space_t

Třída `Space_t` je jádrem této knihovny. Obsahuje obecný počet *n* objektů os definovaný šablonou. Pro každou osu je vytvořena a držena instance třídy `Axe_t`.

Stojí tu za zmínku diskuze nad definicí přístupnosti tohoto objektu. V rámci zapouzdření objektů by přístup k jednotlivým osám neměl být povolen a všechny instance by měli být deklarovány jako private. Na druhou stranu je zde požadavek pro přístup k mnoha metodám os, které nemají žádnou spojitost s prostorem (`Space_t`), ve kterém se osy nacházejí. Jedná se například o nastavení identifikátoru motoru, který je svázán s osou. Možným řešením může být použití atributu protected. To by ale vylučovalo přímé použití této třídy ve složitějších aplikacích. Nakonec byl povolen přístup k těmto osám (identifikátorem public) a ostatní metody třídy `Space_t` byly upraveny, aby s touto možností počítaly.

Motors_t

Tato třída slouží jako přechod mezi znakovými příkazy řídicí jednotky a funkčním programováním v C++. Jedná se o třídu, která je držitelem statických funkcí, kde každá z metod implementuje jeden příkaz. Seznam příkazů (a tedy i funkcí) byl vytvořen dle manuálu k řídicí jednotce CD30x. Tento seznam lze také nalézt v programové dokumentaci vygenerované z komentářů v hlavičkových souborech. Každá funkce navíc provádí kontrolu přijatých parametrů, zda je povolena jejich hodnota. Jako synonymum bylo vytvořeno i logičtější pojmenování `Motors`.



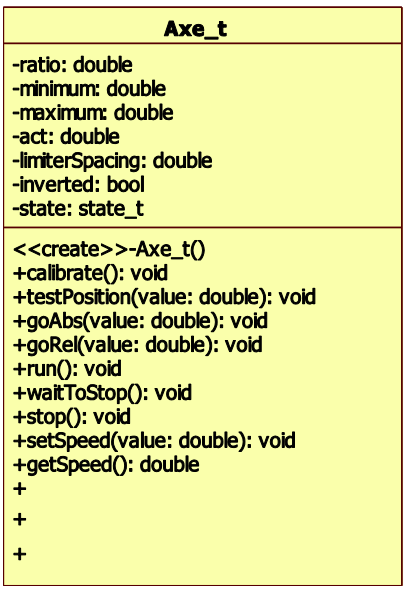
Obrázek 17: Zkrácený model třídy Motors_t.

Motor_t

Třída `Motor_t` má velmi podobný název se třídou `Motors_t`. `Motor_t` je odvozena od této třídy a navíc přiřazuje identifikátor motoru. Jedná se tedy pouze o zapouzdření všech statických metod do jednoho objektu. Objektový model zde proto chybí. Již se jedná o konkrétní objekt simulující motor.

Axe_t

Osa prostoru je dalším specifikací motoru. Na této úrovní se již motor řídí nastavováním polohy na ose v rozumných jednotkách - mm. Třída obsahuje i vnitřní strukturu definující aktuální nastavení motoru, která obsahuje údaje o jeho zrychlení, rychlosti pohybu, počtu mikrokroků a další, které specifikují, jak se motor bude chovat. Při inicializaci jsou všechny tyto údaje



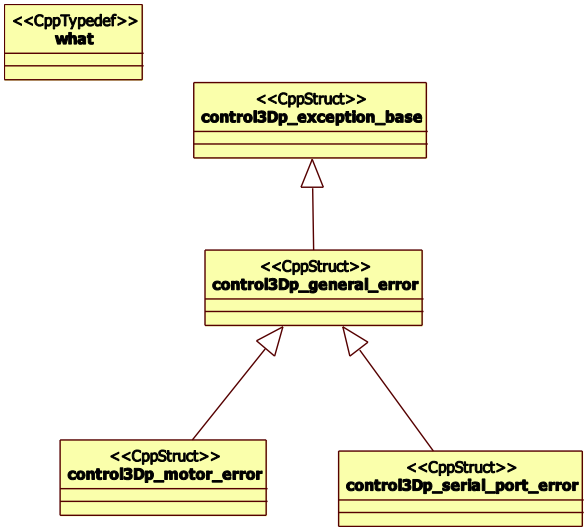
Obrázek 18: Zkrácený model třídy Axe_t.

poslány do řídicí jednotky. Ta se již dále chová jako stavový stroj, takže do jejich další změny není třeba je znovu posílat.

Součástí třídy jsou samozřejmě i metody pro serializaci a obnovení stavu objektu. Po obnovení stavu objektu je třeba znovu volat inicializaci motoru. Takové chování bylo zvoleno, protože ne vždy je tiskový stroj programu dostupný.

Výjimky exception_base, general_error, motor_error, serial_port_error

Strukturu dědičnosti těchto objektů lze nalézt na obrázku 17 vpravo. Základní typ výjimky – `control3D_exception_base` je odvozen od obecného typu výjimky `boost::exception` definovaného knihovnou Boost a zároveň i od `std::exception` z knihovny STL. Pro popis výjimky se používá objekt what se syntaxí danou knihovnou Boost. Veškeré tyto výjimky a objekt je popisující jsou tímto korektně definovány, takže při jejich vytváření a zpracování nemůže dojít k výskytu výjimky.



Obrázek 19: Objektový model výjimek.

Realizace

Při realizaci byly vytvořeny zdrojové a hlavičkové soubory pomocí generátoru C++ kódu v programu StarUML. Tento program byl také použit pro vytvoření digramu tříd zobrazených výše. Zpočátku vývoj probíhal v prostředí wxDevCpp. V průběhu se však přešlo na Visual Studio 2008 s překladačem VC9. K tomuto kroku bylo nutné přistoupit vzhledem ke stále zvyšujícímu se rozsahu kódu a stále častějším pádům wxDevCpp. Přispěla k tomu i potřeba upraveného překladu knihovny wxWidgets s podporou výjimek, vláken a Unicode kódování znaků. Komentáře byli v souladu se „Style Guide“ psány do hlavičkových souborů a jejich struktura umožňuje vygenerování programové dokumentace pomocí programu doxygen. K tomuto účelu je součástí řešení i soubor s konfigurací pro toto generování ve složce ./doc/doxygen. Vygenerovanou programovou dokumentaci lze také nalézt v přílohách na DVD. V *cpp* souborech jsou komentáře psány zřídka a slouží zde spíše jako dodatečné vysvětlení složitějších konstrukcí.

5.3 Knihovna práce s 3D modely

Tato knihovna je koncipována jako velmi jednoduchá. Jejím účelem je poskytovat hlavně operace načtení modelu z co nejvíce formátů souborů a řezání modelu. Pro vlastní funkčnost byla použita knihovna VCG. Implementace spočívá v zapouzdření a zjednodušení celého rozhraní knihovny. Při implementaci bylo třeba překonat mnoho problémů způsobených prakticky neexistující dokumentací. Komentáře v hlavičkových souborech knihovny byly zase často v Italštině. S výslednou funkčností se zde nelze spokojit, protože v pozdních fázích vývoje totiž byly zjištěny závažné nedostatky ve funkčnosti knihovny VCG.

Knihovna podporuje načítání modelu z mnoha různých formátů:

- PLY – Polygon Data File
- STL – stereolitography
- OFF – Object File Format
- OBJ – 3D Object
- VMI – VCGlib mesh image

Ve fázi testování se ukázalo, že tato funkce byla implementována naivním způsobem. Při změně jazyku aplikace přestala knihovna načítat modely ze souborů. To je způsobeno přístupem k parsování datového souboru, kdy jsou využity funkce podporující internacionalizaci. Tyto funkce jsou ovlivněny nastavením lokalizace programu. Formát čísel v datových souborech (např.: OBJ) však má přesně definovaný formát vycházející z americké jednotkové soustavy. Desetinná čísla jsou oddělena desetinnou tečkou. Pokud je však aplikace například v české lokalizaci, tak se pro reprezentaci lokálního oddělovače desetinných čísel používá čárka. Knihovna pak nekorektně čte souřadnice vertexů. Výsledkem je pak „rozbitý“ model. Možným řešením by bylo přepínat lokalizaci při volání této knihovny.

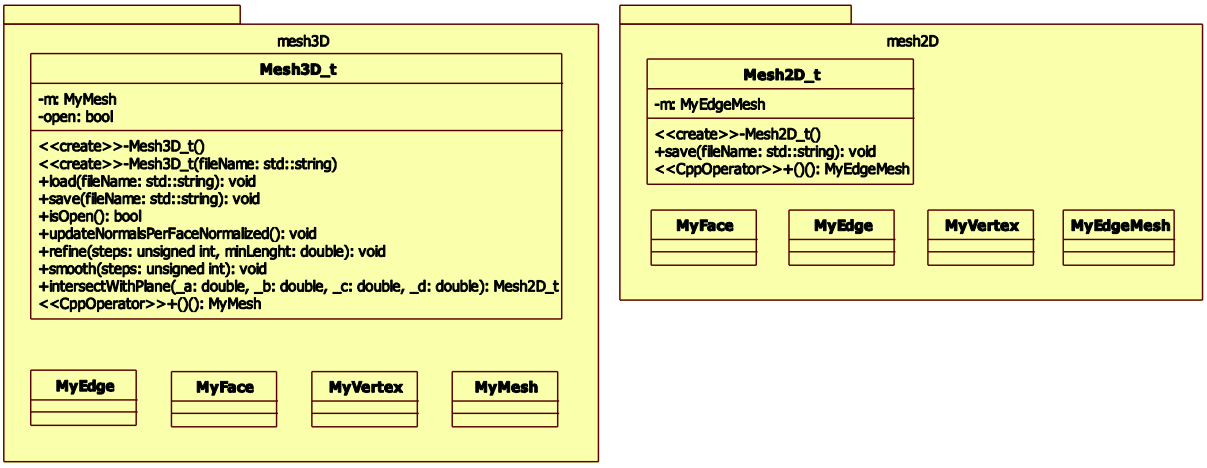
Jako vážnější nedostatek se ukázala funkce průniku modelu s rovinou. Při testování knihovny tato funkce pracovala korektně. Při implementaci se však ukázalo, že tomu tak není ve všech případech. Výsledkem průniku byla často množina bodů na přímce a to i přes to, že rovina byla protínána s koulí. Výsledkem by tedy měla být prázdná množina, bod či kruh. Dále byly tyto body nelogicky posunuty oproti počátku. V některých případech však tato funkce fungovala přesně tak jak má. Po intenzivním testování bylo zjištěno, že funkce funguje, pouze pokud je rovina řezu kolmá na rovinu definovanou osami y, z. Pravděpodobně je chyba v převodu souřadnic z 3D do 2D po vypočítání průniku. Každopádně tato závada je vážná a vedla k nepoužitelnosti tohoto řešení.

Z důvodu pozdního odhalení těchto nectností se již nestihlo přenést celé řešení pod jinou knihovnu. Celá knihovna je tak spíše prázdnou obálkou. To však neznamená, že by tato práce byla zbytečná. Rozhraní knihovny je kvalitní a mělo by být dále použito. Mezi zvažovanými alternativami knihovny byla často zmiňována knihovna CGAL, jejíž licence je open source. Jedná se však o

mnohem větší projekt a křivka učení (learning curve) jistě nebude tak strmá jako v případě knihovny VCGlib. Jiným typem řešení je implementace těchto funkcí od základu.

Rozhraní

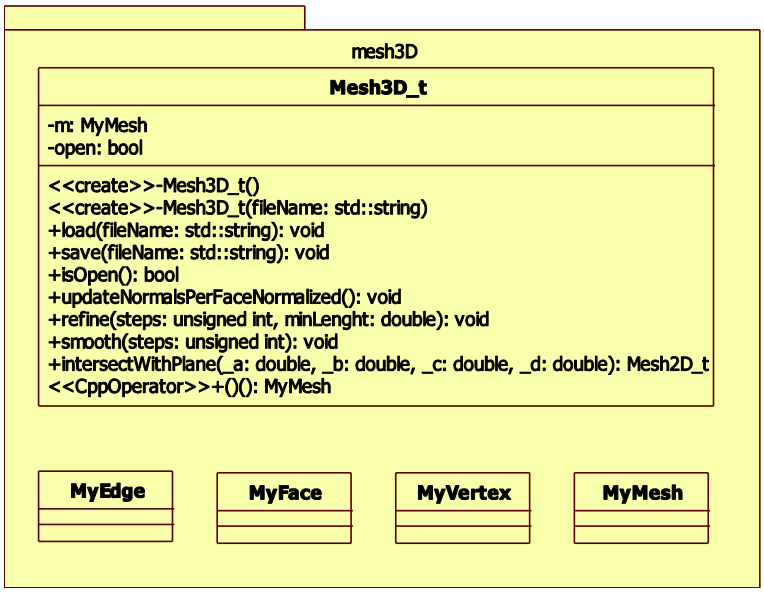
Rozhraní knihovny pro práci se 3D modely bylo navrženo s ohledem na použití v aplikaci tisku 3D objektů. Z tohoto důvodu se může zdát velmi jednoduché až triviální. Cílem aplikace však není provádět různé komplexní operace nad modely objektů. Není to ani cílem této práce. Rozhraní poskytuje dvě třídy: První reprezentující model objektu sítí trojúhelníků ve třírozměrném prostoru. Druhá reprezentující síť bodů a hran je spojujících ve dvourozměrném prostoru. [7] Rozhraní knihovny lze vidět na objektovém modelu na obrázku 20 níže.



Obrázek 20: Rozhraní knihovny libModel3D.

Model

Třídy v této knihovně nemají mezi sebou žádné vazby s výjimkou výjimek. Celkový objektový model knihovny je proto přímo rozložen v podkapitolách, popisujících jednotlivé třídy.



Obrázek 21: Objektový model třídy Mesh3D_t

Mesh3D_t

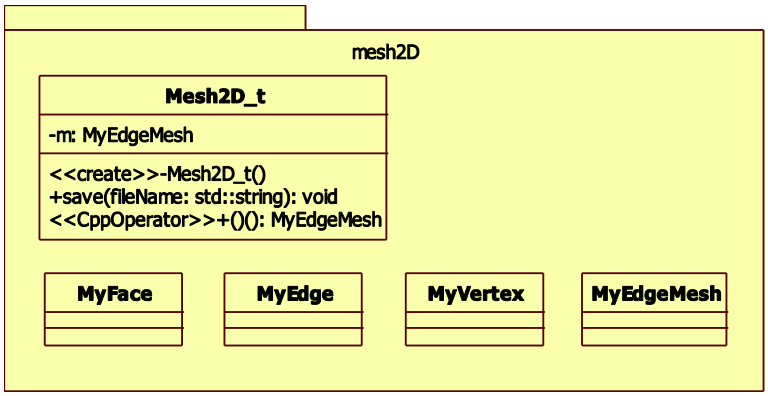
Tato třída je základním prvkem knihovny práce s 3D modely. Zastřešuje a sjednocuje přístup k práci s modelem. Mezi její základní operace patří načtení

modelu ze souboru a analogicky i jeho uložení. Další, ne však v důležitosti, je operace protnutí modelu s rovinou. Tato operace se aplikuje na instanci třídy, která již má otevřený model. Vstupem je obecně zadaná rovina. V praxi se však bude používat rovina kolmá na jednu z os. Tato metoda vrací objekt `Mesh2D_t`, který vektorově reprezentuje výsledek této operace. Podrobněji je rozebrán v následující kapitole.

Další funkce jsou nadstandardní a slouží k úpravám modelu před vlastním tiskem. Jedná se o funkce refine a smooth, která vyhlazuje trojúhelníkovou síť modelu. Dle dokumentace knihovny VCG se jedná o variantu Laplacova vyhlazování. [13] [7] Třída také poskytuje metodu pro přímý přístup k síti trojúhelníků. Ta je využívána při vykreslování modelu.

Mesh2D_t

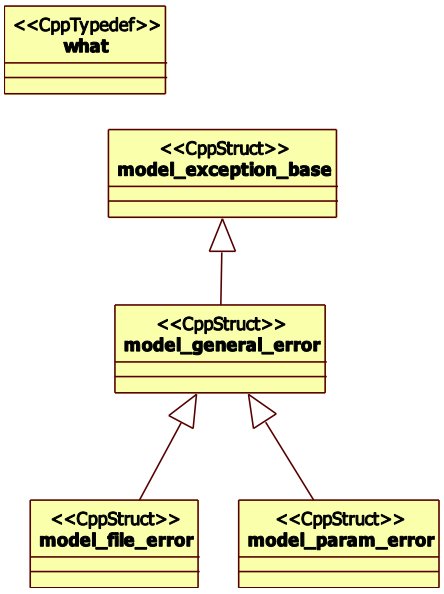
Druhou třídou této knihovny je `Mesh2D_t`, která analogicky k `Mesh3D_t` reprezentuje model ve 2D prostoru. Její hlavní náplní by měla být funkce rasterizace, která vrátí její rastrovou reprezentaci s vyplněnými uzavřenými hranami. Z důvodů nekorektního chování knihovny VCGlib rozebíraných výše v úvodu kapitoly 5.3 – Implementace knihovny práce s 3D modely nebyla tato funkce implementována. Je to velká škoda. Dále tu zůstala pouze funkce zaobalující uložení vektorové interpretace této sítě. [7]



Obrázek 22: Model třídy Mesh2D_t.

Výjimky exception_base, general_error, param_error, file_error

Struktura výjimek je velmi podobná předchozí knihovně libControl3D. Základní je třída je odvozena jak od `std::exception` tak i `boost::exception`. Z ní se potom odvíjejí i specializované třídy obecné chyby, chyby parametru či práce se souborem. Obsahuje také třídu what, která bezpečně přidává informaci o popisu chyby.



Obrázek 23: Objektový model tříd reprezentujících výjimky.

Realizace

Z objektového modelu vytvořeného v programu StarUML byly exportem vytvořeny hlavičkové soubory a nevyplněné kostry *cpp* souborů. Ty byly následně doplněny kódem. V hlavičkových souborech byly dopsány komentáře tříd, funkcí a parametrů. U komentářů bylo dodržováno formátování kompatibilní s programem doxygen. S přechodem ostatních částí práce ke kompilátoru VC9 byla i tato knihovna převedena do projektu ve Visual Studiu. V *cpp* souborech jsou komentáře psány zřídka a slouží zde spíše jako dodatečné vysvětlení složitějších zápisů.

5.4 Grafické uživatelské rozhraní

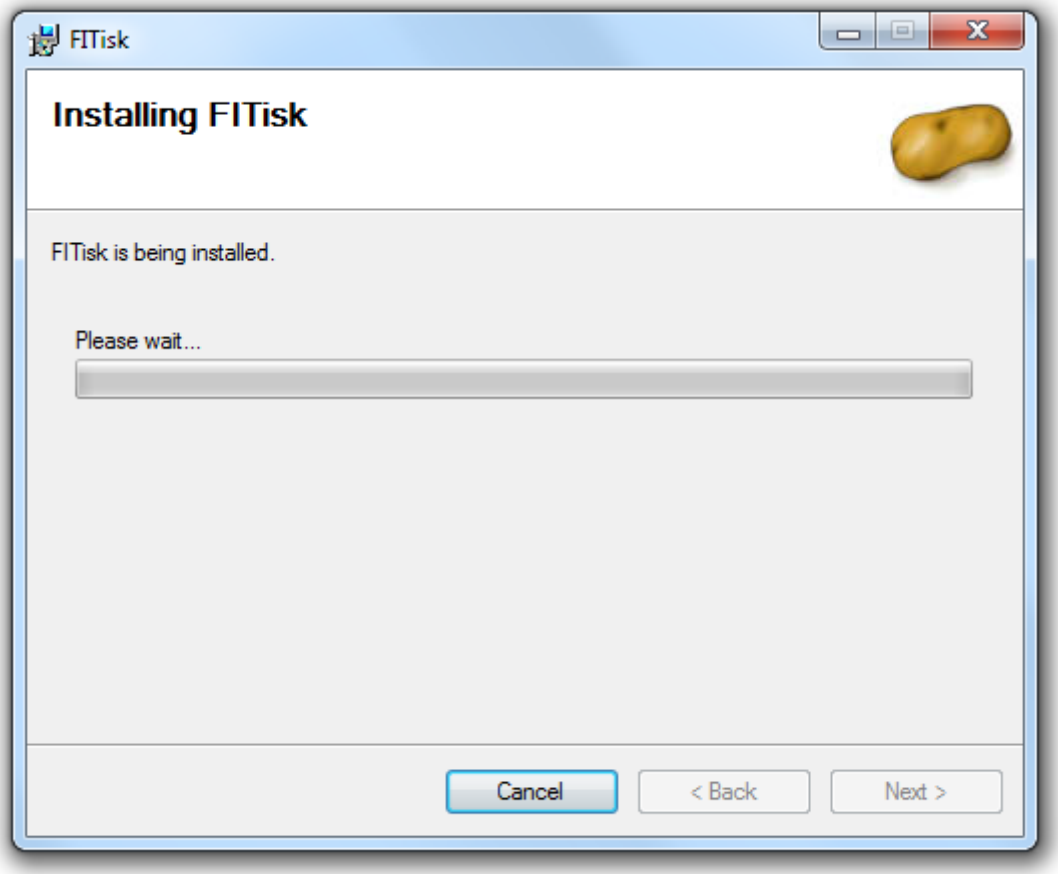
Jak již bylo zmíněno v kapitole 4.4 – návrh Grafického uživatelského rozhraní, tak k sestavení GUI bylo použito programu wxFormBuilder. Tímto programem byly také vygenerovány třídy reprezentující jednotlivé prvky GUI. Jedná se například o hlavní okno aplikace, dialog nastavení, dialog vyhlazování a mnoho dalších. Celkově je v tomto projektu 18 hlavičkových souborů definujících minimálně stejně tolik tříd. Z tohoto důvodu nejsou v této kapitole rozbírány jednotlivé třídy. Tvorbou a kompozicí GUI se zabývají jiné práce. Tato kapitola se omezuje pouze na výjimečné části.

Při implementaci bylo třeba vytvořit množství ikon, tlačítek a piktogramů (dále označováno jako ikony). Jejich kompletní sbírka je k nalezení na přiloženém DVD ve složce rc. Přechod projektu pod Visual Studio také umožnil využití dalších funkcí, které tento velký balík software obsahuje. Umožnil tak sjednocení jednotlivých projektů libControl3D, libModel3D, libCartridge a FITisk do jednoho kompaktního celku nazývaného Solution (řešení). Výsledkem je velmi intuitivní práce s celým projektem. Dále byl přidán projekt typu Windows instalátor, který byl navázán na produkt (spustitelný soubor) předchozích projektů. Instalátor je náplní druhé kapitoly. [14] Poslední kapitola se zabývá strukturou tříd, jejich členěním a charakteristikou.

Instalátor

Jedním z příjemných přínosů vedení projektu pod Visual Studiem bylo i nenáročné vytvoření instalátoru. Nebylo zde použito žádných nadstandardních funkcí. Ovlivnit lze pouze cestu kam se má aplikace nainstalovat. Samotná instalace pak sestává z nakopírování spustitelného *exe* souboru, složky s ikonami a složky s jazykovými překlady. Soubor obsahující konfiguraci aplikace je analogicky nakopírován do složky aplikačních dat. Konkrétně do podsložky s názvem aplikace. Následně jsou ještě vytvořeny zástupci na ploše a v nabídce start. I přes to, že se jedná o kopírování několika málo

souborů, tak celá instalace trvá spíše desítky sekund. Je dáno tím, že před vlastní instalací je vytvářen bod obnovení systému. Vzhled okna instalátoru při provádění instalace lze vidět na obrázku 24. [14]



Obrázek 24: Okno instalačního programu při probíhající instalaci.

Struktura tříd

Celá aplikace má za úkol synchronizaci volání knihovních tříd a správu chybových stavů. Chybové stavy indikované výjimkami jsou zachycovány na úrovni událostí. Zde dojde k zpracování výjimky a následnému zobrazení varování s popisem chyby. Pokud se lze z chyby zotavit bez interakce s uživatelem, tak je výjimka zpracována již v obsluze události (tedy uvnitř události). Třídy jsou v aplikaci rozděleny do tří kategorií:

- Třída GUI
- Pomocná třída
- Knihovní třídy

Třídy GUI přímo reprezentují grafický prvek a obsahují zpracování událostí. Kostry těchto tříd vznikly vygenerováním z programu wxFormBuilder. Všechny dědí z odpovídajících bázových tříd. Bázová třída je plně generována programem wxFormBuilder. Bázová třída reprezentuje vzhled. Odvozená třída pak přidává funkčnost. Výsledný kód je takto velmi přehledný.

Dále jsou zde třídy pomocné, jako například třída reprezentující nastavení lokalizace. Tyto třídy nezpracovávají žádné události. Jejich držitelem jsou třídy GUI, ke kterým logicky patří. Například v případě nastavení lokalizace je to třída `MyApp` reprezentující aplikaci.

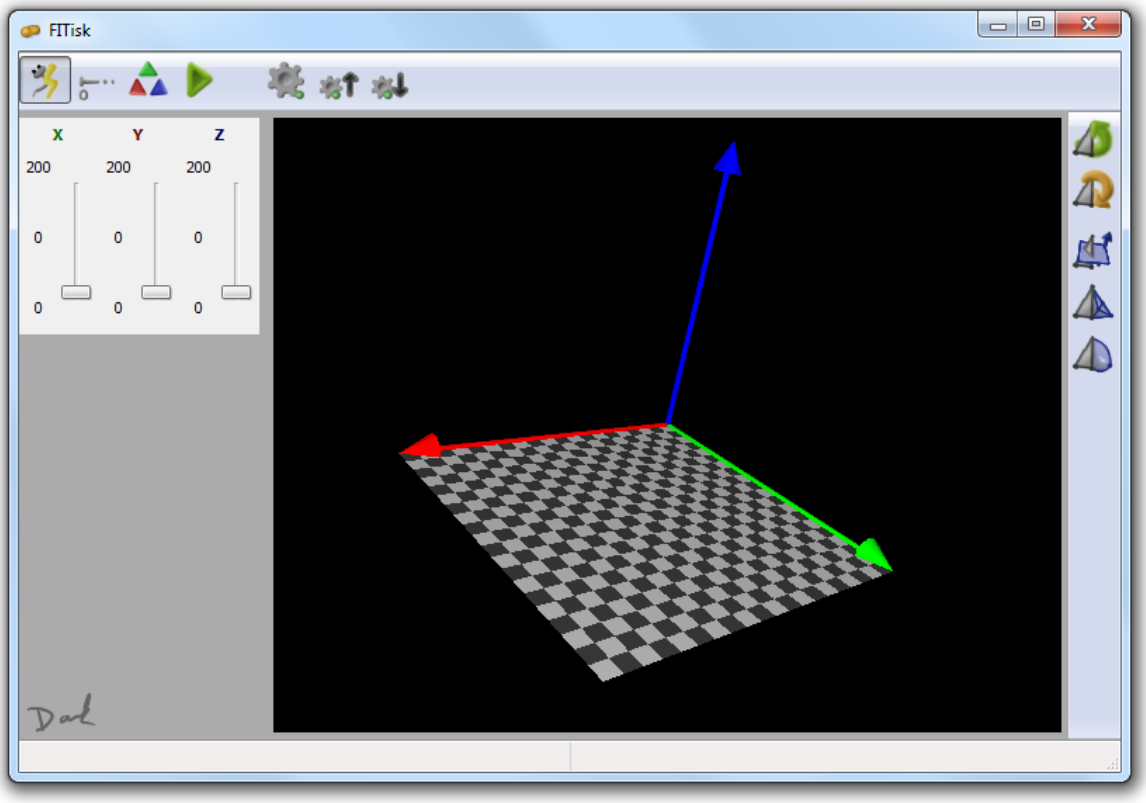
Posledním druhem jsou třídy knihoven. Použití je obdobné s pomocnými třídami. Držitelem jsou třídy GUI. Volány jsou většinou v reakci na události. Kromě případů kalibrace a tisku modelu.

6 Výsledky

Výsledky této práce jsou hodnoceny ve dvou rovinách.

- Splnění požadavků kladených na knihovnu řízení tiskového stroje
- Měření přesnosti pohybu a celkové kvality tiskového stroje

Knihovna řízení tiskového stroje (libControl3D) se ukázala jako funkční. Řeší komunikaci s obecným počtem řídících jednotek jednotlivých motorů. Z hlediska použití na konkrétním stroji ve FIT na UPGM je její použití bezproblémové. Její další vývoj je usnadněn díky programové dokumentaci, která je generována přímo z komentářů ve zdrojovém kódu. Objektový přístup k celé knihovně by měl také usnadnit přidávání dalších funkcí pro specializované použití. Současná funkčnost byla vytvářena s ohledem na použití knihovny v tiskovém stroji. Avšak i při potřebě naprosto odlišného přístupu k řízení řídících jednotek je zde možnost nepoužít vyšší specializace objektů jako `Space_t`, `Axe_t` a použít pouze základní statické funkce pro realizaci příkazů na úrovni komunikačního protokolu s řídícími jednotkami. Jejich implementace nikterak neomezuje komunikaci s řídícími jednotkami. Probíhá zde pouze kontrola, zda jsou dodržovány požadavky kladené na parametry.



Obrázek 25: Hlavní okno výsledné aplikace FITisk.

Při implementaci bylo dosaženo vytyčených cílů u knihovny libControl3D a vlastní aplikace FITisk. U knihovny libModel3D je funkčnost neúplná a provází ji chyby zmíněné v kapitole 5.3 – Implementace knihovny pro práci s 3D modely. Tyto chyby byly způsobeny chybami v knihovně VCGlib. Jejich odstranění tedy nebylo možné. Možná řešení jsou rozebírána ve stejné kapitole. V aplikaci FITisk se na druhou stranu podařilo dosáhnout i relativně nadbytečných funkcí jako je výše zmíněná vícejazyčnost či vytvoření instalátoru.

V budoucnu je třeba se zaměřit na knihovnu práce s 3D modely. Je zde otázka, zda funkce implementovat od základů, či použít jinou knihovnu pro realizaci rutinních operací a implementovat pouze specializované funkce jako je rasterizace. Vhodnější se zdá být použití knihovny, protože tento způsob dovoluje v budoucnu snadněji rozšiřovat funkčnost. Nabízí se i otázka nahrazení vizualizace tiskárny s modelem některým ze standardních toolkitů. Mezi možné varianty je možné zahrnout například VTK (Visualization Toolkit). Poskytuje nástroje pro silnou vizualizaci.

6.1 Měření přesnosti

Měření přesnosti vycházela z předpokládaného použití stroje. Definován byl test přesnosti opakovaného vystavení polohy. Při tomto testu najíždělo čelo vozíku (místo, kde bude připevněna tisková hlava) na předem určenou pozici. Měřidlem potom byla změřena pozice, kam vozík přijel. Nájezd na tuto pozici byl prováděn z různých směrů a vzdáleností. V provozu bude tato přesnost potřeba například při návratu hlavy na počátek souřadnic před zahájením tisku nové vrstvy.

Druhým testem bylo měření odchylky při pohybu jedním směrem. V tomto testu je vozík postupně vystavován na vzrůstající pozici a opět je měřena přesnost, s jakou je této pozice docíleno. Tento pohyb je obdobou pohybu při nastavování pozice pro tisk jednoho pásu materiálu.

Pro měření bylo použito digitální posuvné měřidlo. Jeho parametry jsou shrnuty v následující tabulce 4. Obrázek měřidla lze zase vidět na obrázku 26.

Rozsah	0 – 150mm
Nejmenší rozlišení	0,01mm
Přesnost	± 0,02mm (< 100mm) ± 0,03mm (100 - 150mm)
Typ měření	Lineární kapacitní měřící systém
Pracovní teplota	5ºC - 40ºC

Tabulka 4: Parametry digitálního posuvného měřidla.

Obrázek 26: Digitální posuvné měřidlo

Postup měření

Připojíme tiskárnu k počítači pomocí sériového kabelu nebo přes USB redukci. Pokud není přítomen program FITisk, tak jej nainstalujeme pomocí instalátoru z přiloženého DVD. Ověříme nastavení poměru, rychlosti, zrychlení, síly, počtu mikrokroků, charakteristiky řídícího proudu a kvalifikace řízení. Jejich hodnoty mohou ovlivňovat přesnost stroje, proto jsou součástí podmínek měření. Po spuštění programu zmáčkneme tlačítko připojit k tiskárně. Pokud připojení proběhlo korektně, mělo by se povolit tlačítko kalibrace. Klepneme na něj a vyčkáme, než proběhne kalibrace stroje.

Nyní si uděláme měřící značky na čele vozíku a hraně bloku převodovky. Tyto značky musí být rovnoběžné s osou šneka pohybujícího čelem vozíku. Nyní přiložíme posuvné měřidlo na měřící značky a pomocí tlačítka zero (či jeho obdoby) posuvné měřidlo vynulujeme.

Nyní budeme měřit opakovanou přesnost vystavení. Zvolíme si polohu, kde budeme měření provádět, např. 50mm. Pomocí posuvníku v programu posuneme hlavu na tuto pozici a změříme aktuální polohu. Nyní s hlavou odjedeme náhodným směrem o náhodnou vzdálenost. Následně se vrátíme zpět na měřenou polohu a opět změříme polohu posuvného měřidla. Měření opakujeme 10x.

Druhým testem je měření přesnosti vystavení při lineárním pohybu. Vrátíme čelo vozík na počáteční souřadnice (0). Pokud nemáme kalibrované měřidlo, tak jej kalibrujeme. Nyní popojedeme o 10mm a změříme polohu. Tento postup opakujeme, než dojedeme na 100mm. Nyní vrátíme čelo vozíku na počátek a zopakujeme celé měření 10x.

Poslední je orientační test zvlnění pracovní desky vzhledem k pozici čela vozíku. Nastavíme vozík na počátek souřadnic jak osy x, tak i y. Nyní kalibrujeme posuvné měřidlo na vzdálenost od pracovní desky. Pohneme vozíkem o 20mm ve směru osy x a opět změříme vzdálenost hlavy od

pracovní desky. Toto měření zopakujeme 10x, tak abychom pokryli celou délku pracovního prostoru na ose x. Nyní posuneme hlavu ve směru osy y k maximu a zopakujeme měření pro tuto pozici.

Podmínky měření

Měření probíhalo za pokojové teploty 22ºC ±7º. Řídící jednotka, resp. řídící program FITisk byl nastaven s parametry v tabulce 5.

Poměr	0,0001953 mm / µkrok
Rychlost	7000 µkrok / s
Zrychlení	15000 µkrok / s^2
Síla (start/stop rychlost)	1000 µkrok / s
Mikrokroků	1
Kvalifikace	1
Charakteristika	1 - sinus wave

Tabulka 5: Nastavení programu FITisk.

Parametry posuvného měřidla již byly zmíněny výše.

Výsledky měření

Výsledky měření opakované přesnosti vystavení jsou v tabulce 6 a tabulce 7. Dle vztahu (1) byla vypočítána relativní odchylka (δ p).

p_0	p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8	p_9	p_{10}	$\overline{p_{1-10}}$
50	50,00	50,00	49,99	49,99	50,00	49,98	49,97	50,01	50,00	49,99	49,99

Tabulka 6: Hodnoty měření opakované přesnosti vystavení.

$$\delta p = \left|\frac{p_0 - \overline{p_{1-10}}}{p_0}\right| \times 100\ [\%] \qquad (1)$$

p_0	$\overline{p_{1-10}}$	p_{min}	p_{max}	δ p [%]	Δp_0^+	Δp_0^-
50,00	49,99	49,97	50,01	0,02	0,01	0,03

Tabulka 7: Výsledky měření opakované přesnosti vystavení.

Při opakovaném vystavování polohy (p_0) 50mm byla naměřena průměrná poloha ($\overline{p_{1-10}}$) 49,99mm. Maximální odchylka (p_{max}) od požadovaných 50mm byla 0,03mm směrem k počátku a 0,01mm opačným směrem (p_{min}). Celková relativní odchylka (δ p) byla 0,02%, což ukazuje na velmi vysokou přesnost vystavování hlavy. Pro účely pohybu s přesností na mm je to naprosto dostačující. Přesnost dostačuje i pro pohyb v jednotkách desetin milimetru.

Měření přesnosti pohybu při lineárním pohybu jsou shrnuta v následující tabulce 8.

p_0	p_{1-10}	δ p [%]	Δp_0
0	0,00	0	0,00
10	10,00	0	0,00
20	20,01	0,05	0,01
30	30,01	0,03	0,01
40	40,00	0	0,00
50	49,99	0,02	0,01
60	59,98	0,03	0,02
70	69,98	0,03	0,02
80	79,99	0,01	0,01
90	89,97	0,03	0,03
100	99,97	0,03	0,03

Tabulka 8: Výsledky měření přesnosti při lineárním pohybu.

Z tabulky 8 lze vyčíst, že si stroj zachovává svoji vysokou přesnost po celé měřené dráze. Je však třeba říci, že měřená dráha je menší, než skutečné maximum. Je to způsobeno omezením ze strany měřicího přístroje, který nebyl schopen měřit vzdálenosti větší jak 100mm. Zvyšující se absolutní odchylka při větších vzdálenostech (90mm, 100mm) je pravděpodobně způsobena umístěním měřidla na stroji. Odchylku pravděpodobně způsobovala odchylka osy měření od osy pohybu.

Poslední měření mělo za cíl získat mapu zvlnění pracovní desky od hlavy. Mapa je zobrazena v podobě tabulky 9 níže. Vlevo nahoře je počátek souřadnic 0,0. Směrem dolů pak narůstá hodnota na ose x. Doprava potom skokově roste hodnota osy y na maximum. Zvýšené hodnoty v levém dolním rohu tabulky jsou pravděpodobně způsobeny pevně utaženým šroubem držícím pracovní desku.

Statistické výsledky zvlnění jsou poté v tabulce 10. Absolutní odchylka od průměrné hodnoty (Δp) nepřesahuje 0,06mm. Vzhledem k chybějící poslední desce, která jistě bude obsahovat i kalibrační uchycení to je velmi dobrá hodnota.

	y_0		y_{max}
0mm	0,00		0,00
10mm	-0,02		0,01
20mm	0,01		0,01
30mm	-0,01		0,00
40mm	0,02		-0,01
50mm	0,01		-0,01
60mm	0,02		-0,02
70mm	0,04		-0,01
80mm	0,02		-0,03
90mm	0,06		-0,05
100mm	0,05		-0,04

Tabulka 9: Zvlnění pracovní desky vzhledem k tiskové hlavě.

p_0	$\overline{p_{1-20}}$	p_{min}	p_{max}	$\lvert\Delta p\rvert$
0	0,00	-0,05	0,06	0,06

Tabulka 10: Výsledky měření zvlnění pracovní desky vzhledem k hlavě.

Celkově lze říci, že naměřené parametry jsou na velmi dobré úrovni. Naměřené odchylky v řádech setin milimetrů již spíše narážejí na možnosti měřidla a jeho uchycení k tiskovému stroji.

7 Závěr

V práci byl teoreticky rozebrán prototyp tiskového stroje na UPGM FIT. Byla zde diskutována jeho konstrukce a její vlastnosti. Následně bylo zaměřeno na řízení krokových motorů, konkrétně na pohony firmy Microcon s řídící jednotkou M1486. Rozebrán byl komunikační protokol a jeho použití při zapojení více řídících jednotek na jednu linku. Na základě tohoto rozboru byla potom navržena aplikace pro tříosé ovládání tiskového stroje. Aplikace byla rozdělena do několika částí – knihoven. Experimentování s navrženým řešením vedlo k několika nezásadním změnám, které byly zapracovány. V závěru jsou diskutovány dosažené výsledky jak z pohledu rozhraní knihovny a aplikace, tak i z pohledu přesnosti řízení. Vytvořen byl i plakát prezentující tuto diplomovou práci, její cíle a výsledky, který je součástí příloh.

Navržená knihovna se projevila jako plně funkční a plnící veškeré požadavky na ní kladené. Mimo jiné to bylo prokázáno v demonstrační aplikaci. V době práce nad tímto projektem (říjen 2009 - květen 2010) bohužel nebyla k dispozici knihovna pro řízení tiskové hlavy, takže na hmatatelné výsledky práce je třeba počkat. Demonstrační aplikace však byla úspěšně použita v jiném projektu, který vyžadoval přesný pohyb kamerou nad deskou s obrazcem.

Projekt 3D tiskárny existuje je teprve na svém počátku a toto je první ucelená práce na něm provedená. Bylo zde třeba definovat standarty a knihovny tak, aby byl možný jeho další vývoj na bázi ucelených celků nezávisle na sobě. Celá aplikace je tvořena spíše s ohledem na další výzkum a vývoj. Tomu odpovídají jak vnitřní konstrukce, kdy je použito šablonového návrhu tříd, tak i fronted v podobě aplikace, která poskytuje možnosti jako vypnutí omezovačů pracovní dráhy. Další vývoj je možné postavit na této aplikaci (FITisk), která bude sloužit jako ústřední bod pro všechny knihovny.

Literatura

[1] WEINBERGER Benjy, SILVERSTEIN Craig, EITZMANN Gregory et al.: Google C++ Style Guide [online], Revize 3.154, [cit. 2009-10-23]. Dostupné na URL: http://google-styleguide.googlecode.com/svn/trunk/cppguide.xml

[2] Wikipedia, the free encyclopedia, Linear motor [online], poslední revize 2010-05-16 [cit. 2009-12-24]. Dostupné na URL: http://en.wikipedia.org/wiki/Linear_motor

[3] DAWES Beman, ABRAHAMS David, RIVERA Rene et al., *Boost Libraly Documentation* [online], poslední revize 2010-05-06 [cit. 2010-05-06]. Dostupné na URL: http://www.boost.org/doc/libs/

[4] Komunita vývojářů knihovny wxWidgets, wxWidgets [online], poslední revize 2010-09-04 [cit. 2009-12-22]. Dostupné na URL: http://www.wxwidgets.org/

[5] ORSÁG Filip, Série přednášek k předmětu ROB, říjen – prosinec 2009, Fakulta informačních technologií VUT v Brně

[6] Microcon, Katalog výrobků 2009 [online], 2009, [cit. 2009-10-10]. Dostupné na URL: http://www.microcon.cz/

[7] Visual Computing Lab of the ISTI - an institute of the Italian National Research Council (CNR), VCG lib, 2009, poslední revize 2010-01-11 [cit. 2010-02-11]. Dostupné na URL: http://vcg.sourceforge.net/index.php/Main_Page

[8] Komunita vývojářů projektu wxFormBuilder, wxFormBuilder web[online], poslední revise 2009-06-11[cit. 2009-12-31]. Dostupné na URL: http://wxformbuilder.org/

[9] Komunita vývojářů aplikace poEdit, *poEdit features* [online], poslední revize 2010-03-22 [cit. 2010-03-03]. Dostupné na URL: http://www.poedit.net/

[10] Microcon, *Stepper motor controler Microcon 1486*, verze č. 2.39, 2005

[11] Craig Peacock, *Interfacing the Serial/ RS-232 port* [online], poslední revize 2005-05-15 [cit. 2010-05-15]. Dostupné na URL: http://www.beyondlogic.org/serial/serial.htm

[12] GAMMA Erich, HELM Richard, JOHNSON Ralph et al., *Design Patterns: Elements of Reusable Object-Oriented Software*, Oregon, 1995, ISBN 0-201-63361-2

[13] Tinka Gammel, Los Alamos National Laboratory, *Grain Evolution in a T-Junction* [online], 2009, [cit. 2009-10-23]. Dostupné na URL: https://lagrit.lanl.gov/tee.shtml

[14] Microsoft Corporation, MSDN Lirary [online], 2010, [cit. 2010-03-05]. Dostupné na URL: http://msdn.microsoft.com/library/

Seznam zkratek

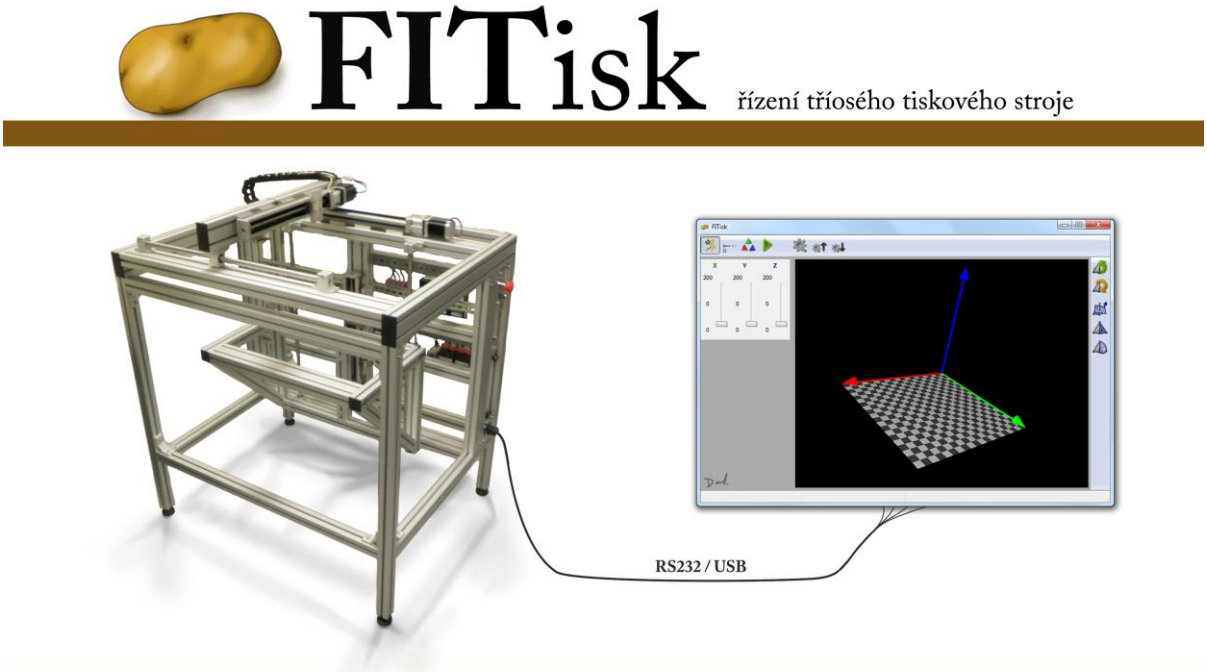
GUI	Graphical User Interface – grafické uživatelské rozhraní
CB	CallBack – zpětné volání
GPL	General Pulic License – všeobecná veřejná licence GPU
VCG	Visual Computing Lab
STL	Stadart Template Library
VTK	Visualization Toolkit

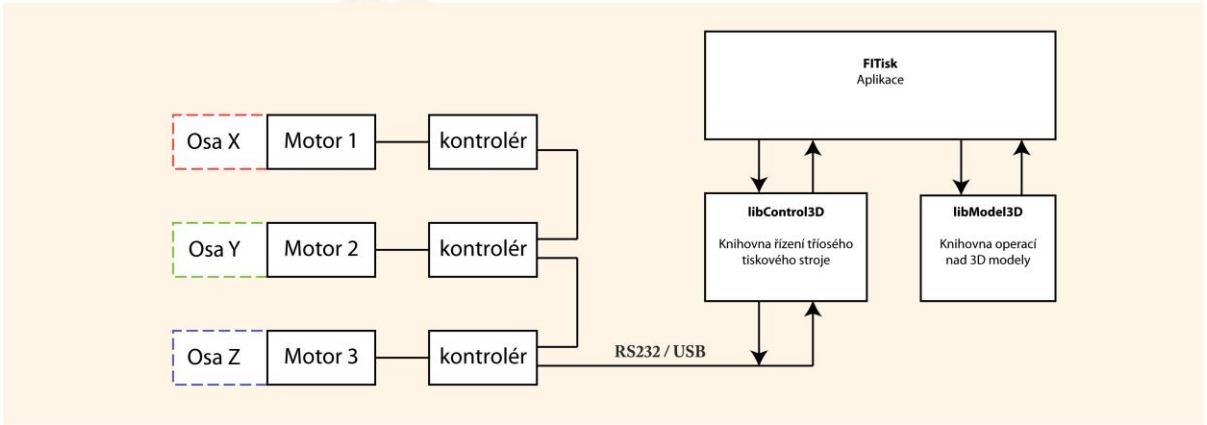
Seznam příloh

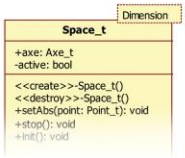
Příloha A: DVD se zdrojovými kódy, programovou dokumentací, přeloženým programem v podobě instalátoru, touto technickou zprávou a plakátem prezentujícím tuto práci.

Příloha B: Plakát prezentující tuto práci.

Příloha B – Plakát







- krokové motory s řídícími jednotkami CD30x firmy Microcon
- znakový komunikační protokol

- modulární struktura, objektový přístup, šablonový návrh
- univerzální, přenositelné řešení
- programovatelné body přerušení

```
class ProgressUpdater_t {
public:
    ProgressUpdater_t():c(f
    virtual void update(int
    virtual void done(int =
    virtual void interrupio
```

Ústav Počítačové Grafiky a Multimédií FIT vedoucí: Ing. Michal Španěl autor: Darius Kočár - xkocar00