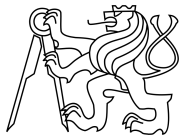
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ



ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Emulátor 3D tiskárny
Student:	Jan Tlamicha
Vedoucí:	Ing. Marek Žehra
Studijní program:	Informatika
Studijní obor:	Softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2016/17

Pokyny pro vypracování

Ve světě běžných stolních tiskáren je běžné, že existují emulátory fyzických zařízení, které například umožňují export do formátu PDF. Svět 3D tisku těmito emulátory ale zatím nedisponuje.
Cílem této práce je vytvořit nástroj, který bude simulovat chování fyzické 3D tiskárny a zobrazovat její stav spolu s možností nastavit základní parametry. Simulátor bude komunikovat po virtuálním seriovém portu a bude umožňovat připojení řídících software pro 3D tisk. Simulátor by měl reagovat na základní i rozšířené GCode příkazy popsané na http://reprap.org/wiki/G-code s možností jejich rozšíření o nové příkazy a definici funkcionality formou modulů.
1. Proveďte rešerši existujících řešení obdobných simulačních programů a firmwarů.
2. Navrhněte architekturu řídícího systému a vyberte realizační prostředky (platforma a programovací jazyk).
3. Implementujte a otestujte software, který bude simulovat chování 3D tiskárny.

Seznam odborné literatury

Dodá vedoucí práce.

L.S.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

prof. Ing. Pavel Tvrdík, CSc.
děkan

V Praze dne 5. ledna 2016

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra softwarového inženýrství

Bakalářská práce

Emulátor 3D tiskárny

Jan Tlamicha

Vedoucí práce: Ing. Marek Žehra

13. května 2016

Poděkování

Děkuji vedoucímu této práce Ing. Marku Žehrovi za připomínky a cenné rady při psaní této práce. Zároveň děkuji i rodičům za jejich neustálou podporu. V neposlední řadě děkuji všem přátelům, kamarádů i vyučujícím, díky kterým jsem se mohl dostat k napsání této práce.

Prohlášení

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

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 13. května 2016 .....

Odkaz na tuto práci

Tlamicha, Jan. *Emulátor 3D tiskárny*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2016.

Abstrakt

Cílem práce je navrhnout a implementovat program simulující chování 3D tiskárny. Bakalářská práce zaměřená na zpracování výstupu ovládacího programu pro 3D tisk emulací (napodobení) chování skutečné 3D tiskárny. Cílem analýzy je připodobnit chování emulátoru vzhledem ke skutečnému firmwaru tiskárny i tiskárně samotné. Využívat bude virtuálního sériového portu ke kterému se připojí ovládací program, a bude vykonávat instrukce ve formátu G-code.

Klíčová slova Reprap, 3D Tisk, Emulátor, Sériový port, G-code

Abstract

Main goal of this work is design and implement program for simulation 3D printer. This Bachelor thesis theme is processing G-Code senders for 3D printer emulating behavior of real 3D printer. 3D print transfer 3D models to real world using plastic fillament. Analysis is targeted for making virtual printer behaving like real printer. It's using virtual serial port for sending slicered instructions in G-Code format sended with G-Code senders program to virtual printer.

Keywords Reprap, 3D Print, Emulator, Serial port, G-code

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Zadání této bakalářské práce se datuje do nedávné doby. Cílem práce je simulovat proces tisku pomocí programu. Význam tohoto programu spočívá ve vývoji, zdokonalení a odzkoušení řídících aplikací pro 3D tisk.

Analýza dostupné literatury nezjistila žádný program, který pokrývá stejnou funkcionalitu jako program, jež je obsahem této práce. Oproti stávajícím řešení má výsledek této práce rozšiřovat možnosti simulace o tisk přímo pomocí vytvořeného virtuálního sériového portu.

Po dokončení implementace je nutné ověřit její funkčnost pomocí testů. Pro testování byla zvolena forma tzv. unit testů. Pro důsledné otestování byla zvolena i další forma testů, jakými jsou testy funkcionální.

Cílem práce je vývoj a otestování funkčního emulátoru 3D tisku. Důležité je propojení ovládacího programu s tímto emulátorem, pomocí virtuálního sériového portu simulovat chování reálného spojení počítače a 3D tiskárny a tím opravovat chyby a kolize, které můžou vzniknout v průběhu tisku. Simulátor umožní napodobit proces tisku bez vlastního zařízení (3D tiskárny), proto umožní snížit čas a náklady na vývoj a výrobu nových výrobků. U stávajících výrobků umožní optimalizovat proces tisku. Také umožní ladění GCodu s vytvořením podpor nebo změnou materiálu. Umožňuje sledovat tisk ve zrychlené formě simulace.

Kapitola 1

Úvod do problematiky

Tato kapitola je zaměřena na technologie a pojmy potřebné k analýze a pozdější realizaci aplikace.

1.1 Sériový port

Sériový port je nejpoužívanějším rozhraním pro řízení a komunikaci s periferním zařízením. Data jsou posílána sériově a mají oddělené vodiče pro příjem i vysílání. Rychlost komunikace udávaná v baudech[1] a další nastavení pro data jsou zásadní pro správnou komunikaci mezi zařízeními. Nastavení se skládá z počtu bitů dat (obvykle 8) startovacího, stop bitu a parity. Obsahuje také další řídící bytové konstrukce, jako historický pozůstatek terminálů.

1.2 Simulátor

Simulace je napodobení nějaké skutečné věci, stavu nebo procesu. Simulací obecně nazveme zobrazení některých klíčových vlastností nebo chování vybraných fyzikálních veličin nebo abstraktních systémů. Počítačová simulace se pokouší o vymodelování reálného světa nebo hypotetické situace, aby bylo možné studovat tento systém. Chování systému je pak předpovídáno v podobě hodnot proměnných. Mimo výpočetní techniku se simulace využívá hojně ve fyzice, chemii, strojírenství. Simulace fyzikálních jevů probíhá pomocí matematických modelů. Jeden z nejsložitější problémů je simulace předpovědi počasí.

[1] Jednotka modulační rychlosti(znaková rychlost) udávající počet změn stavu přenosového média za jednu sekundu.

1.3 Emulátor

Emulátory se všeobecně odkazují na schopnost počítačového programu nebo konkrétního zařízení napodobit, emulovat jiný program či zařízení. Emulátor je druh softwaru umožňující běh počítačových programů na jiné platformě, než pro kterou byla původně vytvořena. Často se používá emulací operačních systémů v podobě virtualizace. Například pomocí Virtualboxu je možno emulovat na počítači s operačním systémem Windows operační systém Linux a naopak. Další typický příklad lze najít ve světě tiskáren. Mnoho tiskáren bylo navrženo k emulování chování tiskáren značky společnosti Hewlett-Packard.

Kapitola 2

3D tisk

2.1 Úvod do 3D tisku

3D tisk prochází dnes bouřlivým vývojem, jak z hlediska nových technologií, tak i možností aplikací. Tiskárny vyrábějí velké a renomované firmy, na druhé straně se na stavbě a vývoji podílejí amatéři. Možnosti tisku sahají od tiskáren použitelných pro výtisk obytného domu až po tiskárny ve vesmíru. Vývoj tiskáren i materiálů je natolik rychlý, že neustále rostou možnosti aplikace 3D tisku v běžném životě. Největší rozvoj a uplatnění nových technologií je v lékařství, zejména v protetice.

3D tisk je forma přenosu počítačem vytvořeného modelu do reálného světa v podobě nanášených tenkých vrstev, které jsou navzájem spojeny fyzikálními nebo chemickými vazbami, dle použitého technologického postupu. Na rozdíl od tradičních výrobních postupů není třeba vyrábět složitou a drahou formu pro odlití výrobku technologií odlévání a tradiční výroby pomocí obrábění materiálu.

Tento virtuální model se vytváří pomocí tzv. CAD, systémy pro tvorbu návrhů, které jsou nástroji pro lepší vytvoření, analýzu, úpravu i optimalizace návrhu. Ten je exportován do formátu obvykle STL a později zpracován slicerem 2.5.

2.2 Stručná historie 3D tisku

Na počátku 3D tisku stála stereolitografie. Ta bylo objevena a zaznamenána v roce 1984 Bc. Charlesem W. Hullem. Tato technologie byla dále zdokonalována. Přibližně v době vznikla FDM technologie, pro kterou je tato práce hlavně určena, a také technologie SLS. V roce 2005 vzniká projekt RepRap z něhož vychází první vydaná verze tiskárny, nazvaná Darwin.

2.3 Technologie 3D tisku

V této kapitole si jednotlivé technologie 3D tisku přiblížíme. Hlavní technologií pro danou úlohu je FDM/FFF.

2.3.1 Stereolitografie

Jedná se o nejstarší formou 3D tisku. Tato metoda spočívá v osvícení tekutého fotopolymeru a tím dochází k jeho vytvrzování. Nejčastějším zdrojem záření je UV laser. Fyzický objekt je tvořen postupným vytvrzováním jednotlivých vrstev fotopolymeru. Tento princip tisku rozdělením na jednotlivé vrstvy je společný pro všechny technologie 3D tisku. Při tisku je tištěný objekt zvednut o výšku vrstvy a jeho místo zaplní tekutý fotopolymer. Vrstvu tvoří nasvícení UV laserem a tím vytvrzení tohoto materiálu. U složitějších výrobků při tisku je nutné vytvořit soustavu nosných vzpěr na tisk mimo nosnou plochu předchozí vrstvy modelu. Tyto podpěry se musí později odstranit. Výtisky mají vysokou přesnost díky možnosti použití malého bodu laseru a výšky vrstvy obvykle 0,05 mm. Jedná se o drahou technologii z důvodu použití drahého polymeru i tiskového zařízení.

2.3.2 FDM/FFF - Fused Deposition Modeling

Dalším typem 3D tisku je metoda FDM, která byla vynalezena a patentována Bc. Scottem Crumpem a jeho ženou, kteří spolu založili společnost Stratasys. Stejný princip používá technologie FFF pojmenovaná členy komunity RepRap. Tato technologie je nejrozšířenější pro 3D tiskárny na trhu, které jsou k dostání od hobby kvality až po profesionální. Princip spočívá v tlačení tzv. „struny“, kterou přestavuje úzké vlákno o malém průměru, z termoplastu, jež je tavena v tiskové hlavě a pomocí trysky vytváří jednotlivé vrstvy výrobku. Mezi nejpoužívanější termoplasty používané v FDM/FFF jsou ABS a PLA (viz. kapitola 2.6).

2.3.3 SLS - Selective Laser Sintering

Metoda, při níž vysoce výkonný laserový paprsek taví práškový tiskový materiál na požadovaný tvar vrstvy výrobku. Při procesu se postupně tiskový materiál nanáší ve výšce odpovídající výšce vrstvy. Materiál i technologie jsou drahé, ale výhodou je tvorba velmi přesných, mechanicky velmi odolných materiálů. Tento princip se v nekomerční sféře téměř nepoužívá.

2.3.4 DMLS - Direct metal laser sintering

Používá velice přesný a vysokoenergetický laser pro spékání práškových kovů a slitin k vytváření funkčních kovových komponent z návrhů 3D objektů. Ob-

GCode	Návratová hodnota	Popis
GXX, MXX	ok	vše proběhlo bez chyby
	ok T:xxx	vše proběhlo v pořádku a teplota extruderu je xxx
	ok B:xxx	vše proběhlo v pořádku a teplota vyhřívané podložky je xxx
	ok T:xxx B:yyy	vše proběhlo v pořádku a teplota extruderu je xxx a teplota podložky je yyy
	rs: xxx	znovu přeposlat řádek xxx
	!!	tiskárna přestala pracovat správně a chyba je neopravitelná

Tabulka 2.1: Návratové hodnoty GCodů[5]

jekty se pak ale nechávají „zapéct“, kde dojde k finálnímu spojení a výrobek se vyrovná frézovanému.

2.4 GCode

GCode vznikl jako podmnožina NGC interpretu. NGC je sada instrukcí pro číslicové řízení pro číslicové obráběcí funkce, které podporují obráběcí stroje mající 3 až 6 os známé jako CNC [3]. GCode je podmnožina NGC příkazů a některých dalších, ale ve stejném formátu jako NGC. Jsou to textové příkazy ve formě kódu instrukce a parametrů, jakými jsou pozice, rychlost tisku, množství materiálu, čekání na správnou teplotu a výměnu tiskových hlav.

V rámci ovládání tiskárny slouží tyto instrukce jako ovládací protokol. Představuje architekturu klient-server, kdy ovládací program jako klient posílá textové příkazy ve formě GCode s kontrolními součty do tiskárny, která na ně náležitě odpovídá potvrzením a někdy i s informativními parametry, jakými jsou teplota tiskové hlavy nebo teplota podložky.

Hlavními kategoriemi jsou pohybové uvozené písmenem G a víceúčelové uvozené písmenem M. Oboje následované číslem instrukce.

Návratové kódy jsou pro instrukce ve formátu dle tabulky 2.1.

2.5 Nástroje pro generování GCode instrukcí

Jsou to programy, které dělí virtuální model, obvykle ve formátu STL, na vrstvy a tyto vrstvy převádí pomocí algoritmů na graf instrukcí, kde uzel je cílová pozice a hrana představuje atributy pro pohyb. GCode instrukce jsou textovou reprezentací tohoto grafu a následně jsou odesílány tiskárně. Tyto programy převádí jednotlivé vrstvy modelu na GCode příkazy, pomocí kte-

Parametr	Popis
Xnnn	Souřadnice na ose X, nnn může být celé číslo i desetinné.
Ynnn	Souřadnice na ose Y, nnn může být celé číslo i desetinné.
Znnn	Souřadnice na ose Z, nnn může být celé číslo i desetinné.
Ennn	Délka struny spotřebovaná po délce vykonávané příkazem.
Fnnn	Rychlost tisku v mm/min.
Snnn	Značka, podle které se rozhoduje se zasažením koncového sensoru, jestli bylo dosaženo konečné souřadnice (S1 má se kontrolovat, S0 ignorovat, výchozí je S0).

Tabulka 2.2: Nejpoužívanější parametry GCody[5]

GCode	Parametry	Popis
G0	Xnnn Ynnn Znnn Fnnn Ennn Fnnn Snnn	Rychlý lineární pohyb s parametry
G1	Xnnn Ynnn Znnn Fnnn Ennn Fnnn Snnn	Lineární pohyb
M106	Pnnn Snnn Innn Fnnn H:nn:nn... Rnnn Tnnn	Zapnout ventilátor, parametry ovládají rychlost ventilátoru, citlivost na teplotu a monitorování teplot pro spuštění.
G92	Xnnn Ynnn Znnn Ennn	Nastavení absolutní pozice. Nestane se žádný fyzický posuv.
M107		Vypnout ventilátor.
M104	Snnn	Nastavit teplotu extruderu. Snnn požadovaná teplota.
G28	X Y Z	Přesunout se na domovskou pozici, pokud jsou nastaveny parametry, poté se tisková hlava přesune na domovskou pozici na dané ose.
G90		Nastavit na absolutní pozicování, souřadnice na osách jsou od počátku, ne od poslední souřadnice.
G91		Nastavit na relativní pozicování, souřadnice na osách jsou od poslední souřadnice, ne od počátku.

Tabulka 2.3: Nejpoužívanější GCody[5]

.

rých se po odeslání tvoří fyzický tisknutý objekt. Tyto typy programů rozhodují o teplotách a rychlosti tisku a tím ovlivňují výslednou kvalitu výrobku. Příklady jsou komerční Cura a Craftware, opensource Skeinforge, Slic3r a MatterSlice.

2.6 RepRap

„*RepRap je free desktop 3D tiskárna schopná tisku plastických 3D objektů.* "[2] Opensource komunitní projekt pro návrh 3D tiskárny, která je schopna se sama replikovat, to znamená co nejvíce dílů je možné pomocí jiné 3D tiskárny vytisknout. Znamená to také, že každý si ji může vyrobit a vylepšit a své vylepšení odeslat zpět ke komunitě. Tiskárny jsou většinou FFF viz. 2.3.2, které používají materiály ABS a PLA.

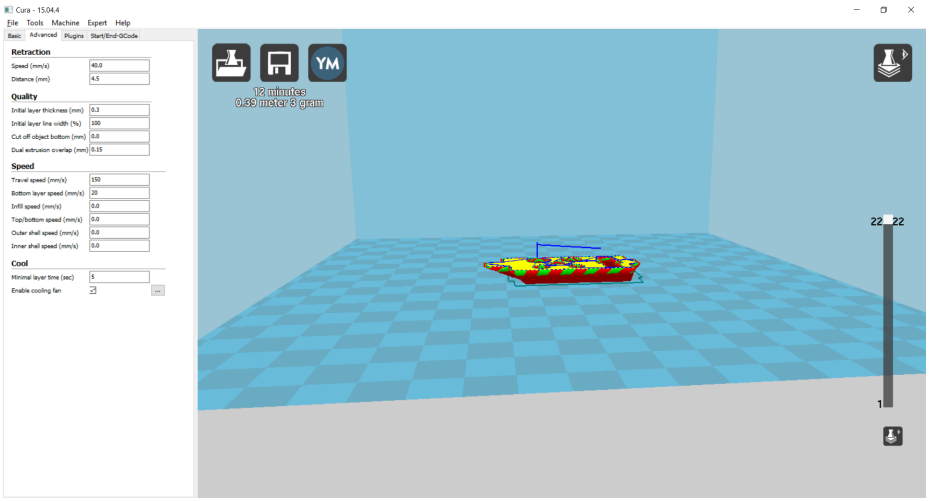
Polylactid acid (PLA) je organický materiál, který je biologicky odbouratelný, ale zároveň není tak mechanicky a teplotně odolný jako ABS. Na rozdíl od ABS ale není při chladnutí tolik náchylný k deformacím způsobených smršťováním. Nepotřebuje pro tisk vyhřívací podložku na správné slévání jednotlivých vrstev. Díky těmto vlastnostem se dají tisknout větší předměty než pomocí jiných materiálů. Tento materiál je zároveň méně pružný a má vyšší stupeň lesku než ABS.

Akrylonitrilbutadienstyren (ABS) je hojně využívaný v průmyslové praxi a v automobilovém průmyslu. Je teplotně a mechanicky odolnější než PLA. Pro jeho tisk je třeba mít vyhřívanou podložku, aby se vlivem smršťování objekt neoddělil od podložky.

Pro své tiskárny si vyvíjí i vlastní verze firmwaru, které se starají o převádění GCode příkazů do fyzického světa v podobě pohybu motor, sepnutí vyhřívání, přečtení hodnoty senzory, apod. Musí být nakonfigurovány a postupně zkalibrovány podle daného vybavení tiskárny.

2.7 Řídící software

Jsou to programy, které slouží ke komunikaci s tiskárnou. Obvykle to nejsou jen hloupé odesílače, starající se o odesílání GCode instrukcí, ale obvykle jej umí zobrazit ve formě 3D modelu nebo náhledu aktuální vrstvy. Řada z nich také podporuje manuální ovládání tiskárny. Přímo komunikují za pomoci příkazů, obvykle za pomoci seriového portu nebo usb, který obsahuje popis pro zařízení typu seriový port.



Obrázek 2.1: Cura

Kapitola 3

Analýza a návrh

Tato kapitola se zabývá analýzou existujících řešení dále pak specifikací funkčních a nefunkčních požadavků pro nové řešení aplikace a jejího návrhu.

3.1 Existující řešení

Z dostupných zdrojů podle analýzy vyplynulo, že existují dvě kategorie programů pro simulaci a zobrazení GCodu, které si nyní určíme.

3.1.1 Zobrazovače GCodu

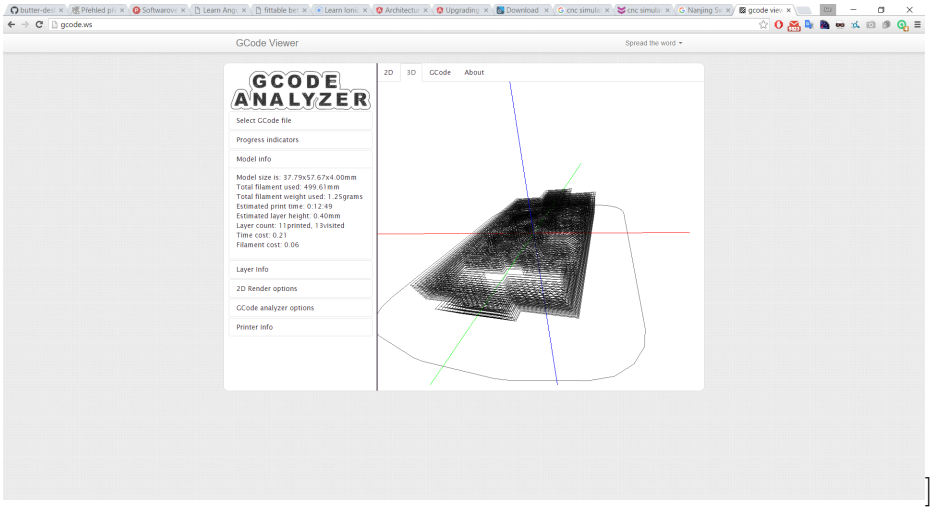
Jsou to programy a webové nástroje pro grafické zobrazení GCode instrukcí. Jejich funkcionalita vypadá následovně. Po vložení souboru s GCodem se obvykle zobrazí GCode v textové a editovatelné podobě. Zároveň vykreslí výrobek samotný v grafické podobě. Vykreslení probíhá najednou a tento program neobsahuje prvky pro postupnou interpretaci příkazů GCode na postupné vykreslování vrstev. Nedá se postupně krokovat ani urychlovat simulace. Zástupci jsou http://gcode.ws a https://nraynaud.github.io/webgcode/ .

3.1.2 Simulátory GCodu

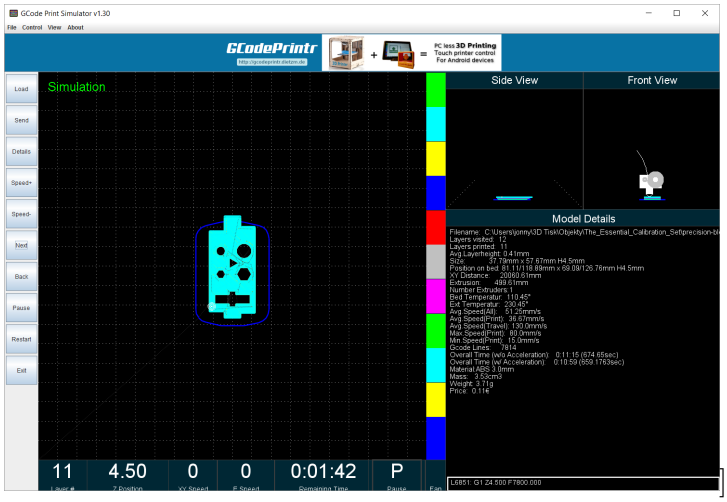
Oproti zobrazovačům simulátory umožňují kontrolu interpretování GCodu. Uživatel vidí průběh tisku a může ovlivnit buď průběh nebo vykonávání. Ovlivnění může probíhat pomocí kontroly vykonávání jako je spuštění, pozastavení, zrychlení, zpomalení, přetočení, zastavení. Fungují taktéž na vložení GCodu v podobě souboru a postupném provádění simulace i zobrazení samotného tisku. Zástupci jsou GCodeSimulator a CNC Simulator Pro.

3.1.2.1 GCodeSimulator

Tato aplikace je přímo vyvinutá pro RepRap tiskárny, ale simuluje i ostatní tiskárny. GCodeSimulator má následující funkce: načtení GCodu, zobrazí in-



Obrázek 3.1: GCode zobrazovač http://gcode.ws

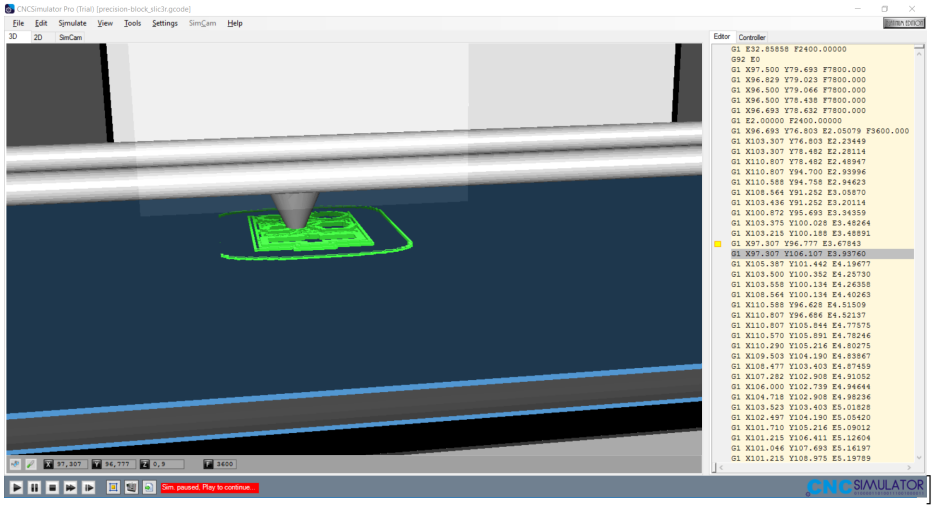


Obrázek 3.2: GCodeSimulator

formace o vloženém GCodu, umožňuje řídit simulaci, obsahuje funkci zoom, umí přejít na další vrstvy, vrátit se z předchozích vrstev, pozastavit simulaci a zastavení a upravit a změnit GCode.

3.1.2.2 CNC Simulator Pro

Simulátor představuje pro všechny CNC obráběcí stroje a také 3D tiskárny. Tento simulátor umí interpretovat GCode. Program je více zaměřen na simulaci obráběcích strojů, kde umožní zadat přímo řídící systém stroje. Chybí mu možnost skutečného připojení k tiskárně.



Obrázek 3.3: CNC Simulator Pro

3.2 Návrh řešení

3.2.1 Výběr programovacího jazyka

Pro vývoj programu nebyl předem určený žádný programovací jazyk. Bylo potřebné, aby tento jazyk umožňoval jistou funkcionalitu v podobě knihoven nebo přímo v jazyku samotném. Důležitou součástí vývoje bylo umožnění snadného vývoje CLI, WebAPI a GUI(zejména ve formě kreslení právě tištěné vrstvy). Jelikož pro tento program nebyla příliš důležitá jeho rychlost, nebylo třeba použití kompilovaného jazyka. Při výběru rozhodovala i možnost rychlého vývoje v daném jazyku a použití volně šiřitelné licence v podobě GNUGPL. Musí taktéž umět vytvořit a obsluhovat virtuální sériový port. To znamená nastavit rychlost přenosu a další nastavení, jakým je délka dat, parita, bitů zastavení a dalších signálů pro řízení toku dat. Typickým nastavením je 8N1, který znamená 8 bitů dat a 1 stop bit bez parity.

- Podle mých osobních preferencí byly vhodné tyto programovací jazyky Java, C/C++, Python, Javascript v podobě NodeJS, C#.
- Jelikož hlavní platforma bude Linux, tím se dá vyloučit C# s frameworkem .NET, protože v podobě platformy Mono bohužel neobsahuje pro program potřebné knihovny, zejména knihovnu pro práci se sériovým portem ani s pseudoterminály. Proto tato platforma není vhodná.
- Java by splňovala kladené požadavky, ale vývoj pomocí ní by nebyl příliš svižný.
- Pomocí C++ přichází v úvahu framework QT, který splňuje všechny podmínky i knihovny pro zobrazení a práci se sériovým portem. Tato

možnost není vhodná pro inteligentní zpracování CLI. Byla by složitá implementace webového API serveru.

- NodeJS je relativně nová platforma, která má velké možnosti, ale problém nastává s vytvořením a obsluhou virtuálního seriového portu. Neobsahuje žádné vysokoúrovňové knihovny a pro práci s virtuálním sériovým portem by bylo třeba vytvořit plugin pomocí C++, protože přímo tuto funkcionalitu neobsahuje.

- Python je dle dostupných informací nejlepší variantou. Splňuje všechny požadavky a navíc má vynikající knihovny pro zpracování CLI, Web API server je taktéž jednoduchý. Všechny mnou používané datové struktury jsou implementované pro nezávislý chod pro vícevláknovou aplikaci. Pro zpracování vstupů je vhodná knihovna Asyncio, která umožňuje čtení ze vstupů pomocí jediného vlákna. Pro zobrazení se používá Tkinter avšak pro tento projekt není tato knihovna příliš vhodná na pravidelné a rychlé překreslování, proto je mnohem vhodnější knihovna PyGame. Velké množství dostupných knihoven, jednoduchost jazyka a jeho velké rozšíření ve sféře 3D tisku přispělo k jeho zvolení.

3.3 Seznámení s jazykem Python

Je to jazyk určený pro rychlý vývoj. Je interpretovaný s možností překladu do tzv. bytekódu, tj. kód, který vzniká při interpretaci a poté je přeložen do strojového kódu spustitelného přímo počítačem. Tento jazyk není určen pro náročné výpočetní úlohy, protože je mnohem pomalejší než Java nebo C++. Poskytuje vícenásobné dědění oproti ostatním. Ačkoliv je tento jazyk typovaný, jsou všechny datové typy kromě primitivních považovány za objekty. V třídách jsou jako privátní metody považovány metody začínající na ___metoda, je to pouze konvence, jinak se chovají jako public metody. Objekty mají i tzv. magické metody pro zpracování jako tzv. iterátory, průchod objektem, operace sčítání, odčítání a podobně.

Existují dvě hlavní verze jazyka Python 2.7.x a 3.x.x, které vzájemně nejsou kompatibilní. Starší je používaná pro zachování starších programů a nová obsahuje lepší konstrukce a podporu kódování. V nové verzi přibyly generátorové notace a další usnadnění vývoje pro tento jazyk. Knihovna Asyncio vyžadovala verzi 3.4 a vyšší, tento program je tedy pro verzi 3.4 a vyšší.

Nyní je třeba představit si knihovny. Začneme knihovnou na asynchronní zpracování vstupů a výstupů.

3.3.1 Asyncio

Knihovna slouží pro běh paralelních úloh pomocí jediného vlákna a událostí se na ní odehrávající. Tyto události mohou být různých typů. Pomocí událostí

začínají zpracovávat data přicházející z tzv. generátorů, kterým se v Pythonu ve verzi 3.4 říká yield. Tento generátor má velice malou paměťovou náročnost a je schopem pracovat ve chvíli, kdy jsou data připravena na zpracování. Po dokončení úlohy lze nastavit funkce, či metody pro další zpracování. Jednou z mnoha důležitých výhod je implementace pro obsluhu vstupů a výstupů, které jsou neblokované. Používány jsou pro přístup z příkazové řádky, webového API i na čtení ze sériového portu a zápisu. Bez použití této knihovny by se obsluha těchto vstupů musela rozdělit do vláken a to tak, že by se každý vstup musel dát do samostatného vlákna, protože jinak poskytují blokovaný přístup pro čtení.

Tato knihovna se ukázala býti nevhodnou pro implementaci, protože nedokáže správně pracovat s blokovacími zařízeními. Při použití se celá smyčka s událostmi zastaví, proto není schopná asynchronně obsluhovat vstupní i výstupní zařízení.

3.3.2 Click

Knihovna se používá pro zpracování CLI, uživatelský vstup pomocí příkazové řádky. Umí zpracovávat parametry i vstupy z příkazové řádky. Při zpracování je třeba zadat datový typ hodnoty, aby se ho knihovna pokusila načíst. Pokud se nedá pomocí takto načtené hodnoty zkonstruovat, daná proměnná zvoleného typu je vytvořena a později je vyvolána výjimka typu ValueError. Zároveň pro argumenty příkazové řádky generuje nápovědu. Zvládá vnořené funkce a tzv. líné zavádění[2] podpříkazů a rutin. Pro výstup do terminálu se používá funkce echo, která má oproti klasické funkci print tu výhodu, že umí správně fungovat i na nesprávně nastaveném terminálu a má stejné rozhraní ve verzích jazyka Python, 2.x a verze 3.x. Podporuje vyspělou formu pro zpracování a validování příkazů s použitím dekorátorů click.command pro nastavení obslužné funkce pro určitý typ příkazu a stejně validátor[3] pro rozhodování o přijetí vstupní hodnoty.

3.3.3 PyGame

Knihovna je zvolena pro vysokou míru optimalizace grafických prvků. Tato knihovna umí používat nízkoúrovňové knihovní funkce pro vykreslování a audio. Zejména ve formě knihovny SDL pro 2D grafiku a pro 3D grafiku OpenGL. Pro naši aplikaci se využívá zejména knihovna SDL, která je zapouzdřená v balíku *pygame.draw*. Kreslení probíhá pomocí *pygame.draw.line*, které používá znalost o poloze bodů pro začátek a konec. Toto kreslení probíhá na abstraktní kreslicí plochu, která se jmenuje Surface. Aby byla animace takto možná při

[2] Líné zavádění(lazy initializing) znamená, že je zavedena do paměti až ve chvíli, kdy je opravdu potřeba pro vykonání příkazu.

[3] Funkce, které rozhodují o přijetí hodnoty podle touto funkcí definovaných kritérií.

větším množství FPS, je třeba využít techniku tzv. „back bufferu“, která spočívá v použití paměťového bufferu, do kterého se kreslí. Vykreslení probíhá pomocí překopírování tohoto vykresleného pomocného bufferu do výstupního bufferu. Vzhledem k potřebným závislostem knihovny je obtížná její instalace a vyžaduje velké množství externích knihoven s jejich vývojovými prostředky.

3.3.4 Tkinter

Je to knihovna pro tvorbu GUI, která je též známá jako Tk. Tato knihovna je multiplatformní pro Windows, Mac a na Unixu založených systémech. Rovněž jí podporují programovací jazyky Python, Perl, Ruby a Obsahuje sadu ovládacích prvků jako jsou tlačítka(button), zaškrtávací pole(checkbox) a tlačítka s možností volby(radio button). Obsahuje též řídící objekty pro grafické rozhraní jako jsou rámy(frame), canvas a další. Nejzajímavější je pro tuto aplikaci objekt Canvas. Slouží ke kreslení 2D grafiky pomocí primitivních grafických objektů jako je úsečka(line), oblouk(arc), obdélník(rect), ovál(oval), bitmapu(bitmap) a další.

3.3.5 Flask

Knihovna Flask je webový server, který nabízí jednoduché použití pro vytvoření WebAPI. Umožňuje jednoduché vytvoření cest (route) pomocí dekorátoru *@app.route(cesta)*, kde cesta může být parametrizována s parametrem peripheral, viz ukázka 3.1. Návratová hodnota této funkce je předána po http protokolu v tomto případě pomocí metody GET a v tomto případě ve formátu JSON.

Ukázka kódu 3.1: Ukázka kódu pro cestu

```
@app.route("/info/<peripheral>")
def api_info(peripheral):
    return printer.get_info_api(peripheral)
```

3.4 Funkční požadavky

Tyto požadavky jsou nejdůležitější pro daný program, či systém. Jsou požadavky, které definují funkcionalitu systému i jeho komponent. Požadavky, které musí program splňovat.

- Program musí být schopen emulovat 3D tiskárnu.
- Program musí umět nastavit poměr času virtuálního k reálnému a tím ovlivňovat, jaký časový úsek virtuálního času má odpovídat jedné sekundě času reálného. Musí být schopen měnit tento poměr v průběhu tisku.

- Program musí umožnit vytvoření virtuálního sériového portu a pomocí tohoto portu připojit řídící software.
- Program musí poskytovat následující rozhraní:
  - Uživatelské rozhraní přístupné z příkazové řádky.
  - Uživatelské rozhraní pro zobrazení aktuálního tisku.
  - Uživatelské rozhraní pro přístup k informacím o probíhající simulaci pomocí webového rozhraní, tzv. Web API.
- Program musí umět načítat nastavení tiskárny z konfiguračního souboru.
- Program musí zobrazovat aktuální tisk v grafické podobě.

3.5 Nefunkční požadavky

Jsou to požadavky, které na rozdíl od funkčních mají doplňující charakter. Jsou to kvalitativní požadavky.

- Program má přijímat parametry v podobě argumentů příkazové řádky při spuštění programu.
- Program má fungovat v prostředí operačního systému Linux.
- Program má při použití určitých argumentů spustit grafické vykreslování simulace.

3.6 Případy použití

Případy použití jsou velice důležitá součást analýzy. Případy užití představují nejčastější případy interakce uživatelů s programem. Jsou popsány v krocích. Slouží k popisu použití funkcí a možností programu. Uživatel je člověk, který používá program. Nebudou zde uváděny všechny případy užití, ale jen ty nejdůležitější. Tento krok je důležitý z hlediska interakce s uživatelem.

- Spuštění simulace z příkazové řádky s parametry.
  - Uživatel si připraví ovládací program.
  - Uživatel zadá parametry, jakými jsou baudrate pro nastavení rychlosti sériového přenosu, dále soubor s nastavením pro tiskárnu a koeficient poměru času virtuálního tisku k reálnému.
  - Uživatel zadá parametry pro spuštění simulace.
  - Uživatel do ovládacího programu zadá parametry sériového portu, tj. rychlost a název portu nebo jeho označení.

  - Uživatel po dokončení simulace dostane informaci o dokončení.

- Spuštění simulace se zobrazením tisku.
  - Uživatel si připraví ovládací program.
  - Uživatel zadá parametry včetně přepínače –gui na spuštění simulace se zobrazením.

- Ovládání simulace z CLI.
  - Uživatel si připraví ovládací program.
  - Uživatel zadá parametry simulace a spustí ji.
  - Uživatel zadává do příkazové řádky příkazy programu: *info motors*.
  - Uživatel dostává informaci o stavu motorů.

- Získávání informací o simulaci pomocí WebAPI.
  - Uživatel si připraví ovládací program.
  - Uživatel zadá parametry simulace a spustí ji.
  - Uživatel přistoupí k webovému API pomocí IP adresy a portu.
  - Uživatel zjistí informace o probíhající simulaci pomocí tvaru koncových bodů v url, jakým je například */info/{motor, bed, . . . }*

3.7 Popis vnitřních vlastností virtuální tiskárny

Z hlediska efektivity pro běh tohoto emulátoru se nejprve zvolí vhodný počet iterací pro výpočet vnitřního stavu tiskárny. Pro určení skutečné doby běhu pohybové instrukce je třeba znát její aktuální parametry pro rychlost pohybu a množství materiálu, které se má při instrukci použít. Pohybové instrukce mají danou rychlost tiskové hlavy v jednotkách milimetrů za minutu (mm/min), které je označováno jako feedrate. Pokud se při této činnosti má provádět tisk, je třeba mít teplotu tiskové hlavy na požadované hodnotě pro začátek tisku. Ohřev tiskové hlavy je založen na lineárním modelu stejně jako pro chladnutí tiskové hlavy. Pokud dochází k nanášení materiálu na tiskovou plochu, instrukce specifikuje délku tiskové struny v milimetrech (mm), která se rovnoměrně taví po zadané trajektorii tisku.

Z funkčních požadavků vyplynula potřeba změny měřítka času. Měřítko umožňuje měnit čas simulace proti reálnému času tisku. Uživatel si nastaví podle svých potřeb poměr času simulace k reálnému času tisku pomocí desetinného čísla. Poměr se vyjadřuje k jedné sekundě reálného času tisku. Změna poměru času se může provést i během probíhající simulace tisku.

Pro zjednodušení celé situace se vybere vhodný počet iterací za sekundu, aby se po změření času výpočtu mohla stanovit doba pro případné uspání

vlákna. Pokud se daný výpočet provede rychleji než pro něj stanovený čas je na zbytek času uspáno. Toto řešení je dostatečně přesné a zároveň šetří procesorový čas. Bylo zvoleno, aby tiskárna svůj stav vypočítávala svůj stav 120 krát za sekundu.

Pro vykreslení se stanoví maximální počet snímků za sekundu, také označováno zkratkou fps(frames per second), aby se vše zvládlo plynule vykreslit a nevznikaly časové prodlevy potřebné pro vykreslení projevující se trháním obrazu. Toto trhání se nejvíce projevuje v počítačových hrách při nedostatečném počtu snímků za sekundu. Minimální hodnota pro plynulý obraz je 24 snímků za sekundu, proto vnitřní běhová smyčka pro výpočet stavu tiskárny musí spočítat svůj další stav minimálně 24 krát za sekundu při 24 snímcích za sekundu. Tato hodnota však není v počítačové grafice vyhovující, proto byla zvolena hodnota 60 fps.

3.8 Rozdělení na komponenty

Virtuální tiskárna je sestavena z jednotlivých komponent s vlastními parametry. Tyto parametry jsou nastavovány pomocí konfiguračního souboru, který pomocí definic v jazyce YAML, který poskytuje člověku dobře čitelný formát uložení počítačem zpracovatelného objektu[4]. Jednotlivé komponenty jsou nastaveny pomocí hodnot z konfiguračního souboru ve formátu YAML.

Každá komponenta má metodu, která se stará o vykonání dalšího přechodu do následujícího stavu, a každá komponenta má vlastní implementaci této metody. Každá komponenta má vlastní vnitřní stav, podle kterého se řídí další chování komponenty. Každá komponenta si eviduje svůj stav. Podle stavu komponenty se řídí její chování v dalším kroku simulace.

Každá komponenta má zároveň metodu info pro zjištění informací pro webové API i CLI rozhraní, ať už ve formě formátovaného textu nebo ve formátu JSON.

3.8.1 Komponenta tiskárna - Printer

Reprezentuje objekt tiskárny. Je definován svými komponenty. Při každé výpočetní smyčce se volá metoda pro výpočet dalšího kroku simulace komponenty, které mají provést přechod do dalšího stavu. Tato komponenta se stará o časové měřítko, připojení ke vstupům i výstupům. Tyto vstupy a výstupy řídí pomocný objekt AsyncInputs obsahující vlastnosti připojení. Virtuální tiskárna tiskne pouze, pokud je v aktivním stavu.

3.8.2 Komponenta motor - Motor

Reprezentuje objekt motoru v podobě pohybu po ose. Při nastavení parametrů pro cílovou destinaci, se rychlost spočítá pomocí zpracování GCode příkazu na čas potřebný k vykonání posuvu a délce jednotlivých kroků v běhové smyčce

tiskárny. Nemůže dojít k vykonávání posuvu, pokud se má nanášet materiál a zároveň tisková hlava a vyhřívaná podložka, je-li přítomná, nemá dostatečnou teplotu.

3.8.3 Komponenta tiskové hlavy - Extruder

Reprezentuje objekt tiskové hlavy a podavače tiskové struny. Pro ohřev trysky se využívá lineární model pro růst teploty dokud nedojde k dosažení požadované teploty, případně spuštění ventilátoru na udržení správné teploty. Tento objekt také sleduje spotřebu materiálu a množství materiálu potřebného v dalším kroku. Chladnutí tiskové hlavy je také lineární.

3.8.4 Komponenta vyhřívané podložky - Bed

Reprezentuje objekt vyhřívané podložky. Pro správný tisk je třeba dosáhnout správné teploty podložky pro daný materiál. Chladnutí podložky je taktéž lineární.

3.8.5 Pomocný objekt GCode

Pomocný objekt GCode provádí pomocné výpočty. Hlavně se to týká výpočtu vzdálenosti a doby potřebné k vykonání instrukce, stejně jako k určení komponent, které se potřebují před vykonáváním předpřipravit, např. dosažení určité teploty a podobně. Tento objekt také řídí celý životní cyklus GCode instrukce.

3.8.6 Pomocný objekt Inputs

Tento pomocný objekt slouží ke sjednocení obsluhy objektů, které se starají o vstupy a výstupy. Startuje a stopuje jejich vlákna pro běh.

Kapitola 4

Realizace

Tato kapitola popisuje technologie a postupy použité při realizaci práce. Bude zaměřena zejména na technickou stránku práce.

4.1 Rozdělení do komponent

Pro přehlednost a snadnou rozšiřitelnost bylo rozhodnuto o rozdělení jednotlivých částí do komponent. Tyto komponenty pro své společné rysy mají rodičovskou třídu Peripheral, od které jsou odvozené. Společné jsou metody *action*, které slouží k vykonání jedné iterace běhové smyčky pro změnu stavů tiskárny. Metoda *config* slouží k nastavení hodnot z konfiračního souboru dané komponenty, kde parametr je objekt instance Dictionary objektu v Pythonu. Společnými atributy jsou stav, *STATE* se stavy *NOT_READY*, kdy komponenta není připravena k vykonávání činnosti, *RUNNING*, kdy vykonává danou činnost a pravidelně je na ni tedy volána metoda *action* této komponenty a *FINISHED*, kdy dokončila zadanou práci.

Provádí se pomocí statické metody uvnitř objektu GCode, která zabraňuje vícenásobnému vložení do slovníku MAP, který mapuje názvy instrukcí na obslužné funkce. Tento objekt GCode slouží ke zpracování a obsluze GCode příkazů.

Výpočty potřebné k správnému vykonávání GCode instrukcí jsou implementovány pomocí obslužných funkcí pro GCode příkazy. Pro zjištění množství vytlačovaného materiálu je třeba zjistit, jak dlouho při dané rychlosti posuvu potrvá cesta do koncového bodu. Množství vytlačovaného materiálu je rovnoměrně rozděleno do tohoto časového úseku. Tím je dána rychlost posuvu materiálu v objektu představujícím aktuální extruder.

4.2 Konfigurační soubor

Konfigurační soubor pro tiskárnu je ve formátu YAML, viz ukázkový soubor 4.1. Tento formát umožňuje definovat parametry jednotlivých komponent. Soubor je uvozen začátkem YAML dokumentu —. Na nejvyšší úrovni jsou názvy jednotlivých komponent. Na nižších úrovních oddělených tabelátorem jsou parametry pro jednotlivé komponenty. Pro komponentu BED se dají definovat její teplotní charakteristiky v podobě maximální teploty a velikost růstu za sekundu ve stupních Celsia. MOTORS struktura udává společné atributy pro osy. Atributy pro každý motor můžou být nastaveny samostatně. Velikost tiskové plochy udává MAX_POSITION u jednotlivých motorů. U komponenty EXTRUDER se dají nastavit stejné parametry jako u komponenty BED až na položku HEATED. Veškeré teplotní jednotky jsou ve stupních Celsia. U extruderu je možné nastavit barvu tiskového materiálu parametrem COLOR. Pokud by bylo více instancí typu EXTRUDER, jsou postupně číslovány. Pro přidání dalších objektů typu EXTRUDER je nutné uvést je v tomto konfiguračním souboru postupně číslované od jedné. Další objekt typu EXTRUDER by se přidal pomocí uvedení direktivy EXTRUDER1:, další s přičítáním ve formě EXTRUDER(číslo extruderu) a dále pokud jsou nutné změnit i parametry, stejně jako u objektu EXTRUDER.

Ukázka kódu 4.1: Ukázkový konfigurační soubor formátu YAML pro tiskárnu Prusai3.

```
---
BED:
    #is the bed heated 1 - true, 0 - false
    HEATED: 1
    #maximal temperature of bed in degrees celsius
    MAX_TEMPERATURE: 120
    #temperature grow in degrees celsius
    TEMPERATURE_GROW: 5
    #cooling in degrees celsius
    COOLING: 1

#sets all motors with this values
MOTORS:
    #speed in rpms
    MAX_SPEED: 600
    #maximal motor position
    MAX_POSITION: 200
    #minimal motor position
    MIN_POSITION: 0
    #mm/sec
    NORMAL_SPEED: 50
```

```
    #home direction 0 - minimal, 1 - maximal position
    HOME_DIRECTION: 0
    MOTORX:
        #MAX_POSITION: 200
        #MIN_POSITION: 0
        #HOME_DIRECTION: 0

    MOTORY:
        #MAX_POSITION: 200
        #MIN_POSITION: 0
        #HOME_DIRECTION: 0

    MOTORZ:
        #MAX_POSITION: 200
        #MIN_POSITION: 0
        # > 0 - max, == 0 min
        #HOME_DIRECTION: 0

EXTRUDER:
    #maximal temperature of extruder in degrees celsius
    MAX_TEMPERATURE: 200
    #temperature grow in degrees celsius
    CONST_TEMPERATURE_GROW: 20
    #color of filament
    COLOR: "black"
```

4.3 Struktura hlavní běhové smyčky

Hlavní běhová smyčka je tvořena pomocí cyklu while viz. následující ukázka kódu [4.2], při kterém se volá metoda *action* na objekt, který je vytvořenou instancí třídy Printer. Protože třída Printer byla odvozena od třídy Peripheral, má vlastní implementaci metody *action*.

Ukázka kódu 4.2: Hlavní běhová smyčka aplikace

```
while self.printer.RUNNING:
        self.printer.action()
```

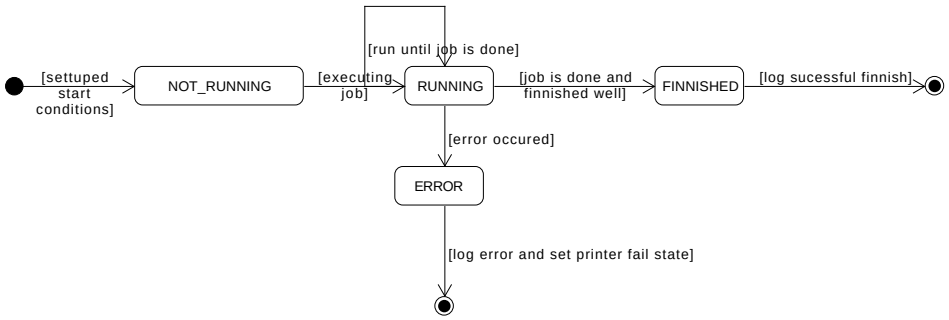
Seznam komponent představuje slovník COMPONENTS, kde klíčem je její jméno napsané v podobě velkých písmen. Pokud se zde může komponenta vyskytovat vícenásobně, jako například tisková hlava v podobě Extruder, pak je automaticky číslována a pomocí GCode se vybírá instrukcí T(číslo nástroje). Tento nástroj je vybrán jako hlavní a provádí zadané činnosti. Motory pro osy jsou taktéž odlišeny jménem pomocí názvu osy.

Implementace metody *action* v třídě Printer vypadá nasledovně 4.3.

Ukázka kódu 4.3: Běhová smyčka tiskárny

```
if self.STATE == "NOT_READY":
        self.gcode.processQ()
        self.STATE = "RUNNING"
        self.t_start_instruction = time.time()
        if self.STATE == "RUNNING":
                period =
                (1000 / self.NUM_OF_ACTIONS_PER_SEC)
                start_t = time.time()

is_running = False
for component in self.COMPONENTS:
        self.COMPONENTS[component].action()
        if self.COMPONENTS[component].is_running():
                is_running = True

if not is_running:
        self.t_end_instruction = time.time()
        self.time_real.append(self.t_end_instruction
                - self.t_start_instruction)
        self.STATE = "NOT_READY"

end_t = time.time()
spended_time = ((end_t - start_t) * 1000)

sleep_time = (period - spended_time)
if sleep_time > 0:
        time.sleep((sleep_time / 1000))
```

Pokud se tiskárna nachází ve stavu *NOT_READY*, vezme se z fronty GCode příkazů čekajících na zpracování první příkaz na řadě. Tento příkaz se zpracuje uvnitř třídy GCode, která se postará o jeho zpracování a zavolání příslušné metody ze slovníku příkazů. Klíčem je název instrukce a hodnotou je zpracovací funkce, která má parametr v podobě předzpracovaného objektu. Po zavolání této obslužné funkce se nastaví startovací hodnoty metody *start* u instance tiskárny. Ta následně zavolá metody *start* na jednotlivé komponenty s příslušnými argumenty. Metoda *start*, která je u každé komponenty implementována zvlášť, nastavuje vnitřní atributy jednotlivých komponent a mění stav na *RUNNING*, viz diagram4.1. Tímto stavem je komponenta považována za aktivní a má přidělenu práci. For cyklus na všechny komponenty volá metodu *action* a zároveň kontroluje pomocí metody is_running, jestli ještě nezůstala nějaká komponenta ve stavu *RUNNING*, tedy běžící, ale že



Obrázek 4.1: Stavový diagram pro Peripheral

přešla do stavu *FINNISHED*. Pokud byl daný výpočet proveden dříve než doba vyhrazená pro iteraci cyklu, je toto vlákno uspáno po zbytek doby.

4.4 Zpracování vstupů

Pro čtení a zápis z I/O, neboli vstupně/výstupních zařízení, existují jich dva druhy podle typu zařízení pro blokovací synchronní a neblokovací asynchronní. Synchronní, neboli blokovací zařízení pracuje po požadavku přístupu k zařízení, poté čeká na dokončení požadované operace, do té doby je vlákno programu pozastaveno. Asynchronní vstup a výstup je možný pouze u neblokovacího zařízení. Tento způsob nevyžaduje vyčkat na dokončení operace, ale při dokončení se zavolá obslužná funkce na zpracování dat.

Vzhledem k tomu, že vstup ze sériového portu a vstup z příkazové řádky jsou blokovací, současně jsou možnosti interpretované jazyka Python v realizaci obsluhy těchto zařízení pomocí jednotlivých vláken pro každé zařízení. Ačkoliv samotný interpret jazyka Python nevytváří další vlákna přímo v operačním systému, ale pomocí svého vlastního mechanismu, který se nazývá GIL, jež je popsán níže.

GIL je mechanismus, který zprostředkuje zámek pro vzájemné provádění kódu(mutual-execution lock). Drží vlákno interpretru jazyka, při jehož provádění se vyhne sdílení kódu, který není bezpečný pro provádění více vlákny najednou(thread-safe). V implementaci s GILem má každé vlákno interpretu vlastní jeden GIL. Používají ho interprety CPython a Ruby MRI. Z tohoto řešení vyplývají limity množství možného paralelismu dosaženého tzv. souběžnost jediného interpretu s mnoha vlákny. Toto řešení ovšem nezvládá využít více jader procesoru a využít tak naplno potenciál počítače. Výhodou je zjednodušení o vypuštění režie pro zámky, které jsou pro vícevláknový běh nutné k vyhnutí střetu s prací se stejnými daty. Díky tomuto mechanismu byla implementace interpretu Pythonu velice zjednodušena. Interpretr jazyka Python, ve

které nepředstavují jednotlivá vlákna knihovny threading systémová vlákna, je ve skutečnosti jednovláknový interpret s událostmi.

Souběžný přístup, neboli concurrency, je vlastnost programu, problému nebo algoritmu, kdy u něj nezávisí na pořadí a výsledek je stále determinovatelný. Pokud tuto vlastnost mají knihovny, lze je bez jakýchkoliv obav použít pro vícevláknovou aplikaci pod interpretem.

4.4.1 CLI

Vstup z CLI daného programu je zpracováván pomocí knihovny Click a probíhá následujícím způsobem. Při zpracování argumentů příkazové řádky se zjistí hodnoty a zapíší se do nastavení tiskárny a pomocí názvu konfiguračního souboru se po načtení nastaví atributy jednotlivých komponent tiskárny. Toto rozhraní poskytuje zároveň nápovědu pro jeho použití a při práci programu sděluje informace o průběhu tisku a stavu komponent tiskárny.

Argumenty pro spuštění aplikace jsou ve formátu krátký, dlouhý argument:

- -b, --baudrate - jsou parametry pro nastavení rychlosti spojení se sériovým portem v celočíselném formátu,
- -c, --config - jsou parametry pro nastavení názvu konfiguračního souboru ze kterého se načítají z adresáře *printer_config* nastavení pro daný typ tiskárny ve formě textu bez přípony yaml,
- -t, --timeeqrt - jsou parametry pro nastavení doby odpovídající jedné sekundě reálného času ve formě desetinného čísla pythonovského datového typu float,
- -g, --graphics - pro spuštění grafického zobrazení,
- -h, --help - pro zobrazení nápovědy.

Při ovládaní pomocí tohoto textového rozhraní se dá virtuální tiskárna po spuštění plně ovládat s těmito příkazy:

- help - zobrazí nápovědu pro dané rozhraní,
- C příkaz[GCode]- dovoluje zadat GCode příkaz,
- exit - ukončí aplikaci,
- info komponent[Název komponenty] - zobrazí informace o zadané komponentě pomocí jejího názvu,
- time nový_čas[float] - umožňuje změnit přepočet virtuálního času odpovídající jedné sekundě reálného času.

4.4.2 Zobrazení tisku

Zobrazení tisku probíhá pomocí knihovny PyGame. Aplikace využívá vykreslení okna s danými rozměry. Pro vykreslení tohoto okna se používá objekt typu Surface, který představuje kreslicí plochu. Při vytváření tohoto objektu se definuje jeho typ. Objekt Surface je reprezentován pamětí o velikosti kreslicí plochy, která je dána jejími rozměry, a nastavením pro daný typ pixelu, jež je reprezentovaný jednotlivými složkami RGB nebo RGBA, jejíž každá složka má velikost 8 bitů, dohromady 24 bitů barevných složek. Tato nastavení jsou udávána při vytváření objektu jako parametr flag, který představuje binární hodnoty pro jednotlivé složky, jako je typ paměti nebo maska pro barevné složky. Tento objekt může být buď hardwarového nebo softwarového typu. Softwarový je jen část systémové paměti a hardwarový představuje část videopaměti. Instance tohoto softwarového typu Surface se využívá pro kreslení. Moje aplikace na ni kreslí pomocí primitivních geometrických tvarů. Nejvíce se využívá úsečka(line). Parametry jsou počáteční a koncový bod, barva a instance objektu Surface. Vykreslují se tyto tvary, dokud se nevykreslí celá plocha. Tato instance je poté překopírována do vytvořené hardwarové instance třídy Surface. Toto se používá protože přímé kreslení do instance Surface hardwarového typu je velice pomalé. Z tohoto důvodu se pouze překopíruje vykreslená softwarová instance Surface do instance hardwarového Surface. Zobrazení je rozděleno na tři pohledy, kde každý pohled bude představovat Surface. Jsou zde pohledy shora, ze strany, zepředu. Jednotlivé vrstvy jsou barevně odlišeny.

4.5 Komunikace mezi programem a sériovým portem

Virtuální sériový port je nastaven jako skutečný sériový port. Je třeba nastavit jeho rychlost a parametry určující kódování a chování na přenosovém médiu. Tento sériový port, ačkoliv se chová jako skutečný, je realizován pomocí tzv. pseudoterminálu (PTY). Tyto tvoří páry pseudo-zařízení textového terminálu. Existují dva typy master(nadřízený) a slave(podřízený). Slave emuluje textový terminál, zatímco proces masteru má kontrolu nad slave terminály. V jazyce Python je pro spolupráci s těmito komponenty připravena knihovna pty. Pomocí ní se při zavolání metody pty.openpty() navrátí data v podobě datového typu N-tice, který má v tomto jazyce označení tuple, ve formátu (master, slave). Popisovače zařízení jsou ve formátu file descriptoru. Do proměnné serial_name se pomocí příkazu os.ttyname(self.SLAVE) uloží cesta k danému pseudoterminálu ve formě /dev/pts/(číslo terminálu). Toto popisuje následující kód 4.4.

Ukázka kódu 4.4: Čtení ze seriového portu

```
self.MASTER, self.SLAVE = pty.openpty()
serial_name = os.ttyname(self.SLAVE)
```

Master pseudoterminál je společný pro všechny pseudoterminály, ale vrací unikátní popisovače. Komunikuji s ním pomocí čtení a zápisu do blokovacího zařízení popisovačem masteru. V následujícím kusu kódu popisuji čtení v podobě čtení ze souborového MASTER deskriptoru. Navrátí přečtený byte informace v kódování ASCII. Toto popisuje následující ukázka kódu 4.5.

Ukázka kódu 4.5: Čtení ze seriového portu

```
def get_byte(self):
        return os.read(self.MASTER, 1)
```

Následující kód 4.6 popisuje výstup do seriového portu, kdy pomocí MASTER deskriptoru zapisuje obsah proměnné data, které jsou předtím převedeny pomocí metody bytes(data, 'utf-8') z kódování UTF-8 do bytů v kódování ASCII. Pokud se celý obsah proměnné data úspěšně zapíše, metoda vrátí hodnotu True jinak vrátí False.

Ukázka kódu 4.6: Zápis do seriového portu

```
def write_data(self, data):
        data += self.ENDL
        data = bytes(data, 'utf-8')
        if os.write(self.MASTER, data) == len(data):
                return True
        return False
```

4.6 Logika aplikace

Hlavní emulační myšlenka je v použití běhové smyčky tiskárny, v níž se volá na jednotlivé komponenty metoda action, která vykonává, krok simulace. Jednotlivé komponenty mají vlastní implementace této metody. Každá komponenta, která se vykonává má nastavenou konečnou hodnotu a jednotlivé kroky vykonává metoda action. Pro komponenty typu extruder nebo bed je třeba přípravy v podobě předehřátí před samotným tiskem. U metod pro zpracování jednotlivých instrukcí se ukazatele na ně postupně zapisují jako položky slovníku pod jmenným pojmenováním odpovídajícímu označení gcode instrukce s parametrem zpracované GCode instrukce. Zpracování GCode instrukce probíhá po jejím přijetí po daném médiu a následně probíhá rozebírání, parsování na objekt typu slovník, kde jsou hodnoty uloženy pomocí jmenných klíčů. S tímto objektem dále pracuje samotná dynamická metoda, která má tento objekt jako parametr metody, jež je uložená ve zvláštním slovníku pojmenovaná podle označení instrukce. Zároveň jsou všechny komponenty pomocí

metody start nastaveny na práci. Po dokončení zadaného úkolu přechází komponenta do stavu *FINISHED*. Pokud jsou dokončeny všechny komponenty, je GCode příkaz dokončen.

4.7 Komponenty aplikace

Z potřeby dělení tiskárny na jednotlivé komponenty se dále prohloubila na implementační úrovni potřeba mít dělení na jednotlivé třídy a pro společné části abstraktní třídy. Návrh aplikace v podobě class diagramu je na diagramu C.3.

4.7.1 Třída Peripheral

Peripheral je bázová třída pro komponentu. Je abstraktní, ačkoliv jazyk Python to neumí syntakticky vyjádřit. Jsou v ní deklarovány metody, jež jsou označované za virtuální, protože mají v těle pouze příkaz pass a jsou přepisovány v rámci dalších komponent. Tato třída obsahuje vlastnosti, atributy stejně jako metody společné pro všechny komponenty. Jsou z metod: action, info, info_api a z atributů name, STATE.

4.7.2 Třída App

Stará se o zpracování argumentů programu a také zavedení třídy Printer pro tiskárnu.

4.7.3 Třída Inputs

Třída definuje chování vstupů a výstupů. Startuje jednotlivá čtecí a zápisová Python vlákna. Pro obsluhu přečtených GCode příkazů se stará fronta ze které se vybírají při zpracování. Zároveň slouží pro čtení a zápis sériového portu, webového api a příkazové řádky.

4.7.4 Třída Printer

Třída pro komponentu tiskárna se skládá z jednotlivých částí popisovaných níže, jejíž diagram je viz. C.1.

4.7.5 Třída Bed

Tato třída zapouzdřuje chování pro simulovanou vyhřívanou podložku.

4.7.6 Třída Extruder

Představuje třídu pro simulaci funkce tiskové hlavy. Rychlost vytlačování materiálu je dána rovnoměrným rozložením množství předepsaného materiálu po dané dráze. Záleží na vypočteném času pro danou dráhu.

4.7.7 Třída Motor

Tato třída představuje a obsahuje chování komponenty motor, která simuluje chování motoru. Jeho rychlost je dána parametrem feedrate a poměru časové konstanty k reálnému času.

4.7.8 Třída Command

Slouží pro zpracovaní příkazů CLI.

4.7.9 Třída GCode

Stará se o zpracování a parsování GCode příkazů i o jejich obsluhu, viz diagram C.2. Zároveň i o načítání kódu v podobě knihoven importovaných a vytvořených. Při vytváření se vkládají obslužné metody do objektu Dictionary. Klíči pro tuto mapu jsou operační kódy GCode instrukcí.

Obsahem následujícího kódu 4.7 je popsání procesu zpracování GCode instrukcí, při kterém dochází ke čtení jednotlivých příchozích dat v podobě bytů pomocí metody self.get_byte(). Tyto byty jsou ve formátu ASCII a to je v této verzi jazyka Python prostý byte a nedá se s ním pracovat jako s řetězcem. Proto je ho třeba převést na kódování odpovídajícímu kódování pro řetězec pomocí byte.decode("utf-8"). Toto čtení probíhá ve smyčce while, pokud je tiskárna v provozu, neboli pokud je hodnota proměnné self.state rovna hodnotě True. Pokud byte nepředstavuje konec řádku, postupně se přidává do pomocného seznamu laterUseBuffer pomocí metody append(byte). Pokud je hodnota proměnné byte rovna konci řádku, zavolá se na celý pomocný buffer laterUseBuffer metoda self.gcode.parse pro zpracování GCode příkazu a uložení do fronty pro pozdější zpracování. Odpověď na příchozí GCode příkaz se provádí okamžitě po jeho přijetí a rozhoduje o něm stav tiskárny. Pokud se na tiskárně něco pokazilo a dostala se do stavu ERROR, odpoví se pomocí metody response pomocného objektu GCode s parametrem odpovídajícím metodě response_err. Pokud proběhlo vše v pořádku, odešle se response s argumentem v podobě metody response_ok.

Ukázka kódu 4.7: Zpracování GCode příkazu

```
while self.state:
    byte = self.get_byte()
    byte = byte.decode("utf-8")
    if byte == self.ENDL:
        self.gcode.parse("".join(self.laterUseBuffer))
        self.laterUseBuffer.clear()
        if self.printer.STATE != "ERROR":
            self.gcode.response(
                self.gcode.response_ok())
        else:
            self.gcode.response(
                self.gcode.response_err())
    else:
        self.laterUseBuffer.append(byte)
```

4.7.10 Třída Logger

Slouží k zaznamenávání událostí v podobě textových záznamů, ať do souboru nebo do jiného souborového popisovače, tzv. log.

4.8 Testování

Tato kapitola popisuje metodiku testování aplikace. Probíhá ve více fázích, tzv. jednotkové testování(unit testy) a funkcionální testování. Testování probíhá za použití ovládacího programu OctoPrint. Po spuštění programu se vypíše název aktuálně vytvořeného virtuálního portu ke kterému se lze připojit.

4.8.1 Unit testování

Při Unit, nebo-li jednotkové, testování se provádí testy, které zkouší správnost chování v rámci jednotlivý kusů zdrojového kódu. Metodě dané třídy se pošle určitý vstup a daná metoda na něj má odpovídat daným výstupem. Je to metoda, při které se provádí tzv. white-box testování. Psaní těchto testů se provádí se znalostí zdrojového kódu. Pro toto testování se mohou používat pomocné a výplňové objekty pro nahrazení skutečné funkcionality a vytvořené na míru pro podmínky testu.

Python má pro daný typ testování framework označovaný unittest. Důležitá byla třída TestCase, která představuje testovaný případ. Má dvě speciální metody *setUp*, která se volá před spuštění každého testu, a *tearDown*, jež se volá po vykonání testu pro úklid po testu. Každá další metoda, která by měla být pro přehlednost pojmenována podle vzoru test_[název metody v dané třídě]. Pro každý test se využívá tzv. assertů, které podle typu metody

vyhodnocují pravdivost testovaného výrazu. Dostupné assertové funkce jsou například *assertEqual*, který vyhodnotí jako úspěšný, když se jeho argumenty rovnají, dále existují asserty pro datové typy podporované jazykem Python jako je Boolean, Dictionary a další. Jak je ze zkrácené ukázky kódu 4.8 patrné byla přetížena metoda *setUp* pro vytvoření výplňového objektu typu Printer se použila funkce *mock_printer*, která vrací instanci třídy MockedPrinter 4.9. Třída MockedPrinter je zjednodušenou implementací třídy Printer s modifikacemi. Z hlediska testování musela být použita tato třída pro zajištění nezávislosti s implementovanou třídou Printer.

Ukázka kódu 4.8: Ukázka unit testu třídy Bed

```
class TestBed(TestCase):

    def setUp(self):
        self.printer = mock_printer()
        self.bed = Bed()

    #...

    def test_action(self):
        # heating
        self.bed.start(90)
        self.bed.TEMPERATURE = 0# start conditions
        iterations = self.printer.action(self.bed)
        # number of iterations needed by default values
        self.assertEqual(iterations, 30)
        self.assertEqual(self.bed.get_temp(), str(90))
        # cooling
        self.bed.TEMPERATURE = 90
        self.bed.start(20)
        iterations = self.printer.action(self.bed)
        self.assertEqual(iterations, 70)
        self.assertEqual(self.bed.get_temp(), str(20))

    #...
```

Ukázka kódu 4.9: Výplňové objekt MockPrinter

```
def mock_printer():
    return MockedPrinter()
```

Pro testování třídy Printer bylo nutné vytvořit výplňové třídy pro typy komponent typu Peripheral z nichž se skládá. Bylo třeba navíc otestovat správné sestavování z komponent a správnost konfiguračního souboru. Zároveň je třeba otestovat získání informací o komponentách ve správném formátu.

4.9 Možnosti pro další vývoj

Jednou z možností budoucího rozšíření programu je změna teplotního modelu na model více odpovídající realitě. Rozšíření o nové GCode příkazy, které zvětšuje rozsah existující funkcionality. Přidání nových druhů komponent pro tiskárnu, které mohou zároveň měnit její funkcionalitu v závislosti na složení a zpracování pomocí GCode příkazů. Další možný vývoj emulátoru by mohl spočívat ve vytvoření nového zobrazení aktuálního tisku ve formě 3D pomocí OpenGL knihovny. Mám v plánu tento program v rámci časových možností dále vyvíjet.

Závěr

Cílem práce je vývoj a otestování funkčního emulátoru 3D tisku. Emulátor se podařilo vytvořit.

Před začátkem samotného vývoje emulátoru jsem vyhledal a analyzoval dostupné řešení emulátorů 3D tiskáren. Nedostatkem těchto emulátorů byla absence možnosti tisku bez použití SD karty při využití sériového portu jako přenosového média pro tiskové příkazy. Toto se v programu povedlo úspěšně vyřešit.

Použil jsem jazyka Python, který je při tomto typu aplikací nejpoužívanější díky své jednoduchosti a rozšíření mezi komunitou projektu RepRap. Tato volba usnadní případný další vývoj ostatním zájemcům o 3D tisk. Díky použití PyGame bylo dosaženo rychlejší vykreslování oproti ostatním dostupným knihovnám.

Testování programu probíhalo po celou dobu vývoje při použití ovládacího programu Printrun a Octoprint. Printrun úplně nedodržuje normu pro GCode, proto bylo lepší použití ovládacího programu Octoprint. Program úspěšně dokončil všechny typy testů.

Při návrhu komponent pro jejich dynamické vytváření zkomplikoval výsledné řešení. Řešení hlavní běhové smyčky jsem mnohokrát upravoval, než jsem přišel na řešení pomocí stavů jednotlivých komponent. Původní návrh počítal s použitím knihovny Asyncio pro práci se vstupem a výstupem, ale kvůli její nedokonalé implementaci jsem musel použít knihovnu threading. Díky tomuto návrhu je program univerzálnější a záleží pouze na jednotlivých komponentech, z nichž se tiskárna složí. Elegantně je vyřešené přidávání nových GCode příkazů, které se načítají ze složky. Toto řešení ulehčí další vývoj emulátoru. Implementace všech požadovaných funkcionalit dopadla úspěšně.

Literatura

[1] Imanica, s. r. o. *Obecně o 3D tisku*[online], Navštíveno 20.11.2015, http://www.o3d.cz/3d-tisk/3d-tisk/

[2] RepRap community. Incomplete beginner's guide. *RepRap Wiki*[online], Navštíveno 23.11.2015, http://reprap.org/wiki/The_incomplete_RepRap_beginner%27s_guide

[3] Thomas R. Kramer,Frederick M. Proctor,Elena Messina. *The NIST RS274NGC Interpreter - Version 3* [online], Navštíveno 25.11.2015, http://www.nist.gov/customcf/get_pdf.cfm?pub_id=823374

[4] The Official YAML Web Site. *YAML*[online], Navštíveno 2.10.2015, http://yaml.org/

[5] RepRap community.G-Code. *RepRap Wiki*[online], Navštíveno 20.10.2015, http://reprap.org/wiki/G-code

[6] Stratasys Direct Manufacturing has nearly five decades of engineering and manufacturing experience in multiple technologies, from Stereolithography to CNC machining. *DMLS*[online] Navštíveno 20.2.2015 https://www.stratasysdirect.com/solutions/direct-metal-laser-sintering/

[7] Imanica, s. r. o. *Obecně o 3D tisku*[online], Navštíveno 20.11.2015, http://www.o3d.cz/3d-tisk/%C5%A1t%C3%ADtky/historie-3d-tisku/

[8] 4ISP spol. s.r.o *Informace o technologiích 3D tisku*[online], Navštíveno 10.02.2015, http://www.easycnc.cz/inpage/informace-o-technologiich-3d-tisku/

[9] Prusa Research s.r.o *Základy 3D tisku*[online], Navštíveno 17.02.2015, http://www.prusa3d.cz/wp-content/uploads/zaklady-3d-tisku.pdf

[10] Python Software Foundation. *Python documentation*[online], Navštíveno 15.03.2015, `https://docs.python.org/3/`

[11] Python Software Foundation. *Unit testing framework*[online], Navštíveno 15.03.2015, `https://docs.python.org/3/library/unittest.html`

[12] Originally by Pete Shinners, now an open source community project. *Pygame framework*[online], Navštíveno 02.04.2015, `http://www.pygame.org/docs/`

[13] Scriptics.com *Tkinter module*[online], Navštíveno 20.03.2015, `http://effbot.org/tkinterbook/tkinter-index.htm`

Příloha **A**

Seznam použitých zkratek

GUI Graphical user interface

XML Extensible markup language

CLI Command line interface

FPS Frames per second

GIL Global interpreter lock

ADT Abstract data type

API Application program interface

Příloha B

Instalační příručka

Tato příručka popisuje kroky nutné k instalaci Emulátoru 3D tiskárny na operačním systému Linux.

1. Pokud máte na vašem systému nainstalovaný interpreter jazyka Python ve verzi 3.4 a vyšší, můžete pokračovat k následujícímu bodu, pokud ne nainstalujte si jej.

2. Pro instalaci pomocí nástroje pip pro danou verzi je připraven soubor *requirements.txt* s jeho pomocí nainstalujete balíčky(knihovny) potřebné pro běh tohoto programu. Instalaci spustíte pomocí následujícího příkazu, pokud je Python verze 3.x.x výchozím interpreterem pro Python:

   ```
   sudo pip install -r requirements.txt
   ```

   Pokud jím výchozí není a v systému se vyskytuje Python 2.7.x i Python 3.x.x, tak se spustí následujícím příkazem:

   ```
   sudo pip3 install -r requirements.txt
   ```

3. Balíček pygame vyžaduje zkompilovanou verzi pygame knihovny. Pokud máte distribuci s Debianem/Ubuntu:
   https://bitbucket.org/pygame/pygame Jinak je třeba nainstalovat ze zdrojových kódů na adrese
   https://bitbucket.org/pygame/pygame.

4. Pokud jste nainstalovaly všechny potřebné balíčky pomocí nástroje pip, tento krok přeskočte. Budete muset všechny nebo zbývající potřebné balíčky nainstalovat ručně. Jsou to tyto balíčky pyserial ve verzi <= 2.7.0, click, pyyaml, pygame. Postupujte podle instalačních pokynů každého balíčku.

5. Argumenty pro příkazovou řádku byly uvedeny v práci. Aplikace se spustí pomocí:

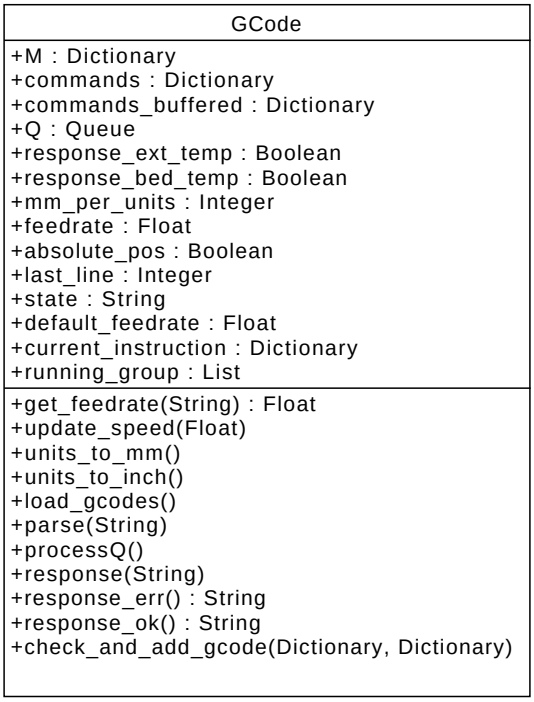
```
python3 App.py [argumenty]
```

Například:

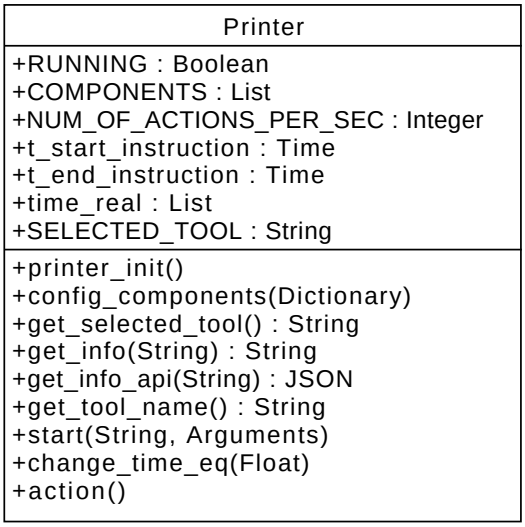
```
python3 App.py --config Prusai3 --timeeq 1
```

Příloha C

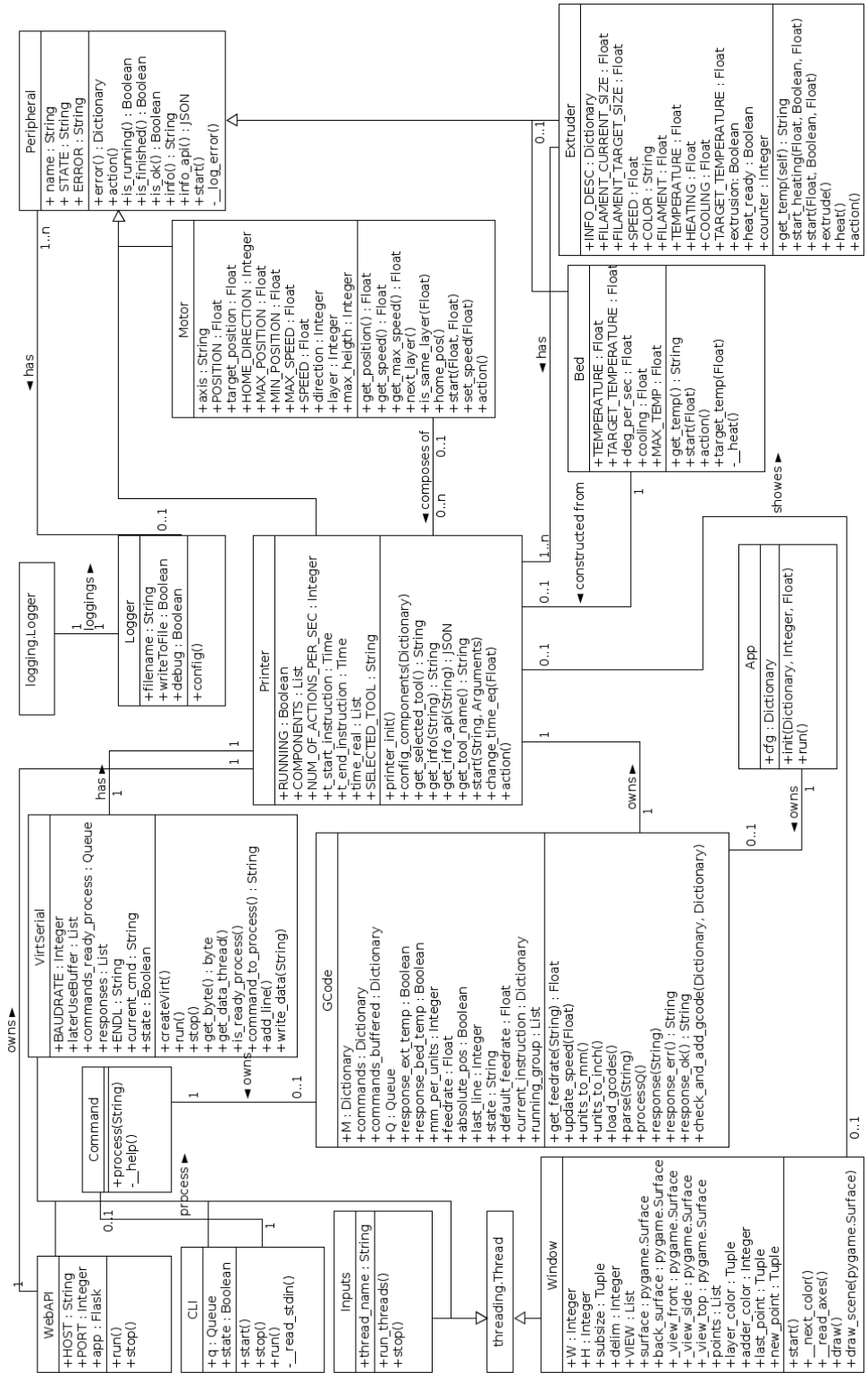
Obrazová dokumentace



Obrázek C.1: Diagram pro třídu Printer



Obrázek C.2: Diagram pro třídu GCode



Obrázek C.3: Class diagram aplikace

Příloha D

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `src`
  - `emuprint` . zdrojové kódy implementace, které jsou zároveň spustitelné
    - `docs` ..... dokumentace programu
    - `tests` ..... unit testy programu
    - `printer_config` ..... konfigurační soubory tiskárny
  - `BAP` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `BP_Jan_Tlamicha.pdf` ..... text práce ve formátu PDF
  - `BP_Jan_Tlamicha.ps` ..... text práce ve formátu PS