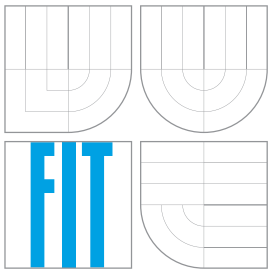
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

OBSLUŽNÝ SYSTÉM PRE FITKIT V PROSTREDÍ PYTHON

FITKIT CONTROL SYSTEM BASED ON PYTHON

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — MARTIN ČERNEK
AUTHOR

VEDOUCÍ PRÁCE — Ing. MICHAL BIDLO, Ph.D.
SUPERVISOR

BRNO 2016

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačových systémů Akademický rok 2015/2016

Zadání bakalářské práce

Řešitel: **Černek Martin**
Obor: Informační technologie
Téma: **Obslužný Systém pro FITkit v prostředí Python**
FITkit Control System Based on Python
Kategorie: Softwarové inženýrství

Pokyny:

1. Seznamte se detailně s existujícím systémem QDevKit. Zaměřte se hlavně na uživatelské rozhraní, správu projektů, integraci překladového systému a externích nástrojů (MSPGCC, Xilinx ISE + simulátor ISIM), nízkoúrovňovou komunikaci s FITkitem a uživatelskou konzoli.
2. Navrhněte multiplatformní aplikaci v prostředí Python, jejímž cílem bude realizace výše zmíněných funkcí, známých ze systému QDevKit.
3. Vypracujte studii zaměřenou na analýzu požadavků aplikace navržené v bodu 2, její návrh, popis funkcí a způsob použití.
4. Aplikaci implementujte v prostředí Python. Zaměřte se na přenositelnost, v případě potřeby využijte zavedené a zdokumentované externí knihovny.
5. Zhodnoťte výsledný systém a diskutujte jeho možná další rozšíření.

Literatura:

- Dle pokynů vedoucího projektu.

Pro udělení zápočtu za první semestr je požadováno:

- Splnění bodů 1 a 2 zadání.

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

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

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

Vedoucí: **Bidlo Michal, Ing., Ph.D.**, UPSY FIT VUT
Datum zadání: 1. listopadu 2015
Datum odevzdání: 18. května 2016

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačových systémů a sítí
612 66 Brno, Božetěchova 2

L.S.

doc. Ing. Zdeněk Kotásek, CSc.
vedoucí ústavu

Abstrakt

Táto bakalárska práca popisuje návrh a implementáciu systému, nazvaného ako PyDevKit, pre vzdelávaciu platformu FITkit. Cieľom je integrovať podporu prekladového systému, správu projektov, editor zdrojových kódov a interakciu s nástrojmi tretích strán do jednej aplikácie. Aby bola zabezpečená dobrá multiplatformná podpora, implementácia je realizovaná pomocou prostredia Python a ďalších potrebných externých knižníc dostupný pre toto prostredie. PyDevKit je vybavený grafickým uživateľským rozhraním ktoré poskytuje užívateľovi nástroj pre pohodlnú prácu s FITkitom.

Abstract

This bachelor's thesis describes the design and implementation of a system, denominated as PyDevKit, for the FITkit educational platform. The goal is to integrate the translation system support, project management, source-code editor and interaction with the third-party tools into a single application. In order to ensure a good multi-platform support, the implementation is performed using the Python environment and some necessary external libraries available for this environment. PyDevKit is equipped by a graphical user interface that provides the user with a tool for a comfortable work with FITkit.

Kľúčové slová

Obslužný systém, Python, FITkit, Softwarové inžinierstvo

Keywords

Control system, Python, FITkit, Software engineering

Citácia

ČERNEK, Martin. *Obslužný Systém pre FITkit v prostredí Python*. Brno, 2016. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Bidlo Michal.

Obslužný Systém pre FITkit v prostredí Python

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením pána Ing. Michala Bidla, Ph.D. Uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

.....
Martin Černek
18. mája 2016

Poďakovanie

Chcel by som poďakovať svojmu vedúcemu Ing. Michalovi Bidlovi, Ph.D. za poskytnutie potrebných informácií a rád pri tvorbe bakalárskej práce.

Obsah

Kapitola 1

Úvod

Platforma FITKit sa využíva na mnohých vyučovacích kurzoch po celú dobu štúdia či už bakalárskeho alebo magisterského študijného programu. Jej cieľom je priblížiť študentom problematiku vstavaných systémov a poskytnúť im okrem teoretických taktiež aj praktické skúsenosti. FITkit je hardware pozostávajúci z mikrokontroléru, hradlového poľa FPGA a niekoľko ďalších periférnych súčastiek. Výhodou FITkitu je použitie reprogramovateľného hardware FPGA, ktorý je možné ľubovolne modifikovať a tým simulovať rôzne typy hardware. Popis tohoto hardware je možné uskutočniť vhodným programovacím jazykom. V prípade FITkitu to je jazyk VHDL. Generovanie dát pre FGPA z jazyka VHDL zabezpečujú rôzne profesionálne návrhové systémy. Aplikácie pre mikrokontrolér vo FITkite sa tvoria v jazyku C. [9]

Celý proces od vytvorenia aplikácie až po naprogramovanie do FITkitu vyžaduje niekoľko nevyhnutných operácií, ktoré je nutné vykonať z počítača. Úlohou obslužného systému je všetky tieto operácie vykonať na pár kliknutí a zobraziť ich stav v priateľskom užívateľskom rozhraní, čím užívateľom umožní pohodlnejšiu prácu s FITkitom.

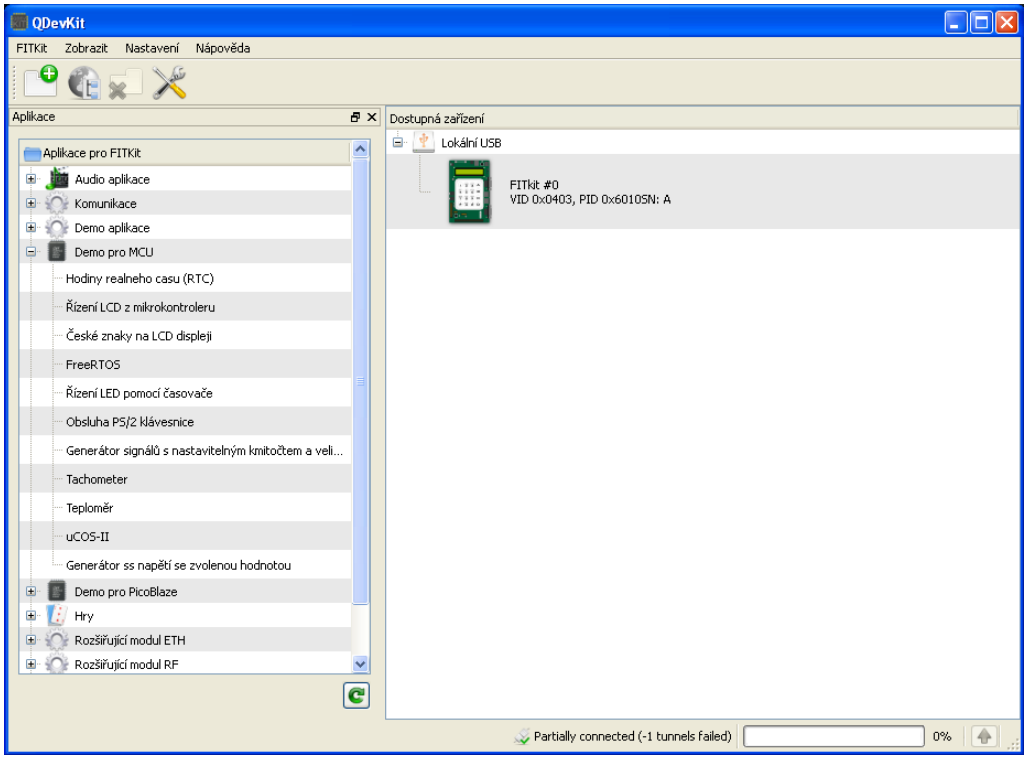
Súčasný stav obsluhy a programovania FITkitu nie je prívetivý pre užívateľa. Stávajúci obslužný systém QDevKit má na najnovších operačných systémoch problémy s kompatibilitou použitých knižníc a stal sa nepoužiteľným. Napriek tomu existujú možnosti ako FITkit obsluhovať, ktoré ale nie sú z pohľadu užívateľa pohodlné.

Cieľom tejto práce je navrhnúť a vytvoriť použiteľný obslužný systém pre prácu s FITkitom, ktorý nahradí a vyrieši problémy stávajúceho systému a poskytne študentom plnohodnotnú náhradu.

Kapitola 2

Analýza požiadavok

Užívateľom je k dispozícií systém pre prácu s FITkitom, ktorý je ale zastaralý a má problémy s kompatibilitou na moderných operačných systémoch. Tento systém, pomenovaný QDevKit, poskytuje základné operácie umožňujúce pracovať s platformou FITkit na školských projektoch. Sprostredkováva vybraný SVN repozitár vo forme zoznamu projektov, s ktorými je možné následne pracovať. Ďalej je do neho integrovaný prekladový systém s podporou vzdialeného prekladu, vďaka čomu nevyžaduje inštaláciu externého systému Xilinx ISE na lokálnom zariadení, čím ušetrí užívateľovi časť dostupnej pamäťe na pevnom disku, ktorej veľkosť rozhodne nie je zanedbateľná. Okrem týchto hlavných funkcií podporuje nízkoúrovňovú komunikáciu s FITkitom prostredníctvom užívateľskej konzoly. Ukážka hlavnej obrazovky systému QDevKit je na obrázku 2.1.



Obr. 2.1: Ukážka súčasného systému QDevKit.

2.1 Súčasný stav z pohľadu systému

QDevKit je navrhnutý ako aplikácia zabezpečujúca obsluhu rôznych externých nástrojov za účelom poskytnutia uživateľského rozhrania aby užívateľovi tieto nástroje nemusel obsluhovať z príkazového riadku. Priamo do seba integruje podporu prekladového systému a komunikácie s FITkitom cez konzolu.

Hlavným problémom systému QDevKit je neschopnosť prekladu pod modernými linuxovými distribúciami. Táto neschopnosť vyplýva z nekompatibility knižníc použitých v implementácií tohoto systému. Problémom je použitie jazyka C++, ktorý vyžaduje pre spustenie aplikácie najprv jej preklad do binárnej podoby pre špecifickú platformu. Pri pokusoch o ručnú úpravu zdrojových kódov dochádza k neustálemu vyskytu nových chýb pri preklade. Použitie systému QDevKit na platforme Windows je v porovnaní s Linuxovými distribúciami na tom o niečo lepšie. Inštalácia z balíčka prebehne vo väčšine v poriadku a v ojedinelých prípadoch sa vyskytne problém.

2.2 Súčasný stav z pohľadu užívateľa

V nasledujúcom texte je popísaný stav, v akom je možná práca so zariadením FITkit na súčasných najrozšírenejších operačných systémoch. Práca a použitie sa líši v závislosti od typu použitého operačného systému.

Na moderných linuxových distribúciach existujú dve možnosti ako pracovať so zariadením FITkit. Prvou z nich je použitie stávajúceho systému QDevKit na virtualizovanom operačnom systéme. Študentom sú k dispozícií predpripravené obrazy operačného systému Windows XP, ktoré obsahujú nainštalované všetky nevyhnutné knižnice a nástroje potrebné pre prácu s FITkitom. Tieto obrazy je možné stiahnuť po prihlásení sa na webových stránkach[8]. Táto možnosť ale prináša so sebou niekoľko problémov. Jedným z nich sú požiadavky na väčší výpočetný výkon počítača, kde pobeží virtualizovaný systém. V niektorých prípadoch hardware počítača nemusí podporovať virtualizáciu. Ďalším problémom je stabilita nástrojov, ktoré umožňujú virtualizovovať operačné systémy, na rôznych linuxových distribúciach. Môžu sa vysknutnúť problémy spojené s hardware, kedy ho nie je možné správne detekovať. Taktiež samotné použitie takmer 15 rokov starého operačného systému v predpripravených obrazoch neprispieva k stabilite. Druhou možnosťou je použiť dostupné nástroje bez obslužného systému vo forme nástrojov ovládaných z príkazového riadku. Pri tejto možnosti už ale užívateľ postráda výhody grafického rozhrania, ktorými je hlavne pohodlie a vizuálne prívetivejšie zobrazenie stavu jednotlivých operácií. Na sučasne najpoužívanejších linuxových distrubúciach je možné použiť aktualizovaný návod na sprevádzkovanie práce s FITkitom pomocou v natívnom prostredí, ktorý je dostupný na webových stránkach[6]. Tento návod kompletne popisuje inštaláciu nástrojov a knižníc potrebných k sprístupneniu obsluhy FITkitu cez príkazový riadok.

V prípade použitia operačného systému Windows je situácia o niečo lepšia. Keďže existuje inštalačný balík zahŕňajúci všetky potrebné knižnice a nástroje, tak je možné systém QDevKit bez problémov nainštalovať a používať. V ojedinelých prípadoch sa môže vyskytnúť problém s jeho používaním. Taktiež tu je možnosť použiť tie isté predpripravené obrazy vo virtualizovanom operačnom systéme, čo prináša rovnaké problémy ako v prípade linuxových distribúcií. Ďalšou možnosťou je použitie dostupných nástrojov z príkazového riadku operačného systému rovnako ako v linuxových distribúciach. Všetky potrebné inštalačné balíky potrebných nástrojov spolu s odkazmi na stiahnutie nástrojov tretích strán sú dostupné po prihlásení sa na webových stránkach[8].

Ak sa užívateľ rozhodne napriek tomu obsluhovať FITkit pomocou systému QDevKit, je vhodné analyzovať jeho interakciu so systémom. Skupinu koncových uživateľov tvoria predovšetkým študenti pracujúci na školských projektoch s platformou FITkit. V nasledujúcich bodoch je stručne popísaná typická práca študenta na projekte:

- Študent vo väčšine prípadoch pri zahájení práce na novom projekte použije zdrojové kódy podobného už implementovaného projektu.
- Zdrojové kódy skopíruje a vloží do SVN repozitára do príslušnej kategórie.
- Inšpiruje sa zo zdrojových kódov iných projektov z repozitára.
- Zdrojové kódy upravuje externým nástrojom.
- Nezaobíde sa bez používania správcu súborov v operačom systéme.

Z predchádzajúcich bodov sa môžu vyvodiť nasledujúce závery. Medzi najväčšie nevýhody stávajúceho systému QDevKit patrí nevyhnutnosť prepínania sa medzi ďalšími externými nástrojmi, ktoré nie sú priamo integrované do systému, a samotným systémom. Vzniká tu teda požiadavka na výskyt ďalších externých nástrojov ako sú editor zdrojových kódov či správca súborov. Toto prepínanie sa medzi nástrojmi a systémom robí prácu užívateľa obmedzujúcou a neefektívnou z pohľadu časovej náročnosti. Ďalší neefektívný prístup vzniká pri používaní už existujúceho projektu ako šablóny. Túto operáciu nie je možné vykonať priamo zo systému ponúkajúceho možnosť úpravy údajov kopírovaného projektu, ktorými sú názov projektu, meno autora, emailový kontakt na autora, číslo revízie a stručný popis.

2.3 Voľba implementačného prostredia, technológií a knižníc

V nasledujúcom texte su rozobraté do detailov dôvody použitia vybraných technológií. Hlavný dôraz pri ich výbere sa kládol na kompatibilitu so súčasnými operačnými systémami, kedže hlavným cieľom je vytvoriť prenositeľný oblužný systém.

Za implementačné prostredie bolo zvolené prostredie Python. Programovací jazyk Python je silný a ľahko naučiteľný. Jeho zdrojový kód je čistý a prehľadný. Medzi jeho silné stránky patrí schopnosť implementovať ten istý kód použitím menšieho počtu príkazov. Jednou z jeho najväčších predností je poskytnutie rozsiahlej štandardnej knižnice. Navyše k tomu si vytvoril silnú komunitu, ktorá vyprodukovala tisíce knižníc tretích strán. Hlavným dôvodom jeho použitia je poskytnutie multiplatformného prostredia bez nutnosti kompilácie zdrojového kódu.[7] Verzia 3 bola zvolená najmä vzhľadom k použitej knižnici implementujúcej grafické rozhranie obslužného systému. Dôvod je popísaný v nasledujúcom texte.

Pri výberu knižníc implementujúcich grafické rozhranie v prostredí Python sa vyberalo z rozsiahleho prehľadu dostupného na webovom portály, kde sú ku každej spomenutej knižnici zobrazené stručné detaily. Po analýze týchto dostupných knižníc prišlo do úvahy použitie práve troch, ktorými sú WxPython, PySide a PyQt. Všetky tri uvedené knižnice sú viazané na knižnice určené pre programovací jazyk C++. Konkrétne WxPython je viazaný s knižnicou WxWidgets a v prípade PyQt a PySide je to knižnica Qt. Taktiež všetky poskytujú stabilitu, multiplatformosť a dostupnosť v open source licencii. V prospech WxPython a PyQt hrá aktuálnosť, kedže obidve poskytujú viazanie na najnovšiu verziu natívnej knižnice pre C++. Tento faktor je dôležitý kedže cieľom PyDevkitu je podpora na nasledujúce roky na operačných systémoch, kde sa predpokladá použitie najaktuálnejších verzií knižníc.

Z tohoto dôvodu bolo použitie knižnice PySide vylúčené. Na základe štúdie dokumentácie obidvoch knižníc PyQt[2] a WxPython[4] dostupnej na ich domovských webových stránkach sa dospelo k záveru, že knižnica PyQt poskytuje čistejší a prehľadnejší kód z pohľadu objektového orientovania a štandardov používaných v Pythone. Napriek tomu hlavným a rozhodujúcim dôvodom, na základe ktorého bola zvolená knižnica PyQt namiesto WxPython je použitie samotnej natívnej knižnice Qt v implementácií nástroja fcmake, ktoré je nevyhnutné zahrnúť do nového systému. Samotné Qt je možné nainštalovať pomocou inštalátoru dostupného na webových stránkach[1]. V prípade PyQt je inštalátor dostupný len pre verziu 3, pre verziu 2 je nutný preklad a inštalácia zo zdrojových kódov. Vzhľadom na čo najväčšiu snahu o bezproblémovosť inštálacie z pohľadu užívateľa bola zvolená verzia 3 implementačného prostredia Python.

V stávajúcom systéme komunikáciu so samotným zariadením FITkit cez uživateľskú konzolu zabezpečuje C knižnica libkitclient, ktorá je aj pre prostredie Python vo forme viazania. V dôsledku, že je naviazaná pre Python vo verzií 2, bude toto riešenie nahradené externou knižnicou pylibftdi poskytujúcou minimálne rozhranie v prostredí Python využívajúcu knižnicu libftdi ku komunikácií s FTDI zariadeniami. Medzi jej vlastnosti patrí podpora prostredia Python ako vo verzií 2 tak aj vo verzií 3, podpora paralelnej a sériovej komunikácie, podpora pre viac súčasne pripojených zariadení a najdoležitejšia vlastnosť ktorou je multiplatformosť.[3]

Do systému je taktiež nevyhnutné integrovanie podporných nástrojov zo starého systému QDevKit ako sú fcmake zo závislosťou na fkflash, ktoré zabezpečujú správu a preklad projektov, generovanie výstupných súborov pre FPGA a následné programovanie do zariadenia FITkit. Postup ich inštalácie a zdrojové kódy sú dostupné po prihlásení sa na webových stránkach.[8]

2.4 Nástroje tretích strán

K činnosti práce s FITkitom je nevyhnutná inštalácia nástrojov tretích strán ktorými sú MSP430-gcc[2] a Xilinx ISE[3]. Okrem týchto nástrojov je potrebné integrovať do systému nástroj pre správu verzií Subversion[4].

Prvý spomínaný nástroj je prekladový systém, ktorý umožňuje preklad zdrojového kódu napísaného v jazyku assembler alebo C pre mikrokontrolér MSP430. Výstupom sú súbory s príponou HEX, ktoré sú určené priamo pre špecifickú rodinu mikrokontroléru použitého vo FITkite. Inštalácia je rozdielna vzhľadom na architektúru použitého operačného systému. Pre niektoré linuxové distribúcie sú dostupné binárne debian balíčky, ktoré je možné nainštalovať okamžite po stiahnutí. Na zvyšných distribúciach, ktoré nedokážu pracovať s debian balíčkami nezostáva iná možnosť ako preklad a inštalácia priamo zo zdrojových kódov. V prípade operačného systému Windows je taktiež dostupný inštalačný balík, ktorý umožňuje inštaláciu bez výrazných problémov.

Xilins ISE je sada nástrojov, ktorých úlohou je preklad zdrojových kódov a syntéza FPGA konfigurácie na základe popisu systému pomocou HDL jazyka. Stiahnutie a inštalácia vyžaduje registráciu na www.xilinx.com aby mohla byť následne vygenerovaná licencia, ktorá je študentom poskytovaná úplne zadarmo. Rovnako ako pre linuxové distribúcie, tak

[1] Dostupný na https://www.qt.io/download/.
[2] Dostupný na http://www.ti.com/tool/msp430-gcc-opensource.
[3] Dostupný na http://www.xilinx.com/.
[4] Dostupný na https://subversion.apache.org/packages.html.

aj pre Windows je dostupný inštalátor, ktorý umožňuje vykonanie inštalácie bez väčších problémov.

Správu obsahu repozitára zabezpečí nástroj Subversion, ktorý je použitý bez knižnice pre Python, pretože žiadna taká nespĺňala kritéria prenositeľnosti medzi operačnými systémami. Príkazy z tohoto nástroja budú volané externe. Inštalačné balíky sú dostupné rovnako ako pre operačný systém Windows tak aj pre Linux na webových stránkach.

Kapitola 3

Návrh systému PyDevKit

Názov systému bol odvodený od pôvodného obslužného systému QDevKit. Kedže implementácia je realizovaná v programovacom jazyku Python, z tohto dôvodu bolo prvé písmeno nahradené slabikou „Py“.

Nevyhnutnou súčasťou implementácie akéhokoľvek rozsiahlejšieho systému je jednoznačne správny návrh. Zvolené ciele nasledované vhodnou dekompozíciou problému na menšie časti zabezpečia plynulejší postup implementácie. V nasledujúcom texte je tento postup návrhu popísaný.

3.1 Cieľové požiadavky systému

Navrhnutie plnohodnotného systému úzko súvisí s určenými cieľovými požiadavkami. Jednou z hlavných požiadavok na cieľový systém je prenositeľnosť a kompatibilita s najnovšími architektúrami operačných systémom, konkrétne typu Linux a Windows, ktoré používa väčšina študentov a taktiež sa vyskytujú na zariadeniach v centre výpočetnej techniky. Kompatibilita medzi operačnými systémami je úzko spätá s použitými technológiami a knižnicami a preto je veľmi dôležité klásť dôraz na voľbu vhodných prostriedkov. Taktiež je cieľom minimalizovať pravdepodobnosť nežiadúcich problémov pri inštalácií obslužného systému. Tohto všetkého je možné dosiahnuť vytvorením systému s čo najmenším počom závislostí na externých knižniciach a nástrojoch tretích strán.

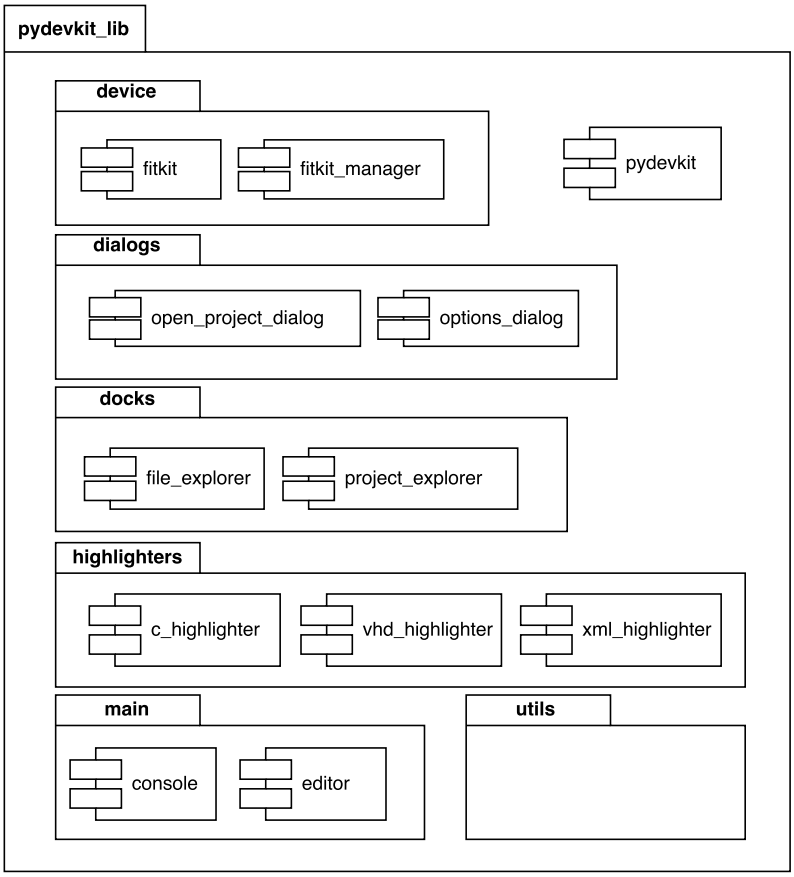
Ďalšou hlavnou požiadavkou je, aby nový systém bol schopný plnohodnotne nahradiť vo výuke stávajúci systém. Očakáva sa teda, že bude poskytovať všetky dostupné funkcie ako sú preklad aplikácie, vzdialený preklad, naprogramovanie do FITkitu, integrovanie operácií s prostredím Xilinx ISE a komunikácia s FITkitom cez terminálové okno.

Cieľom je taktiež ponúknuť užívateľovi rozšírenú funkcionalitu o tie možnosti, ktoré v stávajúcom systéme chýbajú. Na základe obsahu kapitoly 2.2 je vhodné zahrňúť do systému nasledujúcu rozširujúcu funkcionalitu. Obrovskú výhodu pre užívateľa prinesie vstavaný editor zdrojových kódov, kedže viac nie je nútený prepínať okno aplikácie s oknom externého nástroja pomocou ktorého edituje zdrojové kódy. Užívateľ viac nemusí vykonávať operácie, ktoré ho zdržujú a tým sa celý proces tvorby aplikácie urýchli. Výhodu prinesie aj podpora zvýrazňovania syntaxe zdrojového kódu zobrazovaného v editore. Syntax bude zvýraznená vzhľadom na použitý programovací jazyk, vďaka čomu sa bude užívateľ jednoduchšie orientovať v zdrojovom kóde. Pohodlnejšiu prácu pri tvorbe projektu umožní vytvorenie nového projektu nakopírovaním zdrojových kódov už existujúceho s možnosťou úpravy informácií o projekte. Cieľom je aby táto možnosť bola dostupná priamo zo systému

a uživateľ to nemusel robiť manuálne. Zobrazenie súborov patriacich k projektu je možné len pomocou vstavaného správcu súborov v operačnom systéme. Cieľom je aby mal užívateľ dostupné súbory zo systému a mohol ich priamo upravovať, prípadne mať viac aktívnych projektov a prepínať sa medzi nimi.

3.2 Dekompozícia aplikácie

V tejto kapitole je popísaná dekompozícia systému PyDevKit na jednotlivé podproblémy. Prostredie Python ponúka niekoľko konštrukcií ako organizovať zdrojové kódy. Je veľmi jednoduché vytvoriť vlastný modul, pretože samotný súbor napísaný v tomto prostredí predstavuje modul. Každý modul zapúzdruje triedy, funkcie a globálne objekty. Okrem rozdelenia problému do modulov, prostredie Python umožňuje organizovať jednotlivé moduly do balíkov. [5] Je vhodné aby sa všetky významné triedy, spolu s pomocnými triedami alebo funkciami, nachádza v samostatných moduloch, ktoré sú na základe logickej príbuznosti členené do jednotlivých balíkov, vďaka čomu je orientácia v zdrojovom kóde prehľadnejšia. Obrázok 3.1 predstavuje počiatočný návrh rozdelenia implementácie systému PyDevKit do jednotlivých balíkov a modulov.



Obr. 3.1: Návrh štruktúry systému PyDevKit z pohľadu modulov a balíkov.

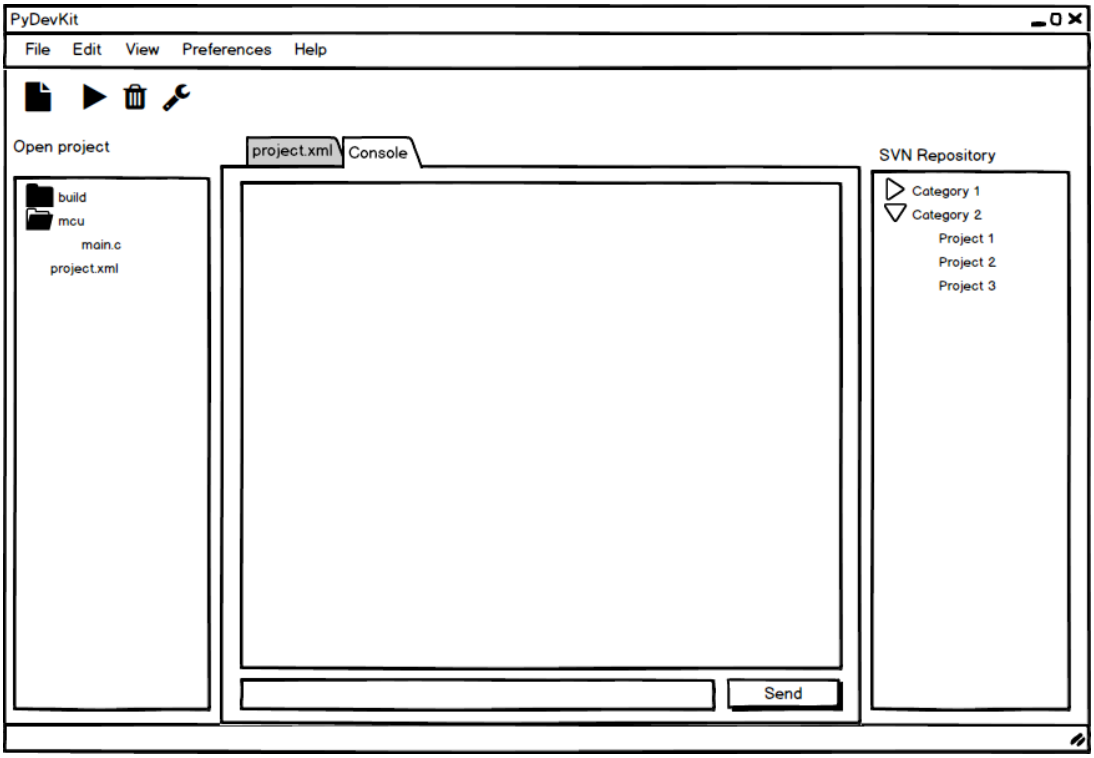
3.3 Návrh interakcie s užívateľom

Návrh vychádza z analýzy požiadavok a stanovených cieľov. Funkcionalita systému pokrýva veľké množstvo operácií, ktoré je možné najlepšie zobraziť prostredníctvom diagramu prípadov použitia. Tento diagram je zobrazený na obrázku 3.2 a poskytuje prehľad o všetkých operáciách s ktorými sa užívateľ stretne pri interakcií so systémom. Pri návrhu grafického rozhrania sa kládol dôraz na jeho intuitívnosť, aby užívateľ bez predchádzajúceho používania systému bol schopný jednoducho obsluhovat zariadenia FITkit. Za účelom čo najefektívnejšieho používania systému PyDevKit, je nutné správne rozvrhnúť pracovný priestor aplikácie. Ten je reprezentovaný celou plochou jej okna. Prvotný návrh grafického uživatelského rozhrania je zobrazený na obrázku 3.2.

Obr. 3.2: Diagram prípadov použití.

Najväčšiu časť pracovného priestoru musia zaberať hlavné prvky systému, ktorými sú editor zdrojových kódov a uživatelská konzola. Toto rozhodnutie bolo vykonané z toho dôvodu, že sa jedná o najčastejšie používané elementy systému. Pri programovaní projektov sa väčšinou pracuje s viacerými súbormi. V dôsledku toho je ľavá strana pracovného priestoru vyhradená pre prehľad súborov v stromovej štruktúre patriacich do aktuálne otvoreného projektu. Zobrazenia SVN repozitára bolo pôvodne zamýšlané umiestniť na ľavú spolu so stromovou štruktúrou otvoreného projektu. Tu však nastal problém pri snahe zobraziť via-

cero otvorených projektov súčasne. V takomto prípade kapacita zobrazovanej plochy nebola dostačujúca a stala sa neprehľadnou. Z tohto dôvodu bol obsah SVN repozitára presunutý na pravú stranu. Každý prvok ktorý slúži na zobrazenie rôzneho obsahu by malo byť možné premiesniť na takú pozíciu, ktorá je pre uživateľa najvhodnejšia. Základné operácie nad projektom ako napríklad preklad zdrojového kódu, naprogramovanie projektu do zariadenia FITkit či zahájenie komunikácie s pripojeným zariadením musia byť dostupné užívatelovi priamo, bez zbytočného preklikávania sa hlavným menu aplikácie. Preto boli realizované formou panelu nástrojov umiestneného pod hlavným menu. Jednotlivé operácie sú jednoducho rozlíšitelné výstižnou ikonou. Aby mal užívatel prehľad o priebehu aktuálne vykonávanej operácie, ako napríklad preklad projektu, bolo pre ne v spodnej časti obrazovky vyhradené miesto vo forme logovacej konzoly.

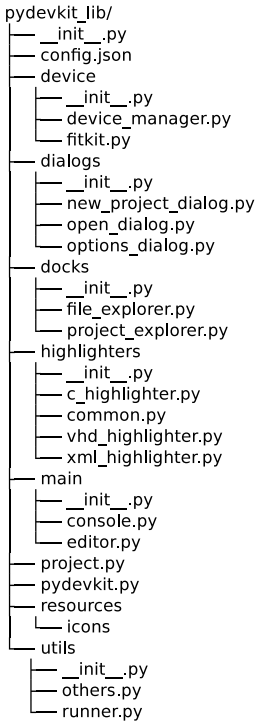


Obr. 3.3: Počiatočný návrh grafického rozhrania.

Kapitola 4

Implementácia

V nasledujúcom texte sú stručne popísané významné časti implementácie hlavných tried a balíkov použitých v PyDevkite. Detailnejšie informácie sú dostupné priamo zo zdrojového kódu aplikácie. Hlavný balík nesie názov `pydevkit_lib` a jeho finálna hierarchia je zobrazená na obrázku 4.1. V súboroch `__init__.py` sú uložené údaje o triedach, funkciách, prípadne konštantách, ktoré je možné z daného balíka importovať. Zdrojový kód spĺňa programovacie návyky a konvencie štandardu PEP 8[1]. Všetky spomenuté triedy v nasledujúcom texte, ktorých názov začína veľkým začiatočním písmenom Q sú použité z knižnice PyQt.



Obr. 4.1: Hierarchia knižnice `pydevkit_lib`.

[1] https://www.python.org/dev/peps/pep-0008/

4.1 Trieda `PyDevkit`

Trieda `PyDevkit` predstavuje hlavné okno aplikácie, dedí z triedy `QMainWindow` a poskytuje metódy s ktorými je zabezpečené ovládanie celého systému. Inicializácia prebieha v niekoľkých krokoch. Na začiatku sa načítajú nastavenia systému nasledované inicializáciou všetkých grafických prvkov s ktorými je možné ovládať systém. Ďalej inicializácia správcu pripojených zariadení triedy `DeviceManager`, ktorej detailnejšia implementácia je popísaná v kapitole 4.3.3. Činnosť správcu zariadení je presunutá do pozadia do vlákna. Na záver inicializácie sa upravia vlastnosti hlavného okna, ktoré je zobrazované užívateľovi. Medzi dôležité atribúty patria `config` uchovávajúci konfiguráciu celého systému vrátane pripojených zariadení, `device_manager` teda správca pripojených zariadení, `connected_devices` pomocou ktorého sú sprostredkované aktuálne pripojené zariadenia. V nasledujúcom texte je stručne popísaný význam a fungovanie hlavných metód a slotov tejto triedy.

Metódy `initialize_menu`, `initialize_main_toolbar`, `initialize_main_panel` a taktiež `initialize_docks` zabezpečujú inicializáciu všetkých hlavných grafických prvkov umožňujúcich ovládanie systému v hlavnom okne.

Metóda `initialize_shortcuts` zavedie do systému všetky dostupné klávesové skratky, ktoré môže užívateľ používať.

Úlohou dvojice metód `load_config` a `save_config` je inicializovať nastavenia systému uložených v atribúte `config`. Ak konfiguračný súbor *config.json* neexistuje, tak sa použijú východzie nastavenia a následne sa tento konfiguračný súbor vytvorí. Dáta v ňom sú uložené vo formáte JSON[2] a na jeho spracovanie sa používa štandardná knižnica `json`.

Slot `manage_connected_devices` je vyvolaný pri detekcií zmeny v pripojených zariadeniach. Jeho úlohou je zaktualizovať dostupné zariadenia v atribúte `connected_devices` a informovať užívateľa o zmene v stavovom riadku.

Zobrazovanie dostupných projektov zo SVN repozitára v systéme je spojené s metódami `change_svn_directory` a `refresh_svn_viewer`. Prvá z nich vyvolá dialógové okno pomocou ktorého môže uživateľ zvoliť cestu adresára s repozitárom. Po zvolení sa vykoná druhá metóda, ktorej úlohou je aktualizovať obsah SVN repozitára zo zvolenej cesty zobrazovaný v objekte triedy `ProjectExplorer`, ktorej detaily sú popísané v kapitole 4.6.1.

Nasledujúca dvojica metód a slot implementujú spravovanie hlavných častí systému ktorými sú editor zdrojových kódov a konzolové okno komunikácie so zariadením. Otvorenie súboru v editore zdrojových kódov zabezpečuje metóda `open_file`, ktorá požaduje ako nepovinný parameter jeho cestu a prípadne aj prislúchajúci projekt. V prípade, že cesta nie je zadaná, tak je otvorená záložka s editorom, ktorý má prázdny obsah a ako prislúchajúci projekt je zvolený práve aktívny projekt v systéme. Hlavnou podstatou otvorenia súboru je vytvorenie instancie triedy `Editor`, bližšie popísanej v kapitole 4.4.2, ktorej predá tieto potrebné informácie. V prípade, že požadovaný súbor je už otvorený, tak prenesie zobrazenie z aktuálne aktívnej záložky na túto záložku s editorom. Podobná metóda `save_file` umožňuje uloženie zmien vykonaných v editore. Otvorenie terminálového okna pomocou ktorého je možné komunikovať s FITkitom je implementované pomocou metódy `open_console`, ktorej je nutné cez parameter predať identifikačného čísla zariadenia s ktorým má začať komunikovať. Metóda vytvára instanciu triedy `Console`, popísanej v kapitole 4.4.1. Úlohou slotu `close_main_panel_tab` je správne uzatvorenie otvorených záložiek v hlavnej časti okna aplikácie. V prípade, že sa jedná o záložku, kde je otvorené spojenie s FITkitom, tak zabezpečuje korektné zatvorenie a ukončenie komunikácie so zariadením.

[2] http://www.json.org/

Otvorenie projektu, teda vytvorenie instancie triedy `Project` je implementované metódou `open_new_project`. Implementácia triedy `Project` je popísaná v kapitole 4.2. Metóda pri volaní vyžaduje nepovinné parametre ktorými sú cesta k projektu a jeho názov. Ak nie je zadaná cesta, je vyvolané dialógové okno `QFileDialog` s požiadavkou na zvolenie súboru, ktorý musí mať presný názov *project.xml*. V prípade, že uživateľ zvolí takýto súbor vykoná sa jeho analýza pomocou statickej metódy `parse_project` z triedy `ProjectExplorer`, ktorá je popísaná v kapitole 4.6.1. Cesta a názov projektu sú dostupné z analýzy, tá prebehne správne ak súbor vyhovuje formátu. Samozrejme ak nie je zvolený žiadny súbor, tak sa nevykoná žiadna analýza nasledovaná otvorením projektu. Následne sa skontrolujú aktuálne otvorené projekty a ak sa medzi nimi nachádza zvolený, tak sa stane aktívnym projektom v systéme. Taktiež je jej úlohou prispôsobiť veľkosť zobrazovaného názvu projektu. Zatvorenie projektu je implementované obdobnou metódou `close_current_project` ktorá odoberie príslušný projekt z aktuálne otvorených a tým ho uzatvorí. Slot `change_current_project` sa vyvolá vždy pri zmene aktívneho projektu v systéme. Zabezpečuje predovšetkým korektné zobrazenie názvu aktuálne aktívneho projektu v hornej lište aplikácie a sprístupnenie dostupných operácií, ktoré je možné vykonávať nad aktuálne aktívnym projektom.

Operácie nad aktuálne aktívnym projektom v systéme vykonávajú nasledujúce metódy volajúce obdobné metódy nad instanciou typu `Project`, ktorej implementačné detaily sú spomenuté v nasledujúcej kapitole 4.2. Preklad projektu, respektíve vzdialený preklad na servery je implementovaný dvojicou metód `project_build` a `project_build_remotely`. Naprogramovanie projektu do zariadenia je možné pomocou metódy `project_load` ktorá vyžaduje identifikačné číslo zariadenia do ktorého sa má vybraný projekt naprogramovať. Otvorenie projektu v prostredí Xilinx ISE, prípadne spustenie jeho simulácie v tomto prostredí zabezpečujú metódy `project_run_ise` a `project_run_isim`. Vyčistenie adresára projektu od súborov vzniknutých po preklade je možné pomocou metódy `project_clean`.

4.2 Trieda `Project`

Reprezentáciu samotného projektu v systéme reprezentuje trieda Project rozširujúca triedu `FileExplorer` popísanú v kapitole 4.6.2. Dôležitými parametrami pri inicializácií sú cesta k danému projektu a jeho názov.

Vykonanie operácií je implementované metódou `run_command`, ktorá vytvorí instaciu triedy `CommandRunner`, presunie jej činnosť do vlákna, aby nebola narušená interakcia s uživateľským rozhraním, a následne spustí jej vykonávanie. Popis implementácie triedy `CommandRunner` je v kapitole 4.7.1. Samotná operácia je zoznam textových reťazcov, kde jeden reťazec reprezentuje príkaz v príkazovom riadku operačného systému pozostávajúci z volania externých nástrojov `make` a `fcmake`. Komunikácia s instanciou vo vlákne je sprostredkovaná pomocou signálov `output` a `running`, vďaka ktorým dostáva informácie o aktuálnom výstupe a stave vykonávania operácie. Hodnoty zo signálu `output` sú zviazané so slotom `log_output`, ktorá všetky výstupy vypisuje do logovacej konzoly aby boli následne sprostredkované užívateľovi. Zmena hodnoty na signále `running` je naviazaná na metódu `set_state`, ktorá vzhľadom na hodnotu zo signálu povoluje, respektíve zakazuje zmenu aktívneho projektu a vykonávanie určitých operácií nad projektom.

Nasledujúce metódy, reprezentujú jednotlivé operácie nad projektom. Všetky z nich požadujú parameter `action`, cez ktorý sa predá odkaz na instanciu triedy `QAction` pomocou ktorej uživateľ zvolil vybranú operáciu. Mapovanie nasledujúcich operácií na príkazy v príkazovom riadku operačného systému sú zobrazené v tabuľke 4.1. Preklad projektu za-

bezpečujú metódy `build` a `build_remotely`, kde v prvá spomínaná preloží projekt lokálne na zariadení, na ktorom sa pracuje a druhá vykoná preklad vzdialene na zvolenom servery. Naprogramovanie aktuálneho projektu do zariadenia zabezpečuje metóda `load`. Táto metóda na rozdiel od ostatných vyžaduje ešte jeden potrebný parameter, ktorým je identifikačné číslo zariadenia, do ktorého sa má aktuálny projekt naprogramovať. Metóda `run_ise` otvorí aktuálny projekt ako projekt v programe Xilinx ISE a obdobná metóda `run_isim` zase spustím simuláciu aktuálneho projektu v programe Xilinx ISE. Metóda `clean` vyčistí adresár aktuálneho projektu od súborov, ktoré vznikli po preklade.

Operácia (metóda)	Zoznam reťazcov v Pythone reprezentujúcich príkazy
`build`	*"fcmake {path}project.xml"* *"make -C {path}"*
`buil_remotely`	*"fcmake {path}project.xml –remote {user}@{server}"* *"fcmake {path}project.xml –remote-build {user}@{server}"* *"make -C {path}"*
`load`	*"make load -C {path} USBDEVID={dev_id}"*
`run_ise`	*"make ise -C {path}"*
`run_isim`	*"make isim -C {path}"*
`clean`	*"make purge -C {path}"*

Tabuľka 4.1: Mapovanie operácií na príkazy v príkazovom riadku operačného systému.

4.3 Balík `device`

V tomto balíku sú zahrňuté všetky triedy a konštanty týkajúce sa obsluhy a komunikácie s pripojenými zariadeniami.

4.3.1 Trieda `FitKit`

Reprezentácia zariadenia v systéme je zabezpečená triedou `FitKit`, ktorá rozširuje triedu `SerialDevice` z knižnice `pylibftdi`. Pri inicializácií nie je potrebné zadávať žiadny parameter, no v tom prípade sa použijú východzie hodnoty. Dôležitou pridanou metódou je `reset_mcu`, ktorej implementácia vykoná reset mikrokontroléru na zariadení. Statická metóda `get_devices` inicializuje instanciu triedy `Driver`, taktiež z knižnice `pylibftdi`, pomocou ktorej vráti zoznam s informáciami o všetkých aktuálne pripojených zariadeniach zavolaním metódy `list_devices`.

4.3.2 Trieda `FitkitWorker`

Trieda `FitkitWorker` komunikuje so zariadením a číta z neho výstup, ktorý ďalej predáva pomocou signálu `output`. Slot `run` implementuje čítanie výstupu zo zariadenia. Čítanie je ukončené až v prípade, že bolo uzatvorené spojenie so zariadením. Inicializácia vyžaduje

odkaz na instanciu triedy `FitKit` ktorú uchováva v atribúte `device`. Táto trieda rozširuje vlastnosti triedy `QObject`

4.3.3 Trieda `DeviceManager`

Správa pripojených zariadení je realizovaná pomocou triedy `DeviceManager`, ktorá v atribúte `driver` uchováva instanciu triedy `Device` z knižnice `pylibftdi` a atribút `device_list` triedy `DeviceList` slúžiaci na uchovanie aktuálne pripojených zariadení vo forme instancií triedy `FitKit`. Trieda `DeviceList` predstavuje typ `dict` ako objekt, pretože pomocou signálov nie je možné tento typ prenášať ale objekt už áno. Detekovania pripojených zariadení umožňuje slot `run` predstavujúci nekonečný cyklus v ktorom sa neustále zisťuje stav zariadení. Stav sa zisťuje pomocou metódy `list_devices` z objektu v atribúte `driver`, ktorá vracia zoznam s informáciami o pripojených zariadeniach. Ak nastane zmena tak sa skontroluje stav zariadení na základe ktorého sa upraví obsah v `device_list`.

4.4 Balík `main`

Tento balík obsahuje triedy objektov, ktoré môže užívateľ otvoriť a pracovať s nimi v hlavnej časti okna aplikácie. Tými objektami sú konzola pre obsluhu FITkitu a textový editor podporujúci zvýraznenie syntaxe. V nasledujúcich podkapitolách je popísaná ich implementácia a fungovanie.

4.4.1 Trieda `Console`

Sprístupnenie komunikácie so zariadením je implementované triedou `Console`. Rozširuje základnú triedu `QWidget` a predstavuje objekt, v ktorom sú usporiadané instacie tried `QPlainTextEdit` na zobrazenie výstupu zo zariadenia, `QPushButton` a `QLineEdit` na zadávanie a odosielanie príkazov. Tieto objekty sú usporiadané tak, aby spolu logicky predstavovali pre užívateľa konzolové okno. Inicializácia vyžaduje odkaz na instanciu triedy `PyDevKit` a identifikačné číslo zariadenia, na základe ktorého zvolí zariadenie z atribútu `connected_devices` z hlavnej aplikácie a otvorí spojenie. Samotnú komunikáciu so zariadením zabezpečuje instancia triedy `FitkitWorker` popísanej v kapitole 4.3.2, ktorá je presunutá do pozadia do vlákna a komunikuje s konzolou pomocou signálu `output`. Výstup zo signálu `output` je zviazaný so slotom `show_output`, ktorý odstráni z neho nepodporované znaky a vloží ho do objektu v atribúte `output_viewer` už spomínanej triedy `QPlainTextEdit`. Zapisovanie príkazov do zariadenia implementuje slot `send_command`, ktorý zoberie vstupný text a pridá k nemu znak `"\n"`. Následne je tento text zapísaný do zariadenia a interpretovaný ako príkaz.

4.4.2 Trieda `Editor`

Trieda `Editor` rozširuje funkcionalitu triedy `QPlainTextEdit` vďaka čomu reprezentuje editor zdrojových kódov s podporou zvýrazňovania syntaxe. Inicializácia vyžaduje cestu k súboru, ktorého obsah sa má sprístupniť uživateľovi. Na základe prípony súboru sa určí o aký typ súboru sa jedná a zvolí sa príslušná trieda, ktorá implementuje jeho zvýrazňovanie syntaxe z balíka `highlighters` popísaného v kapitole 4.5. Ak k rozpoznanému typu súboru nie je možné priradiť vhodnú triedu, zdrojový kód zobrazený v editore nebude zvýrazňovaný. Počas inicializácie metóda `set_style` nastaví vlastnosti editora ako sú typ, veľkosť, farba písma, ďalej farba pozadia a dĺžka tabulátoru. V atribúte `line_number_area`

je instancia triedy `LineNumberArea`, ktorá pridáva do editora grafický prvok zobrazujúci číslovanie riadkov textu. Samotné zobrazenie číslovania riadkov je implementované v metóde `line_number_area_paint_event`, ktorá postupne prechádza jednotlivé bloky, teda riadky textu a priradzuje im číselné hodnoty, ktoré následné zobrazí v grafickom prvku typu `LineNumberArea`. Aktuálna šírku grafického prvku závisí od maximálnej zobrazovanej hodnoty a mení sa s narastajúcim počtom číslic. O jej korektné zobrazenie sa stará metóda `get_line_number_area_width`. Farebné zvýraznenie aktuálneho riadku na ktorom sa nachádza kurzor implementuje slot `highlight_current_line`.

4.5 Balík `highlighters`

Tento balík obsahuje tri hlavné triedy `CHighlighter`, `XMLHighlighter` a `VHDHighlighter`, ktorých účelom je zvýrazňovanie syntaxe podľa vybraného programovacieho jazyka a ďalej obsahuje podporné funkcie a konštanty pre tieto triedy.

Všetky spomenuté triedy dedia zo štandardnej triedy `QSyntaxHighlighter` dostupnej v PyQt, ktorá nezvýrazňuje syntax pre konkrétny programovací jazyk, no zabezpečuje všetku réžiu spojenú so zvýrazňovaním. Dosiahnutie zvýrazňovania syntaxe a prispôsobenie pre konkrétny jazyk je možné dodatočnou modifikáciou a rozšíreným tejto triedy o všetky syntaktické informácie týkajúce sa daného jazyka, ako sú napríklad kľúčové slová, aritmetické, relačné alebo binárne operátory, zátvorky, zápis komentárov a podobne. Tieto informácie sú spracované a uložené vo forme regulárnych výrazov v atribúte triedy `highlighter_rules`. K samotnému zvýrazňovaniu bloku textu je potrebné redefinovať metódu `highlightBlock`, ktorá postupne prechádza všetky pravidlá a hľadá ich výskyty podľa regulárneho výrazu v texte. Následne sú tieto výskyty zvýraznené, tak aby odpovedali vybranému formátu. Zvýrazňovanie syntaxe sa vykonáva okamžite so zmenou textu.

4.6 Balík `docks`

V tomto balíku sú organizované triedy starajúce sa o analýzu a zobrazenie SVN repozitára s projektami a zobrazovanie samotnej stromovej štruktúry súborov, ktoré k danému projektu náležia.

4.6.1 Triedy `ProjectExplorer`

Trieda `ProjectExplorer` rozširuje triedu `QTreeWidget` a je prispôsobená k zobrazeniu projektov z SVN repozitára v stromovej štruktúre. Inicializácia vyžaduje povinný parameter, ktorým je hlavná aplikácia, teda odkaz na instanciu triedy `PyDevKit` a nepovinný parameter cestu k SVN repozitáry. Pri inicializácií sa zavolá metóda `parse_projects`, ktorá rekurzívne prejde všetky súbory a v prípade, že je to súbor s príponou *xml*, tak vykoná jeho analýzu a rozhodne či daný adresár reprezentuje projekt alebo len kategóriu obsahujúcu viac projektov. V prípade, že sa jedná o projekt tak je zavolaná statická metóda `parse_project`, ktorá detailne rozoberie obsah daného *xml* súboru. K spracovaniu obsahu vo formáte XML[3] je použitá štandardná knižnica `minidom` z balíku `xml.dom`. Informácie o rozdelení do kategórií a o jednotlivých projektoch sú uložené vo forme slovníka v atribúte `projects`. Po analýze metóda `fill_explorer` tieto informácie vloží do stromovej štruktúry samotnej instancie tejto triedy. Prvky v štruktúre sú instancie triedy `ProjectExplorerItem`, ktorá

[3] http://www.w3schools.com/xml

môže predstavovať buď kategóriu alebo konkrétny projekt. V prípade, že sa jedná o projekt, tak potom táto trieda uchováva všetky príslušné informácie ako sú cesta, názov, informácie o autorovi, revízia a stručný popis projektu. K zobrazeniu súborov daného projektu po dvojkliku užívateľa, slúži metóda `mouseDoubleClickEvent` vyvolávajúca z instancie hlavnej aplikácie metódu `open_new_project`. Metóda `refresh` znovu vykoná analýzu repozitára a vloží aktualizovaný obsah do stromovej štruktúry.

4.6.2 Triedy `FileExplorer`

Zobrazovanie adresára projektu v stromovej štruktúre je implementované pomocou triedy `FileExplorer`, ktorá rozširuje triedu `QTreeView` o potrebnú funkcionalitu. Pri inicializácií vyžaduje povinný parameter, ktorým je hlavná aplikácia, teda odkaz na instanciu triedy `PyDevKit` a nepovinný parameter cestu k adresáru projektu. V prípade, že už pri inicializácií je definovaná cesta k adresáru projektu, tak okamžite zobrazí obsah adresára. Cestu je možné nastaviť aj neskôr pomocou metódy `set_project`, ktorej jediným parametrom je práve zvolená cesta k adresáru. K zobrazeniu a usporiadaniu súborov v stromovej štruktúre využíva triedu `FileExplorerModel`, ktorá rozširuje vlastnosti triedy `QDirModel`. Obsahuje pravidlá podľa ktorých je príslušný adresár filtrovaný a súbory v stromovej štruktúre usporiadané. Zamedzené je zobrazovanie neviditeľných súborov. Adresáre a súbory sú usporiadané podľa priority, kde adresáre majú väčšiu prioritu ako súbory. V rámci rovnakej priority sú prvky usporiadané podľa názvu, pričom sa nekladie dôraz na veľkosť písmen. Po dvojkliku na zvolený projekt je vyvolaná metóda `mouseDoubleClickEvent`, ktorú bolo nutné redefinovať, tak aby okamžite v systéme sprístupnila obsah zvoleného súboru pomocou metódy `open_file` zavolanej z hlavnej aplikácie.

4.7 Balík `utils`

Balík `utils` poskytuje všetky ostatné pomocné triedy a funkcie použité v implementácií systému PyDevKit.

4.7.1 Trieda `CommandRunner`

Beh externých príkazov zo systému je zabezpečený triedou `CommandRunner`, ktorá dedí všetky vlastnosti z triedy `QObject`. Jej inicializácia vyžaduje zoznam príkazov v poradí, v ktorom sa majú za sebou vykonať. Jednotlivé príkazy musia byť vo forme reťazca predstavujúceho príkaz umožnujúci spustenie v príkazovom riadku operačného systému. Trieda obsahuje dva signály vďaka ktorých je schopná komunikovať s inými objektami. Pomocou signálu `output`, predáva aktuálny výstup bežiaceho príkazu vo forme reťazcu. Signál `running` reprezentuje aktuálny stav behu príkazov, pri začiatku vykonávania predáva hodnotu `True` a v okamžiku, keď skončila činnosť posledného príkazu predáva hodnotu `False`. Vstupné príkazy sa vykonajú pri vyvolaní slotu `run`. O spustenie príkazu a následné čítanie výstupu zo STDOUT prípadne STDERR sa stará trieda `Popen`, ktorá je zo štandardnej knižnice `subprocess`. V prípade, že sa návratový kód vykonaného príkazu nerovná hodnote `0` je ukončené vykonávanie ostatných príkazov.

Kapitola 5

Zhodnotenie výsledného systému a ďalší vývoj

Zhodnotenie výsledného systému je možné vykonať porovnaním cieľov určených v kapitole 3.1 a dosiahnutých výsledkov. Implementovaný systém PyDevKit zatiaľ poskytuje základné operácie umožnujúce prácu s FITkitom. Z časových dôvodov nebolo možné implementovať jeho plnú funkcionalitu, pretože navrhnutý systém bol príliš rozsiahly, ale podarilo sa napriek tomu splniť hlavný cieľ, ktorým bolo vytvorenie multiplatformného nástroja podporujúceho väčšinu dostupných funkcií z QDevKitu. V systéme je integrovaná podpora prekladové systému a kompletná obsluha projektu z hľadiska jeho implementácie a programovania. Systém je vybavený funkčkým editorom zdrojových kódov s podporou zvýrazňovania syntaxe, čo prináša oproti starému systému QDevKit obrovskú výhodu z hľadiska efektivity a pohodlia. Nástroj ďalej disponuje podporou nízkoúrovňovej komunikácie s možnosťou výberu zariadenia FITkit prostredníctvom konzoly.

Systém vo svojej výslednej podobe skrýva aj niekoľko nedostatkov. SVN repozitár je nutné synchronizovať a aktualizovať manuálne z príkazového riadku. Zatiaľ je možné obsah tohto repozitára spracovať a zobraziť ho v užívatelsky prívetivej forme. Správca pri vytváraní nového projektu nie je implementovaný. Stále zostáva nutnosť použiť externé nástroje k vytvoreniu nového projektu a upraveniu jeho parametrov. V prípade potreby meniť nastavenia celej aplikácie je nevyhnutné zastiahnuť do konfiguračného súbory *config.py* a manuálnou úpravou prispôsobiť hodnoty. Dostupné parametre, ktoré je možno meniť sú adresa vzdialeného repozitára, parametre pripojenia FITkitu, potrebné údaje pre vykonanie vzdialeného prekladu a nastavenie ciest externých nástrojov. Okrem toho PyDevKit vo svojej súčastnej podobe obsahuje niekoľko chýb, ktoré je možné postupným testovaním na koncových užívateľoch odhaliť a následne minimalizovať tieto nedostatky. Neočakávané chyby môžeme spojiť so vzdialeným prekladom. Tie menej vážnejšie sú spojené napríklad so zvýrazňovaným syntaxe. Nejedná sa však o kritické chyby, ktoré by závažne obmedzovali funkčnosť systému.

V prípade priestoru pre ďalší vývoj systému by bolo prínosné rozšíriť funkcionalitu vzdialeného prekladu o možnosť tunelovania v prípade, kedy sa užívateľ nenachádza v sieti VUT prípadne odpadne potreba prípajania sa na sieť VUT pomocou služby VPN. Ďalším prínosným prvkom, ktorým by bolo vhodné rozšíriť systém PyDevKit je rozsiahle uživateľské nastavenie, ktoré by zabezpečovalo prispôsobenie si grafického rozhrania svojim potrebám. Určite by bolo veľkým prínosom aby systém PyDevKit podporoval viac jazykov vo svojom grafickom rozhraní.

Kapitola 6

Záver

Cieľom aplikácie bolo vytvoriť plnohodnotnú náhradu za stávajúci obslužný systém, ktorý by poskytoval študentom pohodlné prostredie pre programovanie aplikácií na platforme FITkit.

Vývoj systému bol rozdelený do niekoľkých krokov. Na začiatku bolo potrebné vykonať štúdiu stávajúceho systému QDevKit a zoznámenie sa s jeho funkcionalitou a implementáciou. Dôležitým krokom bolo vykonanie analýzy potrebných technológií a nástrojov aby zostala zachovaná myšlienka multiplatformnej aplikácie. Nasledoval návrh štruktúry systému a grafického rozhrania, pri ktorom sa kládol dôraz na pohodlie pri obsluhe FITkitu. Ďalším krokom bola implementácia podľa vykonaného návrhu. Na záver sa vyhodnotil stav finálneho systému a vykonala diskusia nad jeho nedostatkami a ďalším vývojom.

Určené cieľové požiadavky sa podarilo z časti splniť, vznikol nový použitelný systém schopný nahradiť vo výuke súčasný QDevKit. Oproti nemu navyše ponúka mnoho inovácií ako sú editor zdrojových kódov a pokročilá správa projektov. Výsledný systém je rozsiahli nástroj poskytujúci mnoho rozšírení oproti QDevKitu vďaka čomu spraví prácu s FITkitom na počítači pohodlnejšiu a príjemnejšiu pre užívateľa.

PyDevKit taktiež obsahuje niekoľko oblastí ponúkajúcich priestor pre ďalší vývoj. Tou hlavnou je doimplementovanie zostávajúcich častí. Určite ho je ďalej možné rozšíriť o funkcionalitu tunelovania pripojenia pri vzdialenom preklade pomocou ktorého by bol užívateľ schopný vykonať vzdialený preklad aj mimo sieť VUT bez pomoci technológie VPN. Ďalej by bolo vhodné rozšíriť systém o rozsiahlejšie nastavenia ktorými by si užívateľ bol schopný prispôsobiť uživateľské prostredie pre svoje potreby. Rovnako by bolo prínosné aby systém PyDevKit ponúkal možnosť zobrazenia grafického užívateľského rozhrania vo viacerých jazykoch.

Literatúra

[1] *GUI Programming in Python*. [Online; navštívené 16.5.2016].
URL https://wiki.python.org/moin/GuiProgramming

[2] *PyQt5 Reference Guide*. [Online; navštívené 17.5.2016].
URL https://www.riverbankcomputing.com/software/pyqt/intro

[3] *Welcome to pylibftdi's documentation!* [Online; navštívené 17.5.2016].
URL https://pylibftdi.readthedocs.io/

[4] *wxPython Online Docs*. [Online; navštívené 17.5.2016].
URL http://www.wxpython.org/onlinedocs.php

[5] Lott, S. F.: *Mastering Object-oriented Python*. Packt Publishing Ltd., 2014, ISBN 978-1-78328-097-1.

[6] Michal Bidlo: *Zprovoznění FITkitu v nativním prostředí Linuxu*. Fakulta informačních technologií VUT Brno, 2015, [Online; navštívené 17.5.2016].
URL http://merlin.fit.vutbr.cz/FITkit/docs/navody/qdevkitlinux.html

[7] Summerfield, M.: *Programming in Python 3 : a complete introduction to the Python language*. Addison-Wesley, 2010, ISBN 978-0-321-68056-3.

[8] Zdeněk Vašíček: *Download*. Fakulta informačních technologií VUT Brno, [Online; navštívené 16.5.2016].
URL
https://merlin.fit.vutbr.cz/FITkit/private/web/index.php?pg=download

[9] Zdeněk Vašíček: *Úvod*. Fakulta informačních technologií VUT Brno, [Online; navštívené 17.5.2016].
URL http://merlin.fit.vutbr.cz/FITkit/uvod.html

[10] Zdeněk Vašíček: *Návody*. Fakulta informačních technologií VUT Brno, 2015, [Online; navštívené 17.5.2016].
URL http://merlin.fit.vutbr.cz/FITkit/navody.html

Prílohy

Zoznam príloh

Príloha A

Obsah CD

- `src/` – zdrojové kódy systému
- `doc/` – zdrojové kódy technickej správy
- `projekt.pdf` – technická správa
- `README` – manuál k inštalácií systému

Príloha B

Manuál k inštalácií

K zaisteniu behu systému PyDevKit na natívnom operačnom systéme je nutné zabezpečiť inštaláciu nasledujúcich nástrojov a knižníc:

- Postupovať podľa návodou dostupných na webových stránkach[10], k zaisteniu existencie potrebných nástrojov `fcmake`, `fkflash`, `Xilinx ISE`, `MSP430-gcc` a `Subversion`.
- Nainštalovať prostredie Python 3. Inštalácia sa líši v závislosti od použitého operačného systému. V prípade Linuxu je možné inštaláciu vykonať z príkazového riadku.

  ```
  sudo apt-get install python3 python3-dev python3-pip
  ```

  Prostredie Python 3 pre Windows je dostupné na webových stránkach.

  https://www.python.org/downloads/
- Nainštalovať externú knižnicu pomocou nástroja `pip3`.

  ```
  sudo pip3 install pylibftdi
  ```

- Nevyhnutná je inštalácia knižnice `Qt` pomocou inštalátoru dostupného na webových stránkach.

  https://www.qt.io/download-open-source/
- Na záver je potrebné nainštalovať viazanie `PyQt`. Pri inštalácií je vhodné nasledovať pokyny dostupné na webových stránkach.

  http://pyqt.sourceforge.net/Docs/PyQt5/installation.html

Po nainštalovaní potrebných závislostí je možné systém spustiť pomocou súboru `pydevkit`.