TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

GRAFICKÉ SW ROZHRANÍ VÝUKOVÉHO SYSTÉMU ROBOKIT

Bakalářská práce

Studijní program: B2646 – Informační technologie
Studijní obor: 1802R007 – Informační technologie

Autor práce: **Tomáš Kadleček**
Vedoucí práce: Ing. Zbyněk Mader, Ph.D.

Liberec 2015

TECHNICAL UNIVERSITY OF LIBEREC
Faculty of Mechatronics, Informatics
and Interdisciplinary Studies

GRAPHICAL SW INTERFACES LEARNING SYSTEM ROBOKIT

Bachelor thesis

Study programme: B2646 – Information Technology
Study branch: 1802R007 – Information Technology

Author: **Tomáš Kadleček**
Supervisor: Ing. Zbyněk Mader, Ph.D.

Liberec 2015

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky a mezioborových studií
Akademický rok: **2014/2015**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Tomáš Kadleček**
Osobní číslo:	**M12000141**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**Grafické SW rozhraní výukového systému RoboKit**
Zadávající katedra:	**Ústav informačních technologií a elektroniky**

Zásady pro vypracování:

1. Analyzujte možnosti vizuálního programovacího jazyka mikrořadiče PICAXE.
2. Vytvořte uživatelské rozhraní pro grafické zadání programu určenému mikrořadiči PICAXE s předdefinovanou sadou příkazových instrukcí.
3. Programovací prostředí umožní vytvořit a editovat program v grafické podobě pomocí grafických bloků, zobrazit jej v textové podobě a následně naprogramovat procesor PICAXE připojený k počítači přes USB rozhraní.

Rozsah grafických prací: **Dle potřeby dokumentace**

Rozsah pracovní zprávy: **cca 30 stran**

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

[1] **Hackett, Ron: PICAXE Microcontroller Projects for the Evil Genius, McGraw-Hill Companies, 2011, ISBN 978-0-07-170327-7**

Vedoucí bakalářské práce: **Ing. Zbyněk Mader, Ph.D.**
Ústav informačních technologií a elektroniky

Datum zadání bakalářské práce: **12. září 2014**

Termín odevzdání bakalářské práce: **15. května 2015**

prof. Ing. Václav Kopecký, CSc.
děkan

L.S.

prof. Ing. Zdeněk Plíva, Ph.D.
vedoucí ústavu

V Liberci dne 12. září 2014

Prohlášení

Byl jsem seznámen s tím, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu TUL.

Užiji-li bakalářskou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Bakalářskou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím mé bakalářské práce a konzultantem.

Současně čestně prohlašuji, že tištěná verze práce se shoduje s elektronickou verzí, vloženou do IS STAG.

Datum: 15.5.2015

Podpis: Kaskel

Poděkování

Rád bych poděkoval všem lidem, kteří se na této práci přímo či nepřímo podíleli. Mezi všemi byl největší oporou Ing. Zbyněk Mader Ph.D., kterému děkuji za poskytnuté informace, trpělivost, rady a materiály, které mi poskytl pro vykonání mé bakalářské práce. Nakonec bych rád poděkoval své rodině a přítelkyni pro jejich podporu, ať materiální či duševní.

Abstrakt

Zadáním mé bakalářské práce je vytvořit program, který by byl použitelný pro výuku základního programování. Vytvořený program musí splňovat některé požadavky jednoduchého a uživatelsky přívětivého stylu. Tyto nároky jsou na něj kladeny z důvodu, že bude využit na základních a středních školách pro výuku programování. Výsledná aplikace bude poskytovat studentovi potřebné nástroje pro naprogramování jednoduchého mikroprocesoru Picaxe. Hlavní náplní mé práce bylo navrhnout a vytvořit vhodný design a zabezpečit funkčnost celé aplikace. Program bude schopný pomocí grafických bloků, které sestaví uživatel, vygenerovat zdrojový kód v textové podobě a nakonec jím naprogramovat mikroprocesor Picaxe na elektronickém zařízení. Dále se ve své práci zaměřuji na analýzu dalších vhodných programů pro grafické programování mikroprocesorů Picaxe. Programy, které rozebírám, jsou: Rogic, 12Block, S4A a Logicator. Budu se zaměřovat zejména na funkčnost aplikací a jejich reálné nasazení. Dále ověřuji podporu ze strany výrobce (dokumentace, oprava chyb) a případný další vývoj.

Klíčová slova: grafické programování, výuka programování, mikroprocesory Picaxe

Abstract

The main task of my bachelor thesis was creation of program, which could be useful for elementary teaching of programming. Due to this task the created program will have simple and user-friendly interface. The program will be able to assemble graphical draft and generate source code in text form. Furthermore, the generated source code will be used to program microprocesor of the electronic device. I evaulate the graphic programing. Programs which I will evaluate are: Rogic, 12Block, S4A and Logicator. Main aim of the programs will be their functionality and real application in education. I also deal with the support of the software companies (documentation, error correction), and further development of the application.

Keywords: graphical programming, teaching programming, microprocesor Picaxe

Obsah

Seznam obrázků

Seznam tabulek

Seznam zdrojových kódů

1 Předmluva

Téma mé bakalářské práce je navrhnout grafické rozhraní pro výukový systém RoboKit. Funkční program se stane jeho součástí a bude sloužit k prvnímu seznámení s programováním mikroprocesoru Picaxe. Toto téma jsem si vybral, protože mě zaujalo téma vytvořit jednoduchý programovací editor, který by podporoval výuku programování. Podle mého názoru je nezbytné, aby se výuka programování dostala do osnov výuky, protože v dnešní době se téměř žádný obor neobejde bez zásahu informačních technologií. Dále rozvijí u žaku především kreativitu, analytické myšlení, informatické a matematické dovednosti.

Programování na základních škola se učí spíše výjimečně a programování mikroprocesorů naprosto vůbec. Vstříc jim nepřichází ani nabídka programů, které jsou k dispozici na českém trhu. Většina programů k dispozici jsou buď zastaralé, nebo nefunkční. Vhodné programy existují v zahraničí, ale zde nastává problém jazykové bariéry a zajištění další podpory. V této práci provedu srovnání programovacích editorů, které jsou v současné době k dispozici. Bude jím programovací editor Rogic ve verzi 3.3.0.1. Tento programovací editor je od firmy RoboRobo. Dále dva velice podobné programy 12Bloks a editor S2P. A posledním z hodnocených bude Logicator dodávaný přímo výrobcem mikroprocesorů Picaxe.

Výsledkem má práce bude program, ve kterém budeme schopni pomocí jednoduchého a intuitivního prostředí navrhnout program a posléze nahrát do zařízení. Aplikace by měla řešit nedostatky a nevýhody aktuálně dostupných řešení.

2 Mikroprocesor Picaxe

2.1 Základní informace

Mikroprocesory, kterými se v této práci zabývat a budu chopen naprogramovat, jsou od společnosti MicroChips. Tyto mikroprocesory jsou při výrobě doplněny zavaděčem. Ten z těchto zařízení dělá inteligentní jednočipové mikroprocesory, které se díky tomu jednoduše programují. Bez zavaděče se z Picaxe stává obyčejný mikroprocesor programovatelný v assembleru. Používat se bude konkrétně mikroprocesor 20M2. Program se rozhodně neomezí jen na tento typ. Bude moc být použit pro široké spektrum mikroprocesorů rodiny Picaxe z důvodu univerzálního jazyka, který používá. Tento jednočipový mikrokontrolér je jeden nejrozšířenějších a nejpoužívanějších mikroprocesorů ve výuce a v širokém rozsahu projektů. Hlavním důvodem je jeho cena, která je velice nízká a snadná dostupnost.

2.2 Parametry

Mikroprocesory Picaxe mají několik rozličných parametrů. Jejich hodnoty se liší podle řad. Například počtem pinů, paměti RAM a kapacity. Ve své práci se zaměřuji na konkrétní typ 20M2. Tento mikroprocesor vyniká zejména cenou v porovnání se svými parametry a je nejlepší volbou pro tuto práci. Podrobnější parametry viz *TABULKA 2.1*.

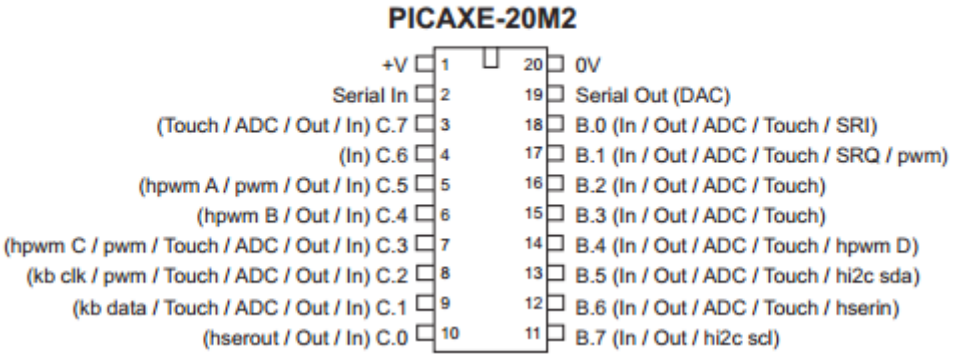
Parametry	Hodnoty
Paměťová kapacita (bytes)	2048
RAM(bytes)	512
Bytové proměnné (bytes)	28
Výstupní/Vstupní piny	18
ADC piny	11
Maximální frekvence(MHz)	32
Sériový Vstup/Výstup	Ano
Infračervený Vstup/Výstup	Ano
I2C	Ano
Melodie (tony)	Ano
Paralelní úlohy	8
Programové sloty	2

Tabulka 2.1 - Parametry mikroprocesoru Picaxe 20M2

2.3 Příkazy v procesorech Picaxe

Procesory Picaxe se programují příkazy, které jsou podobné assembleru. [1] Rozdíl je v tom, že tyto příkazy jsou podstatně jednoduší a názornější. Vycházejí také z jazyka BASIC. Můžeme je dělit do několika skupin podle funkčnosti.

- Práce s digitálními vstupy a výstupy
- Práce s analogovými vstupy a výstupy
- Pozastavení a ukončení programu
- Řízení toku programu
- Přerušení a multitasking
- Práce s proměnnými
- Komunikační rozhraní (Sériové,I2C,SPI)
- Konfigurační příkazy
- Direktivy



Obrázek 2.1 - Schéma mikroprocesoru Picaxe 20M2 (zdroj: [1])

2.4 Programování procesoru

Mikroprocesory Picaxe lze naprogramovat několika různými způsoby. Nejjednodušší přístup je použít vývojové prostředí vyvíjení přímo Picaxe. Toto prostředí obsahuje mnoho užitečných nástrojů, jako jsou debugging zdrojového kódu nebo simulace konkrétního mikroprocesoru. Tímto odpadá nutnost vlastnit hardware. Editor Picaxe také obsahuje základní editor pro grafické programování. Tento styl programování je podobný jako u dále hodnocených editorů, ale nelze ho s nimi srovnávat. Tento editor nesplňuje ani jeden požadavek, které jsme si stanovili na počátku práce a tím je jednoduchost a přímost. Tento editor je velice nepřehledný a práce s ním není vůbec intuitivní. Další

nevýhodou je nelogické řazení bloků a nadbytek funkcí pro základní výuky programování. Pro výuky cílové skupiny tento program nedoporučuji je vhodný spíše pro pokročilejší skupinu uživatelů. Program se specializuje primárně na textové programování a nikoli grafické. Nové verze vývojového prostředí Picaxe přinášejí pokročilejší grafický editor Logicator, kterým se podrobněji zabývám ve své práci. Další možností je samozřejmě využití obyčejného poznámkového bloku nebo jakéhokoli textového editoru, který neformátuje text. V tomto případě je ještě třeba stažení programu, který zdrojový kód zkompiluje a přenese do zařízení.

3 Porovnání dostupných programů

Na trhu je k dispozici několik programů vhodných k jednoduchému programování mikroprocesorů Picaxe. [2] V bakalářské práci se budu zabývat výběrem čtyř - Rogic, 12Block, S2P a Logicator. Aplikace budou porovnávány z několika hledisek.

- funkčnost a uživatelsky přívětivé prostředí
- grafický vzhled
- učební materiály dostupné v českém nebo anglickém jazyce
- dostupnost a cena pořízení
- podpora ze strany výrobce (aktualizace a oprava chyb programu)

3.1 Rogic

Program Rogic je od Korejské firmy RoboRobo. Oficiální sídlo společnosti je v Soulu v Jižní Koreji. Tento software je dodávaná k výukové sadě RoboKit, která obsahuje. Poskytuje základní prostředí pro naprogramování mikroprocesoru. Disponuje vcelku intuitivním prostředím a na první pohled i jednoduchým ovládáním.

3.1.1 Grafické rozhraní

Je rozdělen na několik částí. Největší část vyplňuje editor, do kterého se přesunují bloky a kde se spojují. Na levé straně se nachází nabídka jednotlivých bloků. Po kliknutí na blok dojde k rozbalení dalších funkčních bloků. Pod bloky se nachází tlačítko pro odstranění bloků. Dále v horní pravé části je základní nabídka a v levé nabídka pro nahrání a spuštění kódu na zařízení. Nabídka dále obsahuje tlačítka pro zavření aplikace a odkaz na webové stránky výrobce. V dolním panelu je zobrazena stavová informace o připojení k zařízení. Po grafické stránce je program naprosto v pořádku. Splňuje základní nároky, to znamená jednoduchou a přímou grafiku.

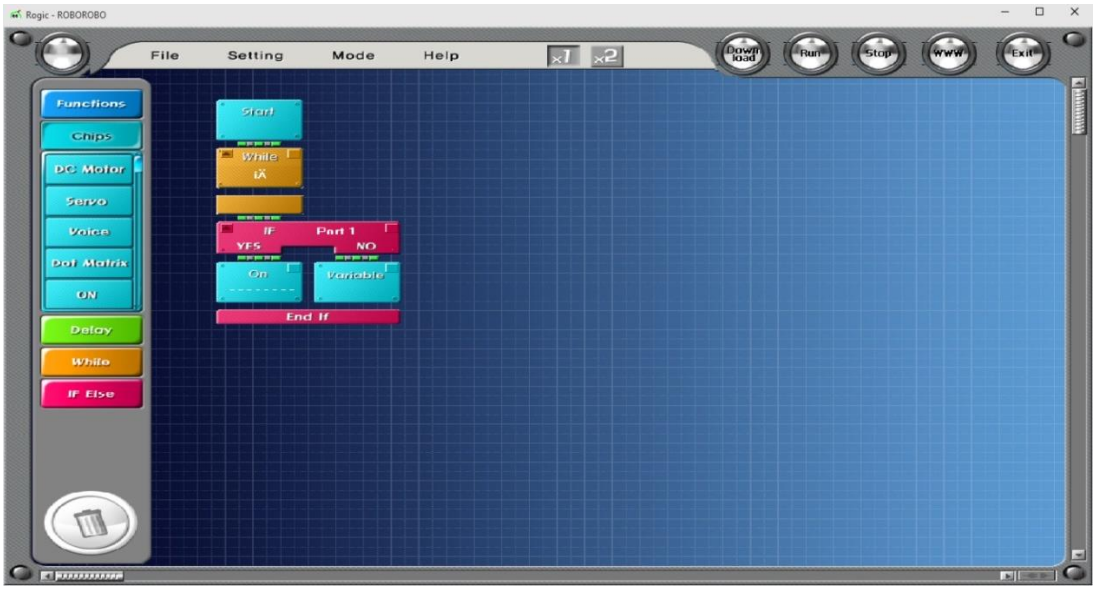
3.1.2 Funkčnost programu

Na první pohled se jeví funkčnost celkem ucházející a vše funguje správně. Bloky, které představují jednotlivé elementy, se přesunují z nabídky panelů do editoru. Při přesunu nastává první problém. Aby se blok přesunul, tak je zapotřebí na něj klinout, přirozenější cestou by bylo držení myši a jeho přetažení do pracovní plochy. V editoru se dále jednotlivé elementy spojují přetažením bloku nad ten, který k němu chceme připojit.

Aktivní připojení je označeno třemi čtverci mezi propojenými částmi. Dalším problém je přesun a mazání. Pro přesun musíme znovu nejprve klinout a pak táhnout, přirozenější by bylo opět držení myši a přesunování. Odstranění je prováněno přenesením bloku nad ikonu „popelnice“. Zde by bylo vhodnější použít klávesu DELETE. Celkový editor je v pořádku. Má téměř neomezenou velikost, takže se nemůže stát že, vyjedeme ovládacím prvkem mimo editor. Další kladem je signalizace připojení k procesoru v pravém dolním rohu. Po vytvoření funkčního programu je třeba ještě tento software nahrát do zařízení. A v tomto začíná hlavní problém celé aplikace. Pokusy o nahrání na novějších systémech Windows Vista a výše byli neúspěšné. Vzhledem k tomu že, operační systémy, na kterých funguje, se téměř nepoužívají, tak se tento program stává nepoužitelný. Domnívám se, že chybu způsobuje nekompatibilita některých ovladačů s operačním systémem.

3.1.3 Celkové hodnocení

Program Rogic po grafické stránce splňuje vymezené cíle, ale z hlediska funkčnosti velice upadá. Tyto nedostatky se objeví zejména při použití novějších operačních systémů. Naproti tomu licence je zdarma, firma Rogic dodává ke svému programu celou sadu hardware pro vytvoření širokého sady robotů, senzorů apod. Další velkou nevýhodou je podpora, která ze strany výrobce naprosto chybí. Učební dokumenty a celková dokumentace k programu chybí také. Většina učebních materiálů je v angličtině nebo korejštině. Další podpora aplikace ze strany výrobce softwaru naprosto chybí.



Obrázek 3.1 - Program Rogic

3.2 12Block

Jméno dalšího programu, kterým se práci zabýváme je 12Block.Tento software je od společnosti OneRobot. Společnost se nezaměřuje pouze na tento typ programů, ale distribuuje celou řadu dalších aplikací například ViewPort a další. Tento program funguje na podobném principu jako minulý rozebíraný. Je schopen graficky, pomocí bloků, programovat několik různých typů procesorů. Například Arduino, Lego Mindstorm NXT, Parallax Propeller a Picaxe.

3.2.1 Grafické rozhraní

Po grafické stránce není program, tak příjemný a přehledný. Mnoho ovládacích prvků je skryto a program obsahuje spoustu nepotřebných nastavení. Je zaměřen spíše pro pokročilé uživatele. Pro cílovou skupinu uživatelů může působit komplikovaně. Okno programu je znovu rozděleno na několik částí. Vlevo je nabídka bloků, která je rozdělena podle typů jednotlivých ovládacích prvků. V horní části okna se nalézá několik ovládacích tlačítek a klasická kontextová nabídka. Pomocí ovládacích tlačítek jsme schopni například spouštět a nahrávat vytvořený program do zařízení. Dále obsahuje tlačítko pro uložení aktuálního rozpracovaného programu a nastavení pro výběr portu. Posledním tlačítkem nastavujeme, zda chceme pracovat v režimu fyzického zařízení, nebo simulace.

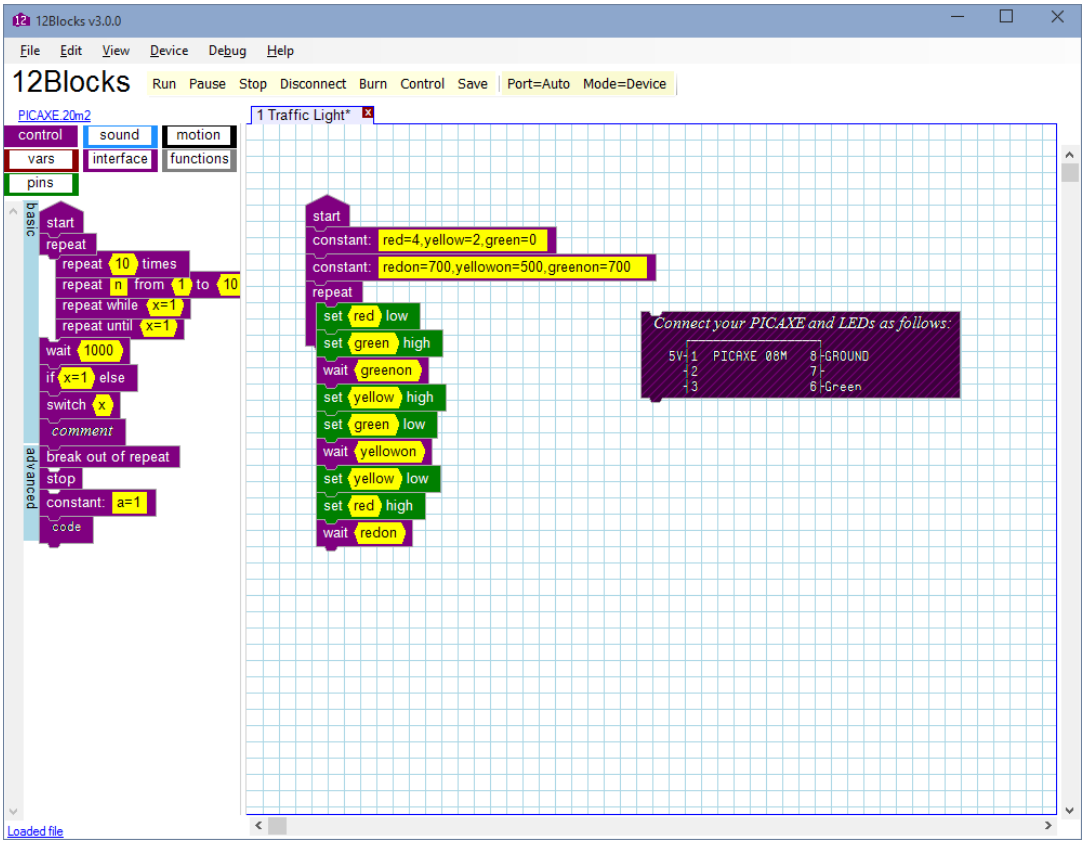
3.2.2 Funkčnost programu

Z hlediska funkčnosti je program 12Block podstatně lepší než Rogic. Práce s ním se sice složitější, ale pro pokročilejší uživatele příjemnější. Poskytuje mnohem širší možnosti úprav zdrojového kódu a ladění programu. Obsahuje také nástroje pro simulaci některých ze základních procesorů. Tyto doplňky se hodí především v případě, kdy k hardwaru nemáme plný přístup a potřebujeme otestovat jen malé změny, které jsme ve zdrojovém kódu provedli. Problémy kompatibility s novějšími systémy u této aplikace nenastaly.

3.2.3 Celkové zhodnocení

Program 12Block je kvalitnější než minulý hodnocený. Obsahuje nějaké prvky navíc, pro základnu uživatelů, na kterou se zaměřuji v mé práci naprosto zbytečných. Program Rogic překonává, jak ve funkčnosti, tak v podpoře a dokumentace. Jediná nevýhoda,

která vyvstává, je požadavek na licenci k programu. Pro použití je třeba program zakoupit. Přičemž základní jedna licence vyjde na 49 dolarů. Což je přibližně 1 200 Kč. V základní verze ale mnoho funkcí chybí například vytvoření vlastní funkce nebo tvoření polí. Plná licence se pohybuje okolo 100 dolarů za jednu. Což je pro základní školy příliš drahé. Z hlediska výukových materiálů a videí program Rogic taktéž naprosto překonává, ačkoliv většina techno učebních materiálů je v angličtině. Z těchto důvodu bych ani tento program není vhodný. Zejména z důvodů ceny a zbytečných funkcí navíc.



Obrázek 3.2 - Program 12Block

3.3 S2P

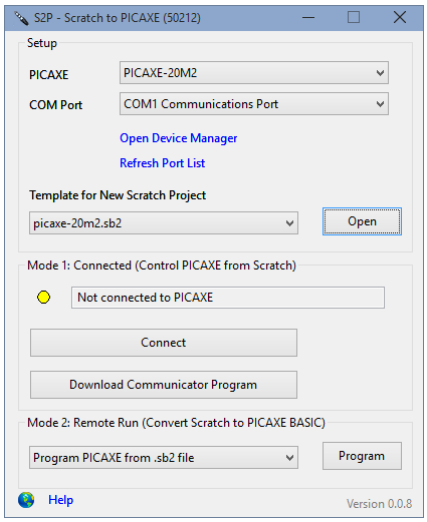
Dalším programem, kterým se práce zabývá je S2P.Tata aplikace je v podstatě doplňkem ke známému programu Scratch. Program Stratch je vyvíjen univerzitou MIT přibližně od roku 2007. Tento program paří mezi špičku grafického programování. Dokonce umožňuje tvorbu grafických animací a různých miniher jen pomocí grafického programování. Doplněk S2P umožňuje vytvořený program ve Scratchi přenést do mikroprocesoru. Nevýhodou je speciální šablona v programu, kterou musíme použít s danými bloky. Jako například zjištění hodnot pinů a jejich nastavení. Program Scratch podporuje širokou škálu možných procesorů, které můžeme pomocí něho programovat. Například velice známý, pokročilejší mikroprocesor Arduino, mnoho verzí čipů Picaxe a další.

3.3.1 Grafické rozhraní

Po grafické stránce je program Scratch velice pěkně proveden. Všechno je přehledné a grafika je velice líbivá pro cílenou skupinu uživatelů. Program se skládá pomocí funkčních bloků. Bloky do sebe zapadající jako „puzzle“. To umožňuje zanořování bloků do sebe, nebo vytvoření vlastní funkce. Další nevýhodou je situace, která nastane při použití bloků, který Picaxe. V aplikaci by se měli zobrazit jen ty, které jsou použitelné. Program Scratch je navrhnutý s cílem mnohem širšího programování například grafických aplikace a různých miniher. Obsahuje také grafické editory pro tvorbu grafiky programu, jednoduchý zvukový editor a také širší možnosti nastavení.

3.3.2 Funkčnost programu

Program Scratch je z funkčního hlediska téměř perfektní a není co mu vytknout. Je vyvíjen několik let a toho důvodu bylo dostatek času na nalezení všech chyb a nedostatků. Vzhledem k tomu, že je určen k vytváření programů ve vyšším jazyce, tak obsahuje mnohem více funkčních bloků. Tyto funkční prvky minimálně využijeme při programování mikroprocesorů, protože velkou měrou přesahují jejich možnosti. Pro naprogramování je třeba k programu Scratch stáhnout doplňkový program S2P, který je schopen vytvořenou aplikaci převést do jazyka Picaxe a posléze nahrát do zařízení.

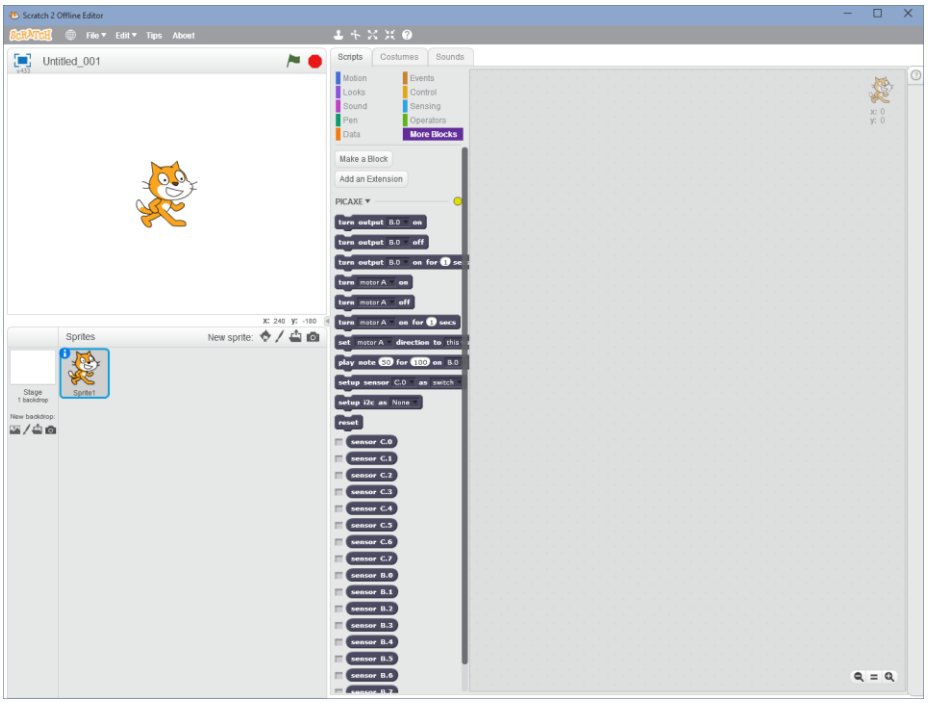


Obrázek 3.3 - Doplňkový program S2P pro Scratch

V programu Scratch je dále nutné otevřít danou šablonu se speciálními funkčními bloky. Po otevření šablony se do programovacího prostředí přidají další ovládací prvky. V této fázi není úplně jasné, jaké funkční bloky mohou být použity pro naprogramování procesoru. Například se může stát, že použijeme prvek, který mikroprocesor nezná. A z toho důvodu pak program nepůjde zkompilovat a nahrát do zařízení.

3.3.3 Celkové hodnocení

Po grafické i funkční stránce funguje naprosto bezchybně. Obsahuje širokou základnu podpory a jistotu dalšího vývoje. Dalším kladem tohoto programu je bezplatná licence a možnost práce ve webovém prohlížeči, čímž naprosto odpadá práce se zavedením programu do školy. To je hlavní důvod, proč je Scratch tolik využívaný. V ohledu na dostupnost materiálů je na tom lépe než minulé hodnocené, ale stále postrádá podrobné návody v češtině, pro konkrétní programování mikroprocesorů. Tento program je z hodnocených programů nejlepší.



Obrázek 3.4 - Program Scratch

3.4 Logicator

Posledním hodnocenou aplikací je Logicator. Tento program je součástí nových verzí editorů od Picaxe. Vzhledem k tomu, že je jeho součástí je doporučovaným programem pro grafické programování společností Picaxe. Aplikace umožňuje programování celé řady procesorů Picaxe.

3.4.1 Grafické rozhraní

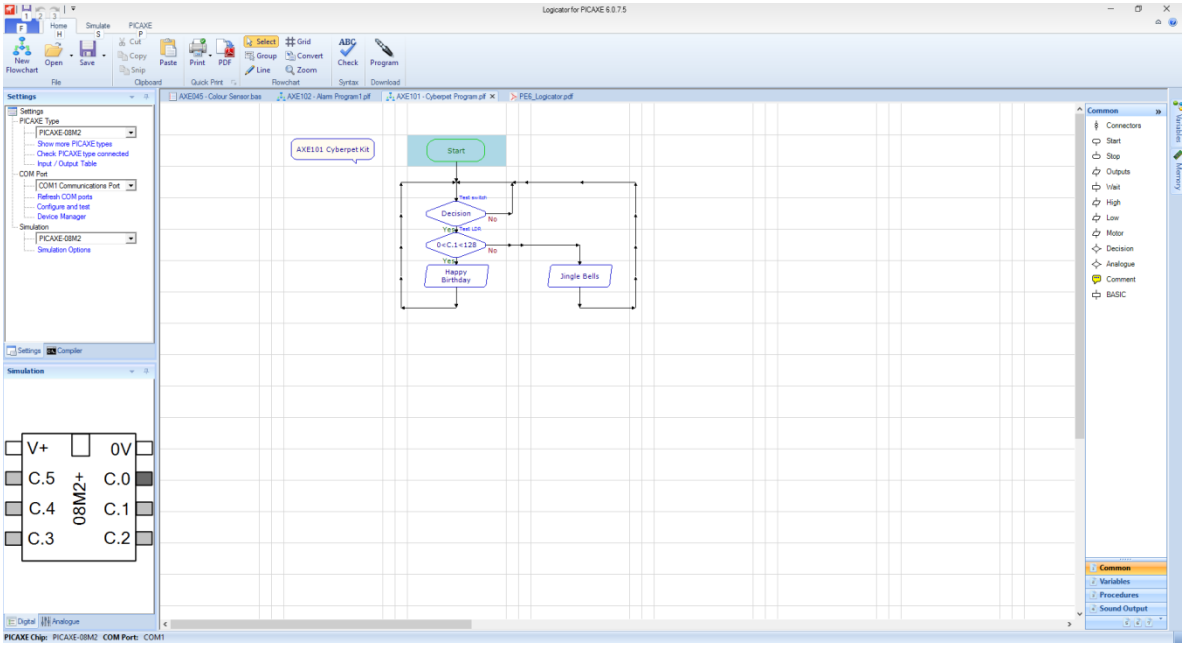
Grafickým rozhraním se mírně odlišuje od ostatních hodnocených programů. Největší rozdíl je v chování editoru, kde se spojují bloky. To funguje na principu spojování bloků šipkami. Editor je rozdělen do tabulky, podle které se zarovnávají bloky. Tabulka představuje určitý druh rastru. Nadále spojování bloků šipkami nefunguje příliš přirozeně. Například tažení od jednoho bloku k druhému. Pro spojování se používá samostatný nástroj *Connection*. Tento vloží do editoru šipku přes jedno políčko rastru. Tudíž například pro spojení bloků, které jsou vzdálenější, musíme tento postup opakovat několikrát. Rozvržení ovládacích prvků se podobá ostatním řešením.

3.4.2 Funkčnost programu

Z funkčního hlediska program funguje bezchybně. Opět pro začínající uživatel může být nepřehledný. Obsahuje některé pokročilé nástroje, které cílová skupina nevyužije. Jako jediná využívá rastr, který zarovnává bloky a jejich spojování to tabulky.

3.4.3 Celkové hodnocení

Program velice dobře použitelný pro grafické programování. Nevýhodou programu, že program postrádá českou lokalizaci. K dispozici je jen ve 4 jazykových verzí, a to v angličtině, němčině, francouzštině a španělštině. Další velkou nevýhodou je nedostupnost učebních materiálů v češtině.



3.5 Výsledné zhodnocení

Program	Klady	Zápory
Rogic	Jednoduché prostředí	Kostrbaté ovládání Nefunkčnost na novějších systémech Učební materiály
12Block	Výborná funkčnost Možnost programování více typů zařízení	Vyšší složitost Nadbytek funkcí pro cílovou skupinu Nadbytek funkcí pro cílovou skupinu Nutnost zakoupení licence Nedostupnost v českém jazyce
Scratch	Výborná ovladatelnost Výborná podpora Vývojové nástroje jsou zdarma	Vyšší složitost Nadbytek funkcí pro cílovou skupinu Nutnost použití doplňkového programu
Ligocator	Dobrá funkčnost Výborná podpora Pro potřeby výuky zdarma	Vyšší složitost Nadbytek funkcí pro cílovou skupinu Nutnost použití hardwaru od Picaxe(připojení zařízení). Nutnost zakoupení licence Nedostupnost v českém jazyce

Tabulka 3.1 - Hodnocení programů

4 Návrh řešení

Návrh a plánování řešení aplikace jsou jedny z nejdůležitějších částí celé vývoje aplikace, proto jsem jimi začal. Na začátku jsem si položil několik základních nároků, které bych chtěl, aby moje aplikace splňovala. A to jednoduchost a snadnou rozšiřitelnost. Těmto nárokům jsem se snažil přizpůsobit grafickou část a funkčního návrh programu. Po grafické stránce jsem z části vycházel z již exitujících aplikací a z části ze svého vlastního návrhu. Co se týče struktury programu, tam jsem se rozhodl pro objektový přístup a MVVM model.

4.1 Programovací jazyk

Nabízelo se hned několik vhodných programovacích jazyků a vývojových prostředí. Volil jsem mezi programovacím jazykem Java a C#. Dalším z návrhů bylo vytvoření webové aplikace s použitím AJAX. Po porovnání možností jazyka s hlavními cíli, které jsem si stanovil a po ohlédnutí mých zkušeností s jednotlivými jazyky jsem dospěl k názoru, že nejvýhodnějším řešením budu použití .NET jazyka C#. Po zvolení vhodného programovacího jazyka, zbývalo vybrat vývojové prostředí. Zvolil jsem nejpoužívanější - MS Visual studio 2012.

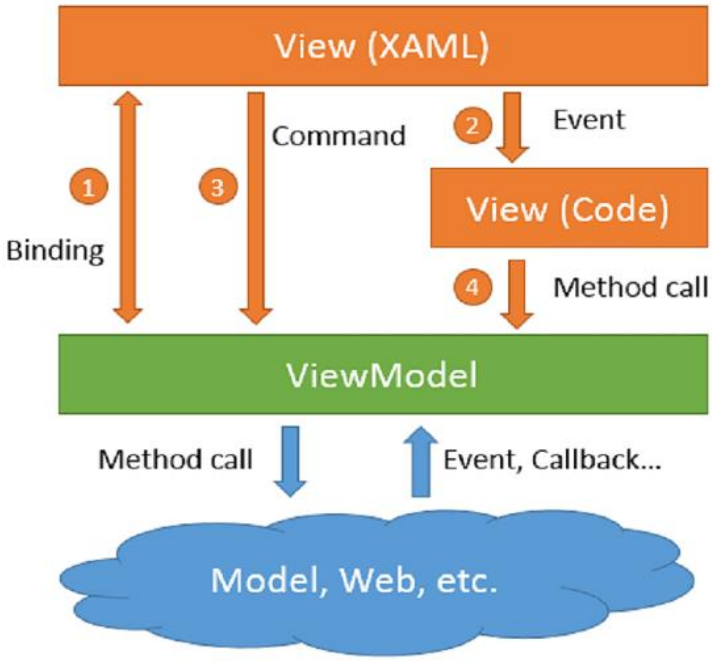
4.2 MVVM model

MVVM [3] model je návrhový model pro WPF aplikace. Odděluje logiku aplikace od uživatelského rozhraní. Použitím tohoto modelu se kód stává přehlednějším a flexibilnějším pro pozdější rozšíření a úpravy. WPF bylo navrženo tak, aby na něm návrhový vzor MVVM využíval pohodlně. Využívá pro to *binding* a *command* – náhrada za uživatelské rozhraní řízené událostmi. K oddělení jednotlivých vrstev se vyžívá několik prvků.

1. Model - popisuje data, se kterými aplikace pracuje
   - nic neví o stavu ovládacích prvků
2. View - uživatelské rozhraní v XAML(aplikace, stránka, ovládací prvek)
3. ViewModel - spojuje Model s View
   -drží stav aplikace (využívání data *Binding* a *Command*)

Nejdůležitějším prvkem je ViewModel, která drží stav aplikace. Poskytuje data pro uživatelské prostředí. Výhodou je, že data poskytuje v takových datových strukturách, kte-

ré vyvolávají události při jejich změně. [4] Tímto je umožněno okamžité zobrazení dat při jejich změně. K tomuto využívá dva základní prvky. Prvním je kolekce *ObservableCollection<T>*, která hlásí, když je přidán nebo odstraněn její prvek. Druhým je interface *INotifyPropertyChanged.* Ta popisuje události, které nastanou, při změně některých vlastností ViewModelu.



Obrázek 4.1 - MVVM model zdroj: [4]

4.3 WPF

WPF [5] (Windows presentation framework) je framework pro tvorbu formulářových aplikací. Součástí .NET od verze 3.0. Obsahuje širokou paletu formulářových prvků a spoustu možností jejich úprav. Technologie WPF je nástupce starší WF (*Windows forms*). Aktuálně se aplikace mohou tvořit v obou systémech, ale pro tvorbu nových aplikací se nic jiného než WPF prakticky nevyplatí používat. To je zejména proto, že *Windows Forms* trpí celou řadou problému. Jedním z hlavních je nízká podpora DPI (počet pixelů na palec). Je to zkratka využívaná u zobrazovacích zařízení a poskytuje informaci o počtu bodů na palec. Konkrétní problém u WF je, že pokud navrhneme aplikace pro určité rozlišení obrazovky, to znamená pro určité DPI, tak při pokusu o zobrazení na zařízení s jiným se program nevykreslí tak, jak bychom očekávali. Může se například stát, že ovládací prvky budou příliš malé, nebo se naopak nebudou vejít do určeného prostoru a budou přetékat do jiného. S tímto problémem souvisí také to že, WF umísťuje prvky absolutně. Tímto vzniká chyba, kterou jsem popisoval výše. Další

nevýhodou je použití původního vykreslovacího rozhraní GDI. Problémem tohoto rozhraní je, že je pomalé a obsahuje mnoho omezení. Naproti tomu WPF využívá Direct3D. To je interakce pro akcelerovanou grafiku. Výhodou tohoto rozhraní je, že aplikace, které ho využívají, jsou svižnější a méně zatěžují procesor. Navíc díky odpoutání od Windows GDI (grafické rozhraní) je umožněno tvořit grafiky mnohem bohatší aplikace. Například úpravou stávajících či vytváření zcela nových formulářových prvků.

Pro popis grafických prvků se využívá značkovací jazyk XAML (obdoba HTML). Vychází z jazyka XML. Vyžívá se zejména v aplikacích společnosti Microsoft nové generace. Využívá od .NET framewoku 3.0 zejména v technologiích WPF , Workflow foundation a Silverlight.

Výhodou tohoto značkovacího jazyka je jeho přehlednost a vysoká flexibilnost. Ve zdrojovém kódu je vše odděleno a řazeno skupin a podskupin. Nástroje jako Visual studio nabízejí také grafický editor této technologie. Tento editor funguje stejně jako v případě Windows Forms. Přesunem prvků do pracovní plochy vzniká výsledný vzhled aplikace. Výhodou je zvýšení jednoduchosti, ale na úkor omezení možností úprav. Pro větší změny prvků je vždy potřeba zásah do zdrojového kódu XAML. Proto grafický editor Visual studia nabízí například rozdělení na dvě okna. v jednom je finální vzhled a ve druhém zdrojová kód. Změny, které provedeme v XAML kódu se okamžitě promítají do grafické podoby.

```
<DataTemplate DataType="{x:Type local:HighBlockView}">
   <Grid>
     <Image Width="100" Source="/MyApp;component/Images/high.png"
       HorizontalAlignment="Center"
       VerticalAlignment="Center" IsHitTestVisible="False"/>
     <Button HorizontalAlignment="Center"
             VerticalAlignment="Center" Margin="3"
             Template="{StaticResource infoButtonTemplate}"
             Command="{Binding ShowDataChangeWindowCommand}" />
   </Grid>
</DataTemplate>
```

Zdrojový kód 4.1 - Ukázka XAML značkovacího jazyka

4.4 Návrh aplikace

Při návrhu jsem si aplikaci rozdělil do několika celků

- Vytvoření GUI
- Vnitřní struktura
- Komunikace za zařízením, přenesení programu

Grafické prostředí by mělo být jednoduché, proto je důležité se na něj zaměřit již v počátcích aplikace. Základní grafický návrh aplikace bude rozvržen do několika bloků. Největší část bude zabírat editor, kde se bude vytvářet výsledný diagram. Dalším blokem bude panel s funkčními bloky, které přesunem bude možné vložit do editoru. Dále horní části budou ovládací tlačítka (Nový, Odeslání do zařízení, Validace). Program by měl také umožňovat zobrazení textové podoby vygenerovaného zdrojového kódu.

Dalším krokem je návrh, jak budu pracovat s vlastními prvky v editoru. To znamená jejich přesun, propojení a nastavení jejich vlastností. K propojení funkčních bloků dojde spojením čarami mezi nimi. Dále pro změnu jejich nastavení se otevře vlastní okno, kde se jejich vlastnosti mohu měnit. Například port, dálka pauzy nebo počet kroků for cyklu.

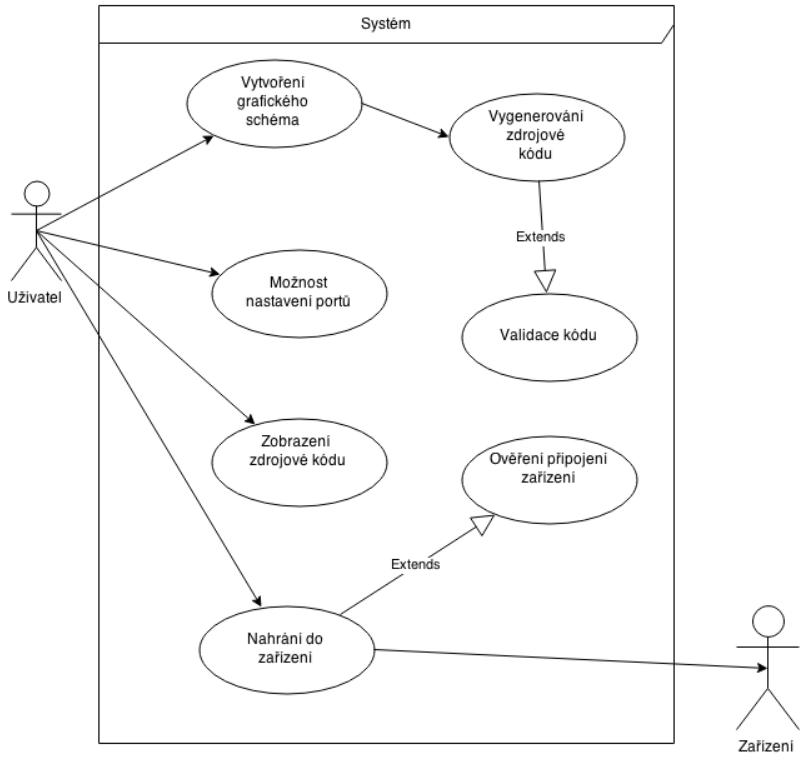
Funkční část programu

Editor. Musí obsahovat seznam prvků, které se aktuálně nacházejí v editoru. Dále bude umožňovat přidávání, výběr prvků a mazání prvků. Důležitá bude samotná práce s prvky například změna jejich pozice při přenosu myší.

Blok: Každý blok bude identifikován podle ID a bude mít nadřazený prvek, kterým bude třída editor. Další požadovanou vlastností bude pozice prvku v editoru a jeho velikost. Dále již každý blok bude mít specifikované vlastnosti podle svého typu. Tyto vlastnosti bude třeba seskupit a zobrazit v dialogovém okně, kde bude pohodlně nastavit jejich hodnota.

Dále bude třeba bloky seskupit do panelu a dále z něho mít možnost z něj přesouvat do editoru. V samotném editoru bude třeba zobrazit cesty mezi připojenými prvky a prvky, které nebudou připojené nezahrnout do generovaného zdrojového kódu.

Komunikace se zařízením budu probíhat pomocí sériového rozhraní. Tudíž v programu budu třeba zjišťovat zařízení na něj připojené. A po té dát uživateli možnost vybrat, jaký port se zařízením vyžaduje. Například v případě, že by k počítači bylo připojeno více zařízení. Pro validaci a naprogramování budu používat kompilátor Picaxe, který lze stáhnout webových stránek. Kompilátor bude také součástí následného instalačního balíčku softwaru.



Obrázek 4.2 - Use Case diagram

5 Realizace navrhnutého řešení

Z důvodu nízké zkušenosti v oblasti návrhu řešení aplikace a použití MVVM modelu se výsledná struktura programu značně liší od prvotního návrhu. Zdrojový kód aplikace se zároveň velice rozšířil. Ve výsledné verzi program obsahuje přibližně 2000 řádků funkčního kódu a podobný počet pro grafické rozhraní.

5.1 Editor

Oblast editoru pro práci s prvky je definovaná pomocí objektu *Canvas* [6]. Umožňuje manipulovat s prvky v ní pomocí souřadnicového systému. To znamená, že prvky v ní jsou relativní vůči její pozici. View model, který této třídě předává data, je třída *DiagramViewModel*. Dědí od *INCPBase* implementuje rozhraní *IDiagramViewModel*.

V tomto rozhraní vytváříme několik metod. Objekty třídy *SimpleCommand*, které budou obstarávat vkládání a odebírání položek z editoru. Dále obsahuje dva seznamy. První je obyčejný list, ve kterém jsou uchovávány aktuálně vybrané prvky. Druhým je objekt *ObservableCollection*. Tento objekt obsahuje seznam všech prvků v editoru. Tento typ objektu distribuuje položky do *View* a zároveň reaguje na změny provedené v editoru. Například smazání položky v editoru se okamžitě promítne to tohoto objektu a danou položku odstraní. Použití tohoto objektu souvisí s MVVM modelem, který v aplikaci používám.

Abstraktní třída *INCPBase* souvisí také se správnou funkci MVVM modelu. Implementuje rozhraní *INotifyPropertyChanged*. Metody, které implementuje, zajišťují správnou funkčnost provázání uživatelského prostředí a dat.

Veškerá data jsou vykreslovány v prvku *DiagramControl*. V něm je vytvořen element *UserConrol*. Používáme jej, pokud chceme vytvořit vlastní ovládací prvek. Toto řešení jsem použil, abych co nejvíce zpřehlednil kód aplikace a oddělil jednotlivé prvky. Pro následné použití prvku stačí vložit tento zdrojový kód (Zdrojový kód 5.1) do výsledného formuláře, který se bude spouštět.

```
<s:DiagramControl Grid.Column="1"
DataContext="{Binding DiagramViewModel}"
Margin="3,1,0,0" />
```

Zdrojový kód 5.1 - Vložení editoru do hlavního programu

5.2 Funkční blok

Z důvodu, že je třeba pracovat bloky bez závislosti na jejich typu, je definována v balíčku *LibraryEditor* abstraktní třída *ItemEditorViewModelBase*. Implementuje rozhraní *IItemEditor*, ve kterém je metoda zajišťující možnost vybrání prvku v editoru. Třída má dva atributy ID a *parent*. Druhý parametr představuje editor, ve kterém se aktuálně pracuje. Tato třída přestavuje základní prvek bloku. A ve stejném balíčku dále definuji abstraktní třídu *DesignerItemInEditorViewModelBase*. Dědí od *ItemEditorViewModelBasePanel*. V této třídě jsou přidány některé více specifické vlastnosti bloků.

- Pozice a velikost v editoru
- Zobrazení připojení – pro zobrazení spojení bloků v editoru
- Připojení – vlastnost zda blok je připojen k ostatním blokům
- Seznam připojených cest

Nakonec se vytváří finální funkční bloky. Například high, low, goto atd. Díky dědičnosti můžu s bloky pracovat společně. Jednotlivým blokům můžeme nastavovat různé parametry. Například návěstí, proměnnou, číslo pinu atd. Tyto hodnoty jsou nadále třeba nastavit. To se provádí otevřením dialogového okna, které zobrazuje aktuálně nastavené hodnoty. Specifikace hodnot, které mohou být nastaven, jsou definovány ve třídě Model každého bloku. Poté v ViewModel stejného bloku se vytváří jeho inscenaci a předává *IUIVisualizerService*. K jejich změně stačí hodnoty přepsat na nové a potvrdit tlačítkem OK. Tím dojde k distribuci změn do *ViewModelu*.

5.3 Panel funkčních nástrojů

Dalším důležitým prvkem v programu je panel funkčních bloků, které mohou být použity v programu. Tento panel se také vytváří jako samostatný *userControl*. Pro prvky v tomto panelu je použit blok *ItemControl*. A v tom se dále vytváří rozvržení prvků a vlastností, které se budou zobrazovat. Důležitou vlastností bylo nastavení možnosti pro Drag and drop operace. To znamená pro přesun pomocí myši. Pro jeho povolení je třeba nastavit tuto vlastnost.

```
<Setter Property="s:DragAndDropProps.EnabledForDrag"
        Value="True" />
```

Zdrojový kód 5.2 - XAML vlastnost Drag and Drop

Bez této vlastnosti by na položkách v seznamu vlastnost Drag and drop nefungovala. K celkové funkčnosti bylo samozřejmě potřeba ošetření několika událostí. A to kliknutí na daný prvek, tažení myší a puštění tlačítka myši na určitě pozici v editoru. Při těchto událostí vytvářím objekt *DragObject*, který obsahuje typ a velikost. Poté při tažení myší vytvářím nový objekt v editoru podle typy na, který bylo kliknuto. Velikost je nastavena kvůli případu, kdy by blok neobsahoval žádné pozadí. Tímto se mu nastaví defaultní hodnota.

Třídou, která drží data a předává panelu je *ToolBoxViewModel*. Obsahuje seznam všech funkčních blokům, které aplikace využívá. Například main nebo goto.

Vkládání bloků do panelu se provádí ve třídě *ToolBoxViewModel*. Pro přidání vytvořeného bloku se vloží do seznamu *ItemToolBox*. Typ vkládaných prvků jsou *ToolBoxData*. Tato třída má dva parametry. Prvním je cestu k obrázku, který se bude zobrazovat v panelu a druhým jeho typ.

```
public ToolBoxViewModel()
{
   ItemInToolBox.Add(new ToolBoxData("../Images/main.png", typeof(MainBlockView)));
   ItemInToolBox.Add(new ToolBoxData("../Images/pause.png", typeof(PauseBlockView)));
}
```

Zdrojový kód 5.3 - Přidání prvků do panelu nástrojů

5.4 Spojení grafických prvků v editoru

K propojení bloků slouží několik prvků. Hlavním z nich je ConnectorViewModel. Tato třída uchovává zdrojový a k němu připojený blok. Dále uchovává body, ve kterých byly spojeny, a podle nich vyhledává cestu mezi nimi. Cesta je uložena jako pole bodů. Vyhledání cesty a její aktualizaci při změně souřadnic zajišťuje třída *PathFinder*. Obsahuje dvě hlavní metody *GetConnectionLine*, která vrací seznam bodů, a *OptimizeLinePoints*, která upraví cestu při změně pozice bloku a vrátí nový list. Výsledné grafické propojení promítne do uživatelského prostředí třída *ConnectorViewModel*.

5.5 Editace zdrojového kódu

V aplikaci byl také požadavek pro možnost textové editace výsledného zdrojového kódu. Například pro některé úpravy a kontrolu. Dále by aplikace měla být schopna tento textový editor skrýt. Pro toto okno jsem použil textový box.

```
<TextBox
        Width="307"
        HorizontalAlignment="Right"
        VerticalAlignment="Bottom"
        Margin="0,0,25,25" Height="264"
        Text="{Binding ViewCode}"
        Grid.Column="1"
        Visibility= "{Binding Path=VisibleCode, Converter = {x:
        Static  s:BoolToVisibilityConverter.Instance}}"
/>
```

Zdrojový kód 5.4 - Zdrojový kód TextBoxu pro zobrazení zdrojové kódu

Pro skrytí a zobrazení jsem použil tlačítko, které mění booleovskou vlastnost *visibleCode* ve třídě *MainWindowViewModel*. Tuto vlastnost je následně pomocí data bindingu poskytnuta pohledu. Pro použití této vlastnosti je třeba ji nejdříve přetypovat na typ, který používá jazyk XAML. Toto přetypování zajišťuje třída *BoolVisibilityConverter*. Použití můžete vidět na obrázku (Zdrojový kód 5.4). Podobné třídy používám také pro přetypování souborové cesty k obrázkům (*ImgaeUrlConvertor*).

5.6 Přesun do zařízení

Připojení zařízení k počítači se realizuje pomocí sériové linky. Pro potřebu naprogramování procesoru je třeba kompilátor. Tento kompilátor je možné stáhnout ze stránek Picaxe. Každý typ mikroprocesoru vyžaduje jinou verzi. Kompilátor je spouštěcí soubor, kterému se předávají parametry. Může přebírat více parametrů, ale nejdůležitější je cesta k textovému souboru, kde je zdrojový kód a dalším neméně důležitým je číslo portu, na kterém je zařízení připojeno.

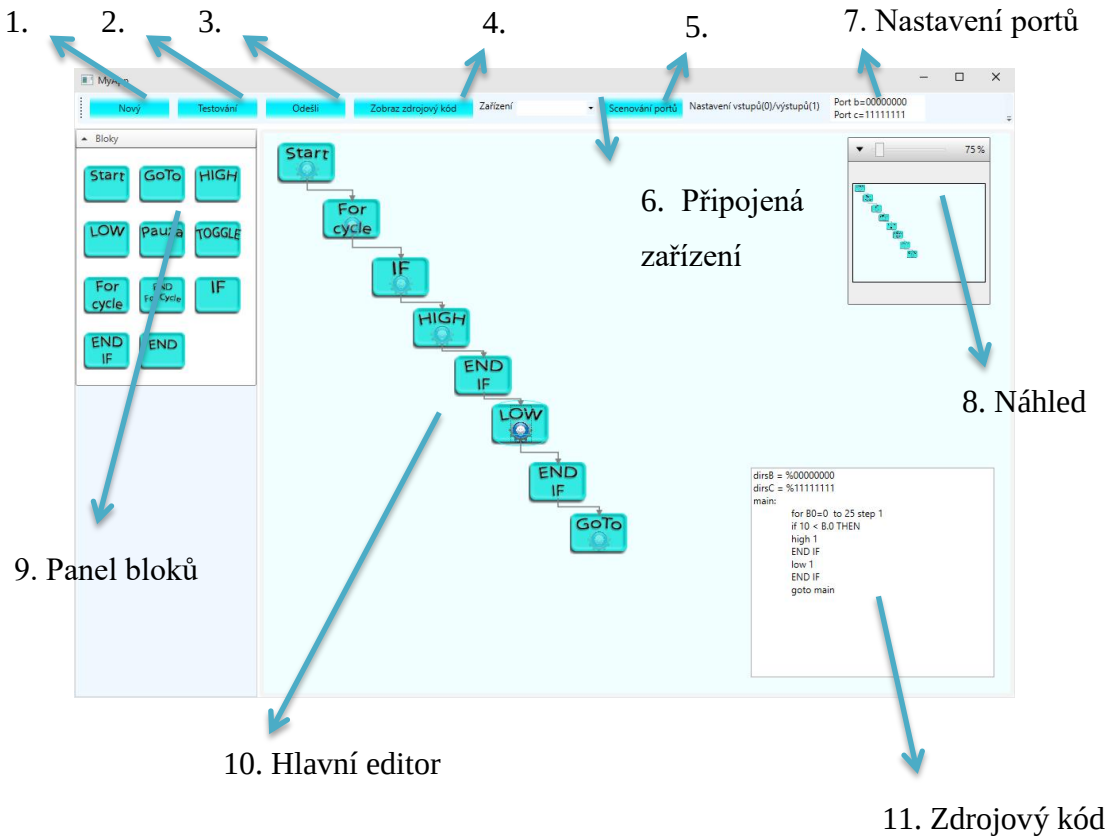
```
1. picaxe20m2.exe -c1 text.bas
2. pause
```

Zdrojový kód 5.5 – Skript pro spuštění kompilátoru

V aplikaci dojde ke spuštění kompilátoru pomocí skriptu, který se vygeneruje při stisku tlačítka „odešli“. Zdrojový kód generuji v metodě *generateSourceCode.*V okamžiku stisku se námi vytvořený program uloží do textového souboru a následně vygeneruje skript s nastavenými parametry. Tento vygenerovaný skript můžeme vidět obrázku Zdrojový kód 5.5. Kompilátor pracuje v režimu příkazového řádku. Při spuštění provede

validaci zdrojového kódu a poté se ověří, zda na námi zadaném portu je připojeno zařízení. Pokud jsou obě tyto podmínky splněny, dojde k nahrání programu do procesoru.

5.7 Grafická podoba aplikace

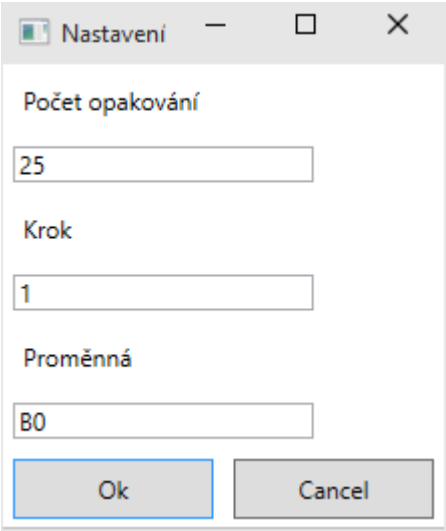


Obrázek 5.1 - Grafické rozhraní

1. Tlačítko pro vygenerování nového editoru.
2. Tlačítko pro vygenerování a aktualizace (v případě změny) zdrojového kódu z vytvořeného diagramu.
3. Tlačítko pro naprogramování zařízení. Stiskem dojde ke spuštění kompilátoru.
4. Tlačítko pro zobrazení a zneviditelnění textové podoby zdrojového kódu.
5. Tlačítko pro aktualizaci aktivních sériových portů v počítači
6. Seznam všech aktivních zařízení připojených k PC pomocí sériového portu
7. Nastavení vstupů a výstupů pinů na mikroprocesoru Picaxe
8. Zmenšený náhled editoru s možností jeho zvětšení a zmenšení měřítka. Změna se provádí bud posuvníkem nebo kolečkem na myši.
9. Panel všech funkčních bloků, které jsou v aplikaci k dispozici.

10. Hlavní část aplikace - editor. Zde se provádí manipulace s bloky a jejich propojování. Bloky jsou zobrazovány jako obrázky.
11. Náhled na textovou podobu vytvořeného grafického programu.

Dále některé funkční bloky obsahují další nastavení. Například for cyklu je třeba nastavit počet opakování, krok a proměnnou, která se bude inkrementovat. Toto nastavení otevřeme kliknutím na poloprůhledný obrázek umístění na každý prvku. Otevře se dialogové okno, kde můžeme měnit jednotlivé parametry.



Obrázek 5.2 - Dialogové okno nastavení bloku

6 Závěr

6.1 Zhodnocení výsledků práce vzhledem k zadání

Prvním bodem mého zadání byla analýza možností vizuálního programovacího jazyka mikrořadiče Picaxe. V první části se zaměřuji na seznámení s tímto mikroprocesorem. V další části se zaměřuji na konkrétní aplikace, které řeší danou problematiku. Hodnotím čtyři programy Rogic, 12Block, Scratch a Logicator. Programy hodnotím z pohledu funkčnosti a jednoduchosti ovládání.

Dalším bodem bylo vytvořit grafické uživatelské prostředí s předdefinovanou sadou příkazů. Vzhledem k cílové skupině, kterou budou žáci základních a středních škol, jsem se snažil prostředí navrhnout s ohledem na jednoduché ovládání a intuitivní grafiku. Důležitým kritériem byla také široká možnost rozšíření programu. Aplikace je vytvořená v prostředí .NET. Bylo použito vývojové prostředí Visual studio. Pro strukturu aplikace jsem zvolil MVVM návrhový vzor.

Posledním bodem mého zadání byla možnost ve vytvořeném uživatelském prostředí vytvořit a editovat grafický návrh programu. Program má možnost ho zobrazit v textové podobě a následně naprogramovat mikroprocesor PICAXE připojený k počítači přes USB rozhraní. S touto částí zadání jsem musel počítat již v předchozím bodě. Aplikace je schopna vygenerovat zdrojový kód z grafického diagramu, připojit zařízení a nahrát zkompilovaný program.

Všechny body zadání se mi podařilo splnit v plném rozsahu. Výsledný program poskytuje jednoduché nástroje a prostředí pro vytvoření programu v grafické podobě a jeho naprogramování do mikroprocesoru.

6.2 Možnosti dalšího rozšíření

Výsledná aplikace má veliký potenciál k růstu. Například rozšiřováním funkcionality jako je přidávání složitějších funkčních bloků, možnost programování více typů mikroprocesorů a možnost ladění. Velkou možnost rozšíření vidím také při přesunu na další platformy. Nejvíce pak na dotykové přístroje, například na tablety a chytré telefony. Tyto rozšíření a vylepšení by byly předmětem diplomové práce, nebo dalších bakalářských prací.

6.3 Výsledné zhodnocení

Výsledná aplikace je funkční a vhodná pro základní výuku programování mikroprocesorů Picaxe. Program bude součástí výukového systému RoboKit a bude sloužit k prvotnímu seznámení uživatele s programováním mikroprocesorů Picaxe.

7 Použitá literatura

[1] Picaxe, „PICAXE Manual,“ 10 2013. [Online]. Available: http://www.picaxe.com/docs/picaxe_manual1.pdf. [Přístup získán 12 2014].

[2] Wikipedia, „Visual programming language,“ Wikipedia: the free encyclopedia, [Online]. Available: http://en.wikipedia.org/wiki/Visual_programming_language. [Přístup získán 09 2014].

[3] Microsoft, „THE MODEL-VIEW-VIEWMODEL (MVVM) DESIGN PATTERN FOR WPF,“ Microsoft, [Online]. Available: https://msdn.microsoft.com/en-us/magazine/dd419663.aspx. [Přístup získán 01 2015].

[4] Microsoft, „Messenger and View Services in MVVM,“ 03 2013. [Online]. Available: https://msdn.microsoft.com/en-us/magazine/jj694937.aspx. [Přístup získán 11 2014].

[5] Microsoft, „XAML Overview (WPF),“ Microsoft, [Online]. Available: https://msdn.microsoft.com/en-us/library/ms752059.aspx. [Přístup získán 01 2015].

[6] Microsoft, „C# Programmer's Reference,“ Microsoft, [Online]. Available: https://msdn.microsoft.com/en-us/library/618ayhy6%28v=vs.71%29.aspx. [Přístup získán 01 2015].

[7] R. Hackett, PICAXE Microcontroller Projects for the Evil Genius, 2011 , ISBN 978-0-07-170327-7.

A. Příloha: Obsah přiloženého CD

Součástí práce je přiložené CD, kde jsou k dispozici:

- Zdrojové kódy aplikace
  - ❖ Zdrojové kódy aplikace
- Spustitelný program
  - ❖ Složka Program - všechny soubory důležité pro chod aplikace
- Tato práce ve formátu PDF
  - ❖ Bakalarska_prace_2015_Tomas_Kadlecek.pdf

B. Příloha: Diagramy

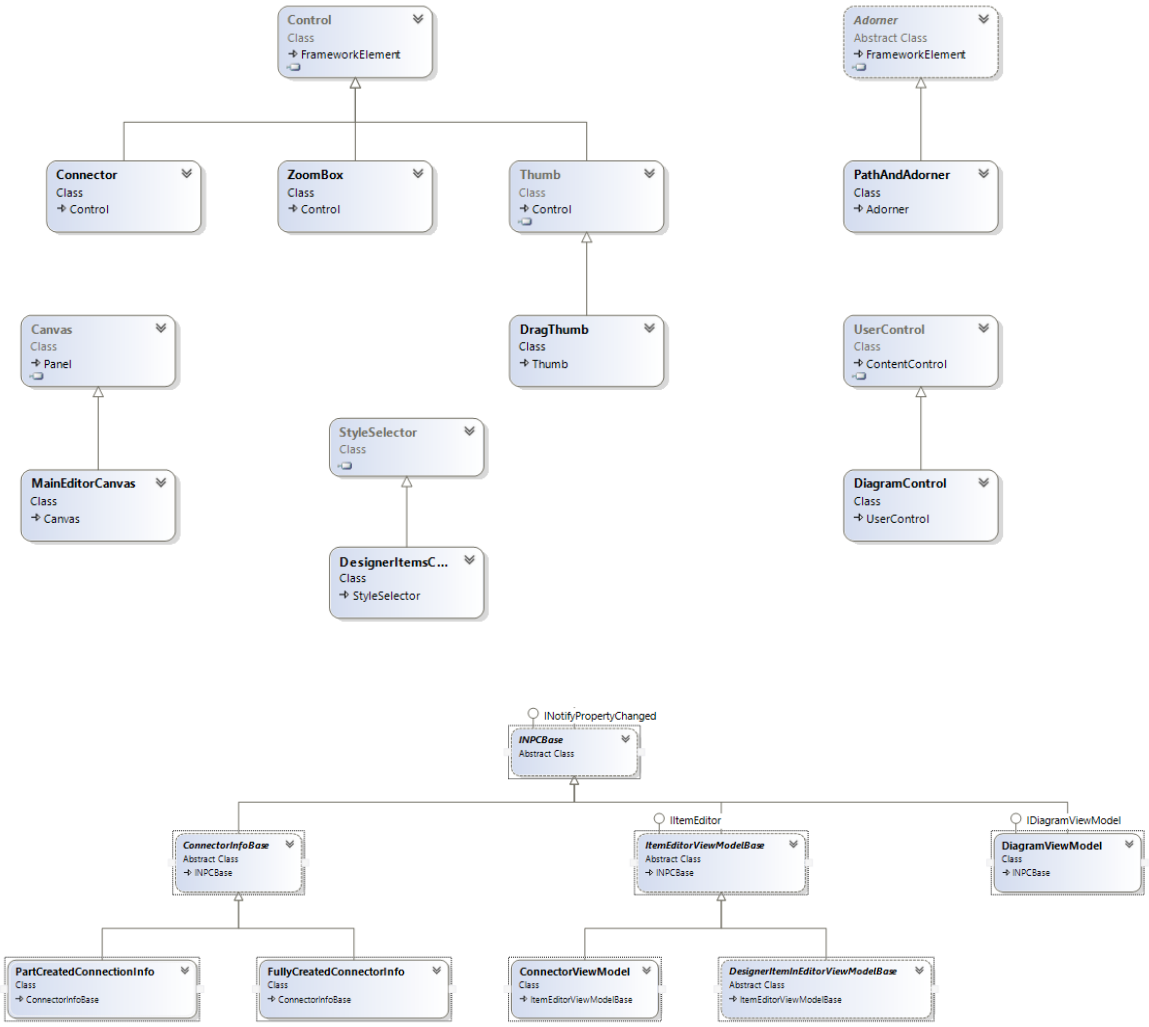


Diagram 2 - Seznam tříd: LibraryEditor

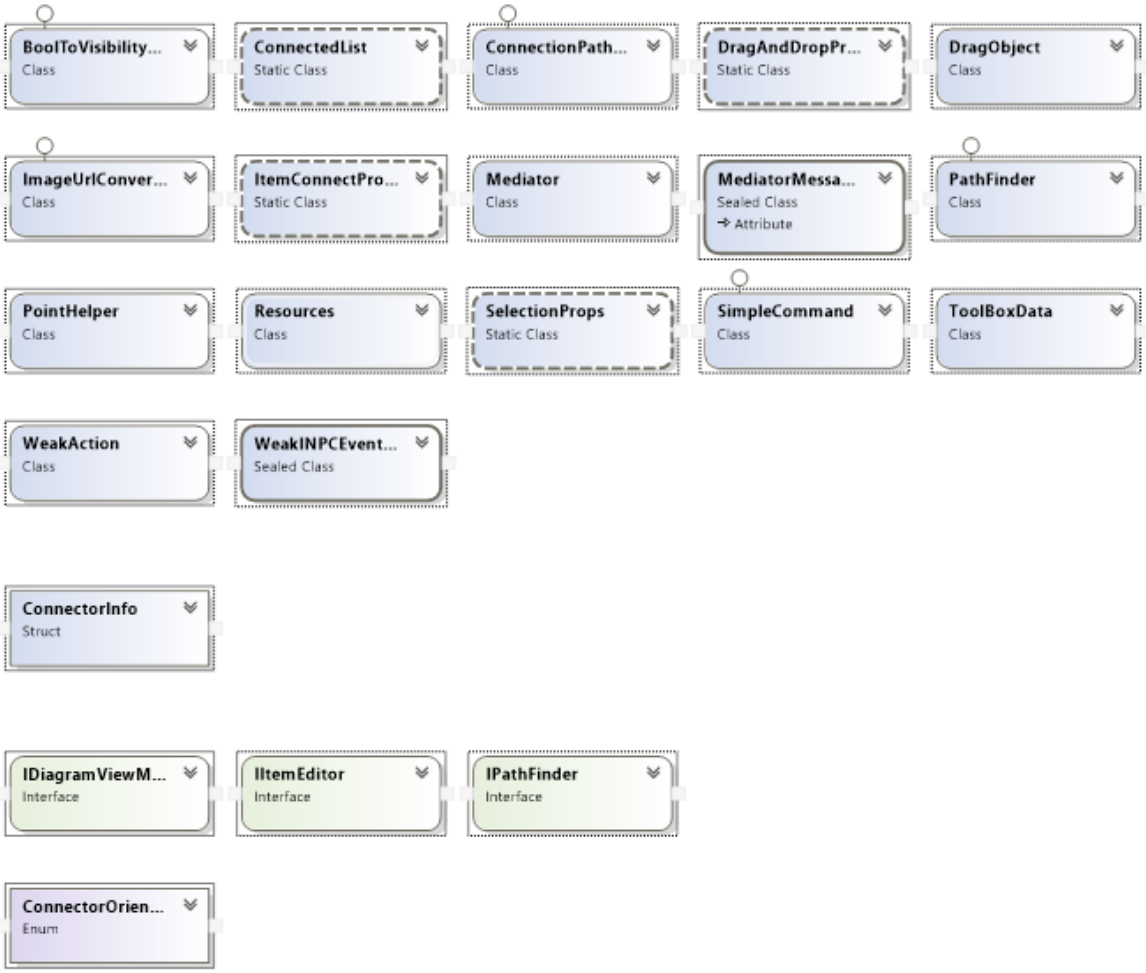


Diagram 3 - Seznam tříd: LibraryEditor

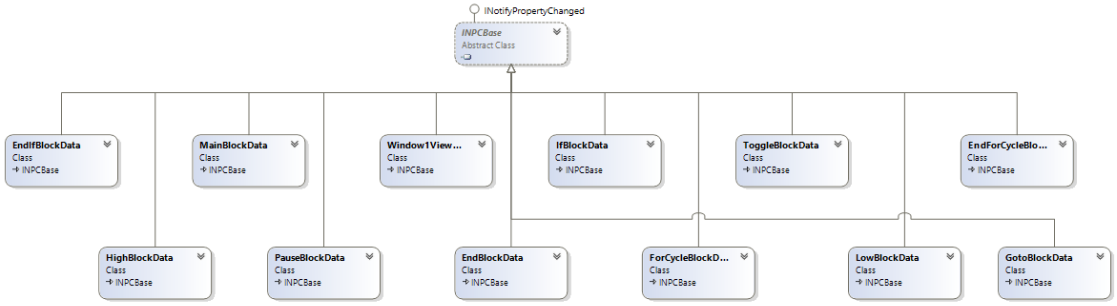


Diagram 4 - Seznam tříd: Hlavní aplikace

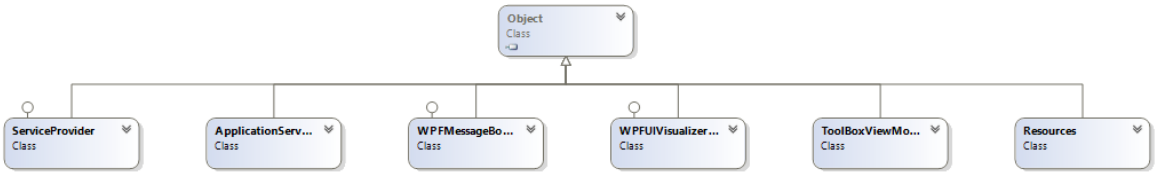


Diagram 5 - Seznam tříd: Hlavní aplikace

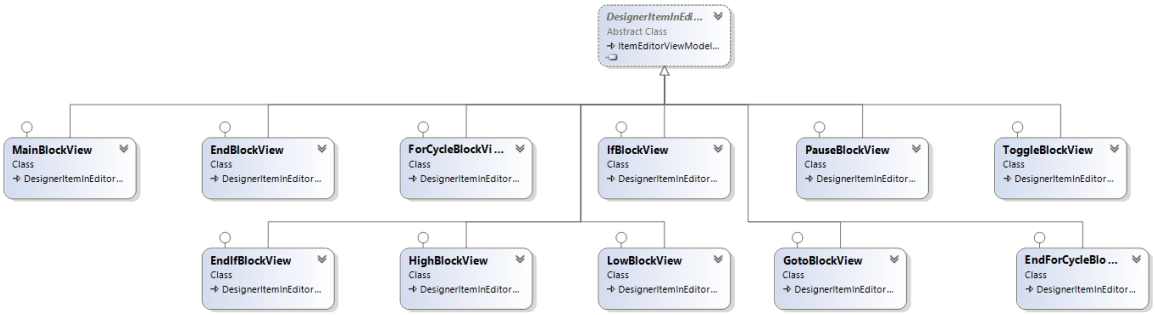


Diagram 6 - Seznam tříd: Hlavní aplikace

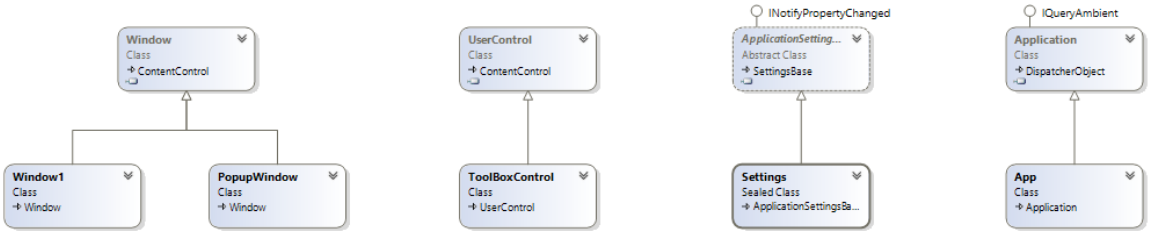


Diagram 7 - Seznam tříd: Hlavní aplikace

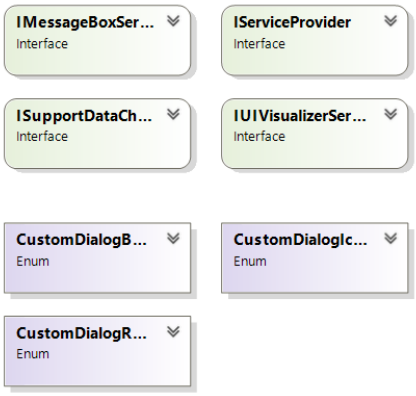


Diagram 8 - Seznam tříd: Hlavní aplikace