VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

WEBOVÉ ROZHRANÍ PRO VIZUÁLNÍ PROGRAMOVÁNÍ ROBOTA

WEB USER INTERFACE FOR VISUAL PROGRAMING OF ROBOTIC TASKS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **ŠTĚPÁN VÍCH**
AUTHOR

VEDOUCÍ PRÁCE **Ing. MICHAL KAPINUS**
SUPERVISOR

BRNO 2019

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2018/2019

Zadání bakalářské práce

Student: **Vích Štěpán**
Program: Informační technologie
Název: **Webové rozhraní pro vizuální programování robota**
Web User Interface for Visual Programing of Robotic Tasks
Kategorie: Uživatelská rozhraní

Zadání:

1. Prostudujte dostupné prostředky pro vizuální programování. Zaměřte se na nástroje určené pro vývoj robotických aplikací.
2. Seznamte se s frameworkem ROS a experimentální platformou ARTable vyvíjenou na naší fakultě.
3. Vyberte vhodné metody a nástroje a navrhněte uživatelské rozhraní, které umožní vizuální programování robotických úloh.
4. Navržené rozhraní implementujte a integrujte do robotického systému ARTable.
5. Proveďte experimenty, demonstrujte a diskutujte vlastnosti vašeho řešení.
6. Vytvořte stručný plakát nebo video prezentující vaši bakalářskou práci, její cíle a výsledky.

Literatura:

- Dle pokynů vedoucího.

Pro udělení zápočtu za první semestr je požadováno:

- Splnění prvních tří bodů zadání.

Podrobné závazné pokyny pro vypracování práce viz http://www.fit.vutbr.cz/info/szz/

Vedoucí práce: **Kapinus Michal, Ing.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2018
Datum odevzdání: 15. května 2019
Datum schválení: 1. listopadu 2018

Abstrakt

Tato bakalářská práce se zabývá vývojem, návrhem a testováním webového uživatelské rozhraní. Toto rozhraní vytváří robotické programy v jazyce Python pro robota PR2 a systém ARTable vyvíjený na VUT FIT v Brně. Rozhraní je navrženo tak, aby jej mohli používat i méně zkušení programátoři a lidé s netechnickým zaměřením. Při návrhu se používá technika vizuálního programování, při které vzniká výsledný kód. Při implementaci se využívá již vytvořené platformy podporující vizuální programování jako je např. Blockly, které celou implementaci značně zjednodušují.

Abstract

This bachelor thesis describes development, design and testing of the web user interface. This interface generates a robotic code for the PR2 robot and ARTable system, that is being developed on the BUT FIT. Interface is designed in the way that anyone could use it, for example less experienced programmer or even a person without technical experience. During the development stage, the visual programming is used to facilitate the program creation. In the implementation stage the visual programming platforms, such as Blockly, are used to facilitate the implementation.

Klíčová slova

web, vizuální programování, Blockly, ARTable, ROS, robot PR2, uživatelské rozhraní, HTML, PHP, JavaScript, jQuery, Python, rosBridge

Keywords

web, visual programming, Blockly, ARTable, ROS, robot PR2, user interface, HTML, PHP, JavaScript, jQuery, Python, rosBridge

Citace

VÍCH, Štěpán. *Webové rozhraní pro vizuální programování robota.* Brno, 2019. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Michal Kapinus

Webové rozhraní pro vizuální programování robota

Prohlášení

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

.....
Štěpán Vích
3. května 2019

Poděkování

Poděkovat bych chtěl hlavně svým rodičům, kteří mě podporují od začátku studia a vždy mi se vším pomáhali. Také velmi děkuji mému vedoucímu práce Ing. Michalu Kapinusovi, který mi pokaždé ochotně poradil, udělal si na mě čas, když jsem potřeboval a vše podrobně vysvětlil. Na závěr chci poděkovat svým blízkým přátelům a přítelkyni, že za mnou vždy stáli a emočně mě podporovali.

Obsah

Kapitola 1

Úvod

V současné době se stále více pracovních postupů robotizuje a nekvalifikovaní lidé jsou postupně nahrazováni stroji, které jim práci značně usnadní nebo je úplně nahradí. To se však již delší dobu neděje pouze u nekvalifikovaných pracovních pozic, ale stále častěji také u profesí kvalifikovaných. Ne nadarmo tak v novinových článcích často vídáme, že nám robotizace a umělá inteligence vezme veškerou práci. Je budoucnost opravdu tak černá? Ve výzkumné skupině robotiky Robo@FIT na VUT FIT v Brně se v to nevěří a raději se doufá, že automatizace odstraní nutnost výkonu banálních postupů, úkoly fyzicky náročné, přinese větší komfort pro celou společnost a umožní větší abstrakci problémů pro nejkvalifikovanější třídu pracovníků. Ti tak zvládnou více práce za kratší dobu. Podobným tématem se zabývá i tato práce.

Konkrétně se zaměřuje na automatické generování kostry robotických programů, pro robota PR2, který je dostupný na VUT FIT v Brně v učebně 0104. Kostra vzniká pomocí vizuálního programování ve webovém prohlížeči a je zajištěno jednoduché uživatelské rozhraní pro její generování. Poté je možné šablonu nahrát do systému ARTable, kde ji může uživatel dále parametrizovat a výsledné programy spouštět na robotovi. Ruční tvorba šablony v jazyce Python je poměrně náročný úkol a v současné době vyžaduje akademického pracovníka, který je obeznámen se systémem a má pokročilé schopnosti programování. Při použití našeho rozhraní bude moct sestavovat a nahrávat šablony do systému i nekvalifikovaný zaměstnanec a celý proces se tím výrazně urychlí. Akademického pracovníka již nebude potřeba a může se věnovat složitějším problémům.

V kapitole 2 je popsána veškerá teorie a prostudované materiály. Je nastíněn vývoj a současný stav webových technologií dnešní doby. Je definováno vizuální programování, naznačeno jeho další dělení a jsou nastíněny hlavní platformy podporující vizuální programování, např. Blockly a Scratch. V kapitole 3 je vysvětlen návrh webového rozhraní a jeho jednotlivé části. Jsou vybrány vhodné nástroje pro následnou implementaci a vylíčeny použité vizuální bloky a jejich kategorie. Také je definován způsob ukládáni vytvořeného programu do databáze systému ROS. V kapitole 4 je objasněna implementace a problémy, na které se při implementaci narazilo. V kapitole 5 je napsán závěr této práce. Jsou shrnuty dosažené cíle, prostudované reference a je navrženo možné další rozšíření bakalářské práce.

Kapitola 2

Studium problematiky

V této sekci se nachází teoretická část bakalářské práce. Jako první je popsána historie a v současnosti nejčastěji používané webové technologie a webová terminologie. Dále je definován vizuální programovací jazyk a platformy, které podporují vizuální programování na různých zařízeních. Poté je popsán systém ARTable, vyvíjený na VUT FIT v Brně a Robotický Operační systém ROS, který ARTable využívá. Poslední a důležitou částí je také popis systému ROS Bridge, který bude v této práci využíván.

2.1 Webové technologie

První veřejně dostupná webová stránka vznikla v roce 1991 [4]. Autorem byl sir Tim Berners-Lee a stránka využívala pouze technologii HTML, pro zobrazování obsahu. Od té doby se technologie výrazně vyvinuly a prostupují do každodenního života již téměř poloviny lidí na planetě [3]. Umožňují především proniknutí informací a znalostí i do méně rozvinutých zemí světa a nabízejí těmto lidem alespoň nějakou formu dostupného vzdělávání [6]. Vznikly technologie jako Flash, JavaScript, jQuery, PHP, MySql a další, které významně ulehčují a rozšiřují možnosti návrhu webových stránek, aplikací a rozhraní.

V současné době převládá při vývoji webových stránek rozdělení implementace na serverou a klientskou část [16]. Serverová část přijímá požadavky od klientské části a vrací odpovědi. Často komunikuje s databází a provádí složitější algoritmy, které není výhodné provádět na klientské části. Na klientské části se provádí vykreslování přijatého obsahu ve webovém prohlížeči. Na každé straně se využívají rozdílné technologie. Serverové a klientské části se také často říká back-end a front-end [23] a tyto pojmy se mnohdy zaměňují a kříží. Následuje výpis nejčastěji používaných klientských a serverových technologií.

Klientské technologie

HTML

Zkratka HTML značí technologii HyperText Markup Language. Jedná se o značkovací programovací jazyk. Dokumenty v něm napsané mají textový formát a jsou tvořeny z html tagů, které tvoří elementy. Rozlišujeme mezi startovním a koncovým tagem [14]. Tyto tagy definují sémantiku textu. Jedna z nejdůležitějších webových technologií, na kterou jsou vázané ostatní. Ze všech možných zde uvedených technologií je nejjednodušší. Společně s technologií CSS a JavaScript tvoří základní stavební kameny internetu [11]. HTML dokumenty jsou načítány ze serveru do klientského webového prohlížeče, kde jsou vykresleny.

CSS

Kaskádové styly jsou standardem konsorcia W3C pro definování vizuální prezentace webové stránky[14]. Využívají se v dokumentech HTML, kdy určují vizuální podobu stránky. HTML technologie určuje obsahovou formu stránky. Jeho hlavním cílem je separovat obsah od designu stránky. Každý CSS soubor se skládá ze selektorů HTML elementů, na které jsou aplikovány CSS pravidla. V moderním webovém návrhu se pracuje s preprocesory CSS, jako je LESS, SASS, Stylus a další [22]. Tyto preprocesory umožňují tvorbu proměnných, definování funkcí, používání matematických výrazů a další doplňky, které zjednodušují tvorbu a správu CSS souborů.

JavaScript

Pod zkratkou JavaScript můžeme nalézt interpretovaný programovací jazyk se základními objektově orientovanými schopnostmi [11]. Jedná se o skriptovací jazyk. Umožňuje použít dynamické prvky na webové stránce. Nejčastěji se jedná o přesuny elementů, změny atributů jednotlivých tagů, přístup k historii procházení či přesměrovávání na jiné stránky. Mezi pokročilé možnosti patří např. asynchronní získávání obsahu jiných stránek (Ajax) [12]. Jádro jazyka bylo poprvé vloženo do webového prohlížeče Netscape 2. V současné době je velmi populární a je pro něj napsáno spoustu knihoven usnadňující jeho používání. Existují také JavaScriptové transpilery, jako CoffeeScript, které se posledních 10 let hojně využívají při vývoji webových aplikací.

jQuery

Knihovnu pro jazyk JavaScript nazýváme JavaScript. Je postavena na principu zjednodušení kódu (write less, do more). Poskytuje množinu funkcí, které nejčastěji píšeme při vytváření dynamického obsahu stránek. Mezi hlavní funkce, které poskytuje, patří vyhledávání HTML elementů pomocí pokročilých selektorů, procházení a manipulování s HTML elementy, animace, zpracovávání událostí a zjednodušení technologie Ajax. U těchto funkcí zajišťuje kompatibilitu s téměř každým používaným prohlížečem [25]. Knihovna je používána na více jak polovině významných stránek [30].

Adobe Flash

Poslední klientskou technologií je platforma firmy Adobe - Adobe Flash. V tomto proprietárním softwaru firmy Adobe lze tvořit animace a interaktivní aplety, které lze poté vkládat do webových stránek. Takové soubory jsou zakončeny příponou swg. Pro spuštění souboru musí být v prohlížeči nainstalován doplněk Adobe Flash Player. V současné době se již na webových stránkách příliš nepoužívá a ustupuje do pozadí. U moderních prohlížečů je často ve výchozím nastavení blokován. Pomocí softwaru Adobe Animator lze však generovat i HTML5 a SVG animace, namísto klasických swg souborů. Tyto animace jsou v dnešní době více doporučované [29].

Serverové technologie

PHP

Jako první serverovou technologií bych uvedl PHP, skriptovací jazyk speciálně designovaný pro web. Do HTML stránky můžeme vkládat PHP kód, který se spustí pokaždé, když

stránku navštívíme. Mezi jeho přednosti patří jednoduchost, snadné napojení na databázové systémy, přenositelnost na různá zařízení a vysoká rychlost. Je napsán v jazyce C++. Licence je Open Source [18].

MySql

Další serverovou technologií je MySql. Jedná se o relační databázový systém, dostupný pod Open Source licencí. Umožňuje efektivně ukládat, číst, třídit a vyhledávat data a poskytuje rozhraní pro tvorbu databázového modelu. Také se využívá s PHP při vytváření webových technologiích. Dříve se místo databázových systémů používalo ukládání do textového souboru, které však nebylo efektivní [18].

MongoDb

Tento databázový systém[1] využívá, narozdíl od MySql NoSQL přístup. Při tomto přístupu se do databáze ukládají celé dokumenty, které jsou nejčastěji reprezentovány jako objekty. SQL dotazy zde neexistují. Pracuje se zde s indexy, které jsou podobné těm z ostatních databázových systémů. Data se získávají filtrováním a běžnými funkcemi daného jazyka, který komunikuje s databází.

Symfony

Poslední zde zmíněnou serverovou technologií je PHP knihovna Symfony určená pro tvorbu webových aplikací[2]. Do tvorby webové aplikace zavádí architekturu MVC, která zpřehledňuje následné rozšiřování celé aplikace. Zlehčuje také používání dalších technik jako je lokalizace, šablony pro formuláře nebo správu sessions. Mezi další podobné knihovny patří Laravel, Nette, Zend a další.

Terminologie

Často se setkáváme s pojmy webová aplikace, webová stránka a webové rozhraní. Rozdíl mezi nimi si nyní nadefinujeme:

- Webová stránka – statická stránka, která pouze poskytuje informace uživateli.
- Webové rozhraní – dynamická stránka, která poskytuje uživateli informace a uživatel s ní může alespoň trochu interagovat.
- Webová aplikace – dynamická stránka, která připomíná desktopový program a v něčem ho i předčí. Poskytuje uživateli maximální komfort a umožňuje ukládání informací.

2.2 Uživatelská rozhraní

V této části bude popsáno, co jsou uživatelská rozhraní a jak se vytvářejí. Uživatelské rozhraní je součást počítačového systému, se kterým uživatel interaguje tak, aby dosáhl svých cílů a splnil všechna zadání. Počítačový systém je kombinace hardwaru a softwaru, které přijímají vstup a dávají zpět výstup. Uživatelské rozhraní je součástí oboru Human-Computer

[1] https://www.mongodb.com/what-is-mongodb
[2] https://symfony.com/about

Interaction (HCI). Obor se zabývá interakcí člověka a počítače. Jedná se o velice široký obor, který se neustále rozšiřuje [8]. V současné době existuje mnoho typů uživatelských rozhraní. Mezi nejčastější patří tyto: [10]

- Příkazový řádek – Anglicky Command line interface (CLI). V dnešní době se u klasických uživatelů již nepoužívá. Je využíván programátory a administrátory pro snadné ovládání systému. Pro běžného uživatele je toto rozhraní příliš složité.
- Menu-Driven – Menu-driven rozhraní je speciální typ rozhraní. Uživateli nabídne řadu funkcí a možností nastavení, ze kterých si může vybrat. Je jednodušší než příkazový řádek. K z reprezentantů menu-driven rozhraní patří například bankovní automat.
- GUI – V dnešní době se nejčastěji používá. S rozhraním se interaguje pomocí myši nebo trackpadu klikáním na grafické prvky nebo ikony.
- Dotykové GUI – Velmi podobné GUI s tím rozdílem, že používáme dotykovou plochu a své prsty k interakci s rozhraním, namísto myši.

2.3 Vizuální programovací jazyky

Podle knižní definice je vizuální programovací jazyk jakýkoliv jazyk, který umožní programátorovi tvořit programy manipulováním grafických bloků spíše než textovým popisem [17]. Manipulování bloků probíhá povětšinou pomocí myší nebo touchpadu. Mnoho vizuálních programovacích jazyků používá techniku boxů a šipek, kde boxy reprezentují entity spojené šipkami, které reprezentují relace. Programátor tak sestavuje vývojový diagram programu, který se poté převede automaticky do textové podoby a buďto se může spustit, nebo jinak zpracovat, například robotem, který zadaný kód splní. Často lze zobrazit koncový kód v reálném čase při sestavování vývojového diagramu. Vizuální programovací jazyky mají velké uplatnění při výuce programování, nebo tam kde nechceme uživatele příliš zaměstnávat s psaním kódu.

Vizuální programovací jazyky mohou využívat ikony, formuláře a diagramy pro generování cílového kódu [1]. Při využívání ikon se jednotlivé ikony spojují do větších celků. Klasickým příkladem vizuálního programování s využitím ikon je systém AppWare. Tento systém využívá ikony k reprezentaci objektů a ke tvoření aplikací na systému Microsoft Windows a Mac OS.

Pokud si chceme vizuální programovací jazyk klasifikovat do stávajících kategorií programovacích jazyků, bude se zajisté jednat o jazyk vysokoúrovňový [5]. Tento jazyk dovoluje uživateli vysokou míru abstrakce. Z velké části se také jedná o jazyk výukový. Vizuálním programovacím jazykům se také někdy říká block-based jazyky. Ve volném překladu to znamená blokově orientované jazyky.

Vizuální programování pomáhá klasickým uživatelům a začátečníkům proniknout do programování a podporovat je ve třech základních bodech [24].

- Zjednodušení syntaxe – Vizuální programovací jazyky využívají bloky, ikony, formuláře, diagramy a další grafické prvky tak, aby snížili potenciální možnost syntaktické chyby při tvorbě programu.
- Zjednodušení sémantiky – Vizuální jazyky se snaží poukázat na principy fungování jednotlivých základních programových struktur a minimalizovat tak možnou sémantickou chybu, které se uživatel může dopustit. Příkladem může být ukázání nápovědy, pokud uživatel přejede myší přes některý grafický blok.

- Procvičování – Vizuální jazyky se snaží demonstrovat fungování systému na názorných příkladech, se kterými může uživatel manipulovat a rozvíjet je do složitějších celků.

2.4 Generátory vizuálních programovacích jazyků

V této části budou uvedeny příklady nejvýznamnějších generátorů vizuálních programovacích jazyků a jejich popis. Vizuální programovací jazyky samotné se využívají ve výuce, video hrách, simulacích, robotice, multimediálních programech a dalších odvětvích [9, 7].

Blockly

Jedná se o JavaScriptovou knihovnu, která umožňuje tvoření vizuálních programovacích jazyků a jejich editorů [13]. Autorem knihovny je společnost Google a vydává ji pod svobodnou open-source licencí Apache 2.0 od roku 20012. Hlavní cílovou platformou je webový prohlížeč. Proto je napsána v jazyce JavaScript. Nicméně existují porty pro operační systémy Android a iOs. Výstupem vizuálního programovacího jazyka je spustitelný kód v cílovém jazyce. Jazyky, které Blockly podporuje nativně jsou JavaScript, Python, PHP, Dart a Lua. Mohou být také vytvořeny vlastní generátory kódu pro libovolný programovací textový jazyk. To znamená, že můžeme vytvořit generátor např. pro programovací jazyk Java.

Pro nainstalování Blockly na web je nutné vložit do webové stránky několik skriptů a HTML elementů, do kterých se vykreslí Blockly editor a vše automaticky funguje. Oblasti, do které se vykresluje hlavní editor skládající se z Toolboxu a oblasti pro manipulaci s bloky se říká Workspace, česky pracovní plocha. Zjednodušeně řečeno se bloky z Toolboxu přetahují do pracovní plochy, kde se spojují a vytváří výsledný program. Výsledný program je vygenerován a může být vložen jako text do kteréhokoliv HTML elementu na webové stránce. To, co je obsahem Toolboxu (jaké bloky a kategorie bloků), je kompletně v režii tvůrce stránek. Na obrázku 2.1 je příklad použití platformy Blockly na hře Maze.

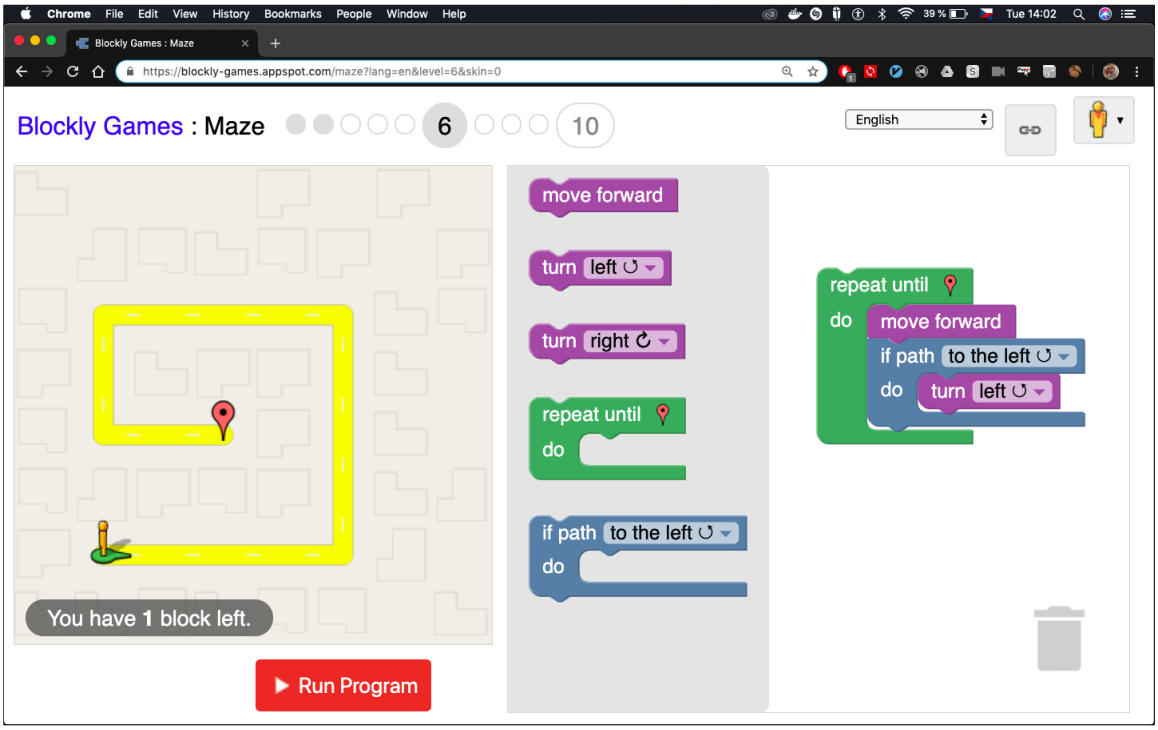
Nové bloky mohou být přidávány podle libosti. Pro přidání nového bloku je potřeba blok definovat a dále poskytnout generátor zdrojového kódu. Mezi další výhodu patří bezesporu to, že je platforma postavena na webových technologiích HTML a SVG, a ne na technologii Flash (oproti platformě Scratch). Vzhled samotných bloků a editoru je silně ovlivněn platformou Scratch, ze které vizuálně vychází.

Samotné Blockly se využívá v poměrně významných aplikacích a projektech jakými jsou například program AppInventor pro tvorbu aplikací na Android, program Dragme IDE pro tvorbu programů v jazyce C, projekt Code.org, který se využívá pro výuku programování či projekt RoboBlockly o kterém, se zmíníme dále a další.

Scratch Blocks

Jedná se o rozšíření platformy Blockly. Autorem je tým MIT Media Lab známý tvorbou platformy Scratch. Projekt je v rané fázi vývoje a dokumentace ještě není zcela dokončena. Projekt funguje tak, že rozšiřuje funkcionalitu Blockly přidáním vlastních bloků (Custom Blocks). Tyto bloky jsou podobné grafickým blokům platformy ScratchJr tj. důraz je kladen na horizontální uspořádání. Cílí na mladší publikum. Platforma může být spuštěna v jakémkoliv webovém prohlížeči a žádné další instalace nejsou nutné[3].

[3] https://github.com/LLK/scratch-blocks/blob/develop/README.md



Obrázek 2.1: Hra Maze s využitím platformy Blockly

Droplet

Droplet je systém velmi podobný platformě Blockly. Za jeho hlavní přednost považujeme to, že umožňuje převádět text do grafických bloků a zpět. Je tedy určen spíše pro pokročilejší uživatele. Jeho vývoj trvá od roku 2014 a je součástí platformy Pencil Code. Editor systému Droplet nepotřebuje žádné instalace a lze jej rozběhnout v libovolném webovém prohlížeči. Pro vizuální programování v tomto systému je nutné definovat vlastní bloky pro výstupní jazyk. Systém je volně dostupný pod licencí MIT. Dokumentace systému však není příliš propracovaná[4].

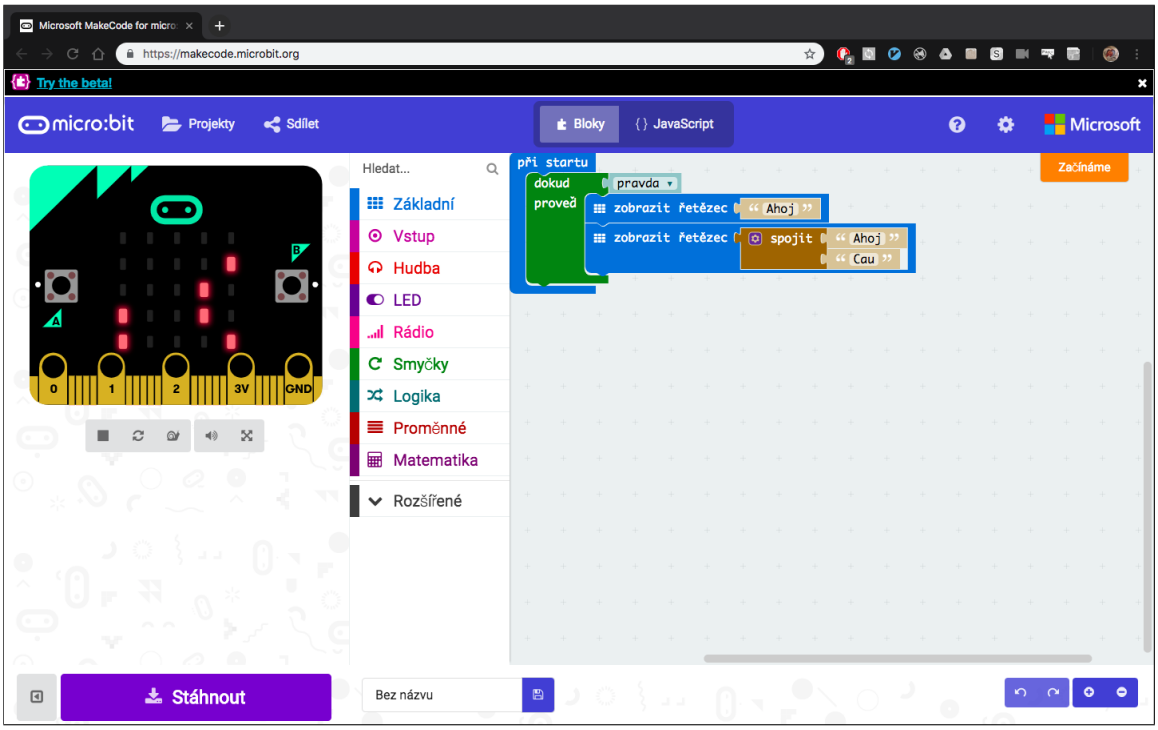
PXT

Platformu PXT lze také nalézt pod názvem Microsoft MakeCode. Jedná se o knihovnu pro tvorbu JavaScriptových programů pomocí vizuálního programování[5]. Je postavena nad platformou Blockly a webovým textovým editorem Monaco. Velkou nevýhodou je, že jediný výstupní jazyk je jazyk JavaScript. Licence platformy PXT je open source, nepotřebuje žádné dodatečné instalace a běží plně ve webovém prohlížeči. Jako příklad využití platformy je vybrán projekt Microsoft Microbit vyobrazený na obrázku 2.2 [21].

2.5 Platforma ARTable

Tato platforma je vyvíjena na VUT FIT pod výzkumnou skupinou RoboFit. Umožňuje interakci člověka s robotem a dotykovým stolem. Celý systém se skládá z několika pro-

[4] https://github.com/PencilCode/droplet/blob/master/README.md
[5] https://makecode.com/about



Obrázek 2.2: Projekt Microsoft Microbit

pojených zařízení, mezi které patří robot PR2, dotyková plocha, projektory, reproduktory, Wi-Fi router, snímač Kinect One, pracovní stůl a přenosné počítače. Tyto zařízení lze navzájem kombinovat a vytvářet sestavy. V současné době existují tři hlavní sestavy. První sestava využívá robota PR2 a dotykovou plochu, na kterou se promítají jednotlivé programy v dané úloze. Tato první sestava je velmi důležitá pro tuto bakalářskou práci a je ukázána na obrázku 2.3. Druhá sestava se používá na konferencích a veletrzích pro demonstrování platformy ARTable. Třetí sestava je určena pro interaktivní hry a programy, které nepotřebují robota PR2 [26]. Pro komunikaci mezi jednotlivými zařízeními se používá platforma ROS, která je více popsána v následující kapitole. Detailní popis jednotlivých sestav a dalších informací o platformě ARTable lze nalézt na stránkách projektu[6].

Platforma se může používat pro promítání a hraní her v rozšířené realitě, promítání aplikací do rozšířené reality, rozpoznávání objektů na dotykovém stole nebo simulování pracovního prostředí např. ve výrobní hale s roboty. Vzniklo již několik aplikací a projektů pro tuto platformu, např. bakalářská práce od Anety Helešicové [15].

Práce navazuje na dřívější výzkum skupiny Robo@FIT v oblasti programování robotických úloh [20]. V uvedené práci je popisováno sestavení pracovní sady ARTable s robotem PR2 a následné testování uživatelského rozhraní, které se promítá pomocí projektorů na dotykový stůl. Uživatelské rozhraní obsahuje kostry programů, které se musí parametrizovat. Parametrizování provede uživatel stojící u stolu. Toto je více popsáno v následujícím odstavci. V této bakalářské práci vytváříme webové uživatelské rozhraní, které umožní tvorbu těchto koster.

Při zapnutí dotykového stolu se zobrazí uživatelské rozhraní se seznamem nahraných programů. Program se skládá z několika bloků a každý blok může obsahovat několik in-

[6] https://github.com/robofit/artable13

Obrázek 2.3: Sestava ARTable s robotem PR2. Převzato z https://github.com/robofit/artable-setup-1

strukcí. Při úspěšném provedení instrukce se vrací hodnota 1 a při neúspěšném hodnota 0. Instrukce mohou požadovat parametry, které musí být nastaveny uživatelem. Takové instrukce se nazývají parametrické. Instrukce, které nepotřebují parametry se jmenují bezparametrové. Mezi základní parametrické instrukce se řadí instrukce `pick_from_polygon`, `pick_from_feeder`, `pick_from_pose` a `apply_glue`. Mezi základní bezparametrové instrukce se řadí `get_ready`, `wait_for_user`, `wait_until_user_finishes`. Seznam všech instrukcí je dostupný v souboru `instructions.yaml` v git adresáři projektu ARTable[7]. Každá sestava může mít svou vlastní instrukční sadu.

2.6 Platforma ROS

Platforma ROS je open-source, meta-operační systém pro robotické aplikace. Poskytuje služby, které se od operačního systému očekávají. Mezi ně patří abstrakce hardwaru, nízkoúrovňové ovládání hardwaru, podpora základních funkcí, posílání zpráv mezi procesy a organizování balíků. uPokud je systém postaven nad normálním operačním systémem, pokládáme ho za meta-operační. Mezi obdobné systémy patří např. Player, YARP, Orocos, Microsoft Robotics Studio a další. Zařízení, která využívají ROS, mezi sebou komunikují zprávami. Existuje několik druhů zpráv a možností komunikace mezi zařízeními. Během komunikace neexistuje centrální server, ale zařízení mezi sebou komunikují na principu peer-to-peer. Systém propojených a navzájem komunikujících zařízení vytváří síť, která je popsatelná grafem. Platformu ROS využívá i ARTable. Podporované operační systémy, na

[7] https://raw.githubusercontent.com/robofit/arcor/master/art_instructions/config/instructions.yaml

kterých ROS běží, jsou například Ubuntu, Mac OS X a ostatní Linux platformy [28]. Více informací o ROSu lze nalézt na wiki stránkách projektu[8].

Hlavním cílem ROSu je poskytnou systém, který je jednoduchý a znovupoužitelný na různých zařízeních a systémech. To vede k zavedení několika cílů [28]:

- Jednoduché rozhraní – aby se mohl ROS propojit s ostatními systémy jako je Player nebo YARP, musí být rozhraní co možná nejjednodušší.
- Nezávislost na jazyku – ROS lze implementovat v hlavních programovacích jazycích jako je C++, Python a Lisp. V budoucnu přibude podpora pro jazyky Java a skriptovací jazyk Lua.
- Jednoduché testování – ROS má zabudovanou testovací knihovnu pro jednotkové a integrační testování.
- Škálovatelnost – ROS je vhodný na velké i malé projekty a není složité rozšiřovat jeho funkcionalitu.

Za hlavní prvek ROSu, který je vhodné popsat, považujeme komunikační model. Procesy, které běží na jednotlivých zařízeních mezi sebou komunikují, a to pomocí několika způsobů. Mezi hlavní prvky komunikačního modelu patří Nodes, Messages, Topics a Services [27].

Nodes

Nodes překládáme do češtiny jako uzly. Jedná se o běžící procesy na jednotlivých zařízeních. Vykonávají výpočty a zajišťují funkčnost systému. Každý uzel zajišťuje jinou funkcionalitu. Například jeden se stará o pohyb robotické ruky, druhý se stará o pohyb hlavy robota a třetí o zvukový výstup do reproduktorů. Každý uzel se implementuje ve zvoleném jazyce (nejčastěji C++ nebo Python) s použitím knihovny roscpp (pro jazyk C++) anebo rospy (pro jazyk Python). Mezi uzly musí existovat alespoň jeden Master uzel. Master uzel se stará o směrování jednotlivých zpráv a zajišťuje správné fungování služeb. Umožňuje také ukládat data do slovníku. Master uzel se chová jako jmenný server DNS. Uchovává informace o jednotlivých tématech a službách. Jakmile se uzel chce přihlásit k odběru nějakého tématu poskytne mu Master uzel informace, k jakému uzlu se má přihlásit.

Messages

Messages, v českém překladu zprávy, může vysílat každý jednotlivý uzel do systému. Pomocí nich jednotlivé uzly komunikují. Zpráva je datová struktura s několika položkami. Podporovány jsou primitivní datové typy položek a pole těchto primitivních datových typů. Každá zpráva je poslána pod určitým tématem (Topic). Dalším způsobem, jak mezi sebou mohou uzly komunikovat je pomocí služeb (Services). Ty jsou popsány níže.

Topics

Topics, v českém překladu témata. Každá zpráva je poslána pod určitým tématem. Uzly mohou naslouchat na tomto tématu a přijímat jednotlivé zprávy. Odpovídat na ně nelze. Na každé téma se může přihlásit libovolný počet uzlů a přijímat zprávy, které se na něm

[8] http://wiki.ros.org/

objeví. Je zde podobnost s předplatným časopisu. Každý uzel se může přihlásit k odběru libovolného počtu témat a sám může generovat libovolný počet zpráv k daným tématům. Architektura je nastavena tak, že jednotlivé uzly neví o odběratelích a naopak.

Services

Tento komunikační prvek překládáme jako Služby a patří mezi další z možných způsobů komunikace mezi uzly. Systém odesílání zpráv pod určitým tématem měl nevýhodu v tom, že na zprávu nešlo odpovědět. Každý uzel tedy může nabízet určitou službu pod zadaným jménem. Proces odešle dotaz na zadanou službu a čeká na odpověď.

Kapitola 3

Návrh

Tato část popisuje existující řešení, návrh webového rozhraní a jednotlivých grafických bloků, které budou používány při vizuálním programování. Každý grafický blok bude uložen v jedné z tématických kategorií. Tento návrh pak bude implementován a otestován.

3.1 Existující řešení

V této části se nachází výpis již existujících vizuálních jazyků a jejich aplikací, především z oblasti robotiky.

Open Roberta

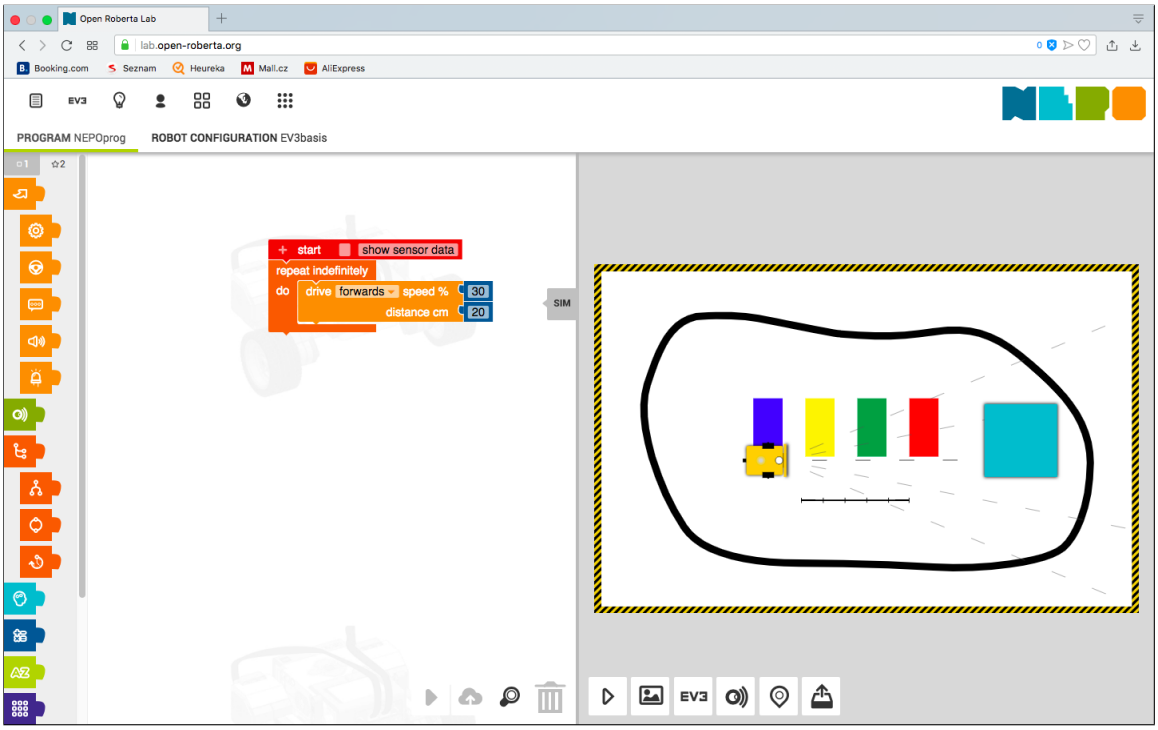
Open Roberta je německý akademický projekt, který učí děti a studenty programovat robotické aplikace. Licencování je volné a je založeno na licenci Apache 2. Hlavním sponzorem projektu je společnost Google. Pro správné fungování nevyžaduje žádnou instalaci. Hlavní částí projektu je Open Roberta Lab, která umožňuje vizuální programování robotických aplikací pro nejrůznější hardwarové systémy jako je LEGO Mindstorms, Calliope mini, NAO a další. Projekt vzniká od roku 2013 a je dále vyvíjen.

Open Roberta Lab je webová aplikace, veřejně dostupná na webové adrese[1]. Na obrázku 3.1 vidíme její screenshot. Jedná se o mezičlánek mezi softwarovými a firmwarovými prvky celé architektury Open Roberta. Vizuální programovací jazyk, který se používá v grafickém editoru webové aplikace se nazývá NEPO. Graficky je editor inspirován platformou Scratch, je však napsán v technologii HTML5 a JavaScript, a to pomocí platformy Blockly. Vizuální programovací jazyk NEPO dělí grafické bloky do několika tématických kategorií.

Jazyk NEPO má celkem 11 základních tématických kategorií s grafickými bloky [33]. Základní kategorie jsou:

- *Action* (Akce) – bloky pro ovládání robota, jeho pohyb, zapínání a vypínání světel, otáčení, zastavování, nastavování hlasitosti a další.
- *Sensors* (Senzory) – bloky pro zjišťování stavu a resetování jednotlivých senzorů robota. Například získání údajů z gyroskopu.
- *Control* (Kontrola) – bloky popisující řídící struktury programu jako if, if–else, while.
- *Logic* (Logika) – bloky pro tvorbu logických výrazů.

[1] https://lab.open-roberta.org/



Obrázek 3.1: Webová aplikace Open Roberta Lab

- *Math* (Matematika) – bloky matematických operátorů a funkcí jako je plus, mínus, krát, děleno, testy sudosti a lichosti a další.
- *Text* – bloky pro výpis textové hodnoty na displej robota.
- *List* – bloky pro tvorbu listů a hledání v nich.
- *Colours* – bloky barevné škály pro kontrolu vstupů jednotlivých senzorů.
- *Variables* – mohou být definovány globální a lokální proměnné, které se mohou využívat napříč programem.
- *Functions* – bloky pro definování vlastních funkcí.
- *Messages* – bloky pro zasílání Bluetooth zpráv.

Kód může být vygenerován přímo ve webovém prohlížeči a poté ručně nahrán do robota, popřípadě lze program přenést do robota přímo přes Wi-Fi síť. Výsledný program lze uložit do webového úložiště a sdílet s ostatními uživateli po síti. Je také možné program přímo spustit v simulátoru. Simulátor je napsán v JavaScriptu a má název Open Roberta Simulator. Lze na něm ověřit základní funkčnost výsledného programu. Simulátor je ve formátu 2D a ukazuje robota z ptačí perspektivy. Používání celé platformy je zcela zdarma. Platforma vyhrála v Německu cenu za nejlepší výukovou aplikaci za rok 2015 [2].

Microsoft Robotics Studio

Tato pokročilá vývojová platforma slouží pro ovládání a simulaci robotů. Je spustitelná pouze na operačním systému Microsoft Windows 7 a vyšším. Cílí na všechny typy uživatelů

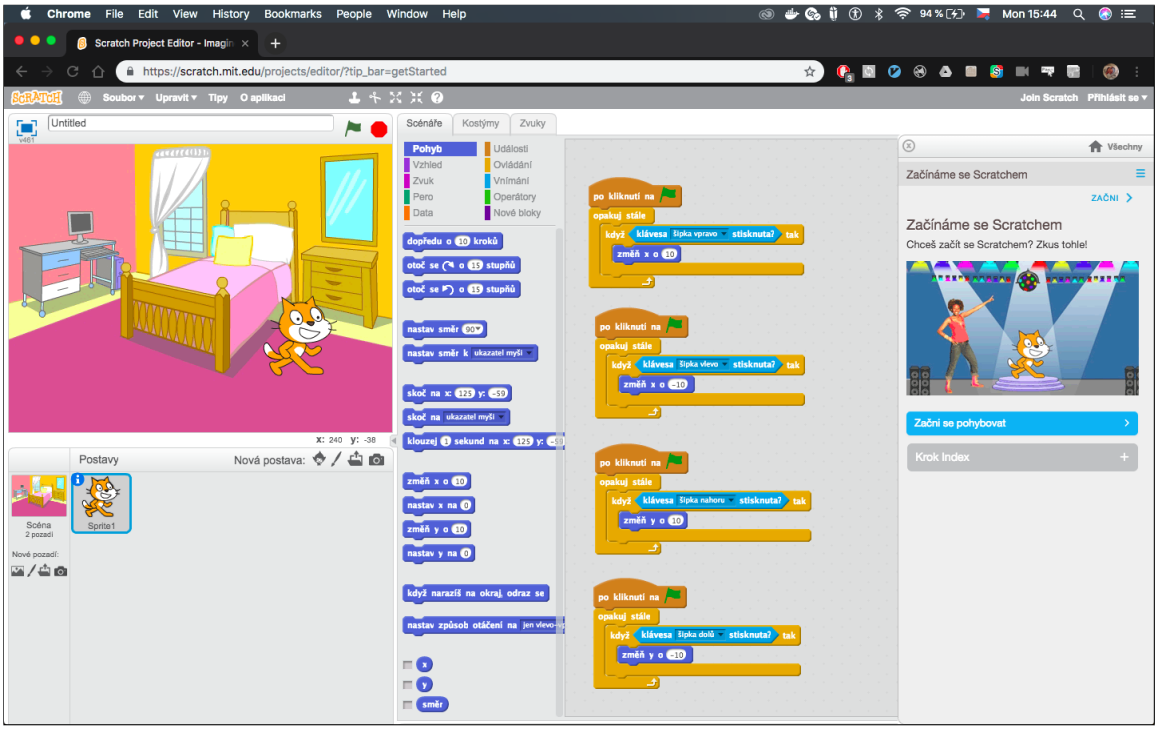
Obrázek 3.2: Webová aplikace RoboBlockly

od studentů až po profesionály. Díky své komplexnosti dokáže komunikovat s velkou řadou hardwaru. Je napsána v jazyce C++. V dnešní době již není dále vyvíjena, ale i tak se jedná o skvělý simulační nástroj, který mohou využívat akademici pro své samostudium. Vývoj byl zastaven v roce 2014 a nic nenasvědčuje tomu, že by měl dále pokračovat. Byl vyvíjen firmou Microsoft od roku 2006. Program se zde tvoří pomocí vizuálního programování, a to jazykem Microsoft Visual Programming Tool, který dovoluje tvorbu robotických aplikací pomocí přetahování a spojování grafických bloků [31].

RoboBlockly

RoboBlockly nazýváme webovou aplikaci postavenou s pomocí knihovny Blockly. Využíváme ji pro výuku programování a matematiky[2]. Vidět jí můžeme na obrázku 3.2. Aplikace umožňuje ovládat a programovat robota, jehož pohyb se vykresluje na SVG Canvasu. Jednotlivé bloky jsou rozděleny do několika kategorií, jako *Logika*, *Matematika*, *Text*, *Proměnné*, *Funkce Robota* atd. Pro výuku slouží speciální lekce, které může student spustit a snaží se vyřešit problém s lekcí spojený. Například přemístit robota na přesně určené místo. Aplikace je přeložená do téměř 50 jazyků, přístupná komukoliv a je zcela zdarma na webové adrese[3].

[2] http://www.roboblockly.org/about.php
[3] http://www.roboblockly.org



Obrázek 3.3: Scratch Editor

Platforma Scratch

Jedná se o vizuální programovací jazyk a komunitu lidí okolo něj utvořenou[4]. Jeho hlavní cílovou skupinou jsou mladí lidé ve věku od 8 do 18 let, kteří se chtějí naučit programovat. S touto platformou lze vytvářet interaktivní příběhy, hry a animace. Pro ještě mladší osoby do 8 let je určena platforma ScratchJr. Celá platforma je bezplatná a editor programů je napsán v jazyce ActionScript od firmy Adobe. Pro jeho fungování tak musíte mít nainstalován Adobe Flash Player pro online editaci programů, popřípadě Adobe Air pro off-line editaci přímo na desktopovém programu. Z dnešního pohledu je řešení touto technologií spíše již zastaralé. Hlavním sloganem je fráze: „Imagine, Program, Share“, v překladu „Představ si, naprogramuj, sdílej“. Tato fráze říká, že je kladen důraz na sdílení vytvořených programů mezi uživateli. Při sdílení je možno vytvářet úpravy a remixy (scratche) programů od jiných uživatelů.

Tvorba programu na platformě Scratch je vcelku jednoduchá a intuitivní. Samotný uživatel výsledný kód nikdy neuvidí, vidí jen výsledný program, který běží přímo v editoru a změny v grafických blocích se projeví ihned. Scratch Editor se skládá z několika základních oken. Mezi ně patří scéna, postavy, scénáře, kostýmy a zvuky. Můžeme jej vidět na obrázku 3.3.

Scéna je okno, ve kterém se vykresluje výsledný program. V horním panelu scény se nachází kontrolní rozhraní. Pomocí tlačítka v levém horním rohu můžeme scénu zvětšit na celou obrazovku, dále můžeme scénu pojmenovat a také můžeme spustit program zelenou vlaječkou, nebo jej zastavit pomocí tlačítka stop, v pravém horním rohu. Jakmile stiskneme vlaječku, vyšle se speciální událost, kterou mohou zachytávat scénáře jednotlivých postav na scéně.

[4] https://scratch.mit.edu/about

V okně Postav je seznam postav, které můžeme přidat do scény. Postavy můžeme přejmenovávat, přidávat nové nebo je mazat. Každé postavě lze přiřadit scénáře. V okně lze také nastavit pozadí scény. To lze také nastavit v okně Pozadí, kde je rozšířenější volba nastavení.

Posledním důležitým oknem je okno Scénáře. Zde vznikají scénáře pro jednotlivé postavy na scéně pomocí vizuálního programování. Bloky, které uživatel přesouvá jsou rozděleny do 10 kategorií: *Pohyb*, *Vzhled*, *Zvuk*, *Pero*, *Data*, *Události*, *Ovládání*, *Vnímání*, *Operátory*, *Nové bloky*. Skládáním bloků v těchto kategoriích se tvoří scénáře. Můžeme například vytvořit scénář, který čeká na událost stisknutí zelené vlaječky a poté ve smyčce čeká na stisknutí pravého tlačítka myši. Pokud k němu dojde, posune postavu o několik pixelů vpravo. Tomuto programování se říká událostmi řízené programování a samotný Scratch je na něm založen. Více informací naleznete přímo v samotné knize popisující práci se Scratch platformou [19].

Scratch je vyvíjen již od roku 2002 a za tu dobu již přilákal okolo 30 miliónů uživatelů. Na vývoji se podílí tým MIT Media Lab ze společnosti Lifelong Kindergarten Group. Na jeho architektuře byli vytvořeny další platformy jako je ScratchJr nebo Snap. Využívá se na základních a středních školách, univerzitách třetího věku a v programátorských klubech. Scratch je dostupný pod licencí GPLv2. Mnoho vizuálních jazyků vychází z návrhu platformy Scratch.

ScratchJr

Za mladší a jednodušší verzi vizuálního programovacího jazyka Scratch považujeme ScratchJr[5]. Primárně je určen pro děti od 5 do 7 let. Jeho hlavní předností je, že uživatelé nemusí umět číst, což se skvěle hodí pro děti v uvedeném věkovém rozpětí. Využívá se především v mateřských školách, kde děti ještě skvěle číst neumí a potřebují rozvíjet kreativitu a systematické a kritické myšlení. Oproti platformě Scratch je značně zjednodušený. Kategorie bloků pro sestavování scénářů jsou zredukovány a samotné bloky jsou velmi zjednodušeny. Z původních 10 kategorií zbylo pouze 6. V jednotlivých blocích se vyskytuje mnohem více ikon. Dalším rozdílem je, že bloky se přichytávají častěji horizontálně než vertikálně. Tento systém je pro děti a začátečníky lépe pochopitelnější. Celý systém byl dotován přes kampaň na crowdfundingovém portálu Kickstarter. První verze aplikace byla vydána v roce 2014 pro zařízení iPad, o rok později i na Android a Chromebook.

Snap

Snap je webová aplikace a vizuální programovací jazyk[6] - rozšířená verze platformy Scratch, zcela zdarma. Má stejný účel jako platforma Scratch, tj. výuka programování. Hlavním rozdílem je, že cílí i na starší a pokročilejší studenty. V nejnovější verzi, je Snap dostupný pouze přes webový prohlížeč a nejsou nutné žádné dodatečné instalace. Je napsán v JavaScriptu a HTML5, tudíž nepotřebuje ani zásuvný modul Adobe Flash od firmy Adobe. Rozvržení editoru je podobné jako u platformy Scratch. Uživatel píše scénáře pro jednotlivé objekty přesouváním grafických objektů. Rozdíl oproti platformě Scratch je v množství a typu tématických kategorií a mírně rozdílných grafických blocích. Vývoj samotné platformy začal v roce 2011.

[5] https://www.scratchjr.org/about/info
[6] https://snap.berkeley.edu/about.html

Stencyl

Výhodou této desktopové aplikace pro tvorbu 2D her[7] je, že převádí výslednou hru do několika formátů. Mezi ně patří HTML5, Adobe Flash, iOs, a Android. Aplikace je dostupná zdarma s několika možnými licencemi k zakoupení. Platforma zajišťuje jak vizuální programovací jazyk, tak i textový programovací jazyk pro tvorbu her. Hry se vytváří pomocí scén, aktérů a definování jejich chování. Vizuální programovací jazyk je inspirován platformou Scratch a vyvíjí se od roku 2011. Aplikace je licencována jako proprietární.

Tynker

Tento vizuální programovací jazyk a výuková platforma, je určen pro děti, které se chtějí naučit programovat hry a aplikace. Tento systém, inspirovaný platformou Scratch, byl napsán v jazyku HTML5 a JavaScript. Platforma je licencována jako proprietární. Hlavní zdroj příjmů pro tento systém pochází z prodeje kurzů. Vyvíjí se od roku 2012 a v roce 2014 byly vydány aplikace pro iPad a Android. Na projekty lze přistupovat jak z webové aplikace, tak z aplikací pro mobilní zařízení [34].

GameFroot

GameFroot je webová aplikace pro tvorbu, hraní a sdílení her[8]. Hry na této platformě se vytváří pomocí vizuálního programovacího jazyka. Je určená pro učitele, kteří chtějí tvořit výukové hry pro své žáky a plně dostupná přes webové rozhraní. Grafický editor webového rozhraní je odvozen od platformy Scratch. Uživatel si proprietární licenci může pořídit u tvůrce platformy, společnosti Gamelab Limited.

Minibloq

Minibloq - počítačový program, umožňující vizuální programovaní platformy Arduino a další podobné platformy. Tento program je primárně určen k učení programování, proto jej můžeme zařadit do výukového softwaru. Používá se na základních i středních školách, a to především v Argentině. Podporované operační systémy jsou Windows a Linux. Program je napsán v jazyce C++.

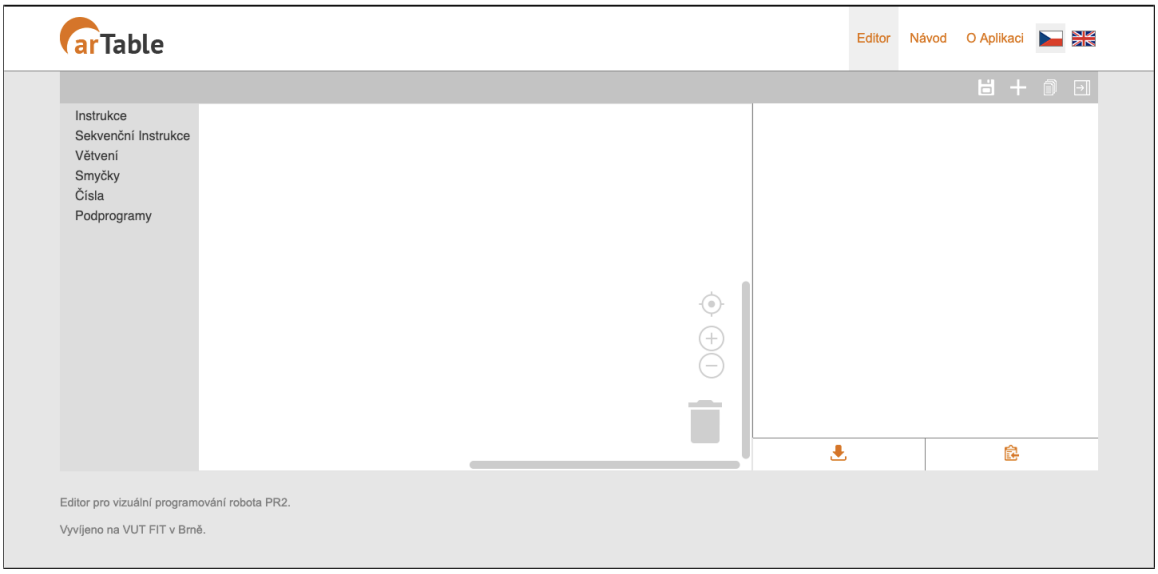
Hlavní komponentou programu je grafický editor, který umožňuje sestavení výsledného programu pomocí grafických bloků. Výsledný kód je většinou vygenerován do jazyka C, který je kompatibilní s platformou Arduino. Každý z bloků je definován pomocí jazyka XML. Pro nahrání výsledného kódu stačí Arduino desku připojit k počítači pomocí USB a spustit synchronizaci.

K tomu, aby mohl uživatel program zkonrolovat přímo v počítači, byl vyvinut simulátor miniSim, podobný technologii Open Roberta Simulator. Jedná se také o 2D simulátor, který ukazuje pohyb robota po desce. Byl napsán v jazyce Python s využitím knihovny PyGame [32].

3.2 Webové uživatelské rozhraní

V této části jsou popsány jednotlivé webové stránky, které budou vytvořeny pro tuto bakalářskou práci. Hlavní stránkou celého webu bude Editor. V editoru bude možné vytvářet

[7] http://www.stencyl.com/
[8] https://gamefroot.com/about-gamefroot/



Obrázek 3.4: Vizuální návrh editoru

šablony pro robota PR2 pomocí vizuálního programování. Mezi jednotlivými stránkami bude možné přecházet pomocí horní lišty, ve které se bude nacházet logo projektu a menu s odkazy na jednotlivé stránky. Toto menu bude dostupné na každé stránce. Mezi další stránky se řadí Návod, Příklady a O Aplikaci. Tyto stránky budou mít spíše informační, než funkční charakter. V menu se také budou nacházet ikony, pomocí kterých bude možné nastavit jazyk celého prostředí. Mezi podporované jazyky bude patřit čeština a angličtina. Rozhraní bude navrženo v nástroji Adobe XD[9].

Editor

Pomocí editoru, hlavní části této bakalářské práce a celého webového rozhraní, vytváříme šablony, které poté nahrajeme do robota PR2. Návrh uživatelského rozhraní editoru může čtenář nalézt na obrázku 3.4. Je silně inspirován webovými aplikacemi jako je Open Roberta Lab a RoboBlockly. Skládá se ze čtyř hlavních části - horizontální lišty, Toolboxu, plátna a generátoru kódu.

V horizontální liště se nachází prvky pro správu programů. Umožňuje uložit aktuální program a zobrazit již uložené, zobrazit nahrané instrukční sady, nahrát nové instrukční sady ze souboru z počítače nebo z URL, a skrýt nebo zobrazit generátor kódu. Možnost uložení programu je přístupná přes ikonu diskety v horizontální liště. Seznam všech uložených programů se otevře při kliknutí na druhou ikonu s názvem zobrazit programy. V tomto seznamu je vizuálně zvýrazněn aktivní program.

V seznamu programů lze vidět všechny aktivní a uložené programy s jejich názvy, s tím že je vždy pouze jeden program aktivní. Každý program lze vymazat kliknutím na ikonu křížku vedle názvu programu. Každý program lze učinit aktivním kliknutím na název programu. Při učinění programu aktivním se vizuální bloky programu nahrají na plátno a přepíší staré grafické bloky. Pokud má uživatel na plátně nějaké neuložené změny může o ně přijít.

[9] https://www.adobe.com/products/xd.html

Obrázek 3.5: Responzivní verze editoru

Třetí ikona v horizontální liště, zobrazuje uložené instrukční sady a nabízí možnost přidat instrukční sadu do kategorie Instrukce a Sekvenční Instrukce do Toolboxu v editoru. Grafické bloky a jejich kategorie budou podrobněji popsány v následující kapitole.

Poslední ikona umožňuje přepnutí viditelnosti generátoru výsledného kódu - zobrazený generátor skryje a naopak. Při ukrytí generátoru se plátno roztáhne do volného místa.

V generátoru kódu je s výsledným kódem možné provést dvě věci. Uložit do počítače pomocí spodní ikony Download a uložit do schránky a nakopírovat do jiného souboru pomocí spodní ikony Clipboard. Každý uložený program se bude ukládat automaticky do systému ARTable.

Responzivní návrh pro mobilní zařízení je vidět na obrázku 3.5. Použije se na monitorech jejichž šířka je menší než 700px. To jsou nejčastěji mobilní telefony. U responzivního návrhu byla využita technika zarovnání bloků pod sebe při níž se jednotlivé části roztáhnou na plnou šířku obrazovky a zařadí se pod sebe.

Návod

Návod je stránka, na které je popsáno, jak pracovat s Editorem. V první části obsahuje popis jednotlivých funkcí jako je popis horizontální lišty, ukládání programů, zobrazování a mazání uložených programů, nahrávání programů na plátno, zobrazování, mazání a nahrávání instrukčních sad do Toolboxu.

V další části stránky je popis vizuálního programování. Jsou vysvětleny funkce jednotlivých bloků, a je popsán kód šablony, která je generována. Na tuto část navazuje stránka O Aplikaci, ve které si uživatel může přečíst informace o systému.

3.3 Srovnání platforem

Pro naší práci je nejdůležitější podpora tvorby vlastních vizuálních programovacích jazyků a kompatibilita s webovým prohlížečem. Dále můžeme do jednotlivých kritérií zahrnout srozumitelnost dokumentace a jednoduchost používání knihovny. Naše uživatelské rozhraní bude mířit na pokročilejší uživatele systému, proto nebudeme volit mezi nejjednoduššími knihovnami.

Mezi platformy, které podporují výše uvedené vlastnosti se řadí Blockly, Scratch Blocks, PXT a Droplet. Platforma Scratch Blocks je spíše cílena na děti a méně zkušené uživatele. Platforma Droplet přidává funkci převodu kódu z textové verze do grafické, není však pro tuto bakalářskou práci vhodná. Platforma PXT se zdá optimální, ale převod do jiného kódu, než JavaScript je složitý a nepřehledný. Po delší úvaze je zvolena knihovna Blockly díky srozumitelné a obsáhlé dokumentaci, možnosti přidávání vlastních bloků, jednoduchého generování výsledného kódu, volné licenci a velké komunitě uživatelů a tvůrců.

3.4 Grafické bloky

Grafické bloky tvoří druhý pilíř této bakalářské práce. Pomocí jednotlivých grafických bloků budeme sestavovat výsledný program pro robota PR2. Každý grafický blok je v právě jedné kategorii v Toolboxu grafického editoru. Pomocí Toolboxu můžeme bloky přetahovat na plátno editoru a spojovat je v posloupnost instrukcí. Návrh bloků byl realizován ve webovém nástroji Blockly Developer Tools[10].

Instrukce pro robota

Každý program má svou sadu instrukcí. Tyto instrukce mohou být rozšiřovány pomocí instrukčních souborů. Příkladem instrukčního souboru je například soubor `instruction.yaml`[11] v repositáři ARTable. Při nahrání instrukční sady budou instrukce dostupné v každém programu. Jakmile jej uložíme, můžeme přidávat instrukce z ostatních souborů. Nahrané instrukce se vždy váží k danému souboru. Jakmile odstraníme instrukční soubor z programu, odstraníme i instrukce s ním spojené. Všechny nahrané instrukce se objeví v kategorii Instrukce a Sekvenční instrukce v Toolboxu jako samostatný grafický blok. Každá instrukce je pak dostupná v obou kategoriích. V každém instrukčním souboru jsou názvy instrukcí a informace kolik má daná instrukce příznaků. Každá instrukce v systému ARTable končí buďto úspěchem nebo neúspěchem. V případě úspěchu se pokračuje na další instrukci, v případě neúspěchu se instrukce opakuje. Programátor může vždy explicitně určit na jaké instrukce se bude skákat v případě úspěchu a neúspěchu. To bude využito při tvorbě větvení, podmínek a smyček v implementaci. Toto chování editoru bude probíhat bez vědomí uživatele.

Kategorie Instrukce obsahuje vstupní bloky pro další bloky z kategorie Větvení a Smyčky. Tyto bloky mohou být přidávány jako podmínky pro bloky *If* a smyčky *While*, *Do…While*

[10] https://blockly-demo.appspot.com/static/demos/blockfactory/index.html
[11] https://github.com/robofit/artable/blob/master/art_instructions/config/instructions.yaml.

a dalších. Vzhled bloků je vidět na obrázku 3.6. Počet bloků závisí na počtu instrukcí v instrukčním souboru.

Kategorie Sekvenční Instrukce, obsahuje bloky představující instrukce, které mohou být řazeny za sebe. Vytvářejí tak posloupnost instrukcí, které mohou být přidávány do těl podprogramů, smyček a jednotlivých větví podmínek. U každé sekvenční instrukce je možné určit co se stane, pokud se jí nepodaří provést. Ve výchozím chování se instrukce zopakuje (repeat). Mezi další možnosti patří skočení na konec bloku (break), opakování od začátku smyčky (continue) nebo ukončení celého programu (exit). Vzhled bloků je vidět na obrázku 3.7. Počet bloků závisí na počtu instrukcí v instrukčním souboru.

Obrázek 3.6: Kategorie Instrukce

Obrázek 3.7: Kategorie Sekvenční Instrukce

Větvení

Větvení obsahuje 4 bloky pro větvení s podmínkami a logické operátory. Do podmínky lze přidat instrukci, nebo logický výraz složený z několika instrukcí. Podmínka je vyhodnocena jako pravdivá, pokud se danou instrukci podaří provést. U logického výrazu je vždy prováděna instrukce za instrukcí. Po vykonání všech instrukcí je daný výraz vyhodnocen. Vzhled bloků lze vidět na obrázku 3.8.

Smyčky

Kategorie Smyčky obsahuje 3 bloky pro podporu základních smyček v programu. Podmínkou pro tyto smyčky bude vždy jedna instrukce, nebo logický výraz instrukcí. Jestli se výraz povede provést bude podmínka vyhodnocena jako kladná a provede se tělo instrukce. Vzhled bloků pro smyčky lze vidět na obrázku 3.9. Do těchto smyček lze vkládat i příkazy `break` a `continue`, které známe z klasických programovacích jazyků. Tyto příkazy lze vyvolat vybráním jedné z akcí u sekvenčních instrukce, která se provede při neúspěchu instrukce.

Obrázek 3.8: Kategorie Větvení

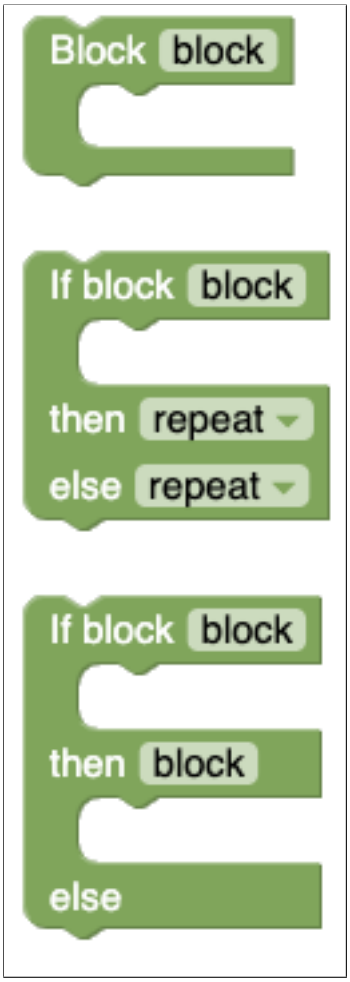
Podprogramy

Všechny instrukční bloky, větvení a smyčky se musí řadit do bloku Podprogram z kategorie Podprogramy. Není povoleno vnořování podprogramů do sebe, mohou se řadit pouze za sebe. Tyto grafické bloky reprezentují blok v programu ARTable. Vzhled grafických bloků lze vidět na obrázku 3.10. Kromě klasického bloku Podprogram, existují speciální grafické bloky určené pro větvení, umožňující přesměrovat klasickou sekvenční posloupnost podprogramů. Pokud se celý blok podaří provést, začne se od zvoleného podprogramu z nabídky.

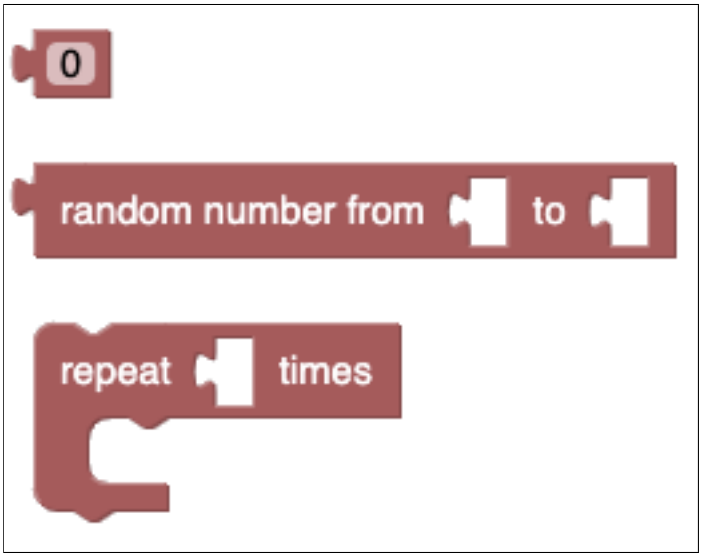
Čísla

Tyto bloky reprezentují čísla v šabloně. Lze je vkládat jako vstup do grafického bloku `Repeat X times {...}`. Tento blok poté zopakuje zadanou posloupnost instrukcí. Čísla nemohou být záporná a je nastaven maximální horní limit na číslo 1000. Je také možné vložit náhodné číslo v zadaném rozmezí. Toto číslo se vypočte při generování šablony. Vzhled číselných bloků lze vidět na obrázku 3.11.

Obrázek 3.9: Kategorie Smyčky



Obrázek 3.10: Kategorie Podprogramy



Obrázek 3.11: Kategorie Čísla

Kapitola 4

Implementace

4.1 Použité technologie

Pro implementaci webového uživatelského rozhraní byla na serverové straně použita technologie `PHP` verze `5.3.29`. Pro klientské vykreslování obsahu v prohlížeči byla zvolena technologie `HTML` a `CSS`. Pro zpřehlednění práce s `CSS` byl vybrán preprocesor `LESS`. Pro dosažení dynamického obsahu na stránce byla aplikovaná technologie `JavaScript`. Pro značné usnadnění práce byla použita knihovna `jQuery`, verze `2.2.4`, která slouží především k pohodlnému označování a procházení `HTML` elementů, vytváření `AJAX` spojení, a spravování událostí, které spouští uživatel nebo prohlížeč. Další významnou `JavaScriptovou` knihovnou byla `Blockly`. Ta byla využita pro zjednodušení vytváření grafických bloků a zjednodušení implementace vizuálního programování. Při inicializaci stačí knihovně říci, do jakých `HTML` elementů se má Toolbox a Editor vykreslit. Pro uchování programů vytvořených v systému ARTable byl použit formát `XML`. Instrukce byly ukládány ve formátu `JSON`. Žádná databázová technologie nebyla využita, ukládání programů i instrukcí tak bylo ve formátu `XML` a `JSON` mnohem přehlednější. Díky tomu lze programy přenášet mezi více Editory mnohem jednodušeji. Uživatel navíc načte stránku pouze jednou (pokud nepřidává další instrukce) a pracuje pouze s editorem. Časovou stránku tedy můžeme zanedbat. Celá platforma běží v současné době na studentské webové adrese[1].

4.2 Hierarchie souborů

V kořenovém adresáři se nachází jednotlivé webové stránky a další podadresáře s pomocnými soubory. Nejdůležitějším souborem je soubor `index.php`. Obsahuje elementy reprezentující editor vizuálního programování, horizontální lištu, Toolbox a vizualizér výsledného kódu. Toolbox se definuje pomocí `XML` elementu, ve kterém jsou popsány jednotlivé kategorie a bloky jenž do kategorií spadají. V kořenovém adresáři jsou dále složky s dalšími důležitými soubory. V seznamu níže jsou uvedeny adresáře a nejdůležitější soubory, které se v nich nachází.

- `/templates/` – Obsahuje hlavní části stránek, které jsou vkládány do stránek v kořenovém adresáři. Patří zde například hlavička (`header.php`), patička (`footer.php`) a navigační lišta (`navbar.php`).

[1] http://www.stud.fit.vutbr.cz/~xvichs00/ibt/

- `/functions/` – Stránky pro zpracovávání formulářů. Formuláře slouží pro přidávání a mazání instrukčních sad uložených v adresáři `/js/instructions/` a přidávání, mazání a čtení programů uložených v adresáři `/artable/programs/`.
- `/js/` – Všechny JavaScriptové soubory, které jsou načítány do stránek. Nejdůležitějším souborem je `artedit.js`, jenž obsahuje zdrojové kódy editoru pro vizuální programování. Je blíže popsán v samostatné kapitole níže. V podadresáři `/js/instructions/` jsou také uloženy definice grafických bloků určující vzhled sekvenční a nesekvenční instrukce.
- `/less/` – Zdrojový kód v jazyce `less`, ze kterého se generuje soubor `main.css` v adresáři `/css/`. Generování se spouští příkazem `make css` v příkazovém řádku.
- `/css/` – `CSS` styly, které jsou načítány do webových stránek. Jedná se o dva soubory, a to `normalize.css` a `main.css`.
- `/images/` – Složka obsahující obrázky, používané v `CSS` stylech a na webových stránkách.
- `/artable/` – Obsahuje uložené instrukce ve formátu `JSON` a programy ve formátu `XML`. Aby se mohly instrukce uložit do Toolboxu jsou u nich uloženy pouze názvy.
- `/downloads/` – Obsahuje jediný soubor a to `program.txt`, jehož součástí je kód v jazyce `Python`, vygenerovaný při vizuálním programování v editoru ARTable. Dynamicky se pomocí JavaScriptu ukládá do tohoto souboru a uživatel si jej poté může stáhnout v prohlížeči, aniž by načítal jakoukoliv stránku. Funguje jen v případě, že prohlížeč podporuje atribut `download` u `HTML` tagu `<a></a>` (jinak se pouze načte stránka s tímto programem).

V odstavcích níže se nacházejí popisy nejdůležitějších souborů, které tvoří jádro vytvořené platformy. Jsou také popsány nejdůležitější algoritmy, které byli použity a je nastíněno, jak se mohou přidávat další grafické bloky do ARTable editoru.

`/js/artedit.js`

Tento soubor reprezentuje JavaScriptový plugin ARTEdit. Tento plugin vytvoří editor pro vizuální programování a vloží jej do stránky. Pro svou funkčnost potřebuje další JavaScriptové pluginy ,a to plugin jQuery a plugin Blockly. Bez nich se vyvolání konstruktoru okamžitě ukončí. Plugin se může inicializovat například tímto způsobem:

```
var editor = new ARTEdit();
```

Výpis 4.1: Inicializace editoru ARTEdit

Tento způsob je nejjednodušší, ale uživatel také může, pokud chce, specifikovat dodatečné parametry do konstruktoru. V současné době lze do konstruktoru zadat tyto parametry:

- *areaEl* – ID HTML elementu, který obaluje Blockly Editor a Blockly Toolbox
- *editorEl* – ID HTML elementu, který definuje Blockly Editor
- *toolboxEl* – ID XML elementu, který definuje Blockly Toolbox

- *outputEl* – ID HTML elementu, kde se generuje výsledný kód
- *programName* – Název programu. Výchozí hodnota je „ROS Program“.

Seznam všech parametrů, které lze specifikovat, je uložen ve veřejné vlastnosti objektu `ARTEdit.defaults`.

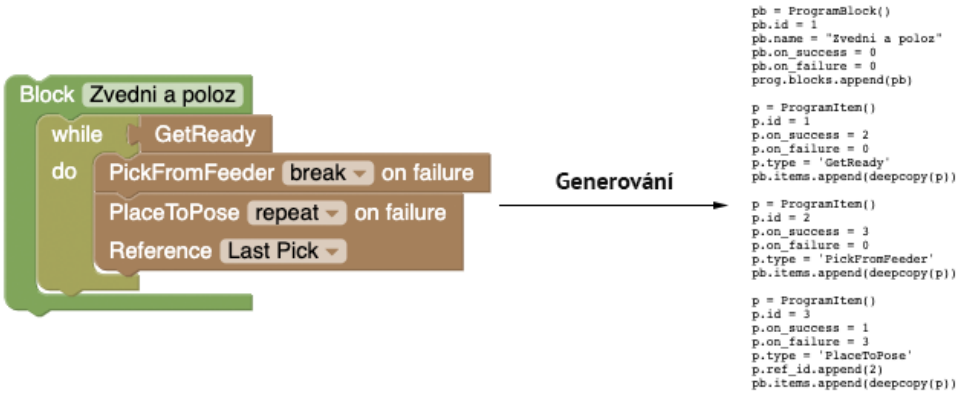
Celý plugin se skládá z několika veřejných a soukromých vlastností a funkcí. Mezi nejdůležitější veřejné metody patří konstruktor, `getPythonCode`, `getRosObject`, `setName`, `getName`, `setId` a `getId`. V privátních metodách jsou poté definovány grafické bloky a jejich generátory kódu. Veřejné proměnné nejsou definovány. Vše se nastavuje přes veřejné `get` a `set` metody. Privátní proměnné slouží jako pomocné při generování kódu. To probíhá v několika fázích. Začne se provádět, jakmile se změní grafické bloky v editoru (na událost `BLOCKS_CHANGED`). Probíhá postupně od nejvyššího bloku (zeleně zbarvené bloky v kategorii Podprogramy) po nejvíce zanořené a je velmi podobné s generováním kódu při syntaktické analýze shora dolů. Je vhodné poznamenat, že bloky, které nejsou spojeny s hlavním blokem, jsou v editoru zašedlé a jejich kód se nevytváří. Vždy tak vznikne jeden souvislý kus programu, který se skládá z bloků a instrukcí řazených za sebe.

Kód programu se negeneruje přímo do textového řetězce (to by bylo příliš nepřehledné a nic dalšího by se s kódem nemohlo dělat), ale vytváří se speciální JavaScriptový objekt. Ten je uložen v privátní proměnné `programObject`. Na začátku generování se nastaví hlavička objektu (funkcí `setHeader`) a poté se přidávají bloky (funkcí `addBlock`) a instrukce (funkcí `addItem`). Tyto funkce se volají z jednotlivých generátorů kódu jednotlivých grafických bloků. Z výsledného objektu se poté může vytvořit kód v jazyce Python (funkcí getPythonCode), nebo objekt, který je posílán přes ROSBridge do systému ROS a robota PR2 (funkcí `getRosObject`).

Problém, který generování řeší je převod instrukcí, které jsou zanořeny do různých grafických bloků, na jejich lineární variantu. V této lineární variantě jsou instrukce v jednom seznamu a o pořadí provádění rozhoduje hodnota natavená ve vlastnosti `on_success` a `on_failure`. Při úspěchu prováděné instrukce se skočí na instrukci s ID rovnu hodnotě `on_success` a při neúspěchu na instrukci s ID rovnou hodnotě `on_failure`. Vlastnosti `on_success` a `on_failure` můžeme popisovat jako ukazatele. Pokud je hodnota některého z ukazatelů nastavena na 0, znamená to, že blok programu je ukončen. Příklad generování je uveden na obrázku 4.1.

Na obrázku 4.1 jsou vpravo vidět grafické bloky a vlevo vygenerovaný kód, který vytvořila funkce `getPythonCode`. Při generování se vytvořil jeden blok s názvem „Zvedni a poloz“ a s `ID` 1. Dále se vytvořili tři instrukce, každá s `ID` 1, 2 a 3, a přidali se do bloku. Během generování byli použity pomocné privátní proměnné, mezi nejdůležitější patří:

- `blockCounter` – Čítač bloků. Používá se při přidávání bloků do objektu `programObject` a výpočtu ID bloku.
- `instructionCounter` – Čítač instrukcí. Používá se při výpočtu `ID` instrukce.
- `negation` – Logická proměnná, které se vždy překlopí hodnota v generátoru kódu grafického bloku `not`. Využívá se při naplňování pole `logic`.
- `logic` – Pole logických objektů. Objekty do něj přidávají jednotlivé grafické bloky. Např. smyčky a podmínky.



Obrázek 4.1: Generování programObjectu z grafických bloků

- `breakHop` – Pole čísel, představují instrukce, na které se má skočit při selhání sekvenční instrukce. Pokud je vybrána možnost `break`, vždy se vybere nejvyšší prvek v poli.
- `continueHop` – Pole čísel, představují instrukce, na které se má skočit při selhání sekvenční instrukce. Pokud je vybrána možnost `continue`, vždy se vybere nejvyšší prvek v poli.
- `programObject` – Hlavní objekt, ze který se sestavuje při generování kódu. Generuje se, jakmile se, jakkoliv změní bloky v editoru. Z tohoto objektu se poté sestavuje kód v jazyce Python a ROS objekt.

Při generování je také důležitá privátní funkce `countInstructions`, která bere jako vstup grafický blok a vrací počet všech instrukcí v tomto bloku (sekvenčních i nesekvenčních). Při přidávání instrukce do proměnné `programObject` je nejnáročnější výpočet hodnoty ukazatelů. Hodnota `ID` je vždy rovna hodnotě privátní proměnné `instructionCounter`. Ta je při každém přidávání instrukce vždy zvětšena o 1 a při přidání nového bloku je vynulována. Pro výpočet ukazatelů se používá 5. kroková metoda ve funkci `instructionWrite`. Tato funkce je veřejná, aby ji mohla zavolat i instrukce, kterou vygenerujeme v php souboru `functions/instruction_handling.php` při přidávání instrukcí.

V pěti krokové metodě se postupně nastavují hodnoty ukazatelů, které začínají na hodnotě privátní proměnné `instructionCounter`. Jednotlivé kroky:

1. Inkrementace ukazatele – Hodnota `on_success` se inkrementuje o 1. Ukazuje tak na následující instrukci. Hodnota `on_failure` zůtane beze změny.
2. Uživatelská změna `on_failure` stavu – Podle nastavení ve výsuvném menu, si uživatel vybere, kde má instrukce skočit v případě neúspěchu. Pouze u sekvenčních instrukcí. Na výběr je repeat (beze změny), break, continue a exit (hodnota 0).
3. Vyřešení logických výrazů – Pokud je nastaven logický objekt `logic`, pak se provede přesměrování na zadaná místa, podle nejvyššího objektu v poli `logic`. Hodnota `on_success` se nastaví na `logic.true` a hodnota `on_failure` se nastaví na `logic.false`.

4. Kontrola překročení meze – V každém bloku se nastavuje mez, přes kterou vnitřní zanořené bloky nesmí ukazovat. Např. u nekonečné smyčky, nesmí nikdy instrukce ukazovat přes konec smyčky. Pokud ukazuje, musí se ukazatel nastavit na začátek smyčky.

5. Kontrola překročení konce bloku – Pokud hodnota ukazatele překračuje přes blok (na neexistující instrukci), nastaví se hodnota na 0. Velikost bloku se spočítá pomocí funkce `instructionCount`. Instrukce nikdy nesmí ukazovat na neexistující instrukci.

Do pluginu je možné přidávat další grafické bloky, a tak plugin rozšiřovat. Pro přidání bloku je nutné blok definovat v poli `Blockly.Blocks` a také naprogramovat jeho generátor kódu v poli `Blockly.Python`. Poté je nutné blok vložit do `XML` Toolboxu. V generátoru kódu je nezbytné používat privátní pomocné proměnné a funkce jako je tomu u ostatních bloků a jejich generátorů. Pokud grafický blok obsahuje nějaké texty, je dobré jej přeložit do anglické a české verze a texty přidat do souborů `/js/en.js` a `/js/cs.js`.

`/js/index.js`

Skript `index.js` zajišťuje dynamický obsah na hlavní stránce, kde se nachází editor. Obsahuje skripty, které inicializují ROSBridge a plugin ARTEdit. Při inicializaci ROSBridge je možné nastavit URL, na které je spuštěný ROS server. Dále se v souboru nachází události, které se spustí při kliknutí na některou z ikon v horizontální liště editoru. Dále umožňuje zkopírovat vygenerovaný kód do schránky, při kliknutí na levou ikonu pod generátorem kódu. Při ukládání programu, vytvořeného v editoru, se program nejprve odešle na ROS server, kde se zkontroluje, jestli je validní. Pokud ano, uloží se na ROS server a jejich `XML` verze se zálohuje na webový server. Pokud program validní není, neuloží se na ani jeden ze serverů. Při mazání programu se program smaže z obou serverů. Pokud se k ROS serveru nejde připojit, nelze programy ukládat ani mazat na žádný ze serverů. Ukládání a mazání instrukcí není závislé na funkčnosti ROS serveru, jako je tomu u programů.

`cs.js a en.js`

Soubory `cs.js` a `en.js` jejichž součástí jsou texty pro českou a anglickou verzi pluginu ARTEdit. Obsahují texty pro Blockly editor a texty pro grafické bloky, uložené v pluginu ARTedit. Při přidávání grafického bloku je nutné vložit přeložené texty do těchto souborů a odkazovat se na ně přes pole `Blockly.Msg`.

`/functions/function.php`

Soubor `function.php` obsahuje definice proměnných, které se používají napříč celou platformou. Inicializuje proměnou `$root`, která určuje kořenový adresář. Pokud budete přesouvat celou platformu a vložíte jí do podadresáře na webovém serveru, je nutné tuto proměnnou přepsat. Také inicializuje proměnnou `$lang`, která určuje jazyk, který si zvolil uživatel. Hodnota proměnné je uložena v poli `$_SESSION["lang"]` po dobu 10 dní. Výchozím jazykem platformy je angličtina. Dále definuje textové konstanty, které se používají při výtisku jakéhokoliv viditelného textu na webovou stránku. V současnosti je podporován anglický a český jazyk.

/functions/instructions_handling.php

Soubor `instructions_handling.php` zpracovává instrukční soubor, který je poslán na php server. Instrukční soubor může být nahrán dvěma způsoby. Můžeme nahrát URL anebo celý soubor z počítače pomocí elementu `<input type="file">`. V případě URL si php server z dané URL soubor stáhne. Poté soubor přečte a zpracuje ho. Zpracování provádí funkce `parse_instructions`, která pomocí regulárního výrazu najde jména všech instrukcí a poté se podívá, jestli instrukce neobsahuje nějaký příznak pomocí dalšího regulárního výrazu. V současné době jsou podporovány příznaky `pick`, `place`, `using_object`, `using_pose`, `using_polygon` a `ref_to_pick`.

Jakmile jsou zjištěny názvy všech instrukcí a jejich příslušné příznaky, uloží se tyto informace do adresáře `/artable/instructions/` ve formátu `JSON` pomocí funkce `saveInstructions`. Při ukládání se také vytvoří jednotlivé grafické bloky a uloží se do adresáře `/js/instructions/`.

/functions/programs_handling.php

Poslední významný soubor, `programs_handling.php`, zajišťuje ukládání programů ve formátu `XML` do adresáře `/artable/programs/`. V prvním řádku programu se nachází `ID`, které je použito k identifikaci programu v systému ROS. Každý program má také své jméno, pod kterým je ukládáno na webovém serveru.

4.3 Implementace grafických bloků

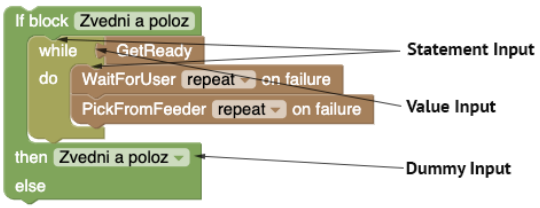
V této podkapitole se blíže podíváme na implementaci jednotlivých grafických bloků, které jsou nastálo definovány v pluginu ARTEdit. K těmto grafickým blokům lze přidat další instrukce načtením instrukční sady, nebo rozšířením samotného pluginu ARTEdit. Při přidání instrukční sady se grafické bloky přidají do kategorie Instrukce a Sekvenční Instrukce. Grafické bloky jsou definovány v souboru `/js/artedit.js`. V současné době čítá jejich počet 13. Níže můžete nalézt jejich seznam grafických bloků a 4 kategorií do kterých patří:

- *Větvení* – if_statement, if_else_statement, not, and_or
- *Smyčky* – while, while_true, do_while
- *Čísla* – number, random_number, repeat
- *Podprogramy* – block_definition, block_definition_if, block_definition_if_then

Každý blok má definován svůj vzhled a svůj generátor kódu. U každého bloku je také definováno, jaké typy bloků může obsahovat. To zabraňuje tomu, aby se například blok podprogramu nemohl zanořit do bloku smyčky. Naopak to však funguje. Každý grafický blok má v knihovně `Blockly` své vstupy, do kterých se zanořují další bloky. Rozlišujeme tři základní typy vstupů:

- `ValueInput` – Vstupem je blok, který se zachytává zprava.
- `StatementInput` – vstupem je sekvence zanořených bloků.
- `DummyInput` – vstupem není žádný blok, ale grafický prvek, např. text.

Typy vstupů jsou vidět na obrázku 4.2. Každý ze vstupů má své jméno, které musí být v rámci grafického bloku unikátní. Vstupy a jejich jména se nastavují při definici grafického bloku. Každý grafický blok může být sekvenční nebo nesekvenční. Sekvenční bloky lze řadit za sebe. V této práci jsou sekvenční bloky například v kategorii Podprogramy nebo Sekvenční Instrukce.



Obrázek 4.2: Typy vstupů grafického bloku

Generátor kódu se poté spustí nad nejvyššími grafickými bloky (v našem případě bloky z kategorie Podprogramy) a z těch se poté mohou volat další generátory kódu nad jednotlivými vstupy grafického kódu, a to pomocí funkce `statementToCode`. Vstupem této funkce je grafický blok a název vstupu (v textové podobě). Blok samotný se předává jako první parametr generátoru. Pokud funkci předáme vstup typu `statementInput`, spustí se generátor pro celý balík (stack) sekvenčních bloků, a to od prvního po poslední. To stejné platí i pro nejvyšší úroveň sekvenční bloků (v našem případě kategorie Podprogramy). V této práci se během generování nevytváří textový kód, ale upravuje se privátní proměnná `programObject`, ze které se poté sestavuje program v jazyce Python a objekt odesílaný do systému ROS přes ROSBridge.

V následujících podkapitolách jsou popsány generátory kódu jednotlivých grafických bloků. Definování jejich vzhledu je triviální a bylo usnadněno pomocí nástroje Blockly Developer Tools[2].

Podprogramy

Podprogramy jsou hlavní grafické bloky, do kterých se obalují všechny ostatní bloky. Při generování se vždy inkrementuje čítač bloků `blockCounter`. Před spuštěním funkce `statementToCode` se přidá do proměnné `programObject` nový kus bloku s hodnotou `ID` rovnou proměnné `blockCounter`, do kterého se následně v dalších generátorech kódu budou přidávat další instrukce.

Instrukce

Instrukce se dělí se na sekvenční a nesekvenční. Nesekvenční instrukce se přidávají do podmínek a logických výrazů. Sekvenční poté do vstupů typu `statementInput` jednotlivých grafických bloků. Jakmile se spustí generátor instrukce, inkrementuje se proměnná `instructionCounter` a vypočítají se hodnoty ukazatelů `on_success` a `on_failure` pomocí pěti krokové metody. Poté se vypočítané hodnoty přidají do pomocné proměnné `programObject` jako další instrukce. Sekvenční i nesekvenční instrukce spouští tu samou funkci `instructionWrite`.

Každá instrukce má nastavený své příznaky. Informace o jednotlivých příznacích je uložena v instrukční sadě. Např. instrukce `PlaceToPose` má nastaven příznak `place`. Ten říká,

[2] https://blockly-demo.appspot.com/static/demos/blockfactory/index.html

že instrukce musí ukazovat na jinou instrukci, která má nastaven příznak `pick`. V editoru je toho dosaženo dodatečným výsuvným menu, ve kterém si uživatel může zvolit instrukci, na kterou potřebuje ukazovat. Ve výchozím nastavení se vždy ukazuje na poslední instrukci s příznakem `pick`. Pokud má instrukce nastaven příznak `ref_to_pick`, musí také ukazovat na instrukci s příznakem `pick`. Ostatní příznaky lze ignorovat. V normálním případě nemá instrukce nastaven žádný příznak.

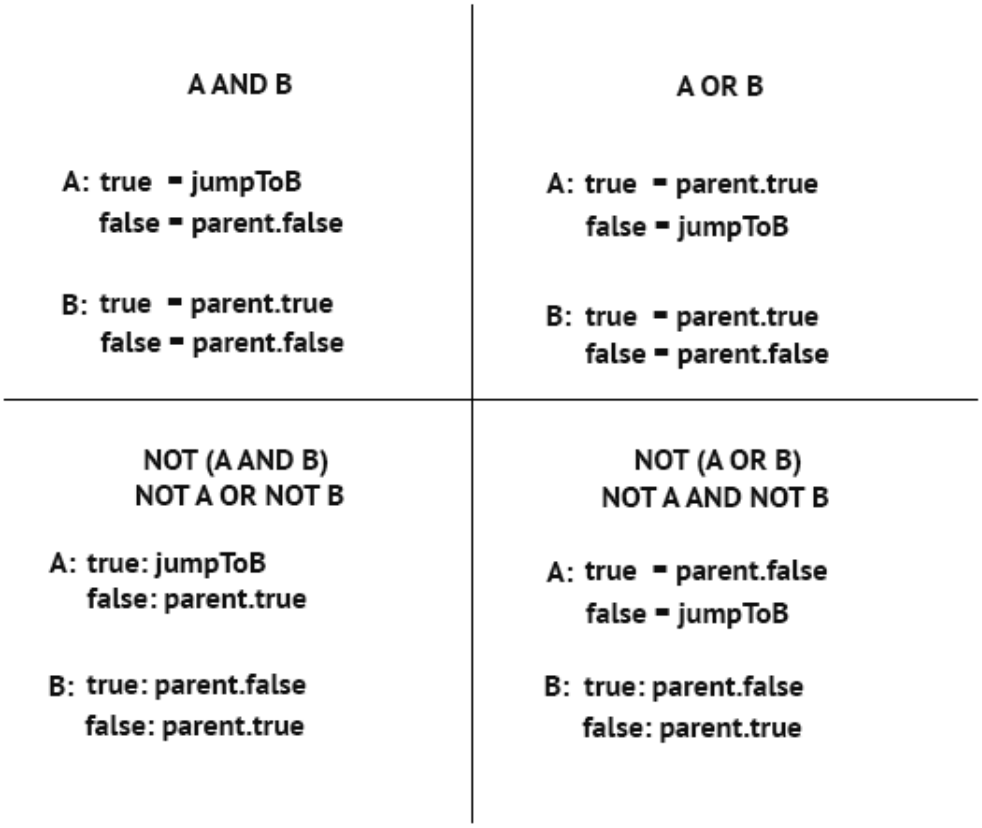
Větvení a smyčky

Kategorie Větvení a Smyčky obsahují nejzajímavější a nejsložitější implementační část této práce. Každý grafický blok v této kategorii, se skládá z podmínky a z těla grafického bloku. Pokud se správně vykoná podmínka, provede se tělo bloku. Do podmínky se může vkládat jedna instrukce, anebo logický výraz s dvěma instrukcemi. Logické výrazy se navíc mohou zanořovat do sebe, a tak může vzniknout podmínka s libovolným počtem instrukcí. Do podmínky je také možné přidávat negace, a to jak k jednotlivým instrukcím, tak k celému logickému výrazu. Implementačně je vše řešeno pomocí privátního pole `logic`. Před začátkem vyhodnocování podmínky se do pole `logic` přidá objekt s hodnotami `true` a `false`. Pokud se poté vyvolá funkce `instructionWrite` hodnoty ukazatelů se vypočítají na základě nejvyššího objektu v poli `logic`. V generátoru kódu logického výrazu ("and_or") se do pole `logic` přidává tento objekt také. Hodnoty `true` a `false` se pak vypočítávají podle vzorců z obrázku 4.3. Pro každou ze dvou částí logického výrazu se vypočítává logický objekt zvlášť. Objekt `parent` značí nejvyšší prvek v poli `logic`. Hodnota `jumpToB` je rovna `instructionCounter + A.length + 1`. Hodnota `A.length` se vypočte pomocí funkce `countInstruction`. Po ukončení vyhodnocení podmínky a logického výrazu se poslední prvek v poli `logic` odstraní.

To však pro zajištění správné funkcionality nestačí. U smyček, které mají tělo bloku za podmínkou může nastat situace, že některá z instrukcí bude ukazovat mimo tuto smyčku. Například poslední instrukce u grafického bloku "while"musí ukazovat na první instrukci v podmínce téhož bloku. To je dosaženo zavedením 4. kroku v 5 krokové metodě a přidávání objektu do privátního pole `logic`, před vyhodnocováním těla bloku. Tento objekt má nastaveny hodnoty `bottomLine` a `bottomJump`. U každé instrukce se ve 4. kroku pěti krokové metody zkontroluje, jestli hodnota ukazatelů neukazuje přes hodnotu `bottomLine`. Pokud ano, bude ukazovat na hodnotu `bottomJump`. Takových objektů může být za sebou více. Například dvě smyčky zanořené do sebe. Proto musíme vyhodnotit každý objekt v poli `logic` a to od nejvyššího po nejnižší. Po zpracování těla bloku se nejvyšší logický objekt odebere.

Repeat blok

Repeat blok, grafický blok, který opakuje instrukce a to tolikrát, jaké číslo mu nastavíme. Číslo může být předané přímo, nebo jako grafický blok vracející náhodné číslo v zadaném rozmezí. V generátoru kódu se nepřidává žádný další objekt do privátního pole `logic`. Vstup do tohoto pole se pouze zpracuje vícekrát než jednou. Kvůli tomuto bloku má funkce `countInstructions` nastaven druhý parametr `repeatFactor`. Ten násobí jednotlivé počty instrukcí, které jsou obsaženy uvnitř bloku "repeat".



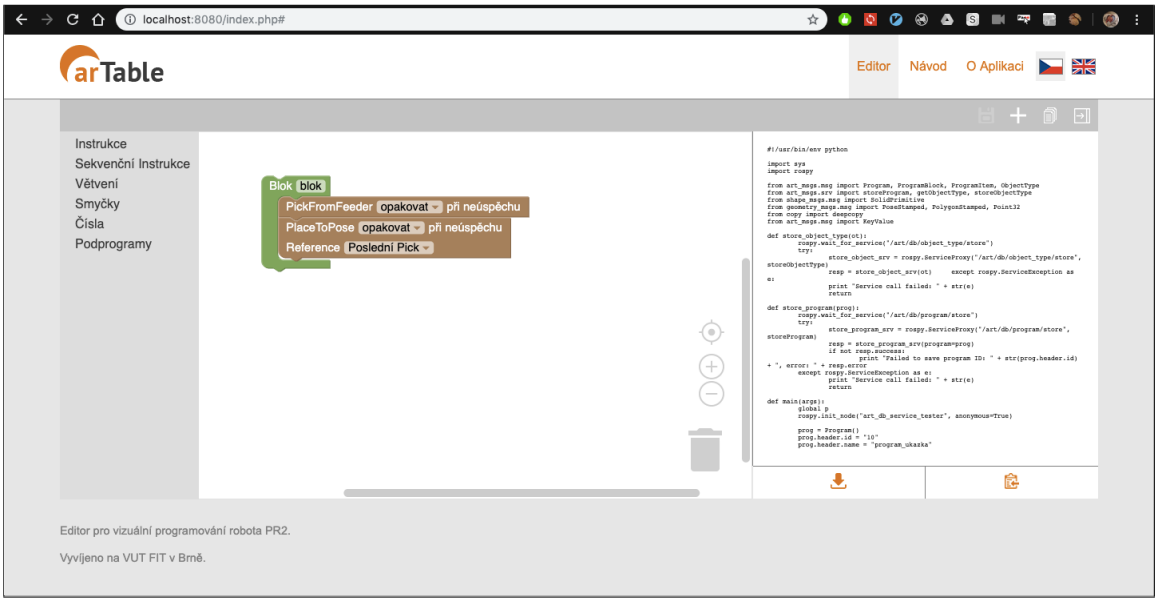
Obrázek 4.3: Nastavení logického objektu v generátoru kódu "and_or"

Čísla

Kategorie Čísla je jedna z nejjednodušších kategorií pro implementaci. Generátor kódu vrací číslo, které zadá uživatel do textového pole. Grafický blok "random_number"vrací náhodné číslo v zadaném rozpětí pomocí JavaScriptové funkce `Math.random`.

4.4 Výsledný vzhled

Po implementaci návrhu vznikla platforma která je viditelná na obrázku 4.4. Na obrázku 4.5 je vidět mobilní verze tohoto editoru. Celkem bylo vytvořeno 13 grafických bloků, které se vkládají do ARTable editoru. Do editoru se poté mohou vkládat další grafické bloky, které reprezentují instrukce z nahrané instrukční sady. Ukládání programů a jejich mazání funguje pouze v případě, že je připojený systém ROS přes plugin ROSBridge. Pokud se spojení nepodaří, ukáže se uživateli informační vyskakovací okénko, které může uživatel zavřít a prohlížet si programy již uložené. Toto okénko se také ukáže, pokud uživatel nemá ve svém prohlížeči zapnutý JavaScript , nebo nemá v prohlížeči povolené cookies. Na obrázku výsledné verze je vidět program, který nastaví robota do přípravné polohy (`GetReady`), vezme jeden objekt z boxu pro kostičky (`PickFromFeeder`) a položí ho na stůl (`PlaceToGrid`).



Obrázek 4.4: Výsledný vzhled desktopové verze

Pro vyzkoušení funkčnosti byla do webového systému nahrána instrukční sada z veřejné webové adresy[3]. V instrukční sadě se nachází 11 instrukcí, které byly uloženy a nahrány do editoru. Poté bylo v robotické laboratoři O104 v areálu školy VUT FIT ustanoveno spojení se systémem ROS. Připojení proběhlo tím způsobem, že jsme se připojili na Wi-Fi systému ARTable a spojili jsme se pomocí frameworku ROSBridge se systémem ROS. Adresa pro připojení byla `ws://192.168.104.200:9090`. Poté jsme do systému nahrány 3 referenční šablony programů a byla vyzkoušena jejich funkčnost. Šablony šli lehce nastavit a nic nenasvědčovalo chybovému chování.

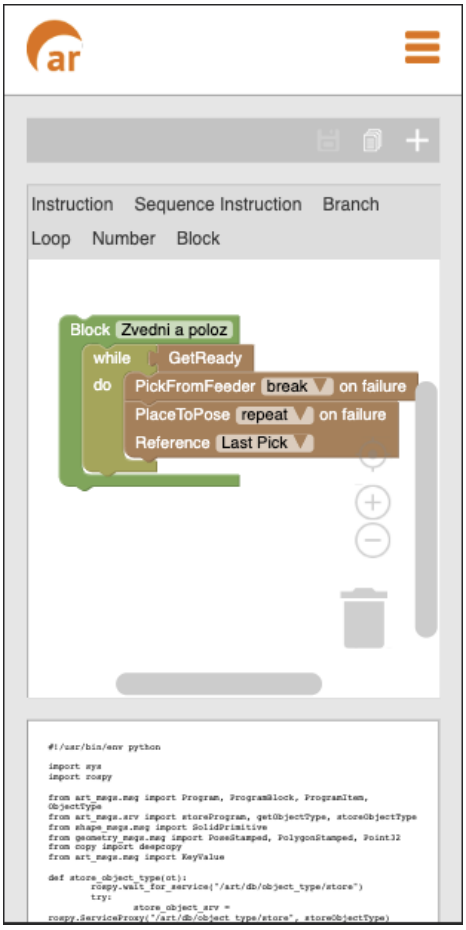
4.5 Testování

Testování se skládalo z celkem tří fází. V první fázi proběhlo zaškolení, v druhé fázi samostatné úkoly, a v poslední fázi vyplnění dotazníku SUS. Celkem bylo otestováno celkem 6 respondentů. Během zaškolení byl respondentovi ukázán systém, vysvětlen princip systému a byli popsány jednotlivé grafické bloky. Na konci byl předveden jednoduchý program, který je uložen pod názvem `program_ukazka`. Toto zaškolení trvalo v průměru 15 minut u každého respondenta. Během samostatných úkolů byl kladen důraz na to, aby respondent pracoval sám a nikdo mu nic nevysvětloval. Cílem samostatných úkolů bylo vytvoření jednoduchých programů v testovaném systému. Programů k vytvoření bylo celkem 5 a jsou uloženy ve složce `/artable/programs/`. U dotazníku byla použita standardní metoda pro měření použitelnosti System Usability Scale (SUS). Samotný dotazník, lze nalézt na webové adrese[4].

V tabulce 4.1 lze vidět výsledky testování u jednotlivých respondentů. Průměrná doba vyplňování úkolů je 21,5 minut. Platí, že čím je člověk méně vzdělaný a čím je více starší, tím je doba vyplňování úkolů vyšší. U dotazníku SUS je průměrná hodnota 70,8 a směro-

[3] https://raw.githubusercontent.com/robofit/arcor/master/art_instructions/config/instructions.yaml

[4] https://www.pouzitelnost.info/2015/06/03/system-usability-scale-cesky/



Obrázek 4.5: Výsledný vzhled mobilní verze

datná odchylka je 17,72. Z tabulky je patrné, že rozdíl mezi mužem a ženou je při používání systému minimální. Hlavními faktory jsou dosažené vzdělání, druh vzdělání a věk. Respondenti s technickým vzděláním vyřešili úkol výrazně rychleji.

Platilo, že čím je respondent starší, tím je méně obeznámen s technickým prostředím obecně a se systémem se mu hůře zaučovalo a pracovalo. To lze vyřešit například delším časem zaškolení a větším množstvím příkladů. Také platí, že čím méně je respondent vzdělaný, především v technické oblasti, tím hůře se mu v systému pracuje. V následujících podkapitolách je popsáno testování jednotlivých respondentů podrobněji.

Respondent A

Respondent A má technické vzdělání, je to velmi mladý muž a studuje informatiku na technické univerzitě. S počítači je tedy velmi dobře obeznámen a je s nimi v každodenním kontaktu. Během zaškolení vše chápal a neptal se na nic zásadního. Tvoření jednotlivých programů mu šlo velmi rychle, vše dělal sám a výsledný úkol měl vždy splněn správně. Na systému nenašel žádné chyby a vše mu něm vyhovovalo.

Respondent	Pohlaví	Věk	Vzdělání	Doba provádění	SUS
A	M	22	Vysokoškolské technické	5 min	85
B	Ž	24	Vysokoškolské humanitní	19 min	50
C	M	24	Vysokoškolské technické	5 min	77.5
D	Ž	67	Střední škola	50 min	42.5
E	M	52	Střední škola	20 min	85
F	Ž	55	Vysokoškolské technické	30 min	85

Tabulka 4.1: Výsledky testování u jednotlivých respondentů

Respondent B

Respondent B je žena s vysokoškolským vzděláním v oboru psychologie. Je mladá a s počítačem je v každodenním kontaktu. Neumí na něm však programovat a vykonává na něm pouze základní uživatelské úlohy. Při zaškolení jsem musel vše důkladně vysvětlit. Samostatné úkoly zvládla udělat sama, ale trvalo jí to trochu déle. V editoru ji zmátlo, že kategorie, kde se nachází podmínky, se nazývá Větvení. V úkolech si také pletla grafické bloky smyček.

Respondent C

Respondent C je muž s technickým vysokoškolským vzděláním v oboru elektrotechniky. S počítačem je v každodenním kontaktu a vykonává na něm i složitější úlohy. Během zaškolení vše chápal. Úkoly plnil správně a velmi rychle. Po konci testování si stěžoval na některé nedostatky systému. Například mu chyběla nápověda k jednotlivým grafickým blokům. Také mu vadilo, že zobáčky u posledních grafických bloků chybí a grafické bloky na stejné úrovni by měli mít jednotnou šířku. Jinak na systému nenašel žádnou další výtku.

Respondent D

Respondent D je žena v důchodovém věku se střední školou a minimálním přístupem k počítači. Během zaškolení téměř nic nechápala a při zadání prvního úkolu jsem jí musel značně pomoci. Projít všechny úkoly bylo velmi obtížné a vysvětlování trvalo velmi dlouho. Nakonec se ke správnému řešení dobrala. U tohoto respondenta bych doporučoval mnohem delší zaškolení a více názorných příkladů.

Respondent E

Respondent E je muž ve středním věku se střední školou. S počítačem je v každodenním kontaktu v práci, kdy do něj zadává faktury a získává z něj informace o zboží. Během zaškolení uváděl, že vše chápe. Během plnění úkolů mu vadili anglické názvy instrukcí. Ty však přeložit do češtiny nejdou. Po ukončení testování poznamenal, že se mu se systémem pracovalo dobře.

Respondent F

Respondent F je žena ve středním věku a vysokou školou zemědělskou. S počítačem je v každodenním kontaktu, kdy hledá v mapách rozlohy pozemků a upravuje formuláře. Během proškolení moc nechápala, k čemu systém slouží. Plnění úkolů bylo pro ni složité. Nechápala, co má vlastně dělat. Bylo nezbytné, stejně jako u respondenta E, vysvětlit anglické názvy. Během plnění úkolů si stěžovala na nevýrazný zobáček, pomocí kterého se přichytávají grafické bloky.

Nedostatky systému

Během testování vyšlo najevo několik nedostatků systému. Starší lidé spíše nerozumí významu systému a je nutno jim podrobně popsat, že slouží ke tvorbě robotických programů. Rozdělení instrukcí na sekvenční a nesekvenční pochopili snadno, ale měli problém s anglickými názvy instrukcí, neboť většina lidí staršího věku neovládá angličtinu. Posledním hlavním problémem byla neexistence nápovědy u jednotlivých grafických bloků. Na tento fakt si stěžovali i technicky zdatní respondenti.

Mezi další nedostatky patří především vizuální nedokonalosti. Jeden z respondentů si stěžoval, že zobáček u grafických bloků je příliš malý a u posledního zanořeného bloku chybí. Kategorie Větvení by se měla přejmenovat na Podmínky, protože název uživatele plete. Pro lepší vizuální vzhled by se grafické bloky na stejné úrovni měly zarovnat do stejné délky. Vykreslování grafických bloku je však záležitostí knihovny Blockly a programátor nemůže vzhled ovlivnit.

Kapitola 5

Závěr

Tato bakalářská práce se zabývala vytvořením webového uživatelského rozhraní pro vizuální programování robota PR2. Pomocí rozhraní lze vytvářet robotické programy pro systém ARTable.

V teoretické části byli popsány webové technologie, uživatelská rozhraní, byl definován vizuální programovací jazyk a nastíněny současné generátory vizuálních programovacích jazyků. Také byla popsána platforma ARTable a ROS. Byl vypsán současný stav platforem pro vizuální programování, a to především JavaScriptová knihovna Blockly vyvíjená firmou Google.

Během návrhu byl vytvořen prototyp webového uživatelského rozhraní společně s grafickými bloky, které byli poté implementovány. Navržené rozhraní umožňuje vytvářet programy pro robotický systém ARTable s využitím podmínek, cyklů, opakování a dalších programátorských konstrukcí.

Pro implementaci rozhraní byla použita technologie `PHP` a `HTML`. K dosažení dynamického obsahu na stránce byla použita technologie `JavaScript`. Z důvodu značného usnadnění práce byla použita knihovna `jQuery` a `Blockly`. Pro uchování programů vytvořených v systému ARTable byl použit formát `XML`. Instrukce byly ukládány ve formátu `JSON`. Žádná databázová technologie nebyla využita.

Rozhraní bylo otestováno s 6 respondenty, kdy každé testování zabralo přibližně 35 minut a skládalo se ze tří fází. V první fázi proběhlo zaškolení, které trvalo přibližně 15 minut. Během zaškolení se respondentům vysvětloval význam systému a jeho funkce. Poté respondenti sami řešili zadané úkoly. Poslední fází bylo vyplnění dotazníku SUS. Výsledkem testování bylo objevení několika nedokonalostí systému.

Na výsledném systému ARTedit lze provádět několik dalších rozšíření. Lze vylepšovat JavaScriptový plugin `ARTedit` o další grafické bloky. Mezi další možnosti patří převod programu uloženém v systému ROS do XML podoby, vytvoření účtů pro jednotlivé uživatele, vytvoření interaktivního robota v prohlížeči (simulátor), lokalizace do dalších jazyků nebo rozšíření vstupních parametrů pluginu `ARTedit`.

Literatura

[1] Visual programming language.
URL https://en.wikipedia.org/wiki/Visual_programming_language

[2] Open Roberta – Spielerisch Programmieren lernen. 2015.
URL https://web.archive.org/web/20151208064554/https:/www.land-der-ideen.de/ausgezeichnete-orte/preistraeger/open-roberta-spielerisch-programmieren-lernen

[3] Internet Usage Statistics. 2018.
URL https://www.internetworldstats.com/stats.htm

[4] World's First Website. 2018.
URL https://www.huffingtonpost.com/2012/08/06/worlds-first-website_n_1747476.html

[5] Beal, V.: High-level language definition.
URL https://www.webopedia.com/TERM/H/high_level_language.html

[6] Bureau, I. T. D.: ITU Developing coutries statistics. Technická zpráva, 2015.
URL http://www.itu.int/en/ITU-D/Statistics/Documents/facts/ICTFactsFigures2015.pdf

[7] Butler, K. L.; Ehsani, M.; Kamath, P.: A Matlab-based modeling and simulation package for electric and hybrid electric vehicle design. *IEEE Transactions on vehicular technology*, ročník 48, č. 6, 1999: s. 1770–1778.

[8] Debbie Stone, M. W. S. M., Caroline Jarrett: *User Interface Design and Evaluation (Interactive Technologies)*. Morgan Kaufmann, 2005, ISBN 0120884364,9780120884360, 3-5 s.
URL
http://gen.lib.rus.ec/book/index.php?md5=59C475120F6511565530398A2F912C15

[9] Dobesova, Z.: Visual programming language in geographic information systems. In *Proceedings of the 2nd international conference on Applied informatics and computing theory*, World Scientific and Engineering Academy and Society (WSEAS), 2011, s. 276–280.

[10] Falk, C.: Exploring the UI Universe: Different Types of UI.
URL https://www.altia.com/2014/09/22/different-types-of-ui/

[11] Flanagan, D.: *JavaScript: The Definitive Guide Activate Your Web Pages*. O'Reilly Media, Inc., 6 vydání, 2011, ISBN 0596805527, 9780596805524, 1-17 s.

[12] Garrett, J.: Ajax: A New Approach to Web Applications. 2005.
URL http://adaptivepath.org/ideas/ajax-new-approach-web-applications/

[13] Google: Blockly Reference.
URL https://developers.google.com/blockly/reference/overview

[14] Grannell, C.: *The Essential Guide to CSS and HTML Web Design*. Essentials, Friends of ED, 2007, ISBN 9781590599075,1590599071, 1-12 s.
URL https://www.amazon.com/Essential-Guide-HTML-Design-Essentials-ebook/dp/B004P8KHX8

[15] Helešicová, A.: Desková hra v rozšířené realitě na dotykovém stole. 2018.
URL http://www.fit.vutbr.cz/study/DP/BP.php?id=21273

[16] Jackson, J. C.: *Web Technologies: A Computer Science Perspective*. První vydání, 2018, ISBN 978-0131856035, 23-44 s.
URL https://www.amazon.com/Web-Technologies-Computer-Science-Perspective/dp/0131856030

[17] Jost, B.; Ketterl, M.; Budde, R.; aj.: Graphical Programming Environments for Educational Robots: Open Roberta - Yet Another One? 2014.
URL https://ieeexplore.ieee.org/document/7033055?tp=&arnumber=7033055

[18] Luke Welling, L. T.: *PHP And MySQL Web Development*. Sams, třetí vydání, 2005, ISBN 9780672326721,0672326728, 2-6 s.
URL
http://gen.lib.rus.ec/book/index.php?md5=308B40249D989CD18D2DFC0785DB1529

[19] Marji, M.: *Learn to Program with Scratch*. No Starch Press, 2014, ISBN 978-1-59327-543-3.
URL https://nostarch.com/learnscratch

[20] Materna, Z.; Kapinus, M.; Beran, V.; aj.: Interactive Spatial Augmented Reality in Collaborative Robot Programming: User Experience Evaluation. In *Robot and Human Interactive Communication (RO-MAN)*, Institute of Electrical and Electronics Engineers, 2018, ISBN 978-1-5386-7980-7, s. 330–338.
URL http://www.fit.vutbr.cz/research/view_pub.php.cs?id=11619

[21] Microsoft: About PXT.
URL https://makecode.com/about

[22] Musienko, Y.: Benefits of Using CSS Preprocessors Like SASS. 2018.
URL
https://dzone.com/articles/benefits-of-using-css-preprocessors-like-sas

[23] Pluralsight: What's the Difference Between the Front-End and Back-End? 2015.
URL https://www.pluralsight.com/blog/film-games/whats-difference-front-end-back-end

[24] Repenning, A.: Moving Beyond Syntax: Lessons from 20 Years of Blocks Programing in AgentSheets. *Journal of Visual Languages and Sentient Systems*, 2017.
URL https://sgd.cs.colorado.edu/wiki/images/2/21/20YearsofBlockProgramingLessonsLearned_published.pdf

[25] Resig, J.: About jQuery.
URL https://api.jquery.com/

[26] Robo@Fit: Introduction to ARTalbe.
URL https://github.com/robofit/artable/blob/master/README.mds

[27] ROS.org: Concepts of ROS.
URL http://wiki.ros.org/ROS/Concepts

[28] ROS.org: Introduction to ROS.
URL http://wiki.ros.org/ROS/Introduction

[29] Staff, L.: HTML5 vs Flash: Things You Should Know.
URL https://blog.liveedu.tv/html5-vs-flash/

[30] Surveys, W. T.: jQuery Usage.
URL https://w3techs.com/technologies/details/js-jquery/all/all

[31] Wikipedia: About Microsoft Robotics Developer Studio.
URL https://en.wikipedia.org/wiki/Microsoft_Robotics_Developer_Studio

[32] Wikipedia: About MiniBloq.
URL https://en.wikipedia.org/wiki/Minibloq

[33] Wikipedia: About Open Roberta.
URL https://en.wikipedia.org/wiki/Open_Roberta

[34] Wikipedia: About Tynker.
URL https://en.wikipedia.org/wiki/Tynker