TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Interaktivní ovládání humanoidních robotů NAO a Pepper přes webové rozhraní

Bakalářská práce

Studijní program: B2612 – Elektrotechnika a informatika
Studijní obor: 2612R011 – Elektronické informační a řídicí systémy

Autor práce: **Václav Kesler**
Vedoucí práce: Ing. Miroslav Holada Ph.D.

Liberec 2021

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

Interactive control of humanoid robots NAO and Pepper via web interface

Bachelor thesis

Study programme: B2612 – Electrical engineering and informatics
Study branch: 2612R011 – Electronic Information and Control Systems

Author: **Václav Kesler**
Supervisor: Ing. Miroslav Holada Ph.D.

Liberec 2021

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Zadání bakalářské práce

Interaktivní ovládání humanoidních robotů NAO a Pepper přes webové rozhraní

Jméno a příjmení: **Václav Kesler**
Osobní číslo: M18000012
Studijní program: B2612 Elektrotechnika a informatika
Studijní obor: Elektronické informační a řídicí systémy
Zadávající katedra: Ústav informačních technologií a elektroniky
Akademický rok: **2020/2021**

Zásady pro vypracování:

1. Seznamte se s humanoidními roboty NAO a Pepper na pracovišti školitele.
2. Proveďte rešerši stávajícího stavu dostupných interaktivních a webových rozhraní pro ovládání robotů.
3. Navrhněte webové rozhraní, pomocí kterého bude možné ovládat robota. Při návrhu zohledněte možnost snadného ovládání i pro nezaškoleného uživatele.
4. Navržené rozhraní realizujte a ověřte jeho funkcionalitu.
5. V závěru diskutujte výhody návrhu a možná bezpečnostní rizika.

Rozsah grafických prací:	Dle potřeby dokumentace
Rozsah pracovní zprávy:	30-40 stran
Forma zpracování práce:	tištěná/elektronická
Jazyk práce:	Čeština

Seznam odborné literatury:

[1] VANER, Pavel: Spolupráce robotů NAO: NAO Robots collaboration. Liberec: Technická univerzita v Liberci, 2018. Bakalářské práce. Technická univerzita v Liberci.
[2] EICHLER, Miroslav: Využití dostupných senzorů robota Nao pro detekci objektů a mapování okolí: Use available Nao robots' sensors to detect objects and map of its surrounding. Liberec: Technická univerzita v Liberci, 2018. Bakalářské práce. Technická univerzita v Liberci.
https://www.softbankrobotics.com
[3] NENCHEV, Dragomir N. a Atsushi KONNO. Humanoid Robots. 1. Berlin, SRN: Elsevier – Health Sciences Division, 2016. ISBN 9780128045602.

Vedoucí práce:	Ing. Miroslav Holada, Ph.D. Ústav informačních technologií a elektroniky
Datum zadání práce:	19. října 2020
Předpokládaný termín odevzdání:	17. května 2021

L.S.

prof. Ing. Zdeněk Plíva, Ph.D.
děkan

prof. Ing. Ondřej Novák, CSc.
vedoucí ústavu

V Liberci dne 19. října 2020

Prohlášení

Prohlašuji, že svou bakalářskou práci jsem vypracoval samostatně jako původní dílo s použitím uvedené literatury a na základě konzultací s vedoucím mé bakalářské práce a konzultantem.

Jsem si vědom toho, 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 nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu Technické univerzity v Liberci.

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

Současně čestně prohlašuji, že text elektronické podoby práce vložený do IS STAG se shoduje s textem tištěné podoby práce.

Beru na vědomí, že má bakalářská práce bude zveřejněna Technickou univerzitou v Liberci v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů.

Jsem si vědom následků, které podle zákona o vysokých školách mohou vyplývat z porušení tohoto prohlášení.

10. 5. 2021 Václav Kesler

Interaktivní ovládání humanoidních robotů NAO a Pepper přes webové rozhraní

Abstrakt

Práce se zabývá problematikou ovládání robotů NAO a Pepper za pomocí webového rozhraní vytvořeném ve frameworku Django. Cílem bylo zhotovit webovou aplikaci, ve které by programování robota probíhalo jednoduchou formou, které by rozuměl i neprogramátor. Byla proto vytvořena aplikace, ve které se programuje za pomoci skládání grafických bloků za sebe, které reprezentují jednotlivé úkony jako například pohyb nebo mluvení. Bloky pro programování mohou obsahovat textové či numerické pole pro zadávání hodnot. Vytváření samotných bloků je omezeno pouze na uživatele s administrátorským oprávněním, který další bloky vytváří v administrátorském rozhraní aplikace a ukládá je do knihovny funkcí, která je k dispozici všem uživatelům. U programátora přidávajícího další bloky do knihoven, je nutná znalost jazyka Python, ve kterém se skripty píší. Aplikace je dostupná pouze administrátorem vytvořeným uživatelům. Pro přístup k webové aplikaci, je nutno se přihlásit.

Klíčová slova: Roboti, NAO, Pepper, Python, Django, NAOqi

Interactive control of humanoid robots NAO and Pepper via web interface

Abstract

This project is about web application for controlling the robots NAO and Pepper. The application was made using web framework Django. The main goal was to make an application for programming robots, which will be understood by a non-programmer. This was the reason, why is programming done by combining graphical blocks. Each graphical block represents some action that will be performed by robot, such as speaking or moving. Blocks can have text fields or numeric fields. Making new block is done via administration interface only by users with administrator privileges. Programmers need to know programming language Python, which is necessary for adding new blocks. The application is available only for users created by administrator. You must be logged in to access the web application.

Keywords: Robots, NAO, Pepper, Python, Django, NAOqi

Poděkování

Rád bych poděkoval vedoucímu práce panu Ing. Miroslavu Holadovi, Ph.D. za zapůjčení robota, a také za připomínky a rady ohledně směřování bakalářské práce. Poděkovat bych také chtěl i své rodině za neochvějnou podporu po dobu celého studia.

Obsah

Seznam obrázků

Seznam zdrojových kódů

Seznam zkratek

HTML	HyperText Markup Language
CSS	Cascading Style Sheets
SDK	Software Development Kit
JS	JavaScript
URL	Uniform Resource Locator
LED	Light Emitting Diode - Světlo emitující dioda
IP	Internet Protocol
AJAX	Asynchronous JavaScript and XML
JSON	JavaScript Object Notation

1 Úvod

1.1 Cíl projektu

Cílem bakalářské práce bylo seznámit se s roboty NAO a Pepper od firmy Softbank Robotics[4] a se současnými možnostmi jejich interaktivního ovládání za pomoci webového rozhraní. Z těchto poznatků poté navrhnout a vytvořit vlastní webovou aplikaci, která by byla vhodná pro ovládání robotů NAO a Pepper. Ovládání aplikace bylo nutné navrhnout s ohledem na nezaškolené uživatele. Důraz při vývoji aplikace byl také kladen na snadnou rozšiřitelnost o další funkce.

Hlavní výhodou webové aplikace je, že není nutné tuto aplikaci instalovat do zařízení a lze ji otevřít na jakémkoliv zařízení s webovým prohlížečem a přístupem na internet. Tato výhoda se stává i současně nevýhodou, protože skrze aplikaci mají přístup k robotům všechny osoby, které znají webovou adresu, na které je aplikace vystavena. Z tohoto důvodu bylo nutné aplikaci zabezpečit proti přístupu neautorizovaných uživatelů.

1.2 Využití aplikace

Využití vytvořené aplikace je především při prezentování naší univerzity a při výuce, kdy má posluchač možnost otevřít si webovou stránku na svém zařízení a bez nutnosti si instalovat specializovaný software může robota ovládat a programovat ho.

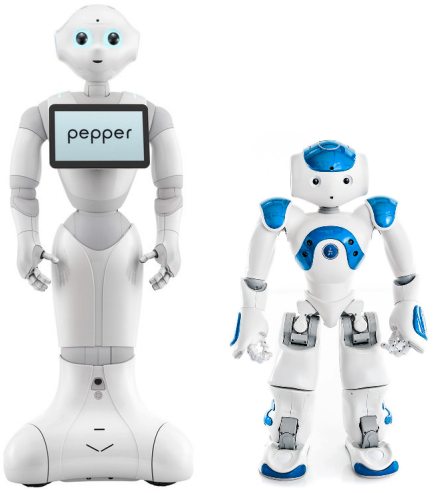
2 Současný stav aplikací pro ovládání robotů NAO a Pepper

2.1 Roboti NAO a Pepper

Roboti NAO a Pepper jsou humanoidní roboti od firmy Softbank Robotics. Tato firma je vedoucí na poli humanoidní robotiky. Kromě robotů NAO a Pepper firma pracuje také na humanoidním robotu s názvem Romeo. Tyto roboti jsou navrženi především pro pomoc lidem a pro výuku. Robot Pepper například sloužil jako recepční v hotelu. Jejich humanoidní podoba pomáhá snadněji navázat sociální interakce s lidmi.

Robot NAO byl prvním robotem firmy. Původně byl NAO vyvíjený francouzskou firmou Aldebran Robotics, kterou v roce 2012 odkoupila japonská společnost Softbank Group a přejmenovala ji na Softbank Robotics.

První generace robota NAO byla uvedena na trh v roce 2008. Robot Pepper byl představen v roce 2014 a již v roce 2015 začaly první prodeje. Oba roboti prošli během let celou řadou vylepšení a úprav. Současná verze robota NAO je verze 6 a pro Pepper je to verze 1.8.[4] Technická Univerzita v Liberci v současné době disponuje dvěma roboty NAO ve verzích 4 a 5 a robotem Pepper ve verzi 1.8.



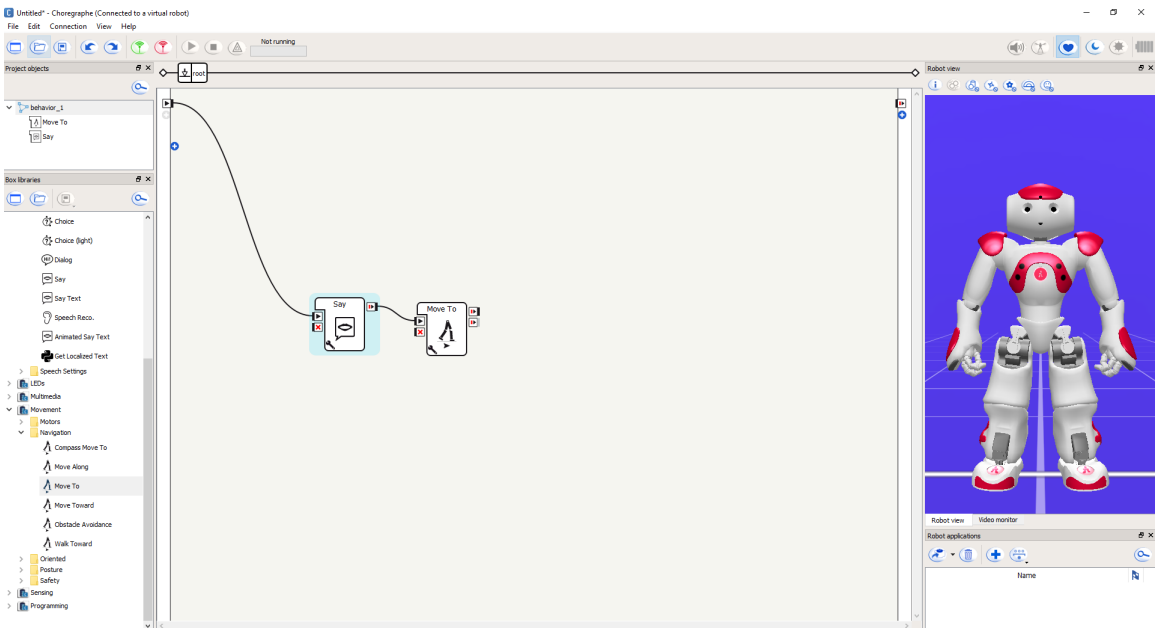
Obrázek 2.1: Pepper (vlevo) a NAO (vpravo) [12]

2.2 Prostředky pro programování

Programátor si může pro programování vybrat z řady nástrojů přímo od výrobce robota. Na výběr je programování pomocí software developer kitu zkráceně SDK a nebo už ve vytvořené aplikaci Choreographe[13]. SDK je balík nástrojů, který může být využit při vývoji aplikace pro určitou platformu nebo operační systém. U programování za pomocí SDK má programátor na výběr z programovacích jazyků Python, C++, Java nebo JavaScript [16]. Pro začátečníky je však nejjednodušší využít aplikace Choreographe (obr. 2.2). Aplikace Choreographe je blíže popsána v kapitole 2.4.1.

V nejnovější verzi softwaru pro robota Pepper je možné programovat robota pouze v programovacím jazyce Kotlin nebo Java v prostředí Android Studio. [17] Aplikace se pak vyvíjí jako klasická aplikace pro zařízení Android a umožňuje tak využít tablet, který je součástí robota Pepper.

Existuje tedy více způsobů jak roboty programovat. Aby bylo možné ovládat oba typy robotů z jednoho rozhraní, je nutné využít verze NAOqi, kterou podporují oba roboti. Problematika verzí frameworku NAOqi je blíže popsána v kapitole 3.5. Při využití jednoho rozhraní pro oba roboty je tedy nutné využít SDK a nikoliv Android Studio, protože NAO Android Studio nepodporuje. Z toho důvodu ztrácí robot Pepper možnost využít tablet.



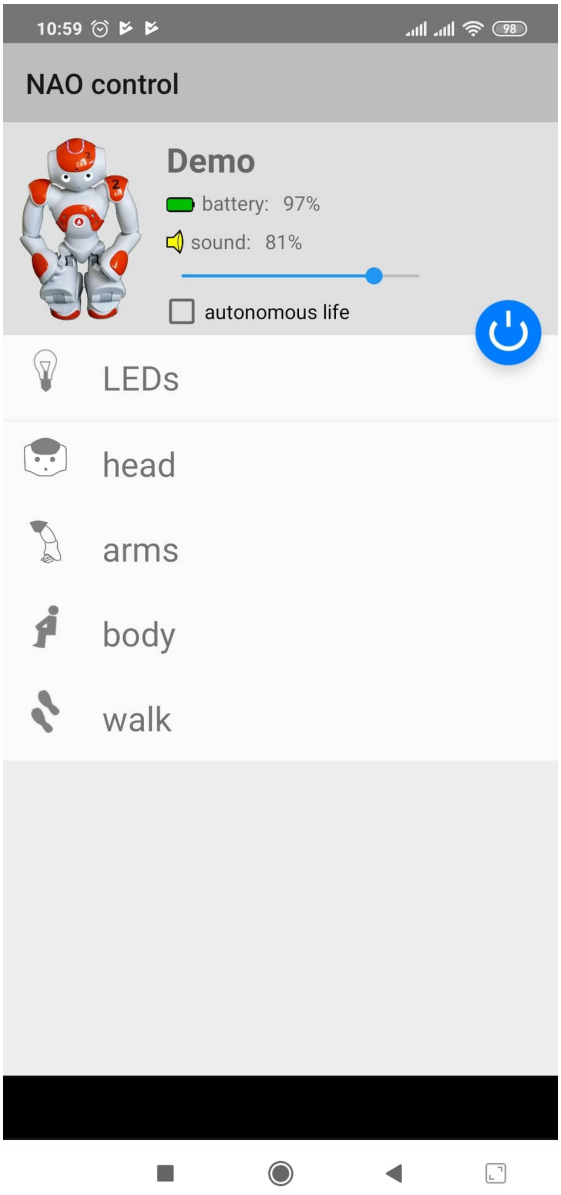
Obrázek 2.2: Rozhraní aplikace Choreographe

2.3 Otestované aplikace

Na internetu je možné nalézt různé aplikace pro ovládání robota NAO. Dvě z nich

jsou dokonce umístěny v obchodu Google Play. Žádná z mnou nalezených aplikací však neumožňovala ovládání robota za pomocí webového rozhraní spuštěného na externím počítači.

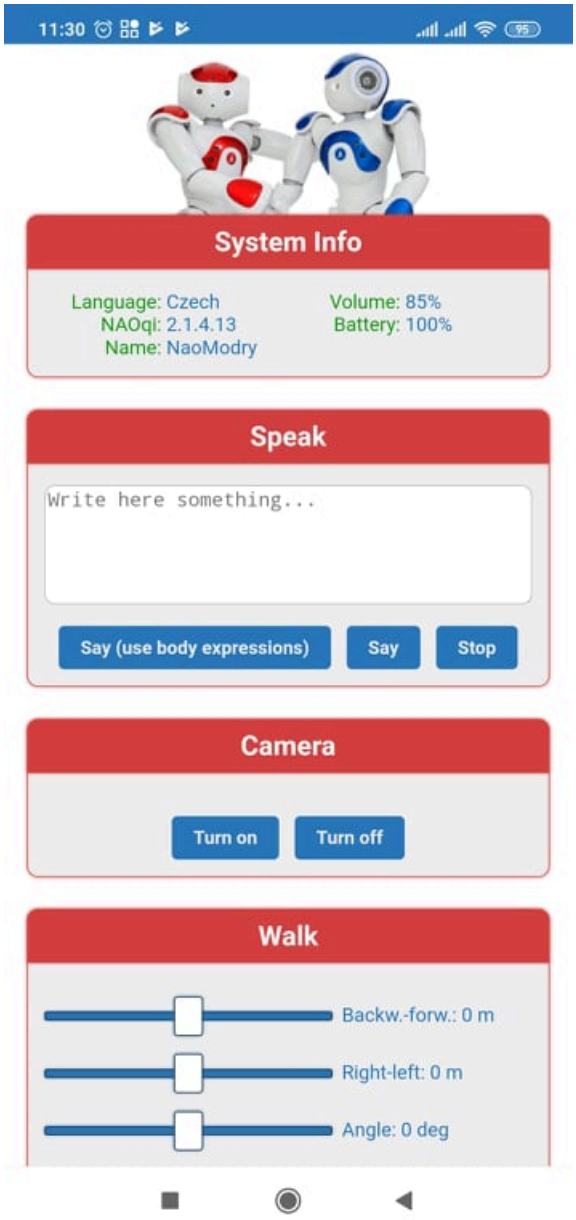
Obě aplikace dostupné na Google Play byly otestovány. První aplikace NAO control [1] umožňuje po přihlášení pomocí IP adresy robota ovládat končetiny, LED diody, nastavovat postoj, chodit, ovládat kameru a mluvit. V horní části aplikace je také možné vidět stav robota a nastavovat jeho hlasitost(obr. 2.3).



Obrázek 2.3: Aplikace NAO control

Druhá aplikace Nao Controller [2] je více pokročilá oproti první zkoušené aplikaci. Umožňuje navíc spouštět interní rutiny pro chování, jako je například smích nebo emoce překvapení. Grafickou podobu aplikace je možné vidět na obrázku 2.4. Obě

aplikace jsou pouze v angličtině. V žádné z těchto aplikací nebylo možné vytvářet sekvence pohybů a událostí, tak jako tomu je možné v aplikaci Choreographe, která je podrobněji popsána v kapitole 2.4.1.



Obrázek 2.4: Aplikace Nao Controller

2.4 Aplikace pro grafické programování robotů

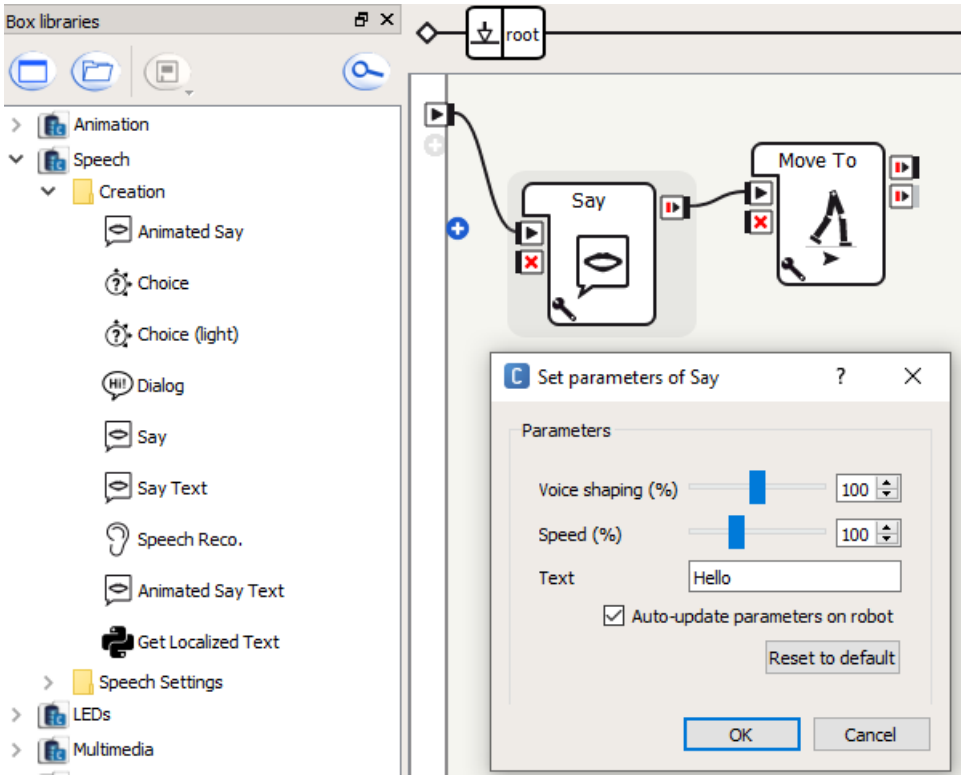
Pro webovou aplikaci bylo stěžejní jednoduché ovládání. Dalším požadavkem byla možnost vytvářet sekvence příkazů a spouštět je najednou. Z těchto důvodů bylo rozhodnuto, že v aplikaci bude probíhat programování pomocí skládání bloků, které

reprezentují kód. Proto byly prozkoumány podobné aplikace řešící problematiku grafického programování robotů.

2.4.1 Prostředí Choreographe

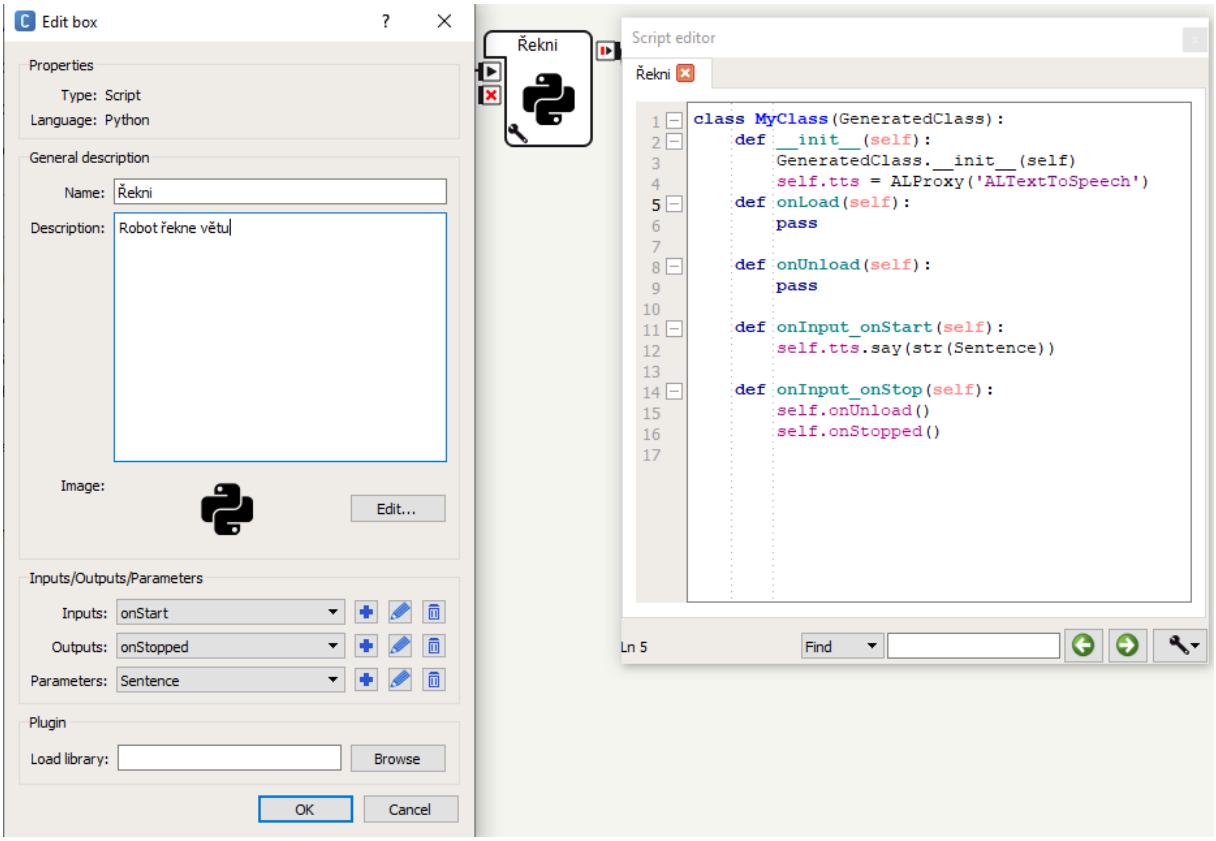
První otestovanou aplikací pro grafické programování robotů je Choreographe[13] od výrobců robota, tedy od firmy Softbank Robotics [4]. Tato aplikace umožňuje programovat roboty NAO a Pepper.

V aplikaci Choreographe probíhá programování pomocí spojování vytvořených bloků k sobě, jak je možné vidět na obrázku číslo 2.5. Jednotlivé předpřipravené bloky jsou přidávány z knihovny, která je k dispozici v levé části aplikace. Bloky je pak možné následně spojovat do sekvencí a vytvářet tak program. Většina bloků obsahuje nějaké parametry, které je možné nastavovat v menu. Toto menu je zobrazeno po kliknutí na ikonu klíče v levém dolním rohu bloku. Na obrázku je zobrazeno menu pro blok "Say", který reprezentuje převod textu na řeč. V dialogovém menu je možné nastavit slovo, které robot řekne, rychlost mluvení a také tón hlasu.



Obrázek 2.5: Vytváření programu v aplikaci Choreographe

Každý blok je reprezentován kódem v jazyce Python. V aplikaci je množné vytvářet i vlastní bloky. Pomocí dialogového okna (obr. 2.7) je zadán název, popis, obrázek, proměnné, vstupy a výstupy. Poté je vytvořen prázdný objekt s již předdefinovanou strukturou v jazyce Python. Pomocí Script editoru, který je také vidět na obrázku, lze předdefinovanou strukturu upravit a vytvořit tak například vlastní blok pro řeč.



Obrázek 2.6: Choreographe - vytváření nového bloku

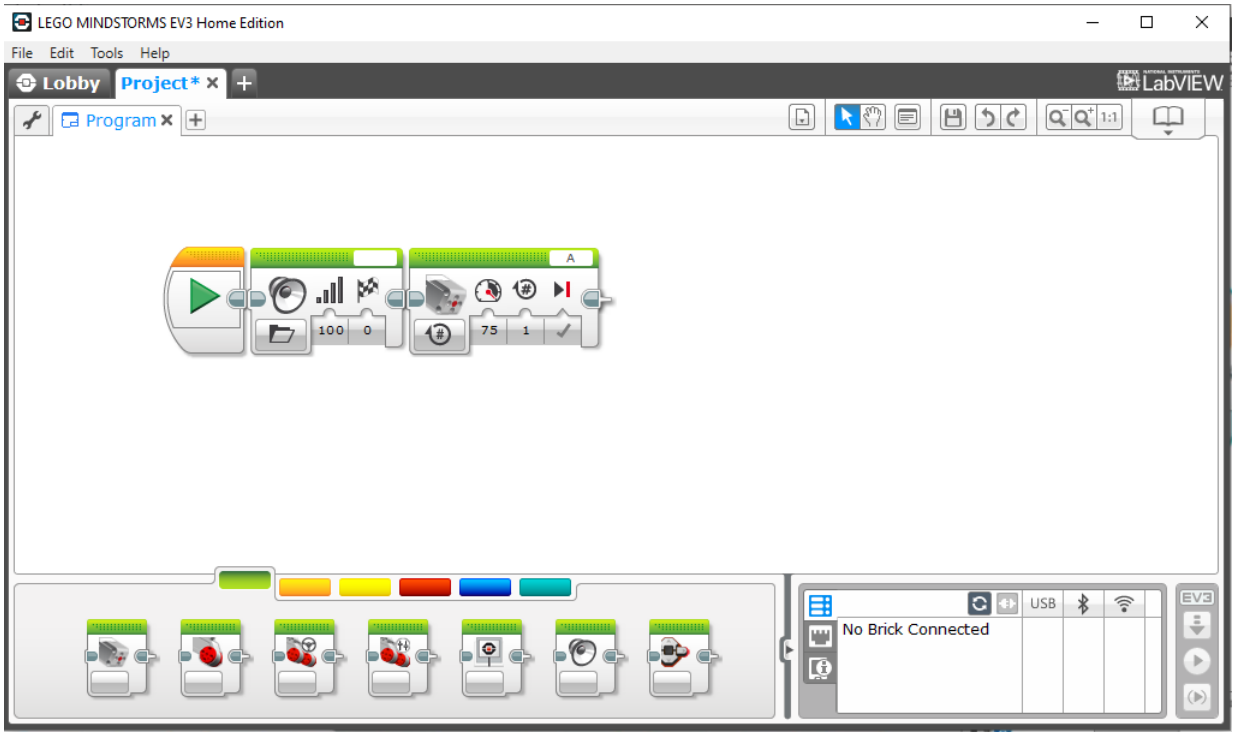
2.4.2 Prostředí Lego EV3 software

Lego EV3 software [6] pro roboty Lego Mindstorms je aplikace umožňující programovat roboty od firmy Lego. Rozhraní EV3 Programmer je možné vidět na obrázku číslo 2.7. Tento software je založen na LabVIEW [18] od firmy National Instruments.

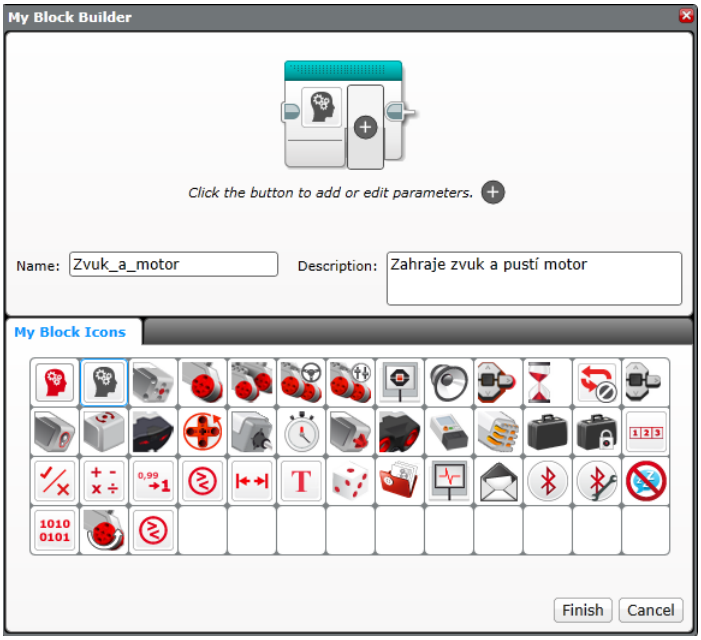
Programování probíhá pomocí skládání bloků za sebe. Jednotlivé bloky reprezentují například motory nebo třeba tlačítka připojená k robotu. Software obsahuje i podmínky a cykly přímo v grafické podobě.

V prostředí Lego EV3 software lze vytvořit vlastní blok z již vložených bloků. Tedy uložit si sekvenci bloků do jednoho bloku. Přetažením myši přes bloky jsou bloky označeny a z těchto bloků je poté vytvořena sekvence. V nabídce ”Tools” v navigačním menu je položka ”My block builder”, která otevírá dialogové okno. Toto okno se nachází na obrázku 2.8. V tomto dialogovém okně je možnost nový blok pojmenovat, napsat mu popis a vybrat obrázek. Do aplikace je také možné přidávat nové bloky pro senzory třetích stran.

K dispozici je i mobilní verze aplikace dostupná na Google Play pro zařízení s operačním systémem Android nebo v App Store pro zařízení s operačním systémem IOS.



Obrázek 2.7: Lego EV3 software



Obrázek 2.8: Lego EV3 software - vytváření vlastního bloku

3 Prostředky pro tvorbu aplikace

3.1 Webové frameworky

Framework neboli aplikační rámec je softwarová struktura, která obsahuje podpůrné programy, knihovny nebo sady knihoven, které dohromady řeší typické problémy dané oblasti. Webové frameworky například řeší problematiku vytváření webových aplikací.

Při výběru webového frameworku bylo rozhodováno mezi dvěma největšími frameworky pro Python a to mezi Djangem[3] a Flaskem[9]. Hlavním rozdílem mezi frameworky je především to, že Django se drží architektury Model-view-controller, což znamená, že datový model aplikace, uživatelské rozhraní a řídicí logika je rozdělena. Modifikace jedné části má minimální vliv na ostatní části. Django tedy cílí na rychlý vývoj, který se zaměřuje na znovupoužitelnost jednotlivých komponentů. Na druhou stranu Flask je klasifikován jako mikro framework, protože nepotřebuje další knihovny nebo nástroje. Flask tedy obsahuje jenom základní nástroje, ale podporuje rozšíření, která mohou přidávat do aplikace další funkce.

3.2 Základní stavební prvky každého webu

Základními stavebními kameny všech webových stránek jsou značkovací jazyk HTML a stylovací jazyk CSS. HyperTextMarkup Language zkráceně HTML je značkovací jazyk používaný pro tvorbu webových stránek. Vychází ze značkovacího jazyka SGML (Standard Generalized Markup Language). Současná verze je HTML5.

CSS neboli Cascading Style Sheets je jazyk, který upravuje způsob zobrazení elementů na stránkách napsaných v HTML. Jazyk CSS byl navržen organizací W3C [5], která ho společně se značkovacím jazykem HTML spravuje. Hlavní využití CSS je oddělení obsahu a struktury od vzhledu dokumentu.

3.3 Framework Bootstrap

Pro rychlejší vývoj webu byl použit framework Bootstrap. [8] Bootstrap je sada nástrojů pro HTML, CSS, a JavaScript. Největší výhodou Bootsrapu je ”flexbox grid”, který umožňuje vytvořit snadné rozložení webových stránek pomocí systému dvanácti sloupců, do kterých je možné vkládat obsah a různě je sdružovat.

Nejnovější verze Bootstrapu je 5.0.0, která je zatím v beta verzi. Nejnovější stabilní verze je 4.6.0. Hlavními rozdíly mezi verzí 5.0.0 a 4.6.0 jsou odstranění závislosti na JQuerry, ukončení podpory Internet Explorer verze 10 a 11 a přidání vlastních vektorových ikon. Pro absenci ikon ve verzi 4.6.0 je doporučené použít například sadu fontů a ikon Font Awesome [10] .

3.4 Skriptovací jazyk JavaScript

JavaScript je objektově orientovaný skriptovací jazyk. V současnosti se používá zejména při tvorbě webových stránek. JavaScriptem jsou pak ovládány různé interaktivní prvky. Používá se zejména pro ovládání komponentů na front endu neboli prezentační vrstvě, která je předkládána uživateli.

3.5 Framework NAOqi

NAOqi je software, který běží na robotovi a ovládá ho. Stejnojmenný framework pak slouží k programování robotů NAO a Pepper. Pomocí SDK umožňuje programovat roboty v programovacích jazycích Python, C++, Java nebo JavaScript.[16]

Starší roboti mají omezenou možnost upgradu NAOqi. Podle stránek výrobce [4] je pro nejnovější verzi NAO V6 k dispozici NAOqi verze 2.8.6, pro starší verzi robotů NAO je doporučená verze NAOqi 2.1. Robot Pepper má doporučenou verzi 2.5. při použití SDK pro Python. Pro programování v Android Studiu je nutné využít QiSDK namísto NAOqi.

4 Realizace

4.1 Vybrané nástroje pro tvorbu aplikace

Při návrhu webové aplikace je brán ohled zejména na jednoduché ovládání. Dalším požadavkem je možnost vytvářet sekvence příkazů a spouštět je najednou. Z těchto důvodů je navrženo grafické rozhraní, ve kterém programování probíhá pomocí skládání bloků za sebe. Podobně tomu je v programech Choreographe nebo EV3 Programmer, které jsou blíže popsány v kapitole 2.4. Aby nebyl uživatel omezený pouze na předpřipravené bloky, je dána uživateli možnost přidávat další bloky pomocí administrátorského rozhraní. V tomto rozhraní se programují jednotlivé bloky stejně, jako při vytváření programu za použití SDK pro Python.

Aby bylo možné zrealizovat navržené rozhraní, jsou vybrány nástroje, pomocí kterých lze webové rozhraní vytvořit. V jednotlivých podkapitolách kapitoly 3 jsou popsané prostředky, které umožňují vytvořit webovou aplikaci.

Prvním krokem je výběr programovacího jazyka. V kapitole 2.1 je výčet všech způsobů jak lze robota programovat. Je vybrán programovací jazyk Python, se kterým mám největší zkušenosti.

Na základě výběru programovacího jazyka je zvolen webový framework. V kapitole 3.1 jsou popsané dva největší webové frameworky pro Python. Z těchto frameworků je vybrán framework Django. Hlavním důvodem výběru Djanga oproti Flasku, je již předchozí zkušenost s frameworkem v semestrální práci. Další z důvodů je to, že Django obsahuje již předpřipravené administrátorské rozhraní a že jeho součástí je i databáze SQLite.

Pomocí Djanga je vytvořen celý server pro webovou aplikaci, který zpracovává veškeré požadavky od uživatele. To znamená, že spravuje databázi a předkládá uživateli předpřipravené HTML šablony nebo položky z databáze.

4.2 Komunikace s robotem

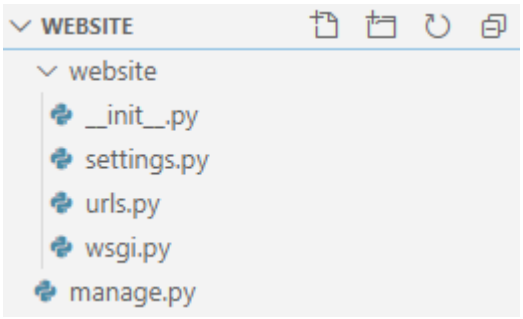
Komunikace mezi webovým serverem a robotem probíhá pomocí frameworku NAOqi. Nejprve je nutné vytvořit proxy server, který je prostředníkem mezi robotem a webovým serverem. Vytvoření první aplikace ”Hello world” pro ověření funkčnosti probíhá následovně - z knihovny NAOqi je naimportován modul ALProxy. Poté je vytvořen zmíněný proxy server, který slouží ke vzdálené komunikaci s robotem. Následně je přistoupeno k metodě ”say”. Tedy je využit předpřipravený převod textu na řeč.

```
1 from naoqi import ALProxy
2 tts = ALProxy("ALTextoToSpeech", "192.168.0.102", 9559)
3 tts.say("Hello, world")
```

Zdrojový kód 4.1: Prvotní aplikace

4.3 Server pomocí frameworku Django

Vytvoření základní struktury projektu ve frameworku Django je následovné. Po zadání příkazu do konzole je vytvořena struktura projektu. Stačí tedy do konzole napsat příkaz ”django-admin startproject website” a framework sám vygeneruje všechny potřebné soubory pro první spuštění vývojového serveru. (obr. 4.1) Při prvotním spuštění vývojového serveru je po přístupu na adresu serveru v prohlížeči pouze zobrazeno, že vše proběhlo v pořádku a server je připraven k dalšímu vývoji.



Obrázek 4.1: Vygenerovaný projekt

Projekt v Django je složený z takzvaných aplikací. Každou aplikaci lze poté použít ve více projektech. Je vytvořena aplikace nazvaná IDE, která je určená pro ovládání robotů. Vytváření aplikací probíhá podobně jako u vytváření projektu. Pomocí příkazu ”python manage.py startapp IDE” framework sám vygeneruje potřebnou strukturu. Dále je nutné doplnit vytvořenou aplikaci do souboru settings.py, aby Django vědělo, že aplikace je vytvořena. V souboru settings.py jsou všechny nastavení ohledně serveru. Každou nově přidanou součást je nutné zde registrovat, aby při spouštění serveru bylo jasné, které všechny součásti se mají načíst, jelikož komponenty v Djangu mohou být využity ve více projektech.

Dalším krokem je vytvoření složky pro takzvané statické soubory. Statické soubory jsou takové soubory jejichž obsah je předán koncovému uživateli bez nutnosti cokoliv upravovat generovat nebo zpracovávat. Statické soubory mohou být jednoduše uloženy do mezipaměti prohlížeče. Jelikož se tyto soubory nemění, není potřeba je při každém přístupu na webovou stránku znovu načítat přímo ze serveru, tudíž je potom přístup na webovou stránku rychlejší. Ve vytvořené složce jsou vytvořeny podsložky pro HTML, CSS a JS soubory a tzv. favicon. Favicon je ikona zobrazovaná na okraji záložky v prohlížeči. Vytvořená složka je také registrována do souboru settings.py, aby s ní Django mohlo pracovat.

4.4 Grafická podoba webu

S již připravenou strukturou projektu, je dalším logickým krokem vytvořit grafický návrh aplikace. Pro návrh rozložení je využito knihovny Bootstrap, která je vložena mezi statické soubory. Bootstrap je nejpopulárnější CSS Framework pro vytváření responzivních webových stránek, jak bylo popsáno v kapitole 3.4. Bootstrap umožňuje vytvořit si kontejner a rozdělit ho na dvanáct buněk na každý řádek. V levé části stránky je vytvořena sekce ”menu” která zaujímá dvě buňky. Zbylých deset buněk zaujímá sekce ”container”, která slouží pro zobrazování boxů a pro samotnou práci s nimi. Sekce menu obsahuje podsekci na zobrazení knihoven a dvě tlačítka. Horní tlačítko je určeno pro zastavování programu a dolní tlačítko pro ukládání sekvencí.

```
<div class="container-fluid h-100">
    <div class="row justify-content-center h-100">
        <div class="col-2 h-100" id = "sidebar">
            <div class="row" id = "siz" style="height:5%">
                <div class="col-12" id = "nav">Menu</div>
            </div>
            <div class="row" style="height:85%">
                <ul id= "Lib"></ul>
            </div>
            <div class="row"  style="height:10%">
               <div class="col-12">
               <button onclick="Stop();">
                  Zastavit program
               </button>
               <button data-toggle="modal"data-target="#SaveModal">
                  Ulozit sekvenci
               </button>
               </div>
            </div>
        </div>
        <div id = "container" class ="col-10 h-100">
        </div>
    </div>
</div>
```

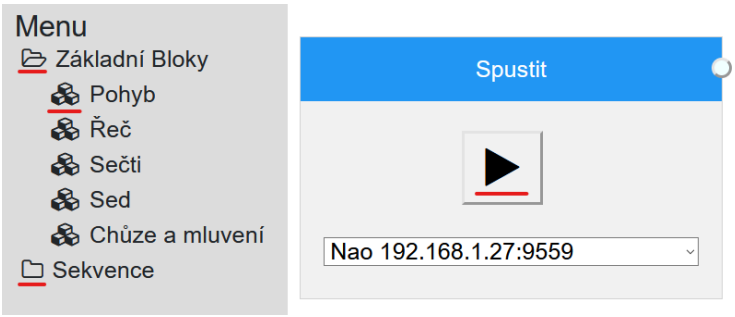
Zdrojový kód 4.2: Rozvržení v HTML

Obrázek 4.2: Ikona webové stránky - favicon

V rámci grafické prezentace webu je vytvořen favicon (obr. 4.2), který je zobrazen na okraji záložky v prohlížeči. Favicon je vytvořen na webové stránce favicon.io[11].

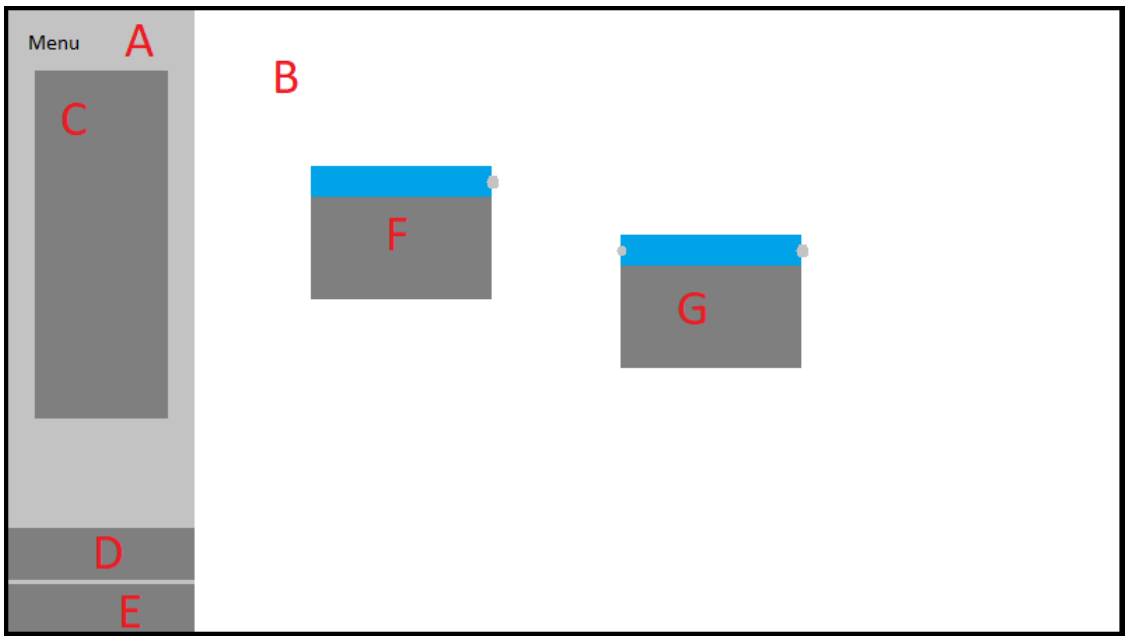
Pro nápis ”NAO” na ikoně je využit stejný font jako v celém projektu a to font ”Lato”. Ikona má rozměry 48 na 48 pixelů.

Součástí webové aplikace jsou i čtyři ikony a to zavřená a otevřená složka, boxy a trojúhelník na spouštěcím tlačítku. Tyto ikony jsou z knihovny Font Awesome. [10] Ikony je možné vidět na obrázku číslo 4.3, kde jsou zvýrazněny červenou barvou.



Obrázek 4.3: Ikony využité na webu

Na obrázku číslo (obr. 4.4) je zobrazen návrh rozložení webového rozhraní. Písmeno A na obrázku označuje sekci ”menu”. Písmeno B na obrázku označuje sekci ”container”. Na místě kde je písmeno C je sekce, kde se zobrazují knihovny a bloky. D je tlačítko na zastavení programu. Pod písmenem E je tlačítko na uložení vytvořené sekvence do databáze. Blok označený písmenem F je spouštěcí blok. Pod písmenem G je blok vytvořený z databáze.



Obrázek 4.4: Návrh rozložení webu

4.5 Využití databáze

Aby stavební bloky, za pomocí kterých je robot programován, byly trvale zachované, je nutné uložit je do databáze. Nejprve jsou vytvořeny databázové modely, které fungují jako vzor pro vytváření databázových položek. Jako první model je vytvořen model knihovny. Každá knihovna má pouze svoje jméno a metodu ___unicode___. Návratová hodnota metody ___unicode___je jméno knihovny. Tato metoda je použita místo metody _str___kvůli podpoře českých znaků v databázi, jelikož metoda _str___pracuje s kódováním ASCII a metoda ___unicode___s kódováním UTF-8. Ukázku modelu knihovny je možné vidět ve Zdrojovém kódu 4.3.

```
1 class Lib(models.Model):
2     name = models.CharField(max_length = 50)
3     def __unicode__(self):
4         return u'%s' % self.name
```

Zdrojový kód 4.3: Model knihovny

Dalším vytvořeným modelem je model bloku. Blok obsahuje jméno, kód který bude vykonán při použití bloku, alternativní text a knihovnu do které je blok přiřazen. Vztah mezi bloky a knihovnou je typu ”mnoho na jednu” z anglického ”many-to-one”. To znamená, že k jedné knihovně může být přiřazeno více bloků. Spojení mezi knihovnou a blokem je vytvořeno pomocí takzvaného ”ForeignKey”, který je vlastně referencí na příslušnou přiřazenou knihovnu. Při smazání knihovny jsou smazány i bloky, které knihovna obsahuje. Podobně jako model knihovny obsahuje i model bloku metodu ___unicode___, která navíc vrací i jméno knihovny, ke které je blok přiřazen. Ukázku modelu bloku je možné vidět ve Zdrojovém kódu 4.4.

```
1 class Block(models.Model):
2     name = models.CharField(max_length = 50)
3     code = models.TextField('code')
4     alternative = models.CharField(max_length = 300,default='')
5     lib = models.ForeignKey(Lib, on_delete=models.CASCADE,default=''
          )
6     def __unicode__(self):
7         return u'%s : %s' % (self.lib.name, self.name)
```

Zdrojový kód 4.4: Model bloku

Aby uživatel mohl ovlivnit program, je potřeba vytvořit modely pro zadávací pole. Celkem jsou v projektu vytvořeny tři typy zadávacího pole a to pole číselné,textové a posuvník. Posuvník je jenom jinou grafickou reprezentací číselného pole. Každé z těchto polí obsahuje jméno, referenci na blok, ke kterému je pole přiřazeno a metodu ___unicode ___. Vztah mezi poli a blokem je typu ”mnoho na jeden” tak jako v případě knihovny a bloku. Návratová hodnota této metody je jméno pole a jméno bloku, ke kterému je pole přiřazeno. Číselné pole a posuvník navíc obsahují

horní a dolní mez a také výchozí hodnotu, která je zobrazena při vytvoření bloku. Ukázku modelu posuvníku je možné vidět ve Zdrojovém kódu 4.5.

```
1 class Slider(models.Model):
2     name =  models.CharField(max_length = 50, default='')
3     minimum = models.IntegerField()
4     maximum = models.IntegerField()
5     default = models.IntegerField()
6     block = models.ForeignKey(Block, on_delete=models.CASCADE)
7     def __unicode__(self):
8         return u'%s : %s' % (self.block.name, self.name)
```

Zdrojový kód 4.5: Model posuvníku

Aby vytvořené modely byly dostupné v databázi, je do konzole zadán příkaz ”python manage.py makemigrations”, který vytváří takzvané nové migrace na základě změn provedených v modelech. V tomto případě jde o vytvoření nových modelů. Poté je zadán příkaz ”python manage.py migrate”, který aplikuje nové migrace do databáze. Aby bylo možné vytvářet nové položky v databázi na základě vytvořených modelů z administrátorského rozhraní, jsou tyto modely registrovány v souboru admin.py. Ukázka registrace modelů do administrátorského rozhraní je zobrazena ve Zdrojovém kódu číslo 4.6.

```
1 from django.contrib import admin
2 from .models import Lib,Block,Slider,ChField,NuField
3
4 admin.site.register(Lib)
5 admin.site.register(Block)
6 admin.site.register(Slider)
7 admin.site.register(ChField)
8 admin.site.register(NuField)
```

Zdrojový kód 4.6: Registrace modelů v souboru admin.py

4.6 Použití skriptovacího jazyka JavaScript

Vytváření programů pro robota, je navrženo jako skládání bloků za sebe. Pro manipulaci s bloky vznikly dva soubory ”box.js” a ”drag.js”. V souboru ”drag.js” jsou umístěny všechny funkce potřebné pro pohyb s bloky a v souboru ”box.js” jsou obsaženy potřebné funkce pro vytváření, spojování a rozpojování bloků.

4.6.1 Vytváření bloků

Po kliknutí na vybraný blok v menu je zavolána funkce pro vytváření bloků. Vstupními parametry funkce jsou jméno bloku a alternativní text. Oba tyto údaje jsou

převzaty z tlačítka, které funkci volalo. Funkce vytvoří oddíl, který se dále dělí na oddíl pro nadpis a oddíl pro ovládací prvky. Součástí oddílu pro nadpis jsou připojovací tlačítka a také samozřejmě samotný nadpis. Hlavnímu oddílu je pak přiřazen alternativní text, který se zobrazí, když uživatel chvíli nechá myš na vytvořeném bloku.

Pro vytvoření potřebných ovládacích prvků je vyslán asynchronní dotaz na server takzvaný AJAX. Pokud vybraný blok má v databázi přiřazené nějaké ovládací prvky, server vrátí JSON odpověď, která obsahuje data o ovládacích prvcích, které se mají vytvořit. Na základě dat získaných ze serveru, jsou zavolány funkce pro vytváření jednotlivých ovládacích prvků. V zdrojovém kódu číslo 4.7 je možné vidět funkci, která vytváří posuvníky. Nejprve je vytvořen oddíl "slider_div", do kterého jsou postupně přidávány další prvky. Ovládací prvek posuvník je tedy složen z textu, který reprezentuje hodnotu na posuvníku a ze samotného posuvníku, který má nastavenou horní a dolní mez a také výchozí hodnotu. Meze a výchozí hodnota posuvníku jsou získány z JSON dat, které vrátil server.

```
1  function Slider(Slider) {
2      var slider_div= document.createElement('div')
3      var text = document.createElement('span')
4      text.innerHTML = Slider.fields.default
5      slider_div.appendChild(document.createTextNode(
           Slider.fields.name));
6      slider_div.appendChild(document.createElement('br'));
7      slider_div.appendChild(text)
8      var range = document.createElement("input")
9      range.setAttribute("type", "range");
10     range.setAttribute("min", Slider.fields.minimum);
11     range.setAttribute("max",  Slider.fields.maximum);
12     range.value = Slider.fields.default;
13     range.setAttribute("oninput", "updateTextInput(this.value,this)
           ;");
14     range.classList.add("form-control-range")
15     slider_div.appendChild(range)
16     spaceholder.appendChild(slider_div)
17 }
```

Zdrojový kód 4.7: Funkce pro vytváření posuvníků

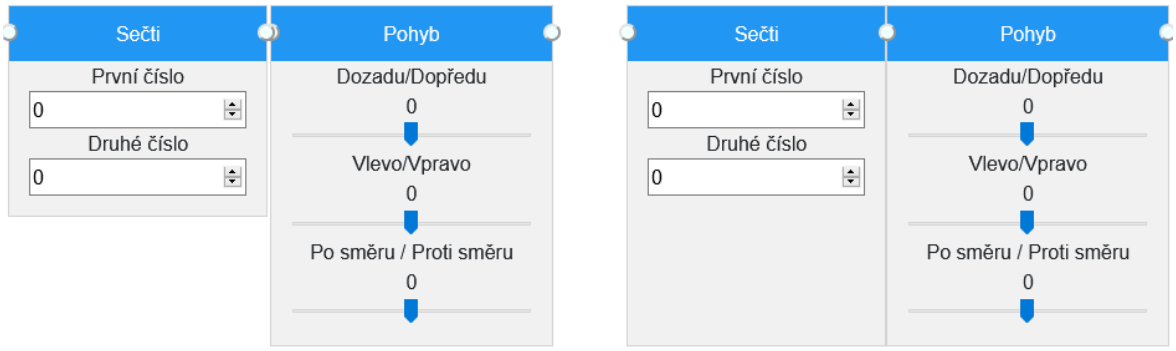
4.6.2 Pohyb bloků

Funkce pro pohybování bloku jsou umístěny do souboru "drag.js". První funkce, která je přiřazena každému bloku se nazývá "dragElement". Vstupním parametrem této funkce je element, který tuto funkci volá. Tato funkce přiřadí funkci pro pohyb bloku do oddílu s nadpisem. To znamená, že se blokem dá posouvat pouze za nadpis.

Samotná funkce pro pohyb je volána po kliknutí na nadpis. Následným posouváním myši po obrazovce je volána funkce "Drag", která vypočítává novou pozici bloku na základě pozice myši. Součástí funkce "Drag" je i ochrana proti pohybu

mimo zakázanou oblast a změna barvy mazacího pole, pokud se v něm blok vyskytuje. Jakmile uživatel pouští tlačítko myši, je zavolána funkce ”CloseDrag”, která korektně ukončuje posouvání.

Tato funkce také volá další funkci, která se nazývá ”combine” a tato funkce kontroluje, jestli se nevyskytuje posouvaný blok v okolí dalšího bloku, se kterým by se měl spojit. Aby se blok připojil k dalšímu bloku nebo ke skupině bloků, musí se levé tlačítko pro připojení posouvaného bloku nacházet v okolí deseti pixelů od pravého tlačítka pro připojení statického bloku nebo obráceně. Pokud tato skutečnost nastane je volána funkce ”combine_elements”, jejíž vstupními parametry jsou bloky, které se mají spojit. Na obrázku číslo 4.5 je možně vidět v levé části situaci těsně před puštěním levého tlačítka myši. Oba bloky ještě nejsou spojeny. V pravé části obrázku jsou již oba bloky spojeny, protože uživatel již pustil levé tlačítko myši.



Obrázek 4.5: Spojování dvou bloků

4.6.3 Vytváření skupin bloků

Všechny vytvořené bloky lze spojovat do skupin. Funkce ”combine_elements”, která spojí bloky, je zavolána, pokud je blok v okolí dalšího bloku nebo skupiny bloků, jak bylo popsáno v předchozí podkapitole 4.6.2. Tato funkce pak rozlišuje, zda je spojována skupina bloků se samostatným blokem, samostatný blok se skupinou bloků, skupina bloků se skupinou bloků nebo dva samostatné bloky.

Nejjednodušší případ je spojení dvou samostatných bloků. Nejprve je vytvořen oddíl, který má třídu ”combine”. Do tohoto oddílu jsou vloženy oba bloky. Dále je změněno pozicování z absolutního na relativní vzhledem k oddílu ”combine”. Pravé spojovací tlačítko levého bloku je zneviditelněno a do levého tlačítka pravého bloku je přiřazena funkce na rozpojování bloků, která se nazývá ”decombine_elements”. Dále jsou obě tlačítka odebrána z ”containeru”, který je pracovní plochou, ve které jsou umístěny všechny bloky. Nakonec je přidán oddíl ”combine” do ”containeru”.

U ostatních variant spojování je nutné dát si pozor, ze které strany pole vložit vkládaný blok. V případě spojování skupin bloků k sobě, skupina bloků z pravé strany zaniká. Všechny bloky, které obsahovala skupina bloků na pravé straně jsou postupně připojeny do levé skupiny z pravé strany.

Rozpojování bloků je inverzní operace ke spojování bloků. Ve funkci ”decombine_elements”, která odpovídá za rozpojování bloků od sebe, jsou rozlišené tři stavy - stav kdy se odpojuje první blok od skupiny bloků, stav kdy se odpojuje poslední blok od skupiny bloků a stav kdy se rozpadá skupina na dvě menší skupiny. V prvních dvou případech se pouze odpojí blok a přidá se do ”containeru”. V případě kdy se skupina rozpadá na dvě menší skupiny, je potřeba vytvořit novou skupinu, do které se vloží všechny bloky, které byly na pravé straně včetně bloku, který rozpojování vyvolal. Následně je nová skupina vložena do ”containeru” s posunem patnáct pixelů vpravo od posledního bloku druhé skupiny.

4.7 Ovládání robota pomocí webového rozhraní

V podkapitole 4.2 je objasněn princip ovládání robota za pomocí SDK. Tento princip je v této podkapitole dále rozvinut a je zde popsáno, jak přesně probíhá ovládání z webového rozhraní od okamžiku kliknutí na spouštěcí tlačítko až do okamžiku, kdy je akce robotem vykonána.

Po kliknutí na spouštěcí tlačítko je zavolána funkce ”Run_send”, která se nachází v souboru ”domload.js”. Tato funkce zjistí, zda je spouštěcí tlačítko spojeno s nějakým dalším blokem. Pokud je tato podmínka splněna, je volána další funkce ”PrepareData”, která vrací připravená data a ta jsou následně odeslána na server AJAXem metodou POST spolu s informací, na kterém robotu se má funkce spustit. Robot je vybírán pomocí rozbalovacího menu, kde jsou zobrazeni všichni roboti uložení v databázi. Funkce ”Run_send” dále vypíná možnost kliknout na spouštěcí tlačítko a čeká do doby, dokud není požadavek robotem vykonán a následně zase spouštěcí tlačítko zpřístupňuje dalšímu kliknutí.

```
1  function PrepareData(){
2    var dta = "";
3    var Run_button = document.getElementById("Run")
4    var children = Array.prototype.slice.call(
         Run_button.parentElement.children)
5    children.shift();
6    children.forEach(element => { //Projdou se elementy
7      dta = dta.concat(element.id)
8      dta = dta.concat(":")
9      var childrenofelement = Array.prototype.slice.call(
           element.children[1].children)
10     childrenofelement.forEach(elmnt => {
11       dta = dta.concat(elmnt.childNodes[0].nodeValue)
12       dta = dta.concat("_")
13       dta = dta.concat(elmnt.querySelectorAll("input")[0].value)
14       dta = dta.concat("^")});
15     dta = dta.concat(";")
16   });
17   return dta}
```

Zdrojový kód 4.8: Funkce pro přípravu dat

Funkce ”PrepareData”, kterou je možné vidět ve zdrojovém kódu číslo 4.8, připravuje data k odeslání na server, jak bylo popsánu výše. Data jsou textovým řetězcem, do kterého je postupně zapisováno jméno bloku, jména ovládacích prvků a jejich hodnota ve formátu:

”Jménobloku:ovládacíprvek1_hodnotaovládacíhoprvku1^;”

Bloky jsou tedy mezi sebou odděleny středníkem. Jméno bloku je odděleno od ovládacích prvků dvojtečkou. Jméno ovládacího prvku a jeho hodnota je od sebe rozdělena podtržítkem a ovládací prvky jsou od sebe odděleny stříškou.

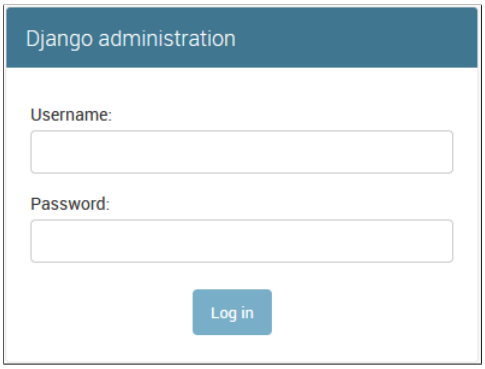
Na serveru je nejprve inicializována instance třídy robot, která zpřístupní všechny proxy servery pro komunikaci a dá je k dispozici uživateli. Inicializace proxy serverů je zabalena do try bloku, který ošetřuje případnou nemožnost vytvořit proxy server. Třída robot je ve zdrojovém kódu číslo 4.9. Poté jsou data, která byla odeslána na server postupně zpracovávána. V databázi má každý blok k sobě přiřazenou databázovou položku ”code” viz. zdrojový kód číslo 4.4. Kód zapsaný v tomto textovém poli je poté spouštěn na serveru. Pro každý blok je tedy v databázi nalezen kód, který je dále upraven. V kódu jsou nahrazeny zástupné znaky daty z ovládacích prvků. Zástupný znak pro ovládací prvek je ”$r”, za kterým následuje číslo ovládacího prvku, ze kterého mají být data použita. Například ”$r3” může být nahrazeno číslem 5, které je nastavené na posuvníku, který je třetím ovládacím prvkem. Takto připravený textový řetězec je pak spuštěn pomocí Python příkazu ”exec”, který je zabalen do try bloku, aby při případné chybě nespadl server. Všechny bloky jsou postupně zpracovávány a nakonec server vrátí uživateli HTML odpověď. Na základě této odpovědi jsou odblokovány tlačítka, která uživateli umožňují opětovné spuštění sekvence.

```
1  class Robot():
2      def __init__(self, ip, port, name):
3          ip = unicodedata.normalize('NFKD', ip).encode('ascii', '
               ignore')
4          PORT = int(port)
5          self.name = name
6          try:
7              self.tts = ALProxy("ALTextToSpeech", ip, PORT)
8              self.pos = ALProxy("ALRobotPosture", ip, PORT)
9              self.mot = ALProxy("ALMotion", ip, PORT)
10             self.memory = ALProxy("ALMemory",ip, PORT)
11             self.sonar = ALProxy("ALSonar", ip , PORT)
12             self.motion = ALProxy("ALMotion", ip, PORT)
13             self.beh = ALProxy("ALBehaviorManager", ip, PORT)
14         except Exception, e:
15             print "Could not init robot"
16             print "Error was: ", e
```

Zdrojový kód 4.9: Třída Robot

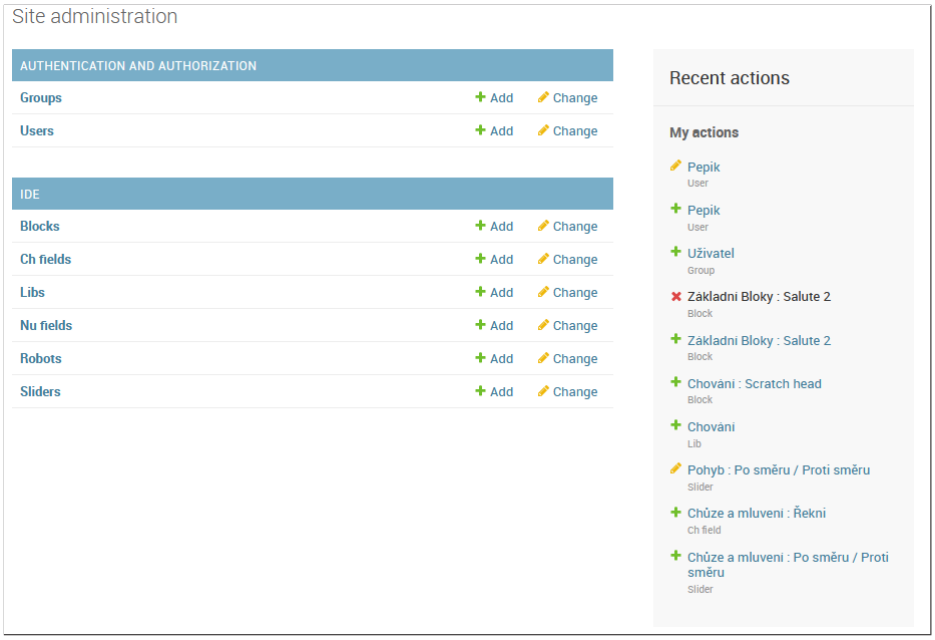
4.8 Práce s administrátorským panelem

Administrátorský panel se nachází na adrese ”webový server/admin”. Prvním krokem je přihlášení. Přihlašovací okno se nachází na obrázku číslo 4.6 Zde uživatel zadá svoje přihlašovací údaje a po přihlášení se mu zobrazí již zmíněný administrátorský panel.



Obrázek 4.6: Přihlašovací okno administrátorského panelu

Uživatelé jsou rozdělení do tří skupin. První skupinou jsou administrátoři. Administrátor může spravovat všechny databáze, to znamená vytvářet nové bloky a vkládat do nich ovládací prvky a také může přidávat a odebírat nové uživatele. Další skupinou jsou programátoři. Ti na rozdíl od administrátorů nemohou přidávat ani odebírat uživatele. Poslední skupinou jsou obyčejní uživatelé, kteří přihlášením získají pouze přístup k webovému rozhraní.

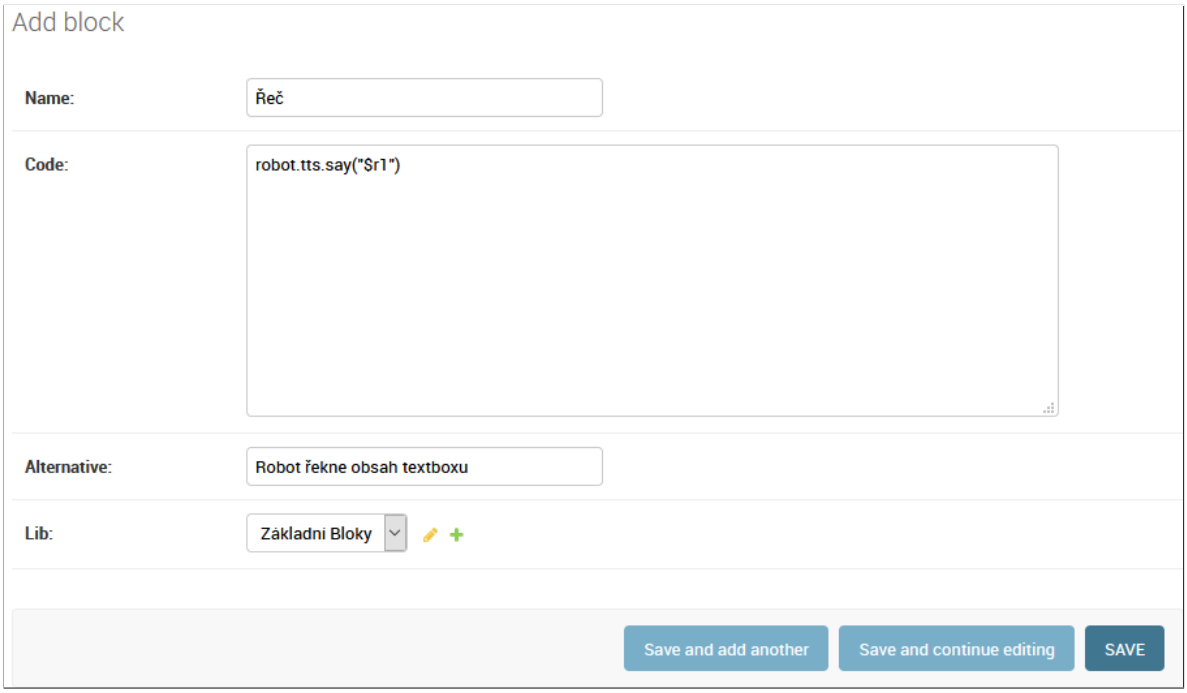


Obrázek 4.7: Administrátorský panel - přihlášení jako administrátor

Každý uživatel, aby mohl využívat jakoukoliv funkci webového rozhraní, musí být nejprve přihlášen do administrátorského panelu. Tímto způsobem je zajištěna ochrana proti neautorizovaným uživatelům. Tudíž bez přihlášení není možno přistupovat k administrátorskému panelu ani do webového rozhraní pro vytváření sekvencí a tím pádem neautorizovaný uživatel nemá přístup k robotům.

Na obrázku číslo 4.7 se nachází administrátorské rozhraní po přihlášení jako administrátor. V levé části se nachází databázové prvky, které může administrátor upravovat a v pravé části je historie akcí, které byly nedávno provedeny.

Pro přidání nového bloku je kliknuto na "Add" vedle položky "Blocks". Toto otevře rozhraní, které je zobrazené na obrázku 4.8. Na tomto obrázku je přidáván blok pro řeč. Je nutné vyplnit všechny položky a vybrat knihovnu, do které bude blok přiřazen. V textovém poli "Code" je kód jako při běžném programování za pomocí SDK. Je vynechána inicializace proxy serverů, protože o to se již stará server, jak bylo popsáno v podkapitole 4.6. Je zde vidět použití zástupného znaku pro ovládací prvek "$r1". Po uložení je ještě nutné vytvořit ovládací prvek a přiřadit ho ke správnému bloku. V tomto případě tedy vytvoření textového pole a přiřazení k vytvořenému bloku "Řeč". V případě, že by nebyl ovládací prvek přiřazen, server nemůže nahradit hodnotu "$r1" a proto by robot řekl "$r1". Při přiřazení více ovládacích prvků do bloku a jejich následnému nevyužití, dochází pouze k zmatení uživatele, protože data získaná z ovládacích prvků nejsou využita a nemají vliv na chod programu.

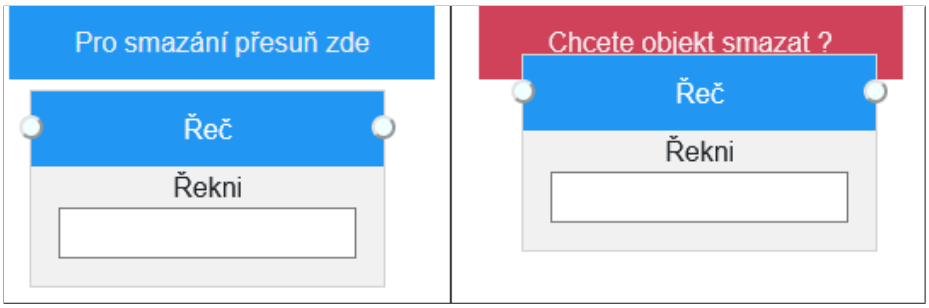


Obrázek 4.8: Přidávání bloku pro řeč

4.9 Další funkce webového rozhraní

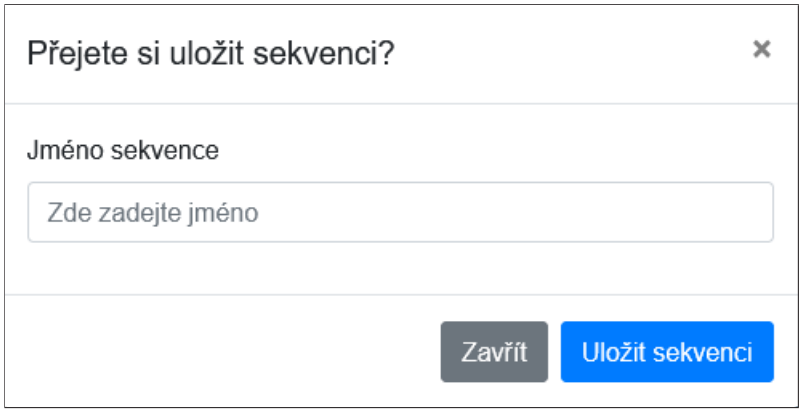
Mezi další funkce, které nebyly zmíněny v předchozích kapitolách, patří mazání bloků, zastavení vykonávaného programu a ukládání sekvencí.

Mazání bloků je vyřešeno pomocí oblasti, která se zobrazí při posouvání jakéhokoliv bloku nebo skupiny bloků. Po přesunutí bloku do tohoto pole se změní jeho barva, jak je možné vidět na obrázku číslo 4.9. Po upuštění levého tlačítka myši v tomto poli, je blok smazán. Jedinou výjimkou je, pokud jsou bloky připojeny ke spouštěcímu tlačítku. Po přesunutí do mazací oblasti se v tomto případě nezmění barva a ani bloky nejsou smazány, protože by došlo ke ztrátě spouštěcího tlačítka. Pro smazání v tomto případě je nutné od spouštěcího tlačítka bloky nejprve odpojit a až poté je smazat.



Obrázek 4.9: Mazání bloků

Z důvodu usnadnění práce, bylo vytvořeno tlačítko pro ukládání vytvořených sekvencí. Toto tlačítko se nachází v levé dolní části obrazovky. Po kliknutí na toto tlačítko je zobrazeno rozhraní(obr. 4.10). Zadání jména je povinné, a proto pokud není zadáno, je zobrazeno upozornění, že toto políčko je nutné vyplnit. Ukládána je sekvence spojená se spouštěcím blokem, pokud ke spouštěcímu bloku není připojen žádný další blok, je také zobrazeno upozornění. Po úspěšném uložení sekvence, je stránka obnovena a uživatel může najít svojí uloženou sekvenci v knihovně "Sekvence" pod jménem, které zadal. Při ukládání sekvence, kvůli nutnosti obnovení



Obrázek 4.10: Ukládání sekvencí

stránky, dochází ke smazání všech vytvořených bloků a rozhraní je uvedeno do původního stavu. Výsledná sekvence je uložena jako samostatný blok, jehož databázová položka "code" obsahuje, veškerý kód ze všech bloků sekvence s pevně nastavenými hodnotami z ovládacích prvků.

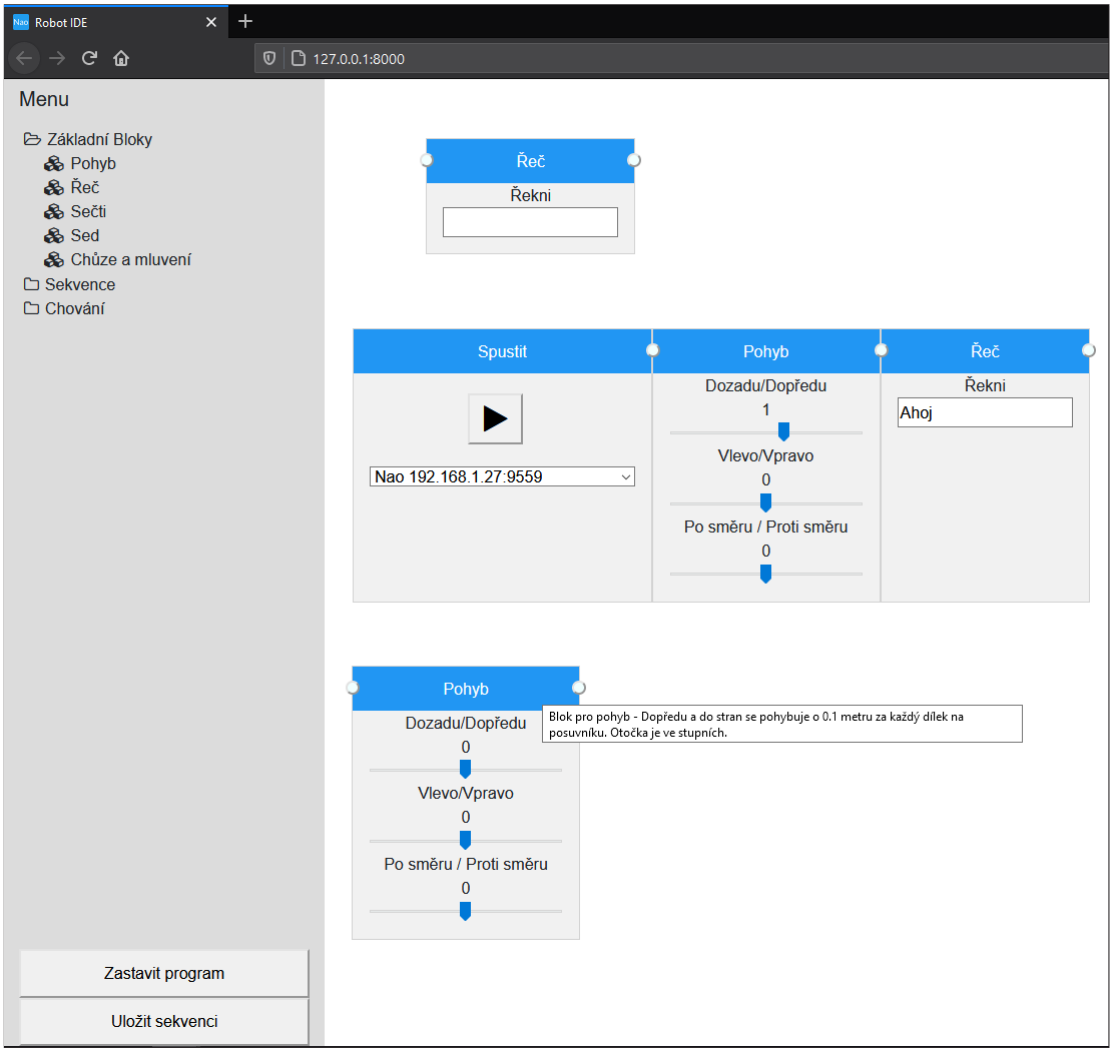
Pro zvýšení bezpečnosti při práci s robotem, bylo přidáno tlačítko pro zastavení vykonávané sekvence. Po odeslání sekvence na server jsou postupně odbavovány jednotlivé bloky sekvence tak, jak bylo popsáno v kapitole 4.7. Po každém vykonaném bloku v sekvenci je kontrolována podmínka, zda uživatel neklikl na tlačítko pro zastavení a při splnění této podmínky není už další blok vykonán.

5 Shrnutí výsledků a diskuze

Webové rozhraní bylo otestováno na robotu NAO V5 Technické Univerzity v Liberci. Všechny prvky webového rozhraní fungují s robotem NAO korektně. Funkčnost vytvořených bloků pro robota NAO nebylo možné otestovat na robotu Pepper, jelikož robot byl aktualizován na verzi 2.9, která nepodporuje programování pomocí SDK NAOqi. Při downgradu na verzi 2.5 by aplikace měla korektně fungovat, protože jediný rozdíl mezi roboty je verze NAOqi spuštěná přímo v robotu. Na straně serveru není žádná změna. Lze usuzovat, že vytvořené bloky budou fungovat i na robotu Pepper, jelikož programování za pomocí SDK je stejné i na novější verzi. Jedinou nevýhodou při použití webového rozhraní s robotem Pepper je nemožnost ovládat dotykový tablet, který je součástí robota Pepper.

Zabezpečení webové aplikace je vyřešeno formou přihlašování, jak je popsáno v kapitole 4.6. Neautorizovaní uživatelé nemají přístup do žádné části webového rozhraní a tudíž ani k robotům. Uživatelé jsou pak rozdělení do tří skupin. Uživatel s běžnými právy je pak nejméně rizikový. Uživatelé s programátorskými a administrátorskými právy jsou potencionálně nebezpeční, protože mohou vytvořit jakýkoliv skript v jazyce Python a následně ho spustit na serveru. Z tohoto důvodu je nutné přidělovat práva s rozvahou a udělovat je pouze důvěryhodným uživatelům. Pokud nejsou žádní administrátoři a programátoři delší dobu aktivní je rozumné odebrat jejich účty, aby nemohlo dojít k jejich zneužití a následně je v případě potřeby znovu vytvořit přímo na serveru z příkazové řádky.

Výsledná aplikace se nachází na obrázku číslo 5.1. Obsahuje všechny položky, které byly popsány v návrhu v kapitole 4.4. Aplikace se dá využít ke vzdálenému ovládání a programování robotů. Například v době pandemie je možné umožnit studentům přístup k webovému serveru a ti bez nutnosti cokoliv instalovat mohou vzdáleně pracovat na robotech ze svých vlastních zařízení.



Obrázek 5.1: Webová aplikace

6 Závěr

V rámci bakalářské práce bylo vytvořeno webové rozhraní pro ovládání humanoidních robotů NAO a Pepper. Pro vývoj webového rozhraní byl použit webový framework Django. Pomocí tohoto frameworku byl vytvořen server, který komunikuje s roboty pomocí SDK NAOqi od výrobců robota Softbank Robotics.

Roboti se ovládají z webového rozhraní, ve kterém se skládají grafické bloky za sebe. Každý z těchto bloků reprezentuje kód, který je spouštěn na serveru. Z webového serveru jsou pak za pomoci proxy serveru odeslány příkazy přímo do robota.

Vytváření nových bloků se realizuje v administrátorském rozhraní. Do tohoto rozhraní je nutné se nejprve přihlásit. Zde má uživatel možnost vytvářet nové bloky s ovládacími prvky jako jsou číselné nebo textové pole. Vytvořené bloky mohou být tříděny do knihoven, které mohou být v administrátorském rozhraní také vytvářeny. Dále je zde možnost vytvářet nové instance robotů, se kterými může server komunikovat.

Zabezpečení webového serveru bylo vytvořeno pomocí autentizačního formuláře administrátorského rozhraní. Obsah se zobrazí pouze přihlášeným uživatelům. Uživatelé jsou tříděni do tří skupin. A to na administrátory, kteří mají plnou kontrolu nad webovým serverem, na programátory, kteří mohou vytvářet nové bloky a knihovny, a na běžné uživatele, kteří mohou vytvářet pouze sekvence z již před připravených bloků a následně je spouštět na robotech.

V delším časovém horizontu by se dal projekt rozvinout například vytvořením více bloků, které by byli k dispozici uživatelům. Dalším vylepšením by mohla být přímá integrace podmínek a cyklů ve formě bloků tak, aby je bylo možné využít přímo v grafickém prostředí. V současném stavu mohou být podmínky a cykly realizovány pouze na úrovni databázové položky pro kód obsažené v jednotlivých blocích.

Zdroje

[1] DABOAPPS. NAO control [Software]. 21. prosince 2015. [cit. 2021-4-11]. Dostupné z: https://play.google.com/store/apps/details?id=de.daboapps.androidnao. Velikost 4,9MB, Vyžaduje Android 2.3 a vyšší.

[2] ATASOYWEB. Nao Controller [Software]. 28. listopadu 2017. [cit. 2021-4-11]. Dostupné z: https://play.google.com/store/apps/details?id=atasoyweb.net.naocontroller. Velikost 877kB , Vyžaduje Android 4.0.3 a vyšší.

[3] Django [online]. Django Software Foundation, 2021 [cit. 2021-04-11]. Dostupné z: https://www.djangoproject.com/

[4] Softbank Robotics [online]. Paris, France: SoftBank Robotics Europe, [c2005-2021] [cit. 2021-5-2]. Dostupné z: https://www.softbankrobotics.com/

[5] World Wide Web Consortium [online]. c2021 [cit. 2021-5-2]. Dostupné z: https://www.w3.org/

[6] LEGO. EV3 software [Software]. 25.7.2020. [cit. 2021-5-3]. Dostupné z: https://www.lego.com/cs-cz/themes/mindstorms/downloads. Velikost 624 MB, Požadavky na systém: Windows Vista (32/64 bitů) nebo novější verze systému Windows.

[7] EICHLER, Miroslav: Využití dostupných senzorů robota Nao pro detekci objektů a mapování okolí: Use available Nao robots' sensors to detect objects and map of its surrounding. Liberec: Technická univerzita v Liberci, 2018. Bakalářské práce. Technická univerzita v Liberci.

[8] Bootstrap [online]. MIT, c2021 [cit. 2021-5-3]. Dostupné z: https://getbootstrap.com/

[9] Flask [online]. Pallets, c2010 [cit. 2021-5-3]. Dostupné z: https://flask.palletsprojects.com

[10] Font awesome [online]. Berlin: ResearchGate, c2008-2021 [cit. 2021--5-3]. Dostupné z: https://fontawesome.com/

[11] SORRENTINO, John. Favicon.io [online]. c2021 [cit. 2021-5-3]. Dostupné z: https://favicon.io/

[12] BIRD, Jordan J. Fig 1 - Pepper (left) and Nao (right) are state-of-the-art robots that can perform general speech recognition but not speaker classification. [online]. [cit. 2021-5-3]. Dostupné z: https://www.researchgate.net/figure/Pepper-left-and--Nao-right-are-state-of-the-art-robots-that-can-perform-general-speech_fig1_340090917

[13] SOFTBANK ROBOTICS. Choreographe 2.5.10.7 [Software]. 25.10.2018. [cit. 2021-5-3]. Dostupné z: https://community-static.aldebaran.com/resources/2.5.10/Choregraphe/choregraphe-suite--2.5.10.7-win32-setup.exe. Velikost 243 MB, Požadavky na systém: Microsoft Windows 7 and 8.1.

[14] VANER, Pavel: Spolupráce robotů NAO: NAO Robots collaboration. Liberec: Technická univerzita v Liberci, 2018. Bakalářské práce. Technická univerzita v Liberci.

[15] NENCHEV, Dragomir N. a Atsushi KONNO. Humanoid Robots. 1. Berlin, SRN: Elsevier - Health Sciences Division, 2016. ISBN 9780128045602.

[16] SDKs. SoftBank Robotics documentation [online]. Francie: SoftBank Robotics Europe, c2006-2017 [cit. 2021-5-10]. Dostupné z: http://doc.aldebaran.com/2-5/dev/programming.html

[17] QiSDK [online]. Paris, France: Softbank Robotics Europe, c2006-2017 [cit. 2021-5-10]. Dostupné z: https://qisdk.softbankrobotics.com

[18] LabVIEW [online]. National Instruments, c2021 [cit. 2021-5-10]. Dostupné z: https://www.ni.com/cs-cz/shop/labview.html

A Obsah přiloženého CD

- Bakalářská práce ve formátu .pdf
- Zdrojové kódy aplikace zabalené ve formátu .zip