

Úlohy pro výuku programování a robotiky

Assignments for the Tuition of the Introductory Courses in
Programming and Robotics

Bc. Jan Vaňhara



Univerzita Tomáše Bati ve Zlíně

Fakulta aplikované informatiky

akademický rok: 2014/2015

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Bc. Jan Vaňhara**

Osobní číslo: **A13468**

Studijní program: **N3902 Inženýrská informatika**

Studijní obor: **Učitelství informatiky pro střední školy**

Forma studia: **prezenční**

Téma práce: **Úlohy pro výuku programování a robotiky**

Téma anglicky: **Assignments for the Tuition of the Introductory Courses in Programming and Robotics**

Zásady pro vypracování:

1. Analyzujte dostupné systémy pro výuku programování a robotiky.
2. Vyberte alespoň jeden systém pro výuku programování na ZŠ/SS ve věkových kategoriích 9–18 let.
3. Pro vybraný systém vytvořte minimálně 10 zadání úloh se stoupající obtížností.
4. Zadání optimalizujte tak, aby jednodušší úlohy bylo možno naprogramovat za 2 vyučovací hodiny. Složitější úlohy se snažte rozložit do modulů, jejichž funkci bude možné samostatně ověřit.
5. Ověřte použitelnost vytvořených zadání v reálné výuce.

Rozsah diplomové práce:

Rozsah příloh:

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

Seznam odborné literatury:

1. **LeJOS: Java for Lego Mindstorms** [online]. 1997–2009 [cit. 2015-01-29]. Dostupné z: <http://www.lejos.org/>
2. **Mindstorms EV3. LEGO** [online]. 2015 [cit. 2015-01-29]. Dostupné z: <http://www.lego.com/en-us/mindstorms/?domainredir=mindstorms.lego.com>
3. **Digital Designer. LEGO** [online]. The LEGO Group, ?2015 [cit. 2015-01-29]. Dostupné z: <http://ldd.lego.com/cs-cz/>
4. **EV3 Curriculum. Carnegie Mellon Robotics Academy** [online]. Carnegie Mellon Robotics Academy, ? 2014 [cit. 2015-01-29]. Dostupné z: <http://www.education.rec.ri.cmu.edu/content/lego/ev3/>
5. **LEGO engineering** [online]. Tufts Center for Engineering Education and Outreach (CEEEO), 2015 [cit. 2015-01-29]. Dostupné z: <http://www.legoengineering.com/>
6. **ROLLINS, Mark. Beginning LEGO Mindstorms EV3.** New York: Springer Science+Business Media New York, 2014. ISBN 978-143-0264-361.
7. **FERRARI, Mario Ferrari and Giulio. Programming Lego Mindstorms with Java.** Rockland, Mass: Syngress Media, 2001. ISBN 19-289-9455-5.
8. **MARIO FERRARI, Giulio Ferrari a Technical editor RALPH HEMPEL. Building robots with Lego Mindstorms the ultimate tool for Mindstorms maniacs!.** [Online-Ausg.]. Rockland, MA: Syngress Pub, 2002. ISBN 19-289-9467-9.

Vedoucí diplomové práce:

Ing. Tomáš Dulík, Ph.D.

Ústav informatiky a umělé inteligence

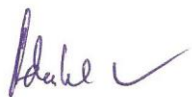
Datum zadání diplomové práce:

6. února 2015

Termín odevzdání diplomové práce:

15. května 2015

Ve Zlíně dne 6. února 2015



doc. Mgr. Milan Adámek, Ph.D.
děkan



L.S.



doc. Mgr. Roman Jašek, Ph.D.
ředitel ústavu

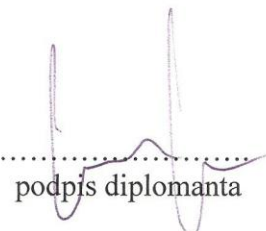
Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomová práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl jsem seznámen s tím, že na moji diplomovou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou práci nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- beru na vědomí, že pokud bylo k vypracování diplomové práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- Že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně 5.5.2015


.....
podpis diplomanta

ABSTRAKT

Práce je zaměřena jako podpůrný materiál pro výuku programování v robotice. Využívá stavebnice Lego Mindstorms EV3. V příloze lze nalézt 3D modely úloh, návody na sestavení modelů, programy v Lego Mindstorms EV3 a programy v programovacím jazyku Java.

Klíčová slova: Lego Mindstorms EV3, JAVA, výuka, úlohy, programování, robotické systémy, robot

ABSTRACT

Purpose of this thesis is the support material for teaching programming in robotics. Uses Lego Mindstorms EV3. The thesis contains 3D models of tasks, building guides, programs in the Lego Mindstorms EV3 and programs in the Java programming language.

Keywords: Lego Mindstorms EV3, JAVA, teaching, tasks, programming, robotic systems, robot

PODĚKOVÁNÍ

Děkuji svému vedoucímu bakalářské práce, Ing. Tomáši Dulíkovi, PhD., za cenné rady a připomínky při tvorbě práce. Chci také poděkovat svým rodičům, rodině a přátelům za trpělivost a podporu při studiu. Jsem rád, že vás mám.

OBSAH

ÚVOD.....	9
I TEORETICKÁ ČÁST.....	10
1 SYSTÉMY PRO VÝUKU PROGRAMOVÁNÍ A ROBOTIKY	11
1.1 PROGRAMOVACÍ JAZYK KAREL	11
1.2 LEGO WEDO	12
1.3 LEGO MINDSTORMS	13
1.3.1 Lego Mindstorms NXT	13
1.3.2 Lego Mindstorms EV3	14
1.3.3 Vývoj Lego Mindstorms – původní vývojové prostředí	14
1.3.4 Vývoj Lego Mindstorms – LeJOS	15
1.4 SESTAVENÍ VLASTNÍHO ROBOTA	16
1.4.1 Obchod určený pro robotiku	16
1.4.2 Raspberry Pi	17
1.4.3 BeagleBone Black	18
1.4.4 JAVA na BeagleBone Black a Rapsberry Pi	18
1.5 3D CAD PROGRAMY PRO KONSTRUKCE ROBOTŮ	19
1.5.1 LEGO Digital Designer.....	19
1.5.2 LDraw	19
1.5.3 FreeCAD	20
1.5.4 SketchUp	20
1.5.5 Rhinoceros.....	21
1.6 SOUHRN A DOPORUČENÍ	22
II PRAKTICKÁ ČÁST	23
2 PŘÍPRAVA PRO VÝVOJ V LEGO MINDSTORMS EV3.....	24
3 PŘÍPRAVA PRO VÝVOJ V LEJOS	27
3.1 POTŘEBY	27
3.2 INSTALACE LEJOS A PŘÍPRAVA MICROSD KARTY.....	27
3.3 INSTALACE VÝVOJOVÉHO PROSTŘEDÍ ECLIPSE	29
3.4 PRVNÍ PROGRAM	34
4 PRAKTICKÉ ÚLOHY PRO LEGO MINDSTORMS EV3.....	35
4.1 ZÁKLADNÍ SEZNÁMENÍ S EV3 BRICK	35
4.1.1 Úkoly.....	35
4.1.2 Sestavení	35
4.1.3 Program	35
4.1.3.1 Lego Mindstorms EV3 Home Edition	35
4.1.3.2 LeJOS.....	36
4.2 ECHO TELEGRAF.....	37
4.2.1 Úkoly.....	37
4.2.2 Sestavení	37
4.2.3 Program	37
4.2.3.1 Lego Mindstorms EV3 Home Edition	37
4.2.3.2 LeJOS.....	38

4.3	ULTRAZVUKOVÝ PARKOVACÍ ASISTENT	40
4.3.1	Úkoly.....	40
4.3.2	Sestavení	40
4.3.3	Program.....	40
4.3.3.1	Lego Mindstorms EV3 Home Edition	40
4.3.3.2	LeJOS.....	41
4.4	SEKAČKA	42
4.4.1	Úkoly.....	42
4.4.2	Sestavení	42
4.4.3	Program.....	42
4.4.3.1	Lego Mindstorms EV3 Home Edition	42
4.4.3.2	LeJOS.....	43
4.5	3-KOLOVÝ ROBOT S DETEKČÍ PŘEKÁŽEK	45
4.5.1	Úkoly.....	45
4.5.2	Sestavení	45
4.5.3	Program.....	46
4.5.3.1	Lego Mindstorms EV3 Home Edition	46
4.5.3.2	LeJOS.....	47
4.6	VYSOKOZDVIŽNÝ VOZÍK	50
4.6.1	Úkoly.....	50
4.6.2	Sestavení	50
4.6.3	Program.....	50
4.6.3.1	Lego Mindstorms EV3 Home Edition	51
4.6.3.2	LeJOS.....	52
4.7	ROBOT SBĚRAČ	54
4.7.1	Úkoly.....	54
4.7.2	Sestavení	54
4.7.3	Program.....	54
4.7.3.1	Lego Mindstorms EV3 Home Edition	54
4.7.3.2	LeJOS.....	55
4.8	SUMO ROBOT.....	58
4.8.1	Úkoly.....	58
4.8.2	Sestavení	59
4.8.3	Program.....	59
4.8.3.1	Lego Mindstorms EV3 Home Edition	59
4.8.3.2	LeJOS.....	59
4.9	AUTO S KLASICKÝM ŘÍZENÍM	63
4.9.1	Úkoly.....	63
4.9.2	Sestavení	63
4.9.3	Program.....	63
4.9.3.1	Lego Mindstorms EV3 Home Edition	64
4.9.3.2	LeJOS.....	66
4.10	HÁDÁNÍ ČÍSEL	68
4.10.1	Úkoly.....	68
4.10.2	Sestavení	69
4.10.3	Program – pouze LeJOS.....	69
ZÁVĚR		73

SEZNAM POUŽITÉ LITERATURY.....	74
SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK.....	77
SEZNAM OBRÁZKŮ	78
SEZNAM PŘÍLOH.....	81

ÚVOD

Velmi mě těší, že můžu spojit užitečné věci v jeden zábavný celek. Jsem rád, že tahle práce má být pomocnou rukou v případě předávání znalostí robotiky a programování dále.

V první části diplomové práce se snažím zachytit, na co jsem přišel při bádání po robotických systémech určených pro výuku. Popisuji jak hardwarové, tak softwarové prvky a snažím se vybrat nejvhodnější dostupný systém a zohledňuji věkovou kategorii 9-18 let.

Druhá část patří deseti modelům pro Lego Mindstorms EV 3, každý se svými úkoly. Modely jsou připraveny s návody na sestavení i 3D návrhu a je možné si je před sestavením prohlédnout. Vývojové kódy jsou dostupné v Lego Mindstorms EV3 Home Edition a programovacím jazyku Java.

I. TEORETICKÁ ČÁST

1 SYSTÉMY PRO VÝUKU PROGRAMOVÁNÍ A ROBOTIKY

Na stránkách jako jsou roboshop.com [20] a Jameco Robot Store [21] se dá nalézt spousta jednoúčelových již postavených robotů, které se dají programovat. Všechny ale nejsou vhodné pro výuku ať už z důvodu jednoúčelovosti, větší náročnosti implementace programů do robota, časové náročnosti, kdy se čeká, než se program zpracuje a spustí, nebo možné špatné kompatibility či chyb vývojového prostředí.

Existují ale i robotické platformy určené přímo pro výuku.

1.1 Programovací jazyk Karel

Karel je programovací jazyk určen k výuce programování pro naprosté začátečníky. [9]

Karel představuje robůtka, který se pohybuje po městě a vykonává vaše příkazy. Dále je tady něco, co je v Karlovi označováno jako "město". Je to v podstatě čtvercová síť, kde se Karel pohybuje. A v této síti může Karel dělat skoro všechno, co se mu líbí, stavět vysoké zdi z cihel, pokládat značky, nebo se jen tak procházet ode zdi ke zdi. [10]

Výhoda Karla je, že používá jen pár základních příkazů a jeho výsledek je vizuálně zobrazován přímo při spuštění kódu.

Karla si lze vyzkoušet na adrese karel.oldium.net [22]. Protože se jedná čistě o software, není možné si na něj sáhnout a vyzkoušet v reálném světě.



Obrázek 1: Prostředí výukového jazyka Karel. [22]

1.2 Lego WeDo

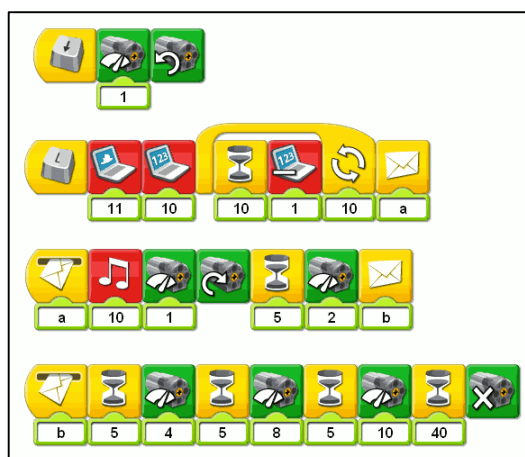
Levná sada od firmy Lego, která se skládá z několika málo součástek, jednoho motoru, senzoru, detekující vzdálenost 15 cm, snímače polohy a usb hubu.



Obrázek 2: Sada Lego WeDo Education [1]

Nejmenší komplexní set, ze kterého je možné postavit v omezené míře vlastní model robota se základní funkcí.

Lego WeDo má vlastní programovací prostředí, ve kterém se programuje tak, že se funkční bloky skládají vedle sebe a spojují se pomocí virtuálních spojů.



Obrázek 3: Programové bloky vývojového prostředí pro lego WeDo [26]

Lego WeDo je možné programovat i v programovacím jazyku Scratch.

Lego WeDo je určeno pro nízkou věkovou kategorii 9-13 let.

1.3 Lego Mindstorms

Lego Mindstorms jsou stavebnice určená pro tvorbu robotů od dánské firmy Lego. Ve stavebnicích se nachází hlavní řídicí jednotka, nazvaná „Brick“, servo-motory a senzory zvuku, doteku, světla a vzdálenosti.

Jedná se o hojně rozšířenou sadu pro výuku robotiky. Díly této sady jsou kompatibilní s Lego Technic a jednotlivé díly se dají na internetu koupit buď na kusy, nebo jako sady dílů vážené na kilogramy. Sestavit lze téměř cokoli.

Fakulta Aplikované Informatiky na univerzitě Tomáše Bati ve Zlíně disponuje Lego Mindstorms ve verzi NXT i EV3.

Jak NXT, tak EV3 umožňuje programování v různých jazycích, jako jsou například nativní grafické vývojové prostředí Lego, C#, Java, Lua, C, Python a další. [8]

1.3.1 Lego Mindstorms NXT

Starší verze Lego Mindstorms uvedená v červenci 2006. NXT Brick je poháněn procesorem Atmel AT91SAM7S256 s frekvencí 48MHz [5] – dnes již hodně omezující a NXT se již neprodává.



Obrázek 4: NXT Brick [4]



Obrázek 5: Robot postavený z Lego Mindstorms NXT [4]

Pro NXT lze nalézt na internetu spoustu návodů na sestavování i programování. Příklady pro NXT je možné nalézt na www.nxtprograms.com [2].

1.3.2 Lego Mindstorms EV3

Novější verze Lego Mindstorms ze září 2013, která obsahuje EV3 Brick s procesorem TI Sitara AM1808 s frekvencí 300 MHz, který umožňuje běh Linux. EV3 Brick obsahuje také i slot pro microSDHC, díky kterému je možné jednodušeji nahrát vlastní software i spustit vlastní firmware (např. LeJOS).

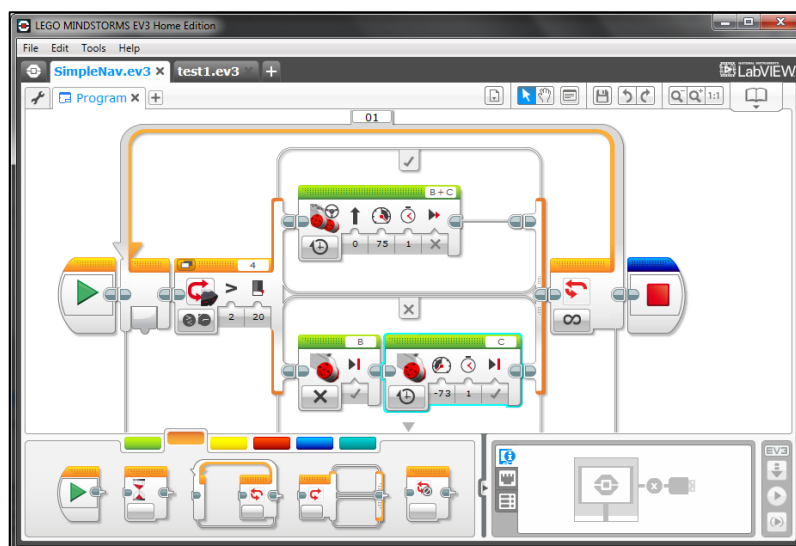


Obrázek 6: EV3 Brick [3]

1.3.3 Vývoj Lego Mindstorms – původní vývojové prostředí

Lego má připraveny nástroje pro programování EV3 Brick s názvem Lego Mindstorms EV3 Home Edition. Jedná se o grafické vývojové prostředí, kde se graficky vkládají a nastavují připravené bloky. Rozšiřitelnost aplikace to značně omezuje, protože se převážně používají

připravené bloky, které jsou především určeny pro ovládání prvků Lega Mindstorms (motory, senzory, atd.).



Obrázek 7: Původní vývojové prostředí pro Lego Mindstorms. [6]

Platforma je dostupná pouze na Microsoft Windows a OS X od Apple. Také programování v blocích se dosti liší od běžného programování (například v jazyce JAVA) a tak může být problém se vyznat v běžném nebo rozsáhlejší kód.

1.3.4 Vývoj Lego Mindstorms – LeJOS

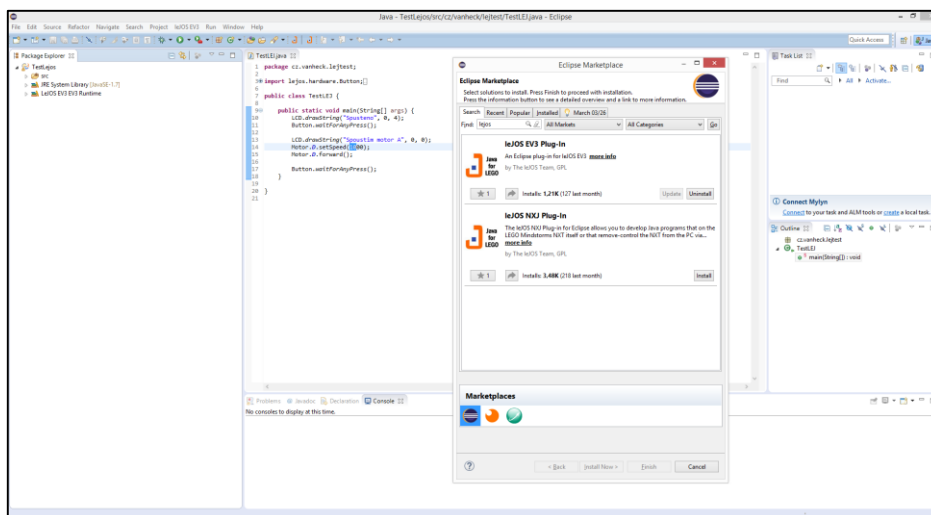
Jedná se o systém pro pokročilé programování Lega Mindstorms. Umožňuje v Bricku běh JRE a tím běh programů napsaných v programovacím jazyce Java.

Vývojáři LeJOS připravili i nástroje pro vývojové prostředí Eclipse, díky kterým lze vytvořený kód přímo spouštět.

Jelikož je možné Eclipse spustit téměř na jakékoli desktopové platformě, není problém vyvíjet na jakémkoli operačním systému, kde se dá Eclipse spustit.

Protože je jazyk JAVA mezi vývojáři softwaru hodně oblíbený, je možné v něm vyvíjet téměř cokoli a bez problému tím jednoduše rozšířit naši robotickou platformu Lego Mindstorm EV3 o další kód.

LeJOS běží zvlášť z microSDHC karty, po vyjmutí karty opět běží původní originální systém a není třeba se tak obávat nemožnosti ovládat Brick originálním softwarem.



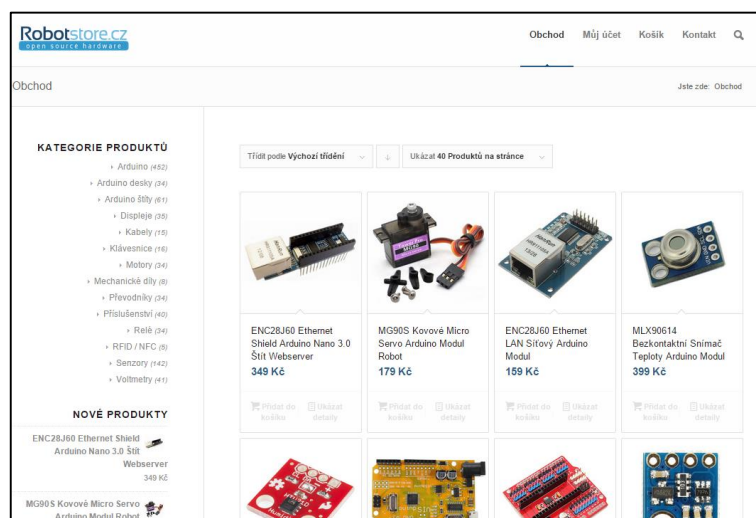
Obrázek 8: Instalace LeJOS Plug-In přímo v Marketplace Eclipsu.

1.4 Sestavení vlastního robota

Tahle možnost je pro ty, kdo chtějí ušetřit nebo sestavit robota podle svého vlastního uvážení a pak provádět výuku na něm. Zde už jsou potřeba pokročilejší znalosti elektroniky a technické dovednosti. Výsledky ale můžou být velmi zajímavé i méně finančně nákladné.

1.4.1 Obchod určený pro robotiku

Tuzemský internetový obchod robotstore.cz [23] se zaměřuje na prodej součástek pro robotiku jak elektronických, tak i mechanických. Je zde možné pořídit gyroskopy, akcelerometry, pohybová, plynová čidla, motory, servo-motory, displeje, kabely i řídicí jednotky.



Obrázek 9: Internetový obchod robotstore.cz [23]

1.4.2 Raspberry Pi

Jde o jednodeskový počítač velikosti zhruba platební karty. Vyvíjí ho britská nadace Raspberry Pi Foundation s cílem podpořit výuku informatiky ve školách. [11]



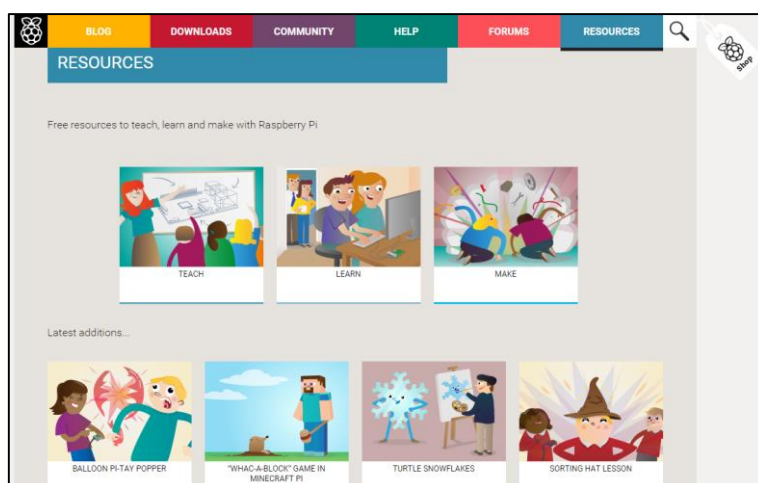
Obrázek 10: Raspberry Pi (model B+) 2014 [11]

Jelikož se jedná o počítač, samotný výrobce nabízí jako operační systémy ARMové verze linuxových distribucí Debian a Arch. Výrobce též ohlásil práce na systému Rasdroid pro Raspberry Pi na bázi Android 4.0. [11]

Lze jej plnohodnotně využít k výuce programování v jakémkoli programovacím jazyku, který funguje pod těmito operačními systémy.

Na stránkách raspberrypi.org [13] v sekci Resources lze nalézt výukové materiály v angličtině pro učitele, učební studijní materiály a manuály pro práci s Raspberry Pi.

K Raspberry Pi lze následně přes I/O (vstupní a výstupní) piny připojit a ovládat součástky nakoupené z robotstore.cz.

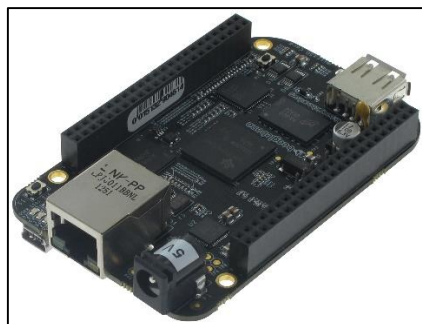


Obrázek 11: Internetová stránka s manuály pro Raspberry Pi [13]

1.4.3 BeagleBone Black

Alternativa k Raspberry Pi je BeagleBone Black. Jde o open-source hardware, vyvíjený firmou Texas Instruments. Oproti Raspberry Pi 2 modelu B má rychlejší procesor, větší počet I/O pinů a mezi nimi lze aktivovat piny přímo pro PWM modulaci a tím ovládat přímo servomotory. Má pouze jeden USB konektor.

Základní manuál v angličtině lze nalézt na beagleboard.org. [12]



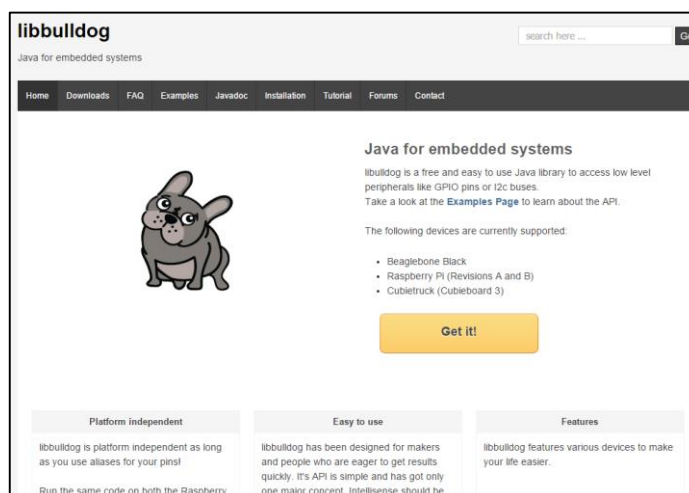
Obrázek 12: BeagleBone Black

1.4.4 JAVA na BeagleBone Black a Raspberry Pi

Raspberry Pi má podporovaný operační systém Raspbian, což je Debian pro Raspberry Pi. Stejně tak BeagleBone Black podporuje operační systém Debian. V něm lze jednoduše instalovat *Java Developer Kit* příkazem [15]:

```
sudo apt-get install oracle-java8-jdk
```

Pro přímý přístup k GPIO v Javě na Raspberry Pi a BeagleBone Black je dostupná knihovna libbulldog [24].



Obrázek 13: Stránky knihovny libbulldog pro přímý přístup k GPIO. [24]

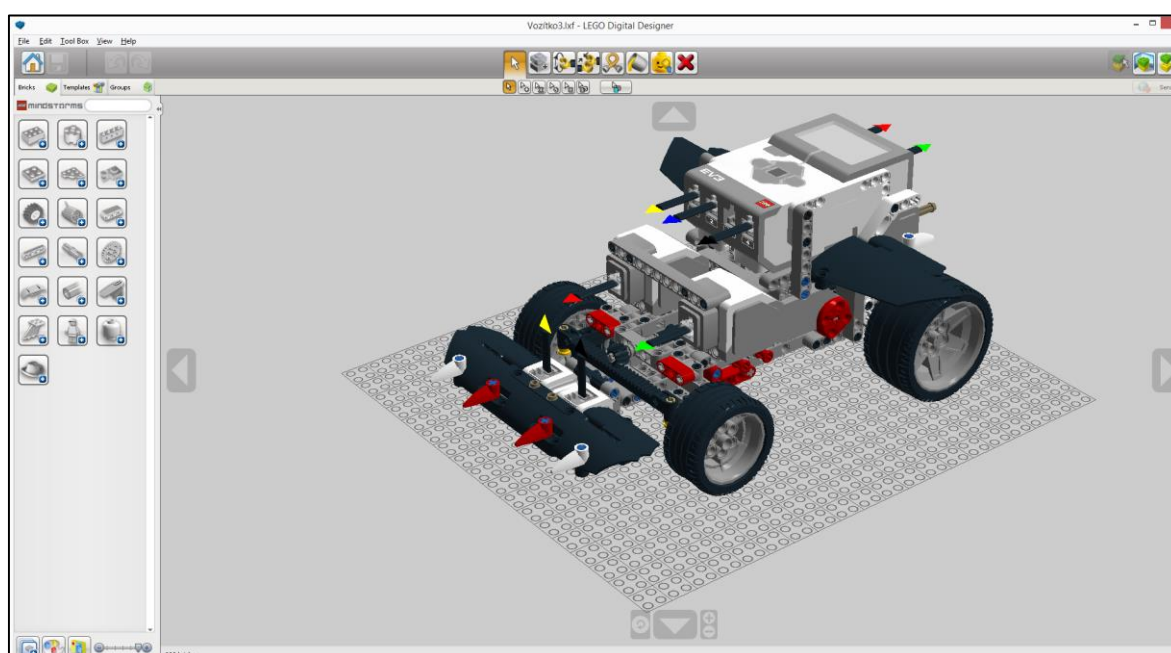
1.5 3D CAD programy pro konstrukce robotů

1.5.1 LEGO Digital Designer

Program pro sestavování robotů od firmy Lego. Podporuje téměř všechny díly Lego Technics, Lego Mindstorms NXT a Lego Mindstorms EV3.

Dokáže z konstruovaného modelu sestavit návod na sestavení modelu po jednotlivých součástkách.

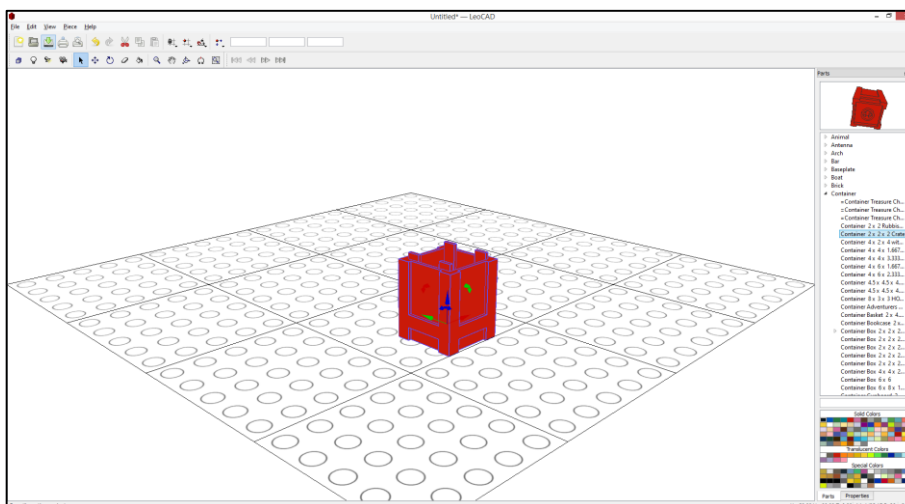
Nepodporuje vlastní vytvořené součástky.



Obrázek 14: Lego Digital Designer

1.5.2 LDraw

LDraw je otevřený standart pro Lego CAD programy, které umožňují vytvoření si vlastních součástek, tvorby modelů a mnoho dalšího. [27] Obsahuje programy jako LDCad, LeoCAD, MLCad.

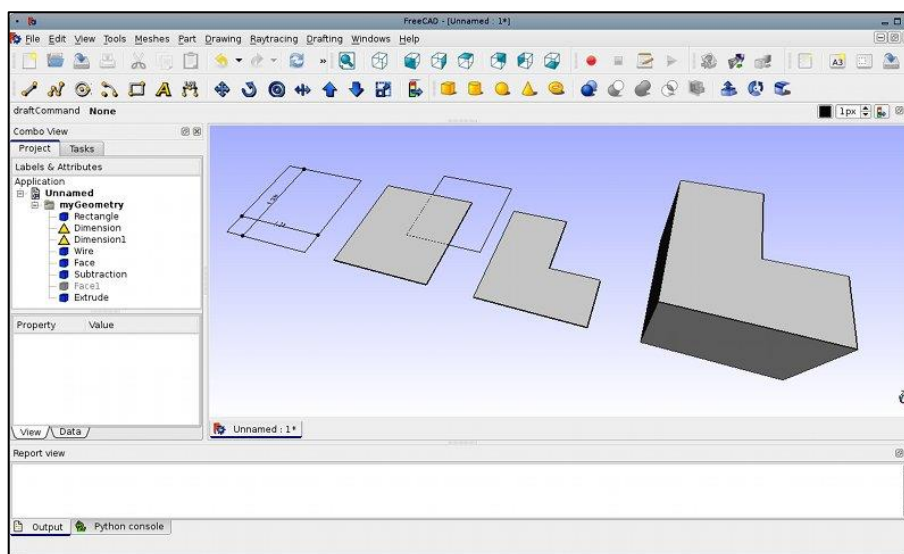


Obrázek 15: MLCad program z rodiny LDraw.

1.5.3 FreeCAD

Open source program pro tvorbu parametrických 3D modelů.

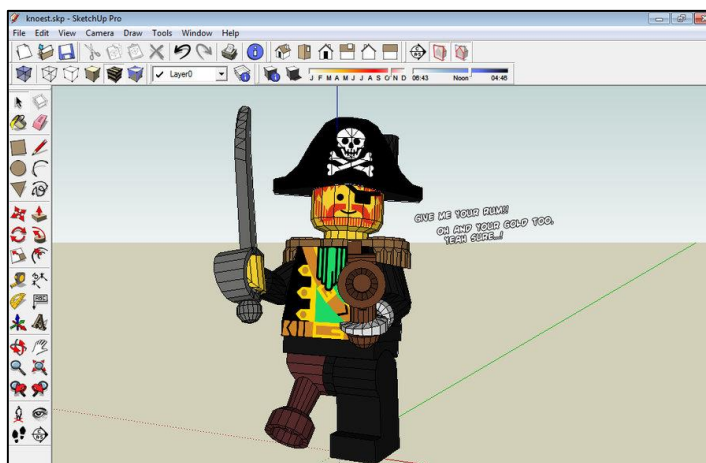
Je multiplatformní a dokáže otevřít mnoho souborových formátů (STEP, STL, atd.).



Obrázek 16: FreeCAD [28]

1.5.4 SketchUp

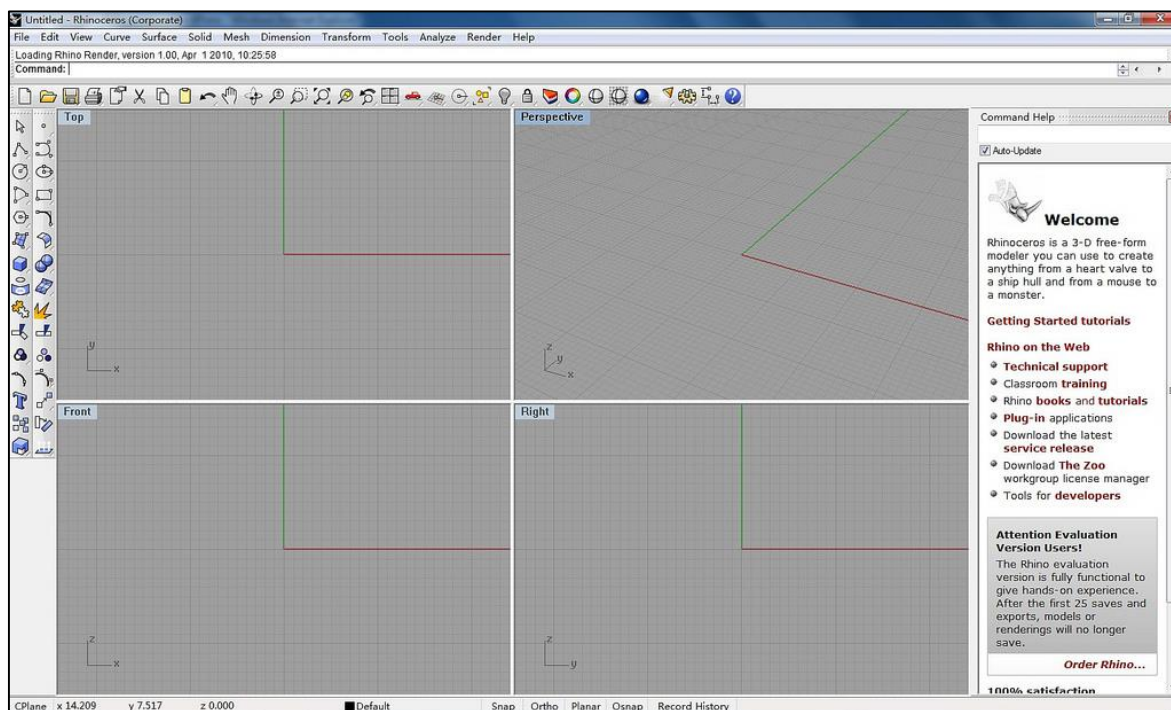
Je pokročilý a hojně využívaný nástroj pro tvorbu 3D modelů. Původně vyvíjen firmou Google. Pro nekomerční nebo studijní účely je zdarma jeho verze Make.



Obrázek 17: Program SketchUp [29]

1.5.5 Rhinoceros

Rhinoceros je alternativa k programu SketchUp. Je oblíbený zejména u modelářů malých papírových modelů. Z 3D modelu dokáže připravit plášť tak, že po vytištění a obstrihnutí lze složit dohromady.



Obrázek 18: Program Rhinoceros [30]

1.6 Souhrn a doporučení

Jednoznačně z analýzy vychází nejlépe Lego Mindstorms.

Lego Mindstorms z pohledu komplexnosti, robustnosti a robotického systému určeného pro výuku opravdu ční z řady.

Tam, kde by člověk musel skládat elektroniku s mechanickými součástkami dohromady a pracně vymýšlet jak co kde uchytit, upevnit a přišroubovat, Lego má řešení.

Jednoduché bezúdržbové spoje, kde všechny díly do sebe bez problému zapadají. Mechanické součástky mají hodně propracované, nechybí díly jako je diferenciál, hřídel, ani různé druhy ozubených koleček.

Pro Lego existují i speciální CAD programy vyvinuty přímo pro modelování součástek nebo modelů. Lego Digital Designer dokáže pak z vytvořeného modelu připravit návod na sestavení součástku po součástce.

Když opomeneme u EV3 elektronickou hardwarovou část, která u dnešních standardů jako je BeagleBone či Raspberry Pi, mírně zaostává, pořád se jedná o špičku v komplexnosti. BeagleBone či Raspberry Pi jsou na jednu stranu lépe hardwarově vybavené, ale EV3 je mechanicky kompatibilní se součástkami Lego Technics a spojení elektroniky s mechanickou částí tyhle nedostatky hardwarové výbavy prozatím lehce nahradí.

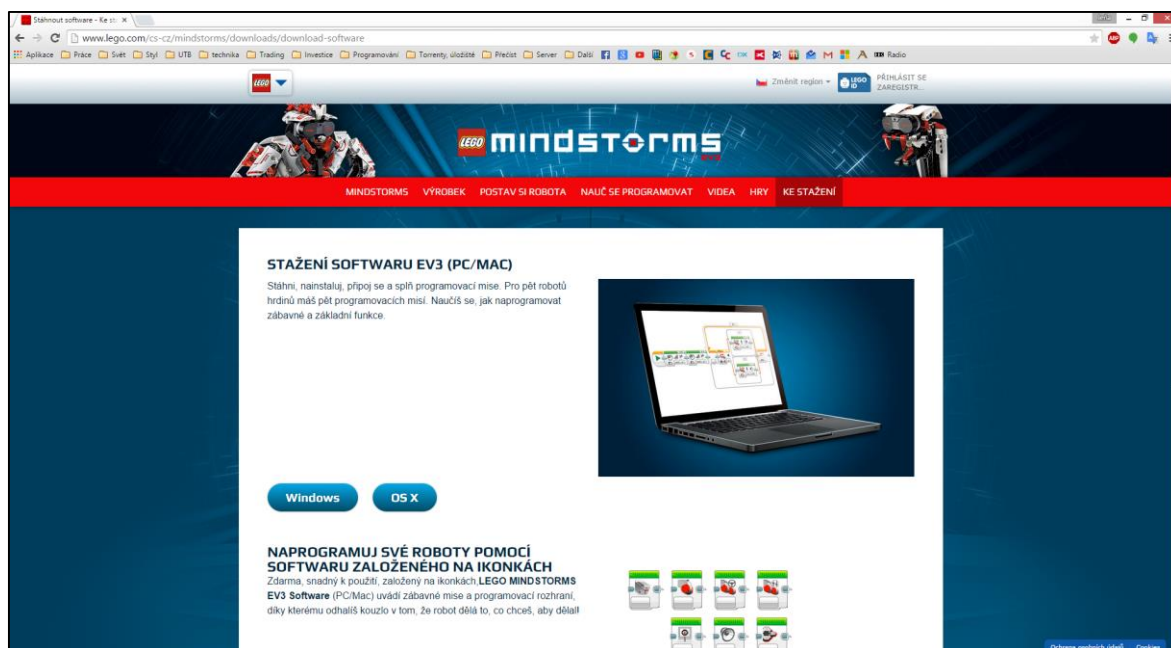
Lego Mindstorms má i skvěle vyvinutý vývojový software a je zvláště vhodný pro nižší věkovou kategorii 9-13 let. Všechno se nastavuje vizuálně, píše se málo a všude je spousta výstižných obrázků. Lego Mindstorms ale umožňuje i vývoj téměř v kterémkoli jazyce a tím možnost vývoje i pokročilých aplikací, což je vhodnější pro starší věkovou kategorii 14-18 let.

Nemá význam dělat multikriteriální analýzu, protože Lego nemá konkurenci.

II. PRAKTICKÁ ČÁST

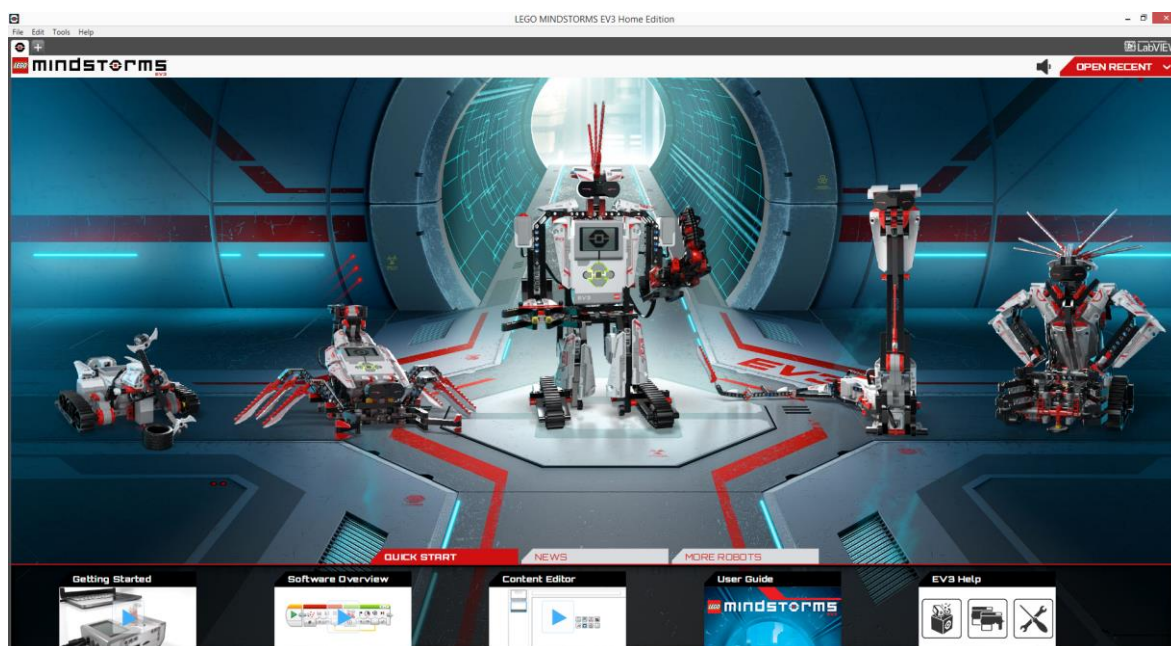
2 PŘÍPRAVA PRO VÝVOJ V LEGO MINDSTORMS EV3

Na stránce Lega Mindstorms [25] v sekci Download je dostupný instalační soubor vývojového prostředí pro Lego Mindstorms EV3. Zde si jen zvolíme, na jaký operační systém chceme program nainstalovat.



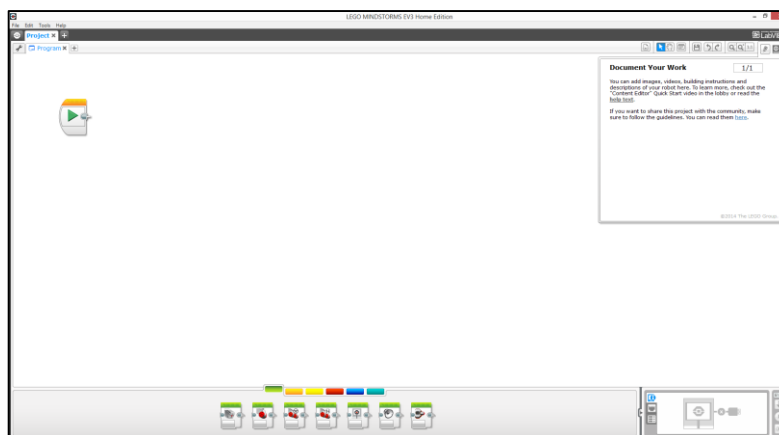
Obrázek 19: Stránka pro stažení vývojového prostředí Lego Mindstorms EV3

Po vybrání operačního systému se stáhne instalační soubor. Pro instalaci soubor spustíme.



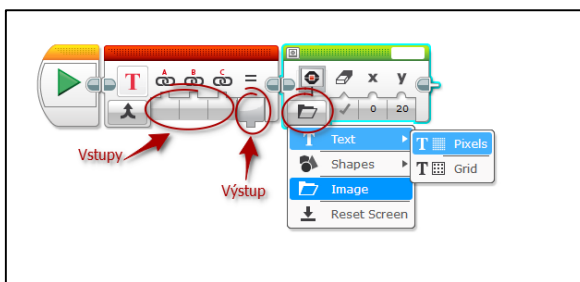
Obrázek 20: První spuštění vývojového prostředí Lego Mindstorms EV3 Home Edition

V menu File → New Project si vytvoříme nový projekt, zobrazí se nám okno, kde hlavní část zabere náš program, kde se vyskytuje pouze jeden blok programu. Na tento blok, který má zelený symbol Play, a který symbolizuje začátek programu, budou napojovat námi zvolené bloky ze spodní části obrazovky. Z těchto bloků budou naše programy sestaveny. Bloky jsou dostupné v kategoriích. Každá kategorie je jinak barevně rozlišená.



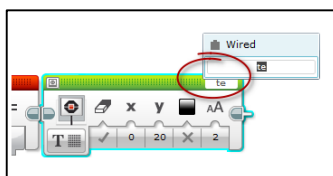
Obrázek 21: Vytvoření nového projektu v Lego Mindstorms EV3 Home Edition

Přidáme z červené kategorie blok Text. Který může mít tři vstupy a jeden výstup. Za něj dáme ze zelené kategorie blok Display. V bloku display nastavíme tlačítkem s obrázkem složky hodnotu Text → Pixels.



Obrázek 22: Základní programování v Lego Mindstorms EV3 Home Edition

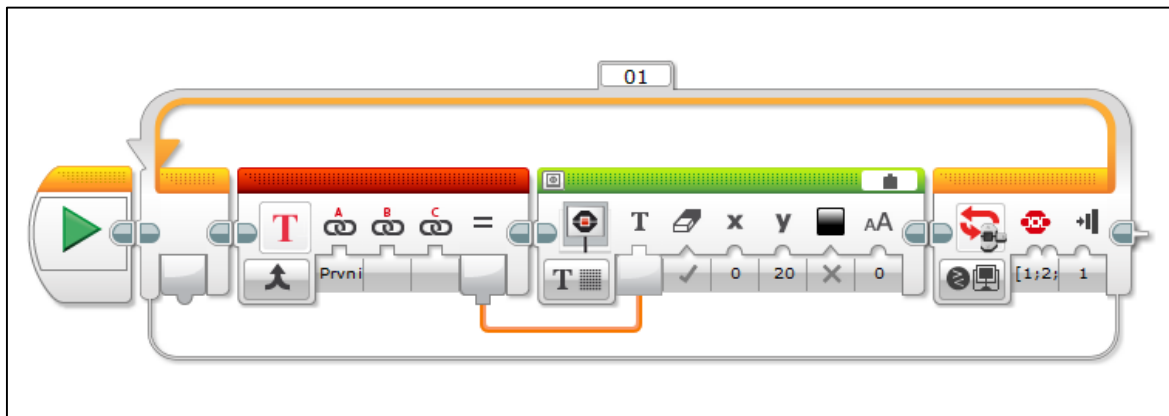
V zeleném bloku ještě nastavíme vlastnost vstupu na Wired.



Obrázek 23: Nastavení vlastnosti vstupu na Wired.

Poté výstup z červeného bloku Text a vstup do zeleného bloku tažením spojíme. Do libovolného vstupu v červeném bloku Text napíšeme „První spuštění“. Poté si přidáme blok

smyčky ze žluté kategorie a nastavíme ji na Brick Buttons a vložíme do ní červený a zelený blok. A máme naprogramován náš první program. Pokud jsme spojeni s Brickem EV3 přes USB nebo Bluetooth, můžeme tlačítkem v bloku Play nahrát a spustit program.



Obrázek 24: Úvodní program v Lego Mindstorms EV3 Home Edition.

V čisté instalaci Lego Mindstorms EV3 Home Edition se nenachází některé senzory (Ultrasonic senzor, Gyro senzor). Musí se stáhnout zvlášť na <http://www.lego.com/en-us/mindstorms/downloads> pod nadpisem EV3 SOFTWARE BLOCK DOWNLOAD. Po stažení souborů s příponou .ev3b je třeba je importovat do vývojového prostředí – v menu Tools → Block Import a poté restartovat vývojové prostředí.

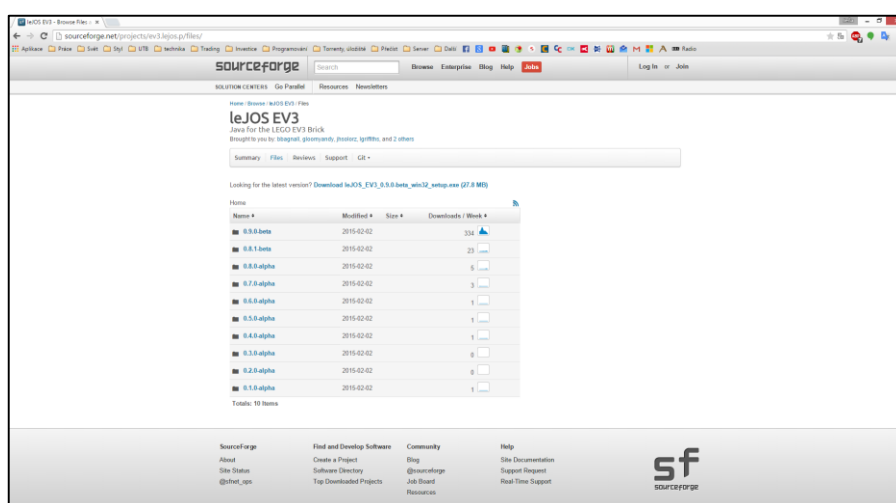
Úvodní kód se nachází v příloze 0.

3 PŘÍPRAVA PRO VÝVOJ V LEJOS

3.1 Potřeby

Pro vývoj na LeJOS budeme potřebovat microSDHC kartu, pro naše účely používám Ridata 8GB. USB WiFi adaptér je výhodou, LeJOS disponuje funkcí implementovat programy z Eclipse přes síť.

Na lejos.org [7] odkaz Download přesměruje na projekt LeJOS na sourceforge.net, kde lze stáhnout všechny potřebný software. Aktuálně v nejnovější verzi 0.9.0 (2.2.2015).

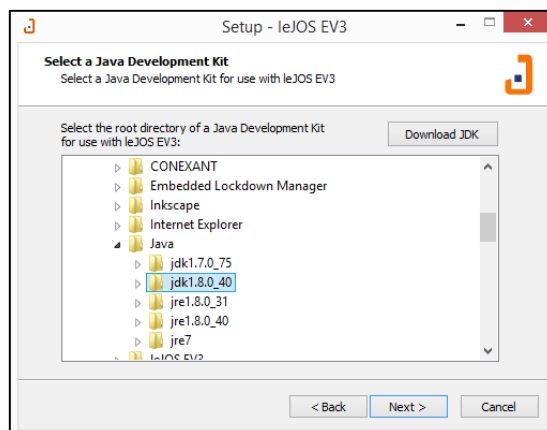


Obrázek 25: Projekt LeJOS na sourceforge.net.

Pro Windows lze stáhnout instalační soubor *leJOS_EV3_0.9.0-beta_win32_setup.exe*. Instalační soubor velmi jednoduše navede a provede potřebné kroky automaticky, aniž bychom museli jednotlivé kroky dělat ručně.

3.2 Instalace LeJOS a příprava microSD karty

Hned druhý krok instalace po úvodním slovu je určení místa nainstalovaného JDK. Pokud JDK nainstalováno není, tlačítko Download JDK otevře v prohlížeči stránku, ze které je možné JDK získat.



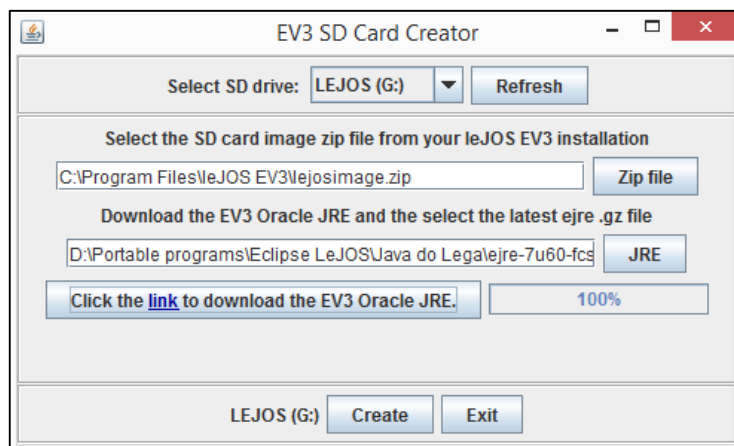
Obrázek 26: Určení JDK pro LeJOS.

Vybereme nainstalované JDK a nainstalujeme LeJOS. Na konci instalace necháme zaškrtnuté políčko Launch EV3SDCard utility.



Obrázek 27: Konec instalace, zaškrtnutím políčka Launch EV3SDCard utility zajistíme spuštění programu pro nastavení microSD karty.

Ve spuštěném programu *EV3 SD Card Creator* nastavíme shora nejprve disk, který reprezentuje microSD kartu, následně vybereme tlačítkem Zip file soubor lejosimage.zip, který se nachází v instalované složce LeJOS, a nakonec tlačítkem JRE vybereme stažený soubor jdk nebo jre javy pro architekturu ARM. JRE pro architekturu ARM lze stáhnout ze stránek Oracle Java, tlačítko *Click the link to download the EV3 Oracle JRE* přesměruje přímo na tuhle stránku. Pro stažení potřebného souboru je třeba mít účet Oracle. Vytvoření účtu u Oracle je bezplatné a celý průvodce je v češtině. Doporučuji stáhnout Oracle Java SE Embedded version 7 Update 60, instalace je jednodušší bez nutnosti dalšího nastavování.

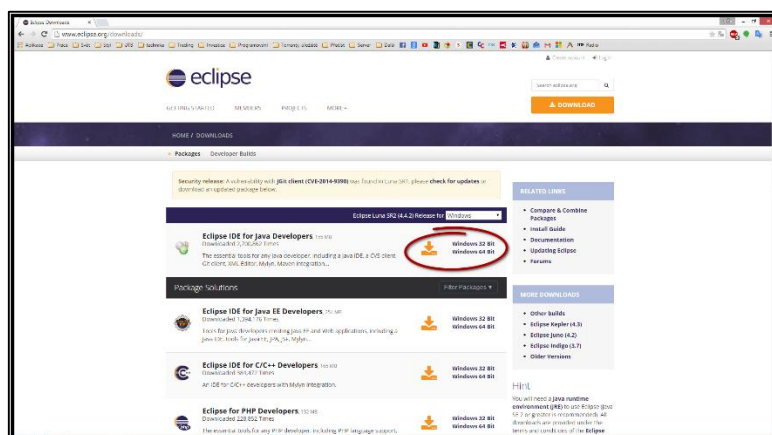


Obrázek 28: Program pro instalaci LeJOS na kartu microSD.

Po instalaci LeJOS na microSD, vložíme kartu do EV3 Bricku, poté spustíme. Instalace LeJOS ještě nějaký čas zabere. Mezitím si můžeme stáhnout vývojové prostředí Eclipse.

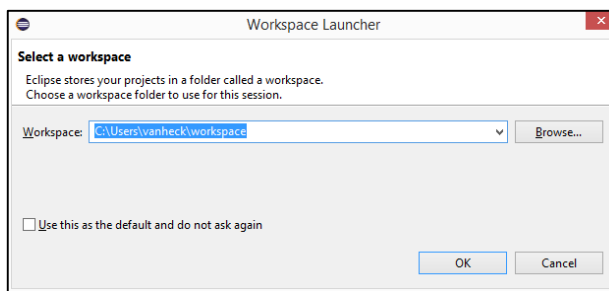
3.3 Instalace vývojového prostředí Eclipse

Ze stránky eclipse.org [17] v sekci Download zvolíme pro jakou verzi Windows chceme Eclipse stáhnout.



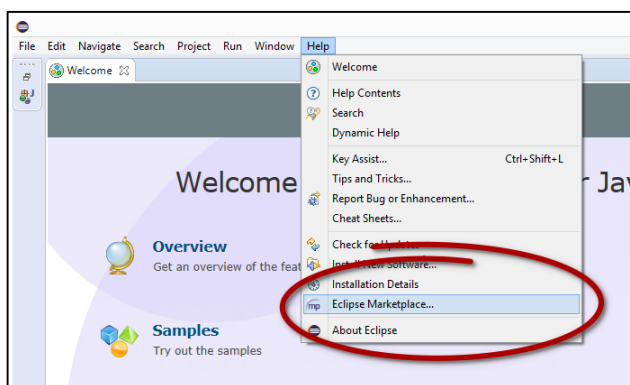
Obrázek 29: Sekce Download na <http://www.eclipse.org/>

Stažený soubor stačí rozbalit a poté spustit *eclipse.exe*. V programu si v prvním okně nejprve nastavíme pracovní adresář a potvrdíme OK.



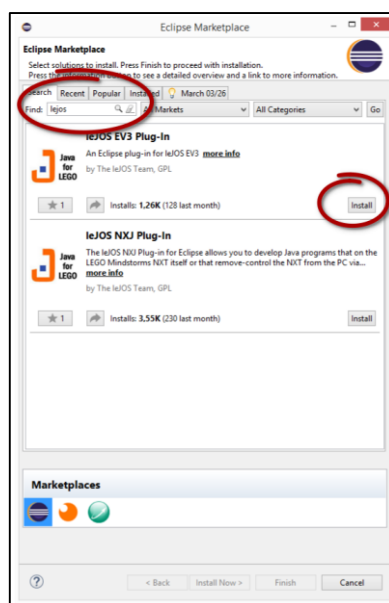
Obrázek 30: Eclipse – v okně po spuštění nastavujeme pracovní adresář.

Nahoře v menu Help → Eclipse Marketplace... můžeme nainstalovat plugin LeJOS.



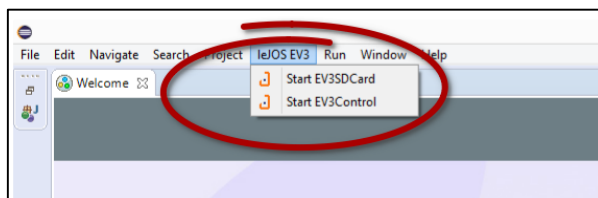
Obrázek 31: V Eclipse Marketplace se nachází plugin LeJOS.

V Marketplace můžeme LeJOS plugin vyhledat a kliknutím na položku Install spustíme instalaci pluginu.



Obrázek 32: Eclipse Marketplace.

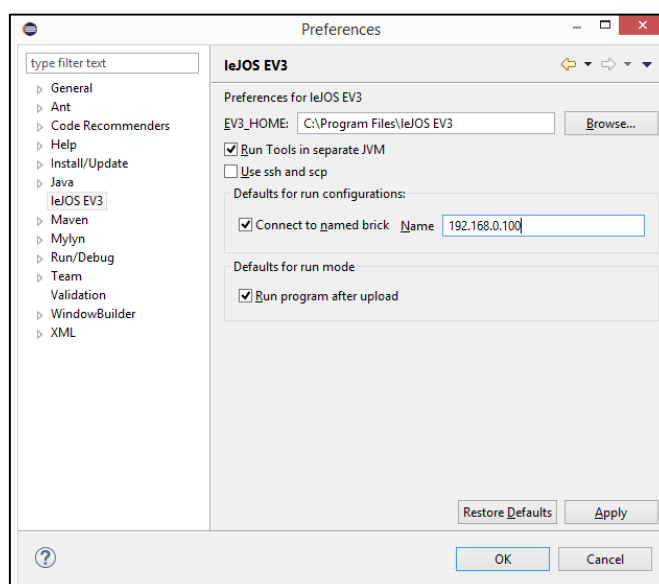
Následné okno stačí potvrdit tlačítkem *Confirm*. Zobrazí se ještě výzva pro souhlas s licenci, akceptujeme licenci a tlačítkem *Finish* pokračujeme dále, nyní se plugin nainstaluje. Po instalaci je ještě třeba Eclipse restartovat.



Obrázek 33: Po instalaci LeJOS pluginu se zobrazí v menu položka s názvem leJOS EV3.

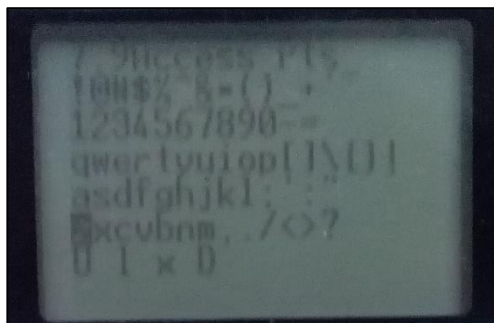
V Eclipse následně můžeme spustit program pro instalaci LeJOS na microSD kartu, pokud bude potřeba.

Pokud máme USB WIFI adaptér a připojený EV3 Brick v síti, v menu Window → Preferences v oddílu *leJOS EV3* nastavíme jeho IP adresu.



Obrázek 34: Eclipse preferences – nastavení leJOS, zadání IP adresy připojeného EV3 Bricku.

Ve spuštěném LeJOS na Bricku se můžeme připojit k síti položkou v menu *Wifi*. Zobrazí se seznam dostupných bezdrátových sítí, po výběru sítě je možné zadat heslo z výběru znaků.



Obrázek 35: Zadávání hesla na wifi v EV3 Bricku.

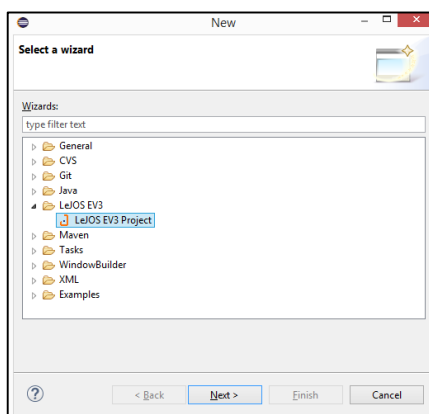
Spodní řádek v zadávání hesla slouží jako ovládací prvky. Další znaky a velká písmena se objeví po vybrání položky *U* ve spodním řádku. Potvrzení zadaného hesla se provádí výběrem položky *D*.

Po úspěšném připojení se IP adresa zobrazí na úvodní obrazovce.



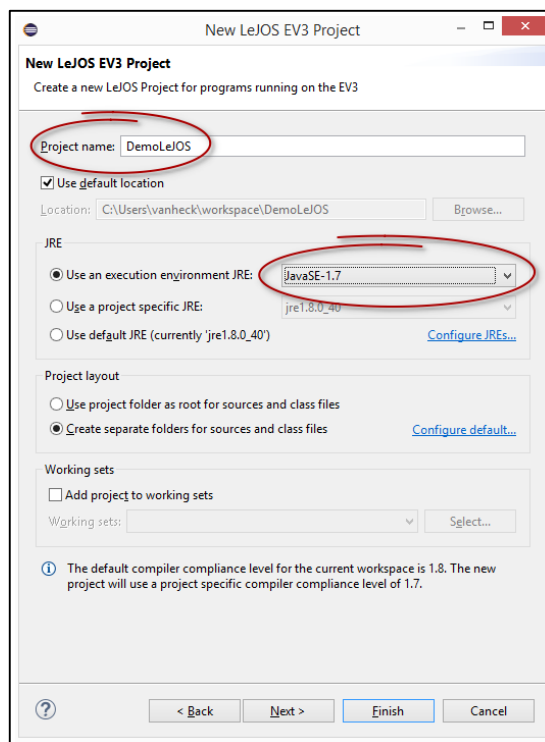
Obrázek 36: Po připojení Bricku do sítě se IP adresa zobrazí na hlavní obrazovce.

Nový projekt pro LeJOS otevřeme pomocí menu File → New → Other..., kde vybereme LeJOS EV3 Project a klikneme na tlačítko *Next*.



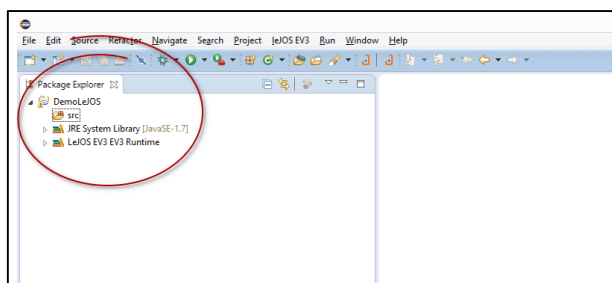
Obrázek 37: Vytváření nového LeJOS projektu.

V další části zadáme název projektu a zkontrolujeme verzi Javy, pro kterou chceme vyvíjet programy. Protože jsme do Bricku nainstalovali verzi Javy 1.7, vybereme i zde verzi 1.7.



Obrázek 38: Vytváření nového LeJOS projektu.

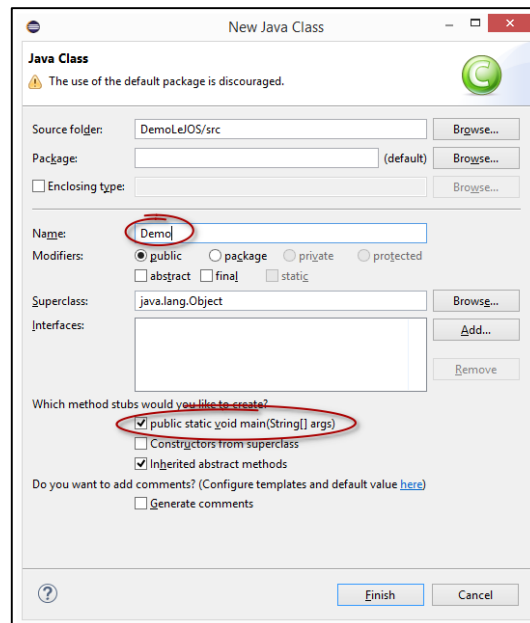
V package exploreru se nám vytvořil nový LeJOS projekt.



Obrázek 39: Nový projekt se zobrazí v package exploreru.

Pravým kliknutím na položku `src` v kontextovém menu vybereme `New` → `Class` se nám otevře pomocné okno, kde zadáme název a zaškrtneme položku `public static void main(String[] args)`. Poté stikneme tlačítko *Finish*. Tím vytvoříme naši první třídu, do které můžeme začít psát kód.


3.4 První program



Obrázek 40: Vytvoření třídy pro programování EV3 Bricku.

V nově vygenerované nové třídy do metody main zapíšeme kód:

```
LCD.drawString("První spusteni", 0, 4);  
Button.waitForAnyPress();
```

V menu Run → Run, nebo tlačítkem  přeložíme a spustíme program. Pokud spouštíme program poprvé, objeví se okno Run As, kde vybereme *LeJOS EV3 Program* a potvrdíme.

Nyní se nám program automaticky spustil přímo v EV3 Bricku, kde můžeme vidět na čtvrtém řádku nápis „První spusteni“. Program ještě čeká na stisk libovolného tlačítka a poté se ukončí.

Úvodní kód se nachází v příloze 0.

4 PRAKTICKÉ ÚLOHY PRO LEGO MINDSTORMS EV3

4.1 Základní seznámení s EV3 Brick

Seznámíme se s ovládáním tlačítek a displeje, zvuku a LED světel pod tlačítka.

4.1.1 Úkoly

1. Vytvořte program, který čeká na stisk kteréhokoli tlačítka, po jeho stisku se rozsvítí LED a na displej se vypíše, které to bylo tlačítko.
2. Přidejte přehrání zvuku po stisku tlačítka.
3. BONUS: Vymyslete své vlastní vylepšení.

4.1.2 Sestavení

Využívá se pouze EV3 Brick.

4.1.3 Program

Zdrojové soubory se nachází v příloze 1.

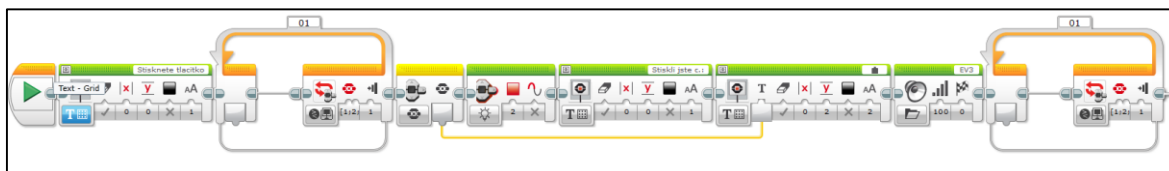
4.1.3.1 Lego Mindstorms EV3 Home Edition

Pro vyspání text na displej se používá blok Display. Nastavíme ho na Text – Grid a v pravém horním rohu bloku nastavíme text, který chceme vypsát, nebo chceme vypsát informaci z jiného bloku.

Blok *Loop* se dá nastavit na vyčkání stisku klávesy.

Z bloku senzoru *Brick Buttons* můžeme získat informaci, které tlačítko bylo stisknuto.

Blok *Sound* dokáže přehrát kterýkoli zvuk.



Obrázek 41: Základní seznámení s EV3 Brick

4.1.3.2 LeJOS

Na displej se vypisuje pomocí metody *drawString()* třídy LCD. Přebírá 3 vstupní parametry, 1. parametr je řetězec, který se má vypsát, 2. parametr je začátek na ose x a 3. parametr je začátek na ose y.

```
LCD.drawString("Stisknete kterkoli", 0, 0);  
LCD.drawString("tlacitko.", 0, 1);
```

Čekání na stisk určité klávesy můžeme pomocí smyčky while a kontroly, zda je dané tlačítko ještě nesepnuto:

```
while(Button.ENTER.isUp());
```

Další způsob, jak můžeme čekat na stisk jakékoli klávesy příkazem:

```
Button.waitForAnyPress();
```

Pokud chceme i zjistit, jaké tlačítko jsme stiskly, vhodnější bude příkaz:

```
int tlacitko = 0;  
while(tlacitko == 0){  
    tlacitko = Button.readButtons();  
}
```

Ten nám zajistí, že ID stisknutého tlačítka budeme mít v proměnné tlacitko.

Přehrání zvuku zajistíme pomocí třídy *Sound*:

```
Sound.twoBeeps();
```

A nakonec LED pod tlačítka zapneme příkazem:

```
Button.LEDPattern(1);
```

4.2 Echo telegraf

Úloha zaměřená pro práci se zvukem. [2]



Obrázek 42: Echo telegraf

4.2.1 Úkoly

1. Sestavte model podle návodu.
2. Naprogramujte EV3 Brick tak, že při stisknutí senzorového tlačítka vydá EV3 zvuk jako telegraf.
3. Naprogramujte EV3 Brick tak, že po zadání Morseovy zprávy, po krátké odmlce tuhle zprávu zopakuje.
4. BONUS: Navrhněte své vlastní vylepšení.

4.2.2 Sestavení

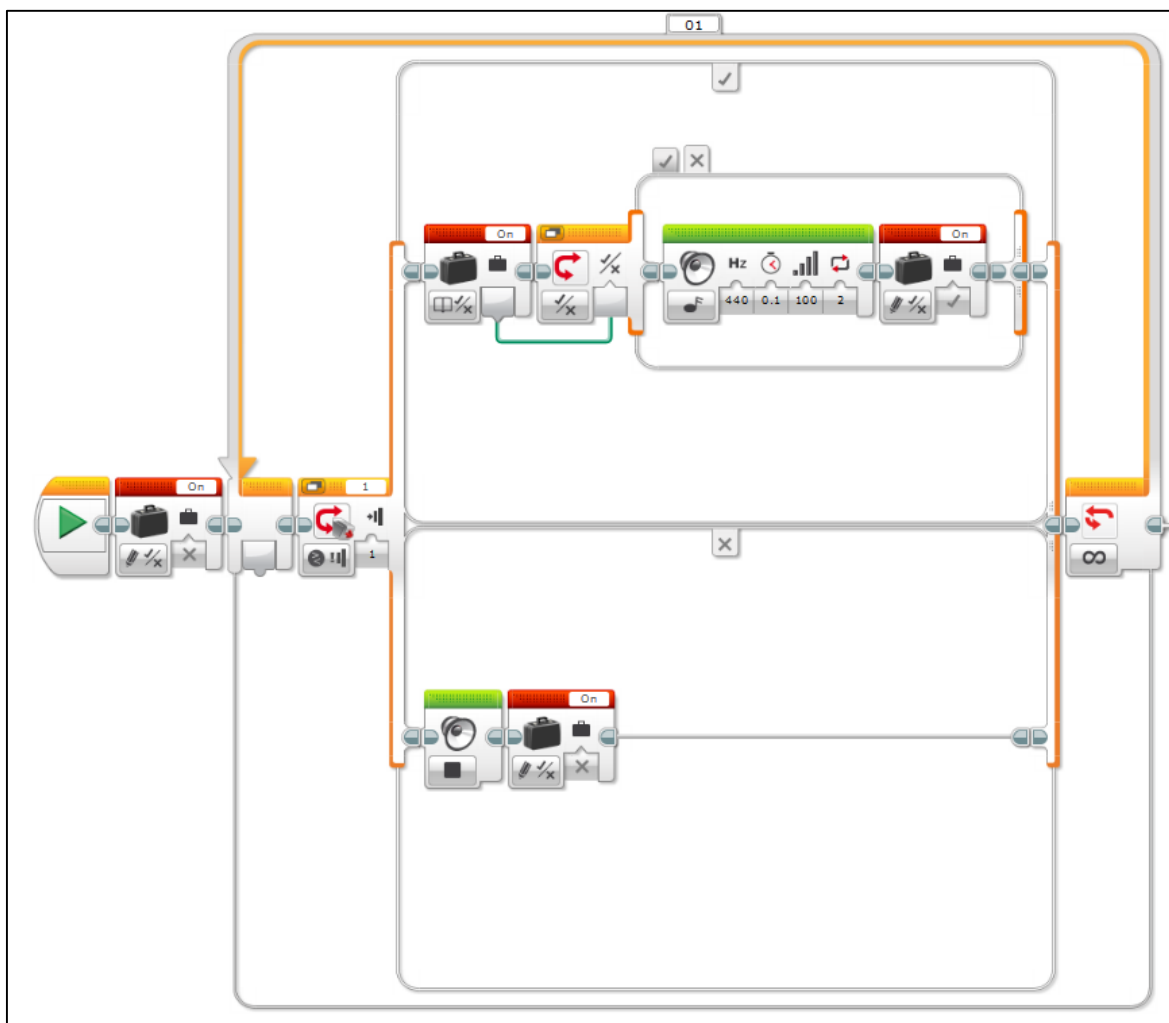
Model i návod na sestavení se nachází v příloze 2.

4.2.3 Program

Zdrojové soubory se nachází v příloze 2.

4.2.3.1 *Lego Mindstorms EV3 Home Edition*

Pro zvuk slouží blok *Sound* v zelené kategorii. Využijeme jeho nastavení *Play Tone*. Díky tomu dokážeme na určitý čas pustit zvuk o určité frekvenci. Abychom mohli ale zvuk držet po dobu, co je stisknut tlačítkový senzor, musíme dát blok *Sound* do smyčky *Loop*. Abychom mohli zvuk zapínat a vypínat, využijeme proměnnou, která nám řekne, zda jsme tlačítko stiskli a stále ho držíme, nebo jsme ho už pustili a musíme tak zvuk vypnout.



Obrázek 43: Program pro echo telegraf.

4.2.3.2 LeJOS

Naprogramovat tento toto zařízení v LeJOS je rychlejší a kód je velmi přehledný. Nejdříve si připravíme senzor. Každý senzor může sbírat data v různých módech, proto musíme ze senzoru získat mód metodou *getTouchMode()*. Metoda vrací interface *SampleProvider*, který popisuje metody pro získávání dat ze senzoru. *SampleProvider* má metodu *fetchSample()*, která právě sběr dat ze senzoru obstará.

```
public static void main(String[] args) {

    EV3TouchSensor touchSensor = new EV3TouchSensor(SensorPort.S1);

    LCD.drawString("Muzete pouzivat", 0, 1);
    LCD.drawString("telegraf.", 0, 2);

    SampleProvider sampleProvider = touchSensor.getTouchMode();

    float sensor[] = new float[sampleProvider.sampleSize()];
    int t = (int) sensor[0];
}
```

```
        while (Button.ESCAPE.isUp() && Button.ENTER.isUp()) {
            sampleProvider.fetchSample(sensor, 0);
            t = (int) sensor[0];
            if (t == 1) {
                Sound.playTone(440, 100, 100);
            }
        }
        touchSensor.close();
    }
```


4.3 Ultrazvukový parkovací asistent

Úloha zaměřená na použití senzoru Ultrasonic. Pomocí senzoru budeme sledovat vzdálenost překážky. Pokud se přiblíží hodně, začne brick pomalu pípat. Čím více se bude překážka přibližovat, tím rychleji bude Brick pípat skoro.

4.3.1 Úkoly

1. Připojte senzor Ultrasonic do senzor portu 1.
2. Naprogramujte EV3 Brick, aby při blízkosti překážky na 30 cm začal pípat.
3. Připravte program, který bude senzorem Ultrasonic sledovat vzdálenost, že čím blíže se překážka ještě přiblíží, tím rychleji začne pípat.

4.3.2 Sestavení

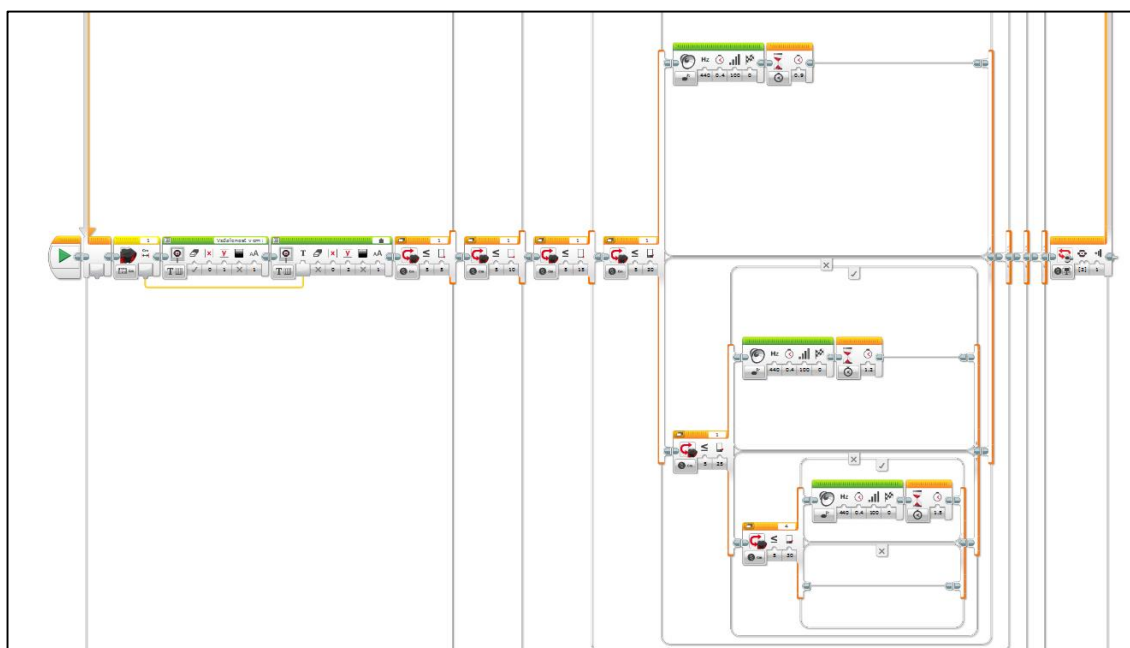
Zapojte senzor Ultrasonic do EV3 Brick, do senzor portu 1.

4.3.3 Program

Zdrojové soubory se nachází v příloze 3.

4.3.3.1 *Lego Mindstorms EV3 Home Edition*

Příklad jednoduchého programu, který je ve vývojovém prostředí Lego Mindstorms hůře čitelný. Celý rozvětvený program by se nevešel na stránku, proto uvádím jen hlavní část:



Obrázek 44: Část programu parkovacího asistentu.

4.3.3.2 LeJOS

Připravíme si senzor a jeho poskytovatele vzorku *SampleProvider*. Poté jen v cyklu budeme sbírat vzdálenost ze senzoru větvením určíme jak často budeme pípat.

```
EV3UltrasonicSensor usonicSensor = new EV3UltrasonicSensor(SensorPort.S1);
SampleProvider spUsonic = usonicSensor.getDistanceMode();
float[] sampleUsonic = new float[spUsonic.sampleSize()];

int vzdalenost;
while (Button.ESCAPE.isUp() && Button.ENTER.isUp()) {
    spUsonic.fetchSample(sampleUsonic, 0);
    vzdalenost = (int) (sampleUsonic[0] * 100);
    LCD.drawString("Vzdalenost: " + vzdalenost + " cm", 0, 3);
    if (vzdalenost <= 5) {
        Sound.playTone(440, 400, 100);
    } else if (vzdalenost <= 10) {
        Sound.playTone(440, 400, 100);
        Delay.msDelay(300);
    } else if (vzdalenost <= 15) {
        Sound.playTone(440, 400, 100);
        Delay.msDelay(600);
    } else if (vzdalenost <= 20) {
        Sound.playTone(440, 400, 100);
        Delay.msDelay(900);
    } else if (vzdalenost <= 25) {
        Sound.playTone(440, 400, 100);
        Delay.msDelay(1200);
    } else if (vzdalenost <= 30) {
        Sound.playTone(440, 400, 100);
        Delay.msDelay(1500);
    }
}
```

4.4 Sekačka

Předělaná verze sekačky na EV3 z nxtprograms.com [2], kde je sekačka provedena ve starší verzi Lego Mindstorms NXT. Jde o základní seznámení se s motory.



Obrázek 45: Sekačka.

4.4.1 Úkoly

1. Sestavte model dle návodu.
2. Naprogramujte model tak, že se bude po stisku tlačítka – ručky model pohybovat dopředu a zároveň se bude točit spodní vrtule, která symbolizuje sekací nože.
3. BONUS: Navrhněte své vlastní vylepšení.

4.4.2 Sestavení

Model i návod na sestavení je v příloze 4.

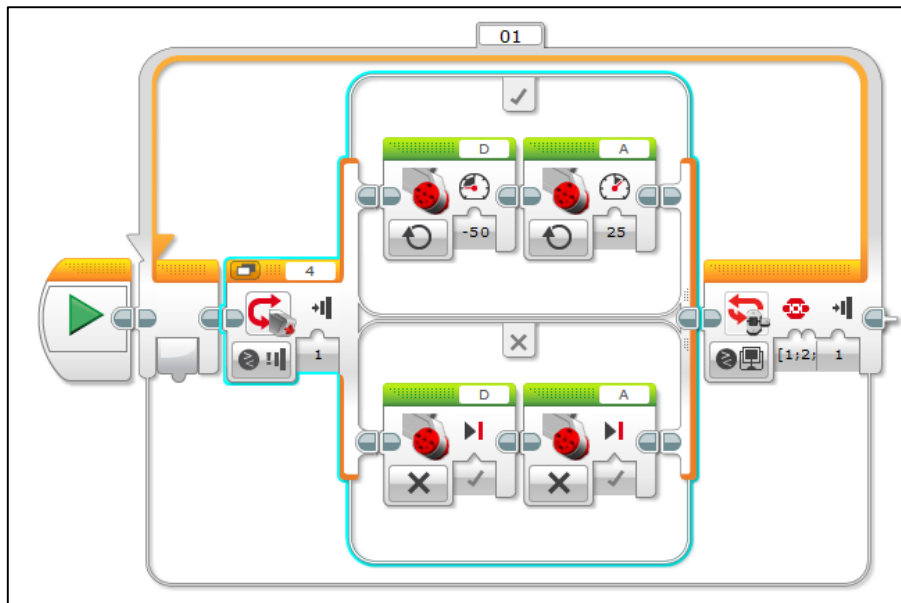
4.4.3 Program

Zdrojové soubory se nachází v příloze 4.

4.4.3.1 *Lego Mindstorms EV3 Home Edition*

Nejprve do programu vložíme smyčku Loop z oranžové kategorie. Do smyčky vložíme blok Switch, který nastavíme na Touch Sensor → Compare → State. Sekačka je provedena z motorů EV3 Large Motor. Těm odpovídá blok Large Motor ze zelené kategorie. Jeden blok Large Motor odpovídá jednomu motoru, proto dáme dva bloky za sebe. Nastavíme

jejich zapnutí a vložíme do připraveného bloku Switch – do vrchní části. Do spodní umístíme stejné bloky, ale nastavíme je na zastavení. Poté můžeme program spustit.



Obrázek 46: Program pro ovládání sekačky.

4.4.3.2 LeJOS

EV3 Large Motors odpovídají třídě *EV3LargeMotors*. Tato třída implementuje interface s názvem *RegulatedMotors*, což je zjednodušeně seznam metod, kterými můžeme ovládat pohyb motorů. Připravíme si tyto objekty:

```
RegulatedMotor m1 = new EV3LargeRegulatedMotor(MotorPort.D);
RegulatedMotor m2 = new EV3LargeRegulatedMotor(MotorPort.A);
```

Připravíme si tlačítko Touch Sensor a jeho „poskytovatele vzorků“ *SampleProvider*:

```
EV3TouchSensor tSensor = new EV3TouchSensor(SensorPort.S4);
SampleProvider spTouch = tSensor.getTouchMode();
float[] sampleTouch = new float[spTouch.sampleSize()];
```

Zbývá ve smyčce rozjet či zastavit motory podle toho, jestli je Touch Sensor zmáčknut nebo ne. Motory se dají do pohybu metodami *forward()* vpřed a *backward()* vzad. Metodou *stop()* je zastavíme.

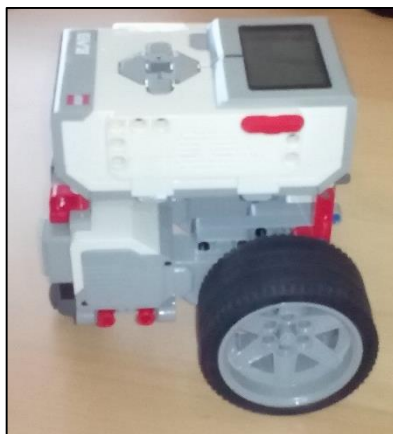
```
LCD.drawString("Sekacka je", 0, 1);
LCD.drawString("nastartovana!", 0, 2);

LCD.drawString("Program ukoncite", 0, 4);
LCD.drawString("stisknutim Escape.", 0, 5);
```

```
int resultTouch = 0;
while(Button.ESCAPE.isUp()){
    spTouch.fetchSample(sampleTouch, 0);
    resultTouch = (int)sampleTouch[0];
    if(resultTouch == 1){
        if(!m1.isMoving()) m1.backward();
        if(!m2.isMoving()) m2.forward();
    }else{
        m1.stop();
        m2.stop();
    }
}
```

4.5 3-kolový robot s detekcí překážek

Jednoduchá úloha na začátek. Cíl úlohy je naučit se ovládat a kontrolovat pohyb robota. V této úloze jde o seznámení se motory. Jak je rozjet, jak ovládat rychlost a polohu.



Obrázek 47: Jednoduchý 3-kolový robot

4.5.1 Úkoly

1. Sestavte model podle návodu.
2. Vyzkoušejte si ovládání modelu. Nejprve rozjed'te robota dopředu – oba motory musí jet stejnou rychlostí.
3. Poté rozjed'te robota dozadu.
4. Nyní si vyzkoušejte změnu rychlosti.
5. Rozjed'te motory každý opačným směrem a vyzkoušejte si tak zatáčení.
6. Nyní aktivujte Ultrasonic senzor pro detekci překážek. Robot pojede dopředu, pokud narazí na překážku, kousek se vrátí, otočí se a bude pokračovat dále dopředu a tak pořád dokola.
7. BONUS - SLALOM: Připravte si prostor a do něj dejte předměty, které bude objíždět. Určete startovací a cílovou pozici a naprogramujte robota tak, aby připravenou dráhu projel.
8. BONUS: Navrhněte své vlastní vylepšení.

4.5.2 Sestavení

Návod i model se nachází v příloze 5.

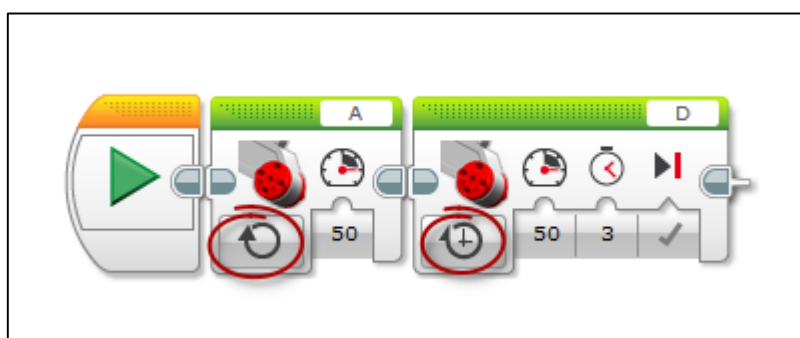
4.5.3 Program

Zdrojové soubory se nachází v příloze 5.

Používáme EV3 Large motory, jež jsou regulované motory, kterým můžeme zadat nejen směr, ale i jejich rychlost nebo i úhel, o který se mají otočit.

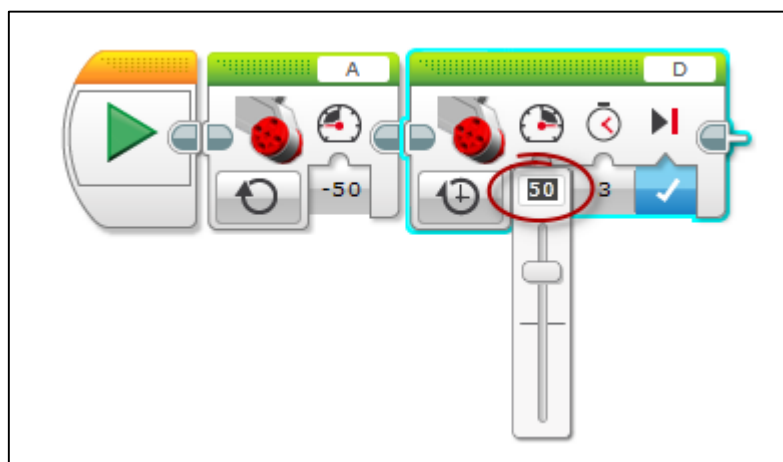
4.5.3.1 Lego Mindstorms EV3 Home Edition

Nejprve si do programu vložíme dva bloky Large Motor. Prvnímu nastavíme On a druhému On for seconds.



Obrázek 48: Program pro spuštění motorů vpřed na 3s.

První vlastností se u motorů udává rychlost motoru v procentech. Pokud se udá v záporných číslech, motor se pohybuje opačným směrem.

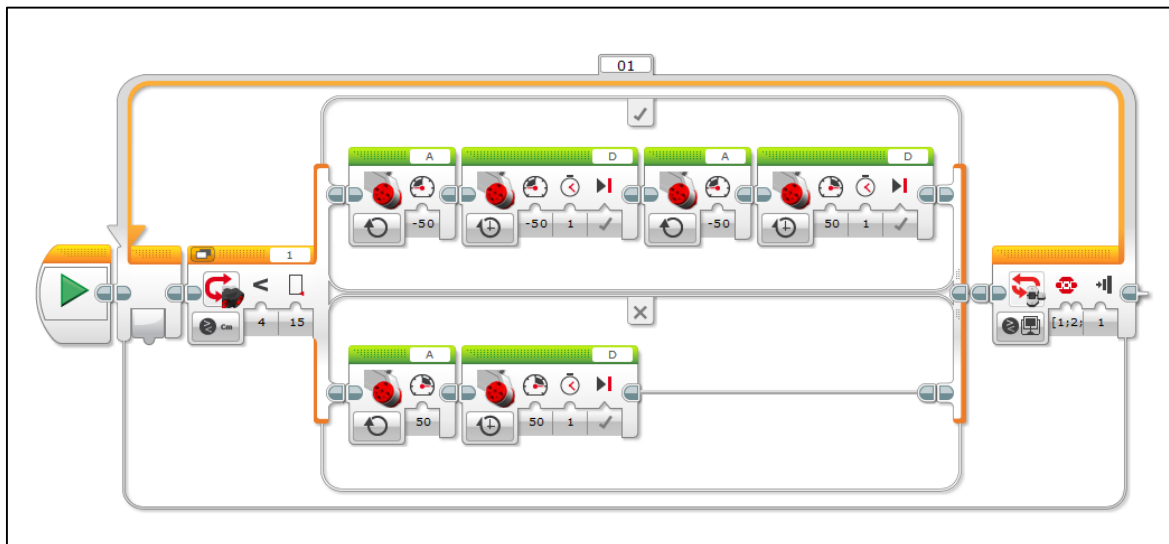


Obrázek 49: Nastavení rychlosti a směru v první vlastnosti motoru.

Pro zatočení je třeba nastavit první vlastnost u jednoho motoru v kladných číslech a u druhého motoru v záporných číslech.

Pro program, který bude kontrolovat vzdálenost a když narazí na překážku, tak couvne a otočí se, budeme potřebovat blok Switch, který nastavíme na porovnávání vzdálenosti

Ultrasonic senzoru. Pokud podmínka projde, zapne zpětný chod motorů, aby couvl a pak zatočí. V případě, že žádnou překážku nedetekuje, pokračuje v pohybu v před. Celý tento kód dáme do bloku Loop, který nastavíme na stisk kteréhokoli tlačítka.



Obrázek 50: Program detekce překážek.

4.5.3.2 LeJOS

LeJOS má pro EV3 Large Motor motory připraven interface s názvem *RegulatedMotor*. Ten slouží pro ovládání pohybu motoru. Připravíme si objekt, abychom s ním pak mohli pracovat. Jako EV3LargeMotor odpovídají objektu třídy *EV3LargeRegulatedMotor* a její konstruktor přebírá jako parametr port, ve kterém je motor zapojen:

```
RegulatedMotor rightMotor = new EV3LargeRegulatedMotor(MotorPort.A);
RegulatedMotor leftMotor = new EV3LargeRegulatedMotor(MotorPort.D);
```

Pro pohyb motoru vpřed a vzad, použijeme metody *forward()* a *backward()*. Kód jen zapne motor a hned vykoná další příkaz. Aby program hned neskončil, pozastavíme ho, k tomu slouží objekt *Delay* a jeho metoda *msDelay()*. Jako parametr uvedeme počet milisekund, na jak dlouho chceme pozastavit vykonávání programu. Motor můžeme zastavit metodou *stop()*, využít metody *stop()* v našem případě ale nemusíme.

```
rightMotor.forward();
leftMotor.forward();

Delay.msDelay(3000);

rightMotor.stop();
leftMotor.stop();

Delay.msDelay(1000);
```



```
rightMotor.backward();  
leftMotor.backward();  
  
Delay.msDelay(3000);
```

Rychlost motoru nastavujeme pomocí metody *setSpeed()*. Jako vstupní parametr zadáme ve stupních za sekundu. Platí, že 100°/s je přibližně na 1V. S AAA bateriemi můžeme dosáhnout maximum cca 900°/s. Maximální možnou rychlost můžeme zjistit metodou *getMaxSpeed()*.

```
rightMotor.setSpeed((int) rightMotor.getMaxSpeed());  
leftMotor.setSpeed((int) leftMotor.getMaxSpeed());  
  
rightMotor.forward();  
leftMotor.forward();  
  
Delay.msDelay(3000);
```

Pro zatáčení se nám bude hodit metoda *rotate()*, jako první vstupní parametr se zadá úhel otočení motoru ve stupních. Přetížení metody *rotate()* zapříčiní, že můžeme/nemusíme zadat druhý parametr typu *boolean*, tedy hodnoty *true* nebo *false*. Pokud druhý parametr nezadáme nebo zadáme hodnotu *false*, vykonávání program vyčká, dokud se nevykoná příkaz – vyčká, dokud se motor opravdu neotočí o zadaný úhel. Zadáním hodnoty *true* jako druhý parametr, program nechá otočit motor a hned bude vykonávat další příkaz.

```
rightMotor.rotate(360, true);  
leftMotor.rotate(-360);  
  
rightMotor.rotate(-720, true);  
leftMotor.rotate(720);
```

Pro detekci překážek slouží Ultrasonic senzor. Reprezentuje ho třída *EV3UltrasonicSensor*, vstupní parametr konstruktoru se zadává opět port, ve kterém je senzor zapojen.

```
EV3UltrasonicSensor usonicSensor = new EV3UltrasonicSensor(SensorPort.S1);
```

LeJOS má svůj vlastní framework pro senzory. Jeden senzor může měřit více způsoby. Tyhle způsoby nazývá LeJOS módy. Ultrasonic senzor může měřit vzdálenost anebo naslouchat jinému ultrasonic senzoru. LeJOS implementuje každý mód jako třídu implementující interface *SampleProvider*, skrze jeho metody můžeme sbírat vzorky dat. Připravíme si *SampleProvider*, pro sběr vzdálenosti:

```
SampleProvider sp = usonicSensor.getDistanceMode();
```

SampleProvider sbírá data a vrací je jako pole *float*. Připravíme si takové pole. Jak má být pole veliké, nám řekne *SampleProvider*. Zároveň si připravíme proměnnou, do které budeme ukládat vzdálenost a můžeme si ji nechat zobrazovat na displeji:

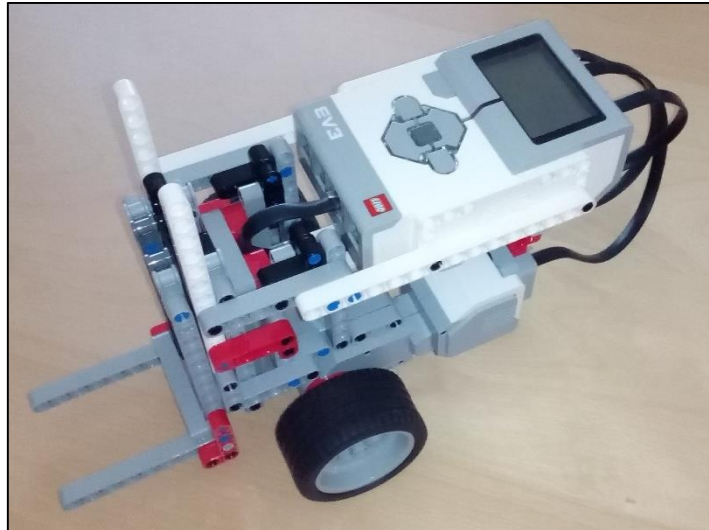
```
float[] sample = new float[sp.sampleSize()];  
int distance;
```

Metodou `fetchSample()` si vyžádáme data ze senzoru. Aktuální data se nám ukrývají v prvním prvku pole. Hodnota výsledku je v metrech, když chceme vzdálenost v cm, musíme výsledek vynásobit 100. Ještě přidáme pohyb robota a dáme do smyčky *while*, která se bude provádět, dokud se nezmačkne tlačítko Escape na EV3 Bricku:

```
while(Button.ESCAPE.isUp()){  
  
    if(!leftMotor.isMoving()){  
        leftMotor.forward();  
        rightMotor.forward();  
    }  
  
    sp.fetchSample(sample, 0);  
    distance = (int)(sample[0]*100);  
    LCD.drawString("Vzdálenost v cm: " + distance, 0, 4);  
    LCD.drawString(" " + distance, 0, 5);  
  
    if(distance<15){  
        leftMotor.backward();  
        rightMotor.backward();  
        Delay.msDelay(1500);  
        leftMotor.rotate(333,true);  
        rightMotor.rotate(-333);  
    }  
  
    Delay.msDelay(10);  
    LCD.clear();  
}
```

4.6 Vysokozdvížený vozík

Pokračování úlohy s motory, kdy se přidává ještě jeden motor pro radlice a při ovládání radlic názorně ukáže, jak můžeme detekovat krajní polohu motoru. Jak poznat, kdy jsme s radlicemi dole nebo nahoře.



Obrázek 51: Vysokozdvížený vozík.

4.6.1 Úkoly

1. Sestavte model dle návodu. Sestavte si paletu.
2. Vyzkoušejte si pohybování s radlicemi – jak poznáme, že je radlice v krajní poloze? Jak poznáme, že je spuštěna dolů? Jak poznáme, že je vysunuta nahoře?
3. Sestavte si paletu, může být podle návodu.
4. Vyzkoušejte si zajet s vozíkem pod paletu, nabrat ji a převést kousek vedle, kde ji položíte.
5. BONUS: Navrhněte a připravte svá vlastní vylepšení.

4.6.2 Sestavení

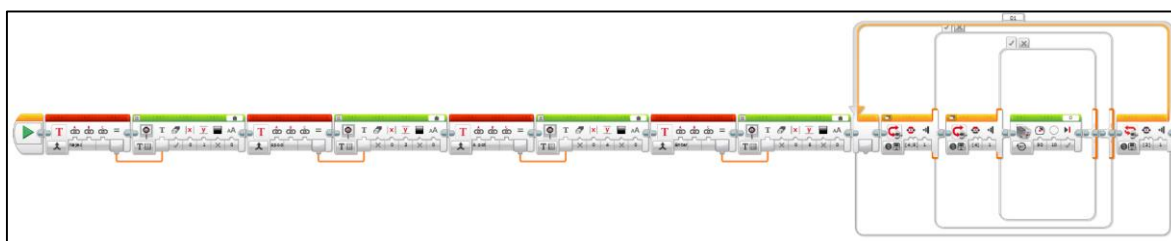
Návod na sestavení vozíku i palety se nachází v příloze 6.

4.6.3 Program

Zdrojové soubory se nachází v příloze 2.

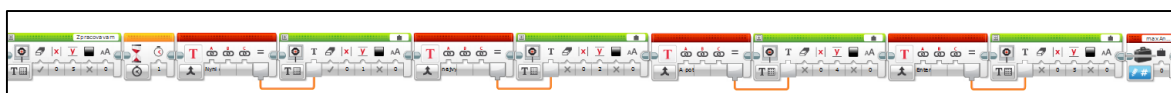
4.6.3.1 Lego Mindstorms EV3 Home Edition

Pro pohyb radlic máme sestavený EV3 Medium Motor, ten reprezentuje blok Medium Motor ze zelené kategorie. Tento blok dáme do splněné podmínky bloku Switch, který nastavíme na tlačítko nahoru. Do nesplněné podmínky bloku Switch dáme další blok Medium Motor. Nyní ho ale nastavíme na pohyb opačným směrem. Celé to dáme do smyčky, kterou nastavíme na tlačítko Enter. Nakonec přidáme na začátek instrukce „*Najedte radlicemi do spodní polohy. A potvrďte stiskem Enter.*“ na displej:



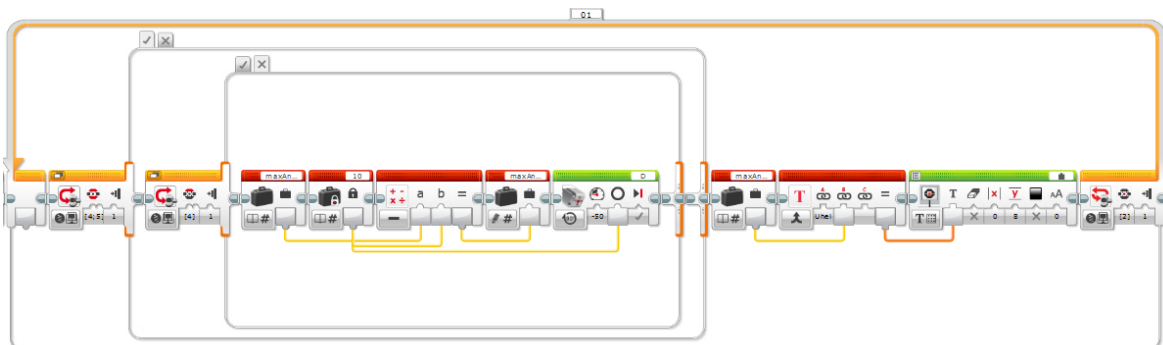
Obrázek 52: První část kódu – nastavení radlic do spodní krajní polohy pomocí tlačítek.

V druhé části kódu vypíšeme další instrukce „*Najedte radlicemi co nejvyše. Potvrďte stiskem Enter.*“ a připravíme si proměnnou, do které budeme ukládat úhel otočení, jejíž počáteční hodnotu nastavíme na 0.



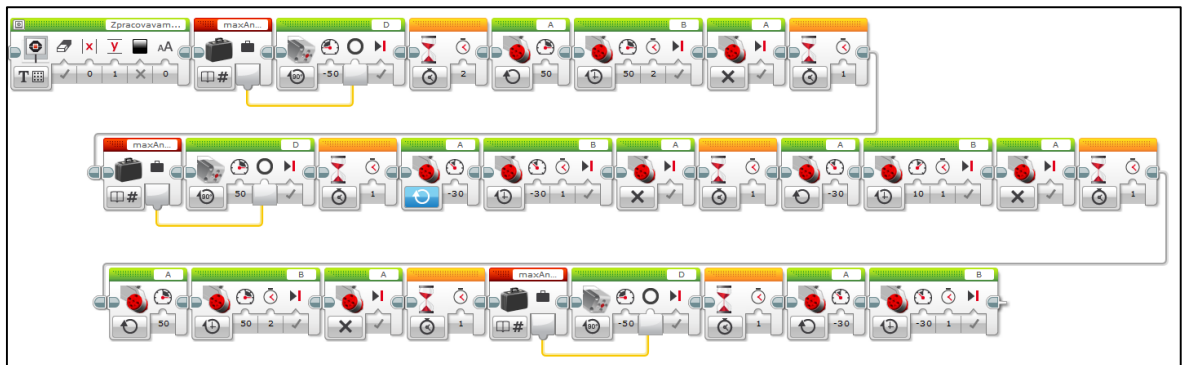
Obrázek 53: Druhá část kódu – vypsání instrukcí na displej a příprava proměnné pro uchování úhlu otočení.

Bude následovat podobná smyčka jako v první části kódu, ale s uchováním úhlu otočení:



Obrázek 54: Třetí část programu – otočení radlic do vrchní krajní polohy

V konečné fázi programu vysokozdvizný vozík najede s radlicemi pod náklad, zvedne ho, převezí na jiné místo, kde ho spustí a nakonec vyjede bez nákladu.



Obrázek 55: Konečná třetí část kódu vysokozdvizného vozíku pro převoz nákladu.

4.6.3.2 LeJOS

Třetí motor má název EV3 Medium Motor. Připravíme si Java objekt pro jeho ovládání:

```
RegulatedMotor ploughshareMotor = new EV3MediumRegulatedMotor(MotorPort.D);
```

Pro nastavení krajních pozic použijeme tlačítka nahoru a dolů:

```
LCD.drawString("Nastavte tlacitky",0,1);
LCD.drawString("nahoru a dolu",0,2);
LCD.drawString("vrchni pozici",0,3);
LCD.drawString("radlic.",0,4);
LCD.drawString("Potvrďte enterem.",0,6);

while(Button.ENTER.isUp()){
    if(Button.UP.isDown()) ploughshareMotor.rotate(20);
    if(Button.DOWN.isDown()) ploughshareMotor.rotate(-20);
}

Button.ENTER.waitForPressAndRelease();

LCD.clear();
ploughshareMotor.flt();

int actualAngle = 0;
int maxAngle = 0;

LCD.drawString("Nastavte spodni",0,1);
LCD.drawString("pozici radlic.",0,2);
LCD.drawString("Potvrďte enterem.",0,6);

while(Button.ENTER.isUp()){
    if(Button.UP.isDown()) {
        ploughshareMotor.rotate(20);
        maxAngle += 20;
    }
}
```

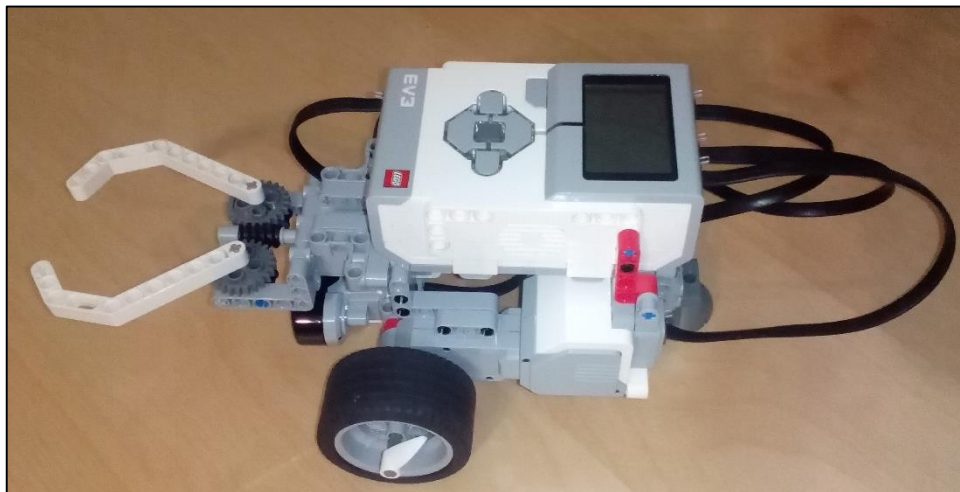
```
    }  
    if(Button.DOWN.isDown()){  
        ploughshareMotor.rotate(-20);  
        maxAngle -= 20;  
    }  
    LCD.drawString("posun o " + Math.abs(maxAngle),0,3);  
    Delay.msDelay(10);  
}  
  
LCD.clear();  
ploughshareMotor.flt();  
maxAngle = Math.abs(maxAngle);  
  
LCD.drawString("Uhel otoceni je:", 0, 1);  
LCD.drawString(maxAngle + " stupnu", 0, 2);  
  
LCD.drawString("Potvrďte enterem", 0, 6);  
  
Button.ENTER.waitForPressAndRelease();
```

Následně můžeme použít pohyb vozíku a radlic, abychom převezli náklad:

```
leftMotor.forward();  
rightMotor.forward();  
Delay.msDelay(1000);  
  
leftMotor.stop();  
rightMotor.stop();  
  
ploughshareMotor.rotate(maxAngle);  
  
leftMotor.backward();  
rightMotor.backward();  
Delay.msDelay(500);  
  
leftMotor.forward();  
Delay.msDelay(500);  
  
rightMotor.forward();  
Delay.msDelay(1000);  
  
rightMotor.stop();  
leftMotor.stop();  
  
ploughshareMotor.rotate(-maxAngle);  
  
rightMotor.backward();  
leftMotor.backward();  
Delay.msDelay(1000);  
  
rightMotor.flt();  
leftMotor.flt();  
ploughshareMotor.flt();  
  
LCD.drawString("Hotovo", 0, 4);  
  
Button.waitForAnyPress();
```

4.7 Robot sběrač

Obdobná úloha jako vysokozdvížený vozík, nyní ale s robotickou rukou namísto radlic. [19]



Obrázek 56: Robot sběrač s robotickou rukou.

4.7.1 Úkoly

1. Sestavte robota podle návodu
2. Připravte si předmět, který bude moci robot uchytit.
3. Naprogramujte robota tak, aby k předmětu přijel, uchytil ho, převezl na jiné místo a pustil.
4. BONUS: Navrhněte svá vlastní vylepšení.

4.7.2 Sestavení

Model a návod na sestavení se nachází v příloze 7.

4.7.3 Program

Zdrojové soubory se nachází v příloze 7.

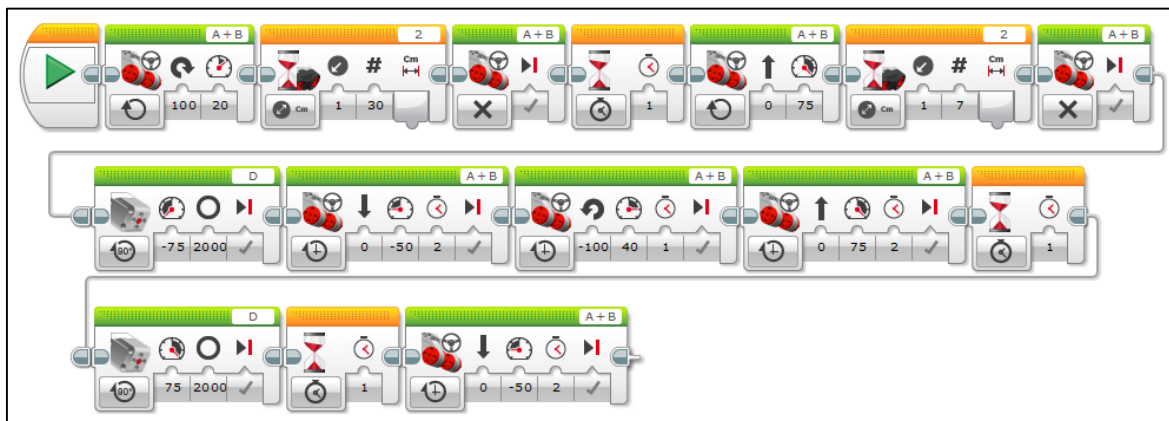
4.7.3.1 *Lego Mindstorms EV3 Home Edition*

Zelený blok *Move Steering* umožňuje synchronizovaně ovládat dva motory. Pokud ho nastavíme, aby jel dopředu, zapne oba dva motory současně a přesně.

Můžeme využít oranžového bloku *Wait*, který nastavíme, aby čekal na vzdálenost od objektu v centimetrech, detekovanou Ultrasonic senzorem.

Následně pomocí totožného principu zadáme přijetí k překážce. Poté uchopíme pomocí Medium motoru předmět. Úhel otočení nastavíme na 2000.

Poté objekt přesuneme s pomocí nám známých bloků.



Obrázek 57: Program robota sběrače.

4.7.3.2 LeJOS

V LeJOS se pro synchronizaci otáčení motoru používají metody *synchronizeWith()*, *startSynchronization()* a *endSynchronization()* z interface *RegulatedMotor*. Metoda *synchronizeWith()* jako vstupní parametr požaduje pole prvků *RegulatedMotor*, což je v podstatě seznam motorů, které mají být mezi sebou synchronizovány.

Připravíme si metody, které nám synchronizovaně zapne motory směrem vpřed, směrem vzad, zatočení a metodu, která motory synchronizovaně zastaví.

```
public static void synchronizeForward(RegulatedMotor motor1, RegulatedMotor
motor2)
{
    RegulatedMotor [] motorsSynchronization = {motor2};
    motor1.synchronizeWith(motorsSynchronization);

    motor1.startSynchronization();
    motor1.forward();
    motor2.forward();
    motor1.endSynchronization();
}
```

```
public static void synchronizeBackward(RegulatedMotor motor1, RegulatedMotor
motor2)
{
    RegulatedMotor [] motorsSynchronization = {motor2};
    motor1.synchronizeWith(motorsSynchronization);

    motor1.startSynchronization();
    motor1.backward();
    motor2.backward();
}
```



```

        motor1.endSynchronization();
    }

    public static void synchronizeLeft(RegulatedMotor rightMotor, RegulatedMotor
leftMotor)
    {
        RegulatedMotor [] motorsSynchronization = {leftMotor};
        rightMotor.synchronizeWith(motorsSynchronization);

        rightMotor.startSynchronization();
        rightMotor.forward();
        leftMotor.backward();
        rightMotor.endSynchronization();
    }

    public static void synchronizeRight(RegulatedMotor rightMotor, RegulatedMotor
leftMotor)
    {
        RegulatedMotor [] motorsSynchronization = {leftMotor};
        rightMotor.synchronizeWith(motorsSynchronization);

        rightMotor.startSynchronization();
        rightMotor.backward();
        leftMotor.forward();
        rightMotor.endSynchronization();
    }

    public static void synchronizeStop(RegulatedMotor motor1, RegulatedMotor
motor2)
    {
        motor1.startSynchronization();
        motor1.stop();
        motor2.stop();
        motor1.endSynchronization();
    }

```

V metodě *main()* si připravíme motory a senzor. Využijeme připravené metody synchronizovaného pohybu v sestaveném programu:

```

    public static void main(String[] args) {

        RegulatedMotor rightMotor = new
EV3LargeRegulatedMotor(MotorPort.A);
        RegulatedMotor leftMotor = new EV3LargeRegulatedMotor(MotorPort.B);
        EV3MediumRegulatedMotor handMotor = new
EV3MediumRegulatedMotor(MotorPort.D);

        EV3UltrasonicSensor usonicSensor = new
EV3UltrasonicSensor(SensorPort.S2);
        SampleProvider sp = usonicSensor.getDistanceMode();
        float[] sample = new float[sp.sampleSize()];
        int distance=1000;

        synchronizeLeft(rightMotor, leftMotor);
    }

```

```
while(distance>30){
    LCD.clear();
    sp.fetchSample(sample, 0);
    distance = (int)(sample[0]*100);
    LCD.drawString("Vzdalenost v cm: " + distance, 0, 4);
    LCD.drawString(" " + distance, 0, 5);
    Delay.msDelay(100);
}

synchronizeStop(leftMotor, rightMotor);
Delay.msDelay(1000);

synchronizeForward(rightMotor, leftMotor);

while(distance>7){
    LCD.clear();
    sp.fetchSample(sample, 0);
    distance = (int)(sample[0]*100);
    LCD.drawString("Vzdalenost v cm: " + distance, 0, 4);
    LCD.drawString(" " + distance, 0, 5);
    Delay.msDelay(100);
}

synchronizeStop(leftMotor, rightMotor);
Delay.msDelay(1000);

handMotor.rotate(-2000);
handMotor.flt();
Delay.msDelay(1000);

synchronizeBackward(rightMotor, leftMotor);
Delay.msDelay(1000);
synchronizeStop(rightMotor, leftMotor);
Delay.msDelay(1000);
synchronizeLeft(rightMotor, leftMotor);
Delay.msDelay(1000);
synchronizeForward(rightMotor, leftMotor);
Delay.msDelay(3000);
synchronizeStop(rightMotor, leftMotor);
Delay.msDelay(1000);

handMotor.rotate(2000);
handMotor.flt();
Delay.msDelay(1000);

synchronizeBackward(rightMotor, leftMotor);
Delay.msDelay(1000);
synchronizeStop(rightMotor, leftMotor);
}
```

4.8 Sumo robot

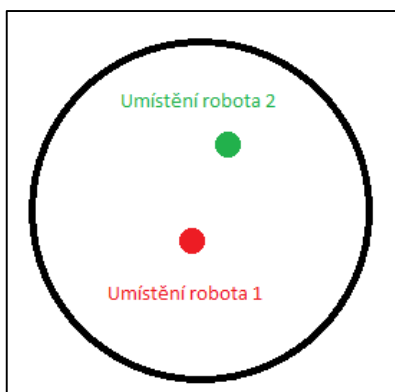
Úloha pro orientaci v prostoru. Úloha je vhodná pro soupeření mezi více roboty. Jedná se o kombinaci sestavení, ovládání pohybu a senzorů za účelem vytlačení překážek z bojového „ringu“.



Obrázek 58: Sumo robot

4.8.1 Úkoly

1. Sestavte robota podle návodu
2. Připravte si hřiště – ring bude jednobarevný, např. bílý, a hranice bude mít barvu odlišnou, např. černou.



Obrázek 59: Příklad hřiště.

3. Naprogramujte robota tak, aby vyhledal ve svém zorném poli překážky nebo robota soupeře a vytlačil ho z hřiště.
4. BONUS: Naprogramujte senzor tlačítka tak, aby na začátku robot čekal s jakoukoli akcí, až se stiskne tlačítko. Tlačítko bude fungovat jako zapnutí.
5. BONUS: Navrhněte a proveďte své vlastní vylepšení robota i programu tak, aby porazil soupeře.

4.8.2 Sestavení

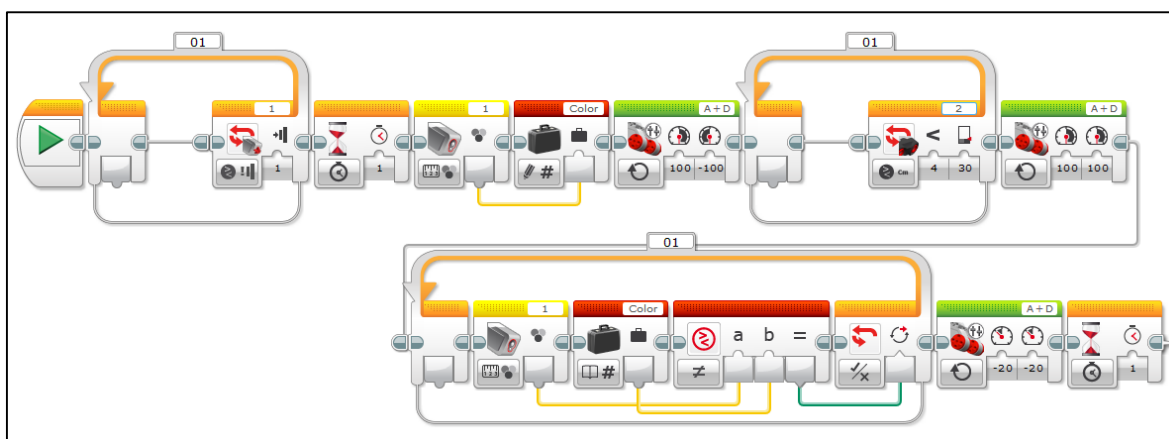
Model a návod na sestavení se nachází v příloze 8.

4.8.3 Program

4.8.3.1 *Lego Mindstorms EV3 Home Edition*

Zde využijeme bloku nazvaného Move Tank. Ten pracuje podobně jako blok MoveSteering, kterému můžeme nastavit i postupné zatáčení. Na rozdíl od něj, Move Tank aktivuje motory pouze vpřed a vzad.

Využijeme jednotné barvy hřiště a na začátku si nastavíme barvu pozadí do proměnné. Barvu hřiště budeme v průběhu porovnávat, pokud narazíme na změnu, musíme provést akci – zastavit motory nebo nabrat zpětný chod.



Obrázek 60: Program sumo robota.

4.8.3.2 *Le.JOS*

Na tomhle příkladu využijeme kód z předchozího modelu synchronizace pohybu motorů.

Abychom ale nemuseli psát tolik kódu, vytvoříme si nyní vlastní třídu, kterou pojmenujeme *MyMotorsManager*. Třída bude obsluhovat synchronní pohyb motorů za nás. Vytvoříme si

tak podobný blok, jako je ve vývojovém prostředí Lego Mindstorms EV3 Move Tank. Pomocí metod třídy pak jen dáme pokyn, jakým směrem chceme, aby se robot vydal, a třída se už o to postará sama. Protože v našem případě je příhodné, aby se motor pohyboval co nejrychleji, připravíme si ještě metodu *setMaxSpeed()* pro nastavení nejvyšší možné rychlosti:

```
public class MyMotorsManager {

    private RegulatedMotor rightMotor;
    private RegulatedMotor leftMotor;

    public MyMotorsManager(RegulatedMotor rightMotor, RegulatedMotor
leftMotor) {
        this.rightMotor = rightMotor;
        this.leftMotor = leftMotor;
        RegulatedMotor [] motorsSynchronization = {leftMotor};
        rightMotor.synchronizeWith(motorsSynchronization);
    }

    public void synchronizeForward()
    {
        rightMotor.startSynchronization();
        rightMotor.forward();
        leftMotor.forward();
        rightMotor.endSynchronization();
    }

    public void synchronizeBackward()
    {
        rightMotor.startSynchronization();
        rightMotor.backward();
        leftMotor.backward();
        rightMotor.endSynchronization();
    }

    public void synchronizeLeft()
    {
        rightMotor.startSynchronization();
        rightMotor.forward();
        leftMotor.backward();
        rightMotor.endSynchronization();
    }

    public void synchronizeRight()
    {
        rightMotor.startSynchronization();
        rightMotor.backward();
        leftMotor.forward();
        rightMotor.endSynchronization();
    }

    public void synchronizeStop()
    {
        rightMotor.startSynchronization();
```

```

        rightMotor.stop();
        leftMotor.stop();
        rightMotor.endSynchronization();
    }

    public void setMaxSpeed() {
        rightMotor.setSpeed((int)rightMotor.getMaxSpeed());
        leftMotor.setSpeed((int)leftMotor.getMaxSpeed());
    }
}

```

Na začátku programu definujeme instanci třídy *MyMotorsManager* a jako vstupní parametry mu předáme motory, které chceme ovládat a zavoláme metodu *setMaxSpeed()*, čímž nastavíme maximální rychlost motorů:

```

MyMotorsManager motorsManager =
    new MyMotorsManager(new EV3LargeRegulatedMotor(MotorPort.A),
                        new EV3LargeRegulatedMotor(MotorPort.D));

motorsManager.setMaxSpeed();

```

Dále si připravíme senzory, jejich „odchytávače vzorků“ a pole, do kterého vzorky budeme odchytávat. Externí tlačítko, zapojené do portu senzorů, je v podstatě také senzor. Jeho název je EV3 Touch Sensor, proto se k němu budeme chovat jako k senzoru:

```

EV3TouchSensor tSensor = new EV3TouchSensor(SensorPort.S4);
EV3UltrasonicSensor usonicSensor = new EV3UltrasonicSensor(SensorPort.S2);
EV3ColorSensor cSensor = new EV3ColorSensor(SensorPort.S1);

SampleProvider spUsonic = usonicSensor.getDistanceMode();
SampleProvider spTouch = tSensor.getTouchMode();
SampleProvider spColor = cSensor.getColorIDMode();

float[] sampleUsonic = new float[spUsonic.sampleSize()];
float[] sampleTouch = new float[spTouch.sampleSize()];
float[] sampleColor = new float[spColor.sampleSize()];

```

Nyní vypíšeme na displej instrukce pro stisknutí senzorového tlačítka pro start celé akce:

```

LCD.drawString("Zadnim tlacitkem", 0, 1);
LCD.drawString("spustite robota.", 0, 2);

int resultTouch = 0;
while(resultTouch == 0){
    spTouch.fetchSample(sampleTouch, 0);
    resultTouch = (int)sampleTouch[0];
    Delay.msDelay(100);
}

```

A zbývá doplnit zbytek programu:

```

LCD.clear();
LCD.drawString("Spusteno, pracuji!", 0, 1);
Delay.msDelay(1000);

```

```
motorsManager.synchronizeLeft();

int distance=1000;
while(distance>50){
    LCD.clear();
    spUsonic.fetchSample(sampleUsonic, 0);
    distance = (int)(sampleUsonic[0]*100);
    LCD.drawString("Vzdalenost v cm: " + distance, 0, 4);
    LCD.drawString("    " + distance, 0, 5);
    Delay.msDelay(100);
}

motorsManager.synchronizeStop();
motorsManager.synchronizeForward();

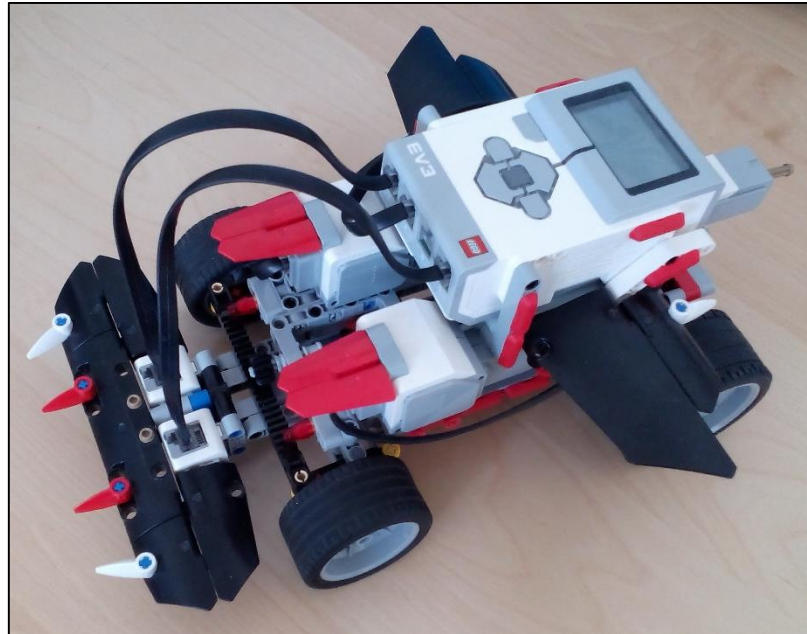
spColor.fetchSample(sampleColor, 0);
int defaultColor = (int)sampleColor[0];

int color = defaultColor;
while(defaultColor == color){
    spColor.fetchSample(sampleColor, 0);
    color = (int)sampleColor[0];
    Delay.msDelay(100);
}

motorsManager.synchronizeStop();
motorsManager.synchronizeBackward();
Delay.msDelay(1000);
}
```

4.9 Auto s klasickým řízením

Auto s klasickým řízením se hodí pro sledování čáry a také jako názorná ukázka užitečných mechanických prvků a sestavení. Model využívá mechaniku zatáčení jako v reálných autech a zadní náhon pomocí diferenciálu.



Obrázek 61: Auto s klasickým řízením

4.9.1 Úkoly

1. Sestavte model auta, který bude mít dva senzory barvy.
2. Připravte pro něj dráhu, čáru, která bude složena ze tří barev.
3. Naprogramujte model tak, aby kopíroval svůj pohyb podle čáry.
4. BONUS: Navrhněte své vlastní vylepšení.

4.9.2 Sestavení

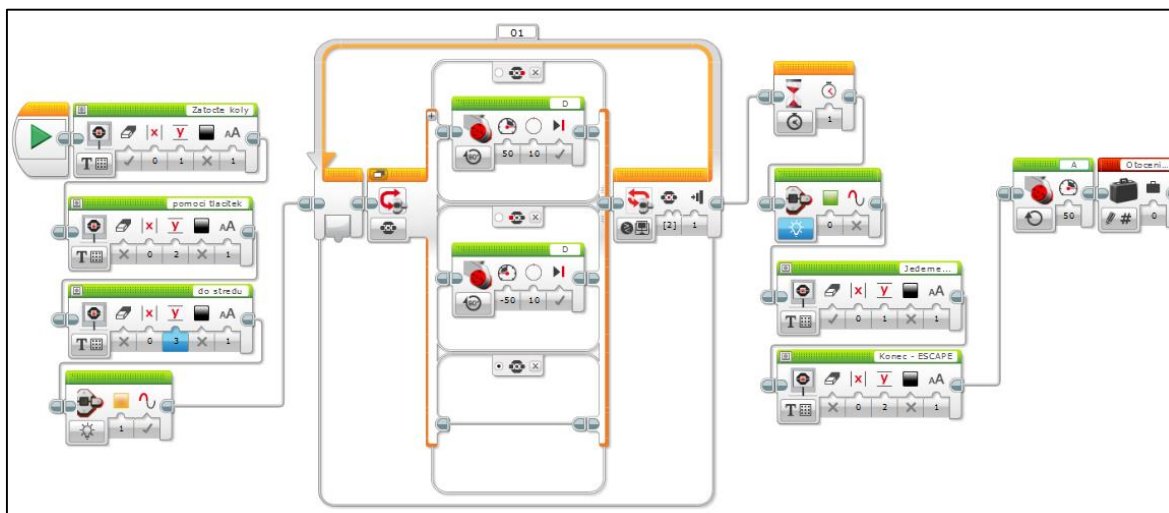
Model i návod na sestavení se nachází v příloze 9.

4.9.3 Program

Zdrojové soubory se nachází v příloze 2.

4.9.3.1 Lego Mindstorms EV3 Home Edition

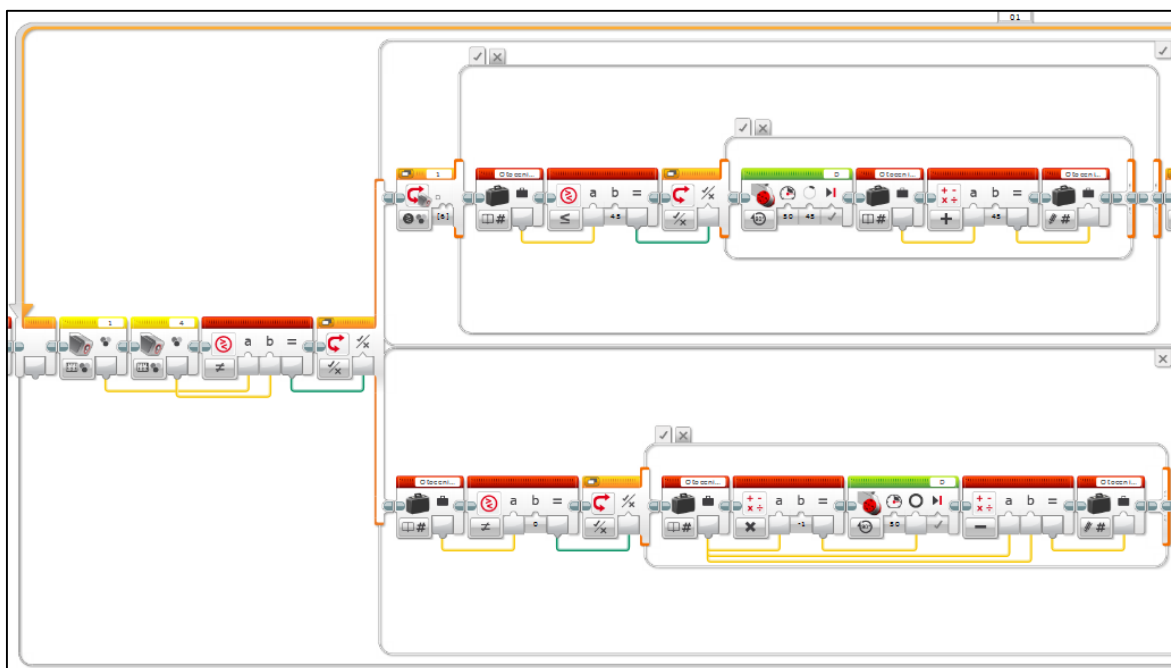
Nejprve vypíšeme instrukce pro natočení kol do středu a zadáme smyčku, která čeká na potvrzovací tlačítko. Ve smyčce budou bloky, které zajistí, že při stisknutí tlačítka vlevo a vpravo se pohne motor pro ovládání předních kol požadovaným směrem.



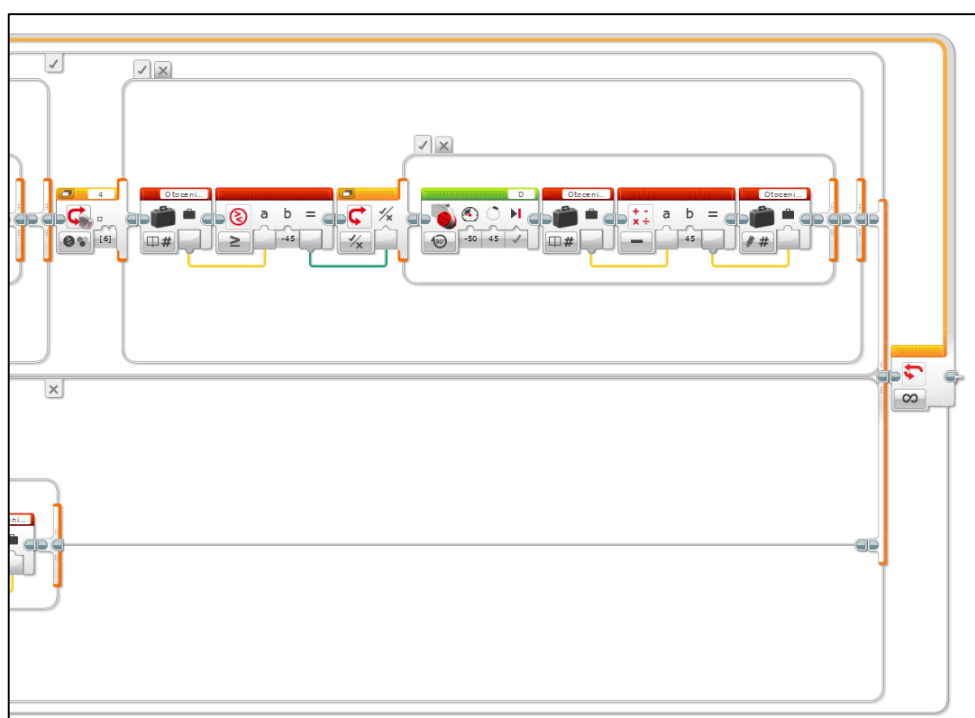
Obrázek 62: První část programu slouží pro nastavení předních kol na střed.

V další části programu je smyčka, která se stará o zatáčení modelu. V proměnné *OtoceniMotoru* se ukládá úhel otočení. Díky této proměnné můžeme zajistit otočení o přesný úhel od středu otáčení.

O zbytek funkcionality se již stará větvení programu bloky Switch.



Obrázek 63: Druhá část programu – část smyčky Loop, která se stará o zatažení podle detekce bílé barvy na senzorech barvy.



Obrázek 64: Třetí část programu – konečná část smyčky Loop, která se stará o zatažení podle detekce bílé na senzorech barvy.

4.9.3.2 LeJOS

Nejprve si připravíme senzory a motory:

```
EV3ColorSensor sensorL = new EV3ColorSensor(SensorPort.S1);
EV3ColorSensor sensorR = new EV3ColorSensor(SensorPort.S4);

RegulatedMotor gearMotor = new EV3LargeRegulatedMotor(MotorPort.A);
RegulatedMotor steeringMotor = new EV3LargeRegulatedMotor(MotorPort.D);
```

Nyní vypíšeme instrukce pro natočení kol do středu (do střední polohy) a pak za pomoci tlačítek uživatel natočí přední kola do středu a potvrdí enterem:

```
LCD.drawString("Zatocte koly", 0, 1);
LCD.drawString("pomoci tlacitek", 0, 2);
LCD.drawString("do stredu.", 0, 3);
Button.LEDPattern(4);

while (Button.ENTER.isUp()) {
    if (Button.RIGHT.isDown())
        steeringMotor.rotate(10);
    if (Button.LEFT.isDown())
        steeringMotor.rotate(-10);
}

LCD.clear();
Button.LEDPattern(0);
Delay.msDelay(1000);
```

A o sledování čáry už se postará program:

```
LCD.drawString("JEDEME...", 3, 1);
LCD.drawString("Konec - ESCAPE", 3, 7);

gearMotor.forward();
DetectedLine detectedLine = DetectedLine.MIDDLE;
while (Button.ESCAPE.isUp()) {
    int colorL = sensorL.getColorID();
    int colorR = sensorR.getColorID();
    LCD.drawString("BarvaL=" + colorL, 1, 3);
    LCD.drawString("BarvaR=" + colorR, 1, 4);
    if (colorR != colorL) {
        if (colorR == Color.WHITE) {
            detectedLine = DetectedLine.RIGHT;
            steeringMotor.rotateTo(45, true);
        }
        if (colorL == Color.WHITE) {
            detectedLine = DetectedLine.LEFT;
            steeringMotor.rotateTo(-45, true);
        }
    }
    if (colorL == Color.WHITE && colorR == Color.WHITE) {
        detectedLine = DetectedLine.MIDDLE;
        steeringMotor.rotateTo(0, true);
    }
    if (colorL != Color.WHITE && colorR != Color.WHITE) {
        switch (detectedLine) {
```

```
        case MIDDLE:
            steeringMotor.rotateTo(0, true);
            break;
        case RIGHT:
            steeringMotor.rotateTo(90, true);
            break;
        case LEFT:
            steeringMotor.rotateTo(-90, true);
            break;
    }
}

gearMotor.close();
steeringMotor.close();
sensorL.close();
sensorR.close();
```

V programu používáme výčet prvků *enum DetectLine*, aby nám zpřehlednil kód a věděli již na první pohled, co se na kterém řádku děje. Tento výčet si připravíme ve třídě *Auto*:

```
enum DetectedLine {
    RIGHT, LEFT, MIDDLE
};
```

4.10 Hádání čísel

Jednoduchá hra, která pracuje pouze s EV3 Brickem a jeho tlačítky. Hra je určená pro výuku programovacího jazyka Java, takže pro LeJOS.

Hra je určena pro dva hráče – zde člověk vs. EV3. Jeden si vymyslí číslo v určitém rozmezí a druhý začne hádat číslo. První porovná číslo se svým vymyšleným, pokud je hádané číslo vyšší, řekne prvnímu, aby hádal dál vyšší číslo a stejně tak pro číslo nižší. Zajímá nás, jak rychle se ten, který hádá číslo, dobere výsledku.

Názorný příklad: Počítač si myslí číslo z rozmezí 0-20 a člověk hádá. Člověk řekne 8, počítač porovná číslo se svým a řekne, že se člověk netrefil a musí hádat nižší číslo. Člověk znovu hádá a nyní volí číslo 4. Počítač opět porovná a odpoví, že člověk musí hádat číslo vyšší. Tohle se opakuje do té doby, než člověk opravdu číslo uhádne.

Zde je možné využít rekurzi, což je pokročilá technika programování a chytré algoritmy rekurzy využívají.

4.10.1 Úkoly

1. Naprogramujte úvodní menu v EV3 Brick tak, že na začátku si tlačítkem vlevo vybereme, že člověk si bude myslet číslo a EV3 ho bude hádat. A tlačítkem vpravo vybereme, že EV3 bude vybírat číslo a člověk ho bude hádat.
2. Naprogramujte část kódu, který následuje po stisku tlačítka vpravo – EV3 vybírá číslo a člověk hádá. Na začátku počítač vybere číslo z rozmezí 0-20. Člověk bude hádat tak, že se zobrazí úvodní číslo, které se bude moct změnit tlačítky nahoru a dolů nebo enterem potvrdit. Po potvrzení EV3 porovná potvrzené číslo se svým a vypíše výsledek – uhodnutí a konec hry, nebo že se má hádat znovu. Pokud se bude hádat znovu, vypíše se i jestli má člověk hádat menší nebo větší číslo.
3. Naprogramujte část kódu, která následuje po stisku tlačítka vlevo – člověk si vymyslí číslo v rozmezí 0-20 a EV3 hádá. EV3 zkusí tipnout číslo a člověk mu tlačítky řekne, zda má hádat vyšší/nižší číslo nebo uhodl. Celé hádání se opakuje, dokud EV3 číslo neuhodne.
4. Pokud jste tak neučinili, zkuste naprogramovat EV3 tak, aby hádala půlením vyhledávaného intervalu.

Například: Člověk si vymyslí číslo 14 z rozmezí 0-20. EV3 hledá polovinu z intervalu 0-20, což je 10. Člověk mu řekne, že má hledat vyšší číslo. Nyní EV3 ví,

že se číslo nachází mezi 10-20, číslo v polovině tohoto intervalu je 15. EV3 řekne 15. Člověk mu řekne, že má hledat menší. Takže EV3 ví, že nyní je hledané číslo mezi 10-15. Nyní chceme vzít polovinu, což by mělo být 12,5, ale pracujeme s celými čísly, proto řekneme 12. Člověk řekne, aby EV3 hledalo vyšší číslo. Nyní je náš interval 13-15. Polovina je 14, EV3 konečně našel výsledek.

Tip: Pro půlení intervalu se využívá rekurze.

5. BONUS: Navrhněte svá vlastní vylepšení.

4.10.2 Sestavení

Budeme potřebovat pouze EV3 Brick.

4.10.3 Program – pouze LeJOS

Zdrojové soubory se nachází v příloze 10.

Na začátku programu vytvoříme úvodní menu. Vypíšeme text na displej a budeme čekat na stisk tlačítek. Pokud stiskneme levé, budeme chtít spustit část kódu, kde bude hádat počítač. Pokud stiskneme pravé, budeme chtít spustit část kódu, kde bude hádat číslo člověk. Obě části kódu od sebe oddělíme tak, že je napíšeme zvlášť. Každá část bude mít svou vlastní metodu. Tím zpřehledníme kód a budeme moci využít pokročilých programátorských dovedností a později naprogramovat rekurzi:

```
public static void main(String[] args) {  
  
    LCD.drawString("Vítejte ve hře", 0, 0);  
    LCD.drawString("HADANI CISEL", 0, 1);  
  
    LCD.drawString("Hadat bude:", 0, 3);  
    LCD.drawString("VLEVO - pocitac", 0, 4);  
    LCD.drawString("VPRAVO - clovek", 0, 5);  
  
    int tlacitko = 0;  
    while (tlacitko != Button.ID_LEFT && tlacitko != Button.ID_RIGHT) {  
        tlacitko = Button.getButtons();  
        Delay.msDelay(100);  
    }  
    LCD.clear();  
    Delay.msDelay(1000);  
    if (tlacitko == Button.ID_LEFT)  
        hraPocitace();  
    else  
        hraClloveka();  
}
```

Nyní naprogramujeme část, kde hádá člověk. Naprogramujeme metodu *hraCloveka()*.
Téměř celý kód se bude odehrávat v jedné smyčce:

```
private static void hraCloveka() {
    LCD.drawString("Pockej, vymyslim", 0, 1);
    LCD.drawString("si cislo", 0, 2);
    LCD.drawString("v rozmezi 0-20", 0, 3);

    int randomNum = new Random().nextInt(20);
    Delay.msDelay(1000);
    LCD.clear();

    int hledaneCislo = 10;
    boolean uhodnuto=false;
    while(!uhodnuto){
        LCD.clear();
        LCD.drawString("Mam vymyslenu,", 0, 0);
        LCD.drawString("hadej!", 0, 1);
        Delay.msDelay(1000);
        LCD.drawString("Tlac NAHORU/DOLU", 0, 5);
        LCD.drawString("ENTER potvrd.", 0, 6);

        while (Button.ENTER.isUp()) {
            LCD.drawString("Hadam c.: " + hledaneCislo + "      ", 0, 3);
            if(Button.UP.isDown()){
                hledaneCislo++;
            }else if(Button.DOWN.isDown()){
                hledaneCislo--;
            }
            Delay.msDelay(500);
        }

        LCD.clear();
        LCD.drawString("Porovnavam...", 0, 3);
        Delay.msDelay(1000);

        if (hledaneCislo == randomNum){
            LCD.clear();
            LCD.drawString("UZASNE", 0, 1);
            LCD.drawString("Jsi jasnovidec!", 0, 2);
            LCD.drawString("Bylo to c. " + hledaneCislo, 0, 3);
            uhodnuto = true;
            Button.waitForAnyPress();
        }else {
            if(hledaneCislo>randomNum){
                LCD.clear();
                LCD.drawString("Bohuzel musis", 0, 1);
                LCD.drawString("zkusit mensi", 0, 2);
                LCD.drawString("cislo.", 0, 3);
            }else{
                LCD.clear();
                LCD.drawString("Bohuzel musis", 0, 1);
                LCD.drawString("zkusit vetsi", 0, 2);
                LCD.drawString("cislo.", 0, 3);
            }
        }
    }
}
```

```

        LCD.drawString("Pokracuj stiskem", 0, 5);
        LCD.drawString("jakehokoli tlacitka.", 0, 6);
        Button.waitForAnyPress();
    }
}
}

```

Zbývá naprogramovat metodu hraPocitace(). Abychom využili vyhledávací metodu půlením intervalů, vytvoříme si ještě metodu hadejInterval(). Tato metoda bude vyhledávat přímo v zadaném intervalu. Tahle metoda nám ušetří spoustu napsaných řádků kódu. Pokud v daném intervalu výsledek nenajdeme, zavolá se sama ještě jednou, ale už s menším intervalem:

```

private static void hraPocitace() {

    LCD.drawString("HADAT BUDE POCITAC", 0, 1);
    LCD.drawString("Vymyslete si cislo", 0, 3);
    LCD.drawString("v rozmezi 0-20", 0, 4);
    LCD.drawString("Potvrďte Enterem", 0, 6);

    while (Button.ENTER.isUp())
        Delay.msDelay(100);

    LCD.clear();
    Delay.msDelay(1000);

    hadejInterval(0, 20);
}

private static void hadejInterval(int min, int max) {
    LCD.clear();

    if (min == max) {
        Button.LEDPattern(Button.ID_ALL);
        LCD.drawString("JE TO CISLO", 0, 1);
        LCD.drawString("" + min, 0, 2);
        LCD.drawString("JUPI, VYHRAL JSEM!", 0, 5);

        LCD.drawString("ENTER = HOTOVO", 0, 7);
        Button.waitForAnyPress();
    } else {

        int hadaneCislo = min + ((max - min) / 2);
        LCD.drawString("HADAM", 0, 1);
        LCD.drawString("Je to cislo " + hadaneCislo + " ?", 0, 2);

        LCD.drawString("Pro cislo vyssi", 0, 4);
        LCD.drawString("tlacitko NAHORU", 0, 5);
        LCD.drawString("Pro cislo nizsi", 0, 6);
        LCD.drawString("tlacitko DOLU", 0, 7);

        int tlacitko = 0;
    }
}

```



```
        while (tlacitko != Button.ID_UP && tlacitko !=  
Button.ID_DOWN && tlacitko != Button.ID_ENTER) {  
            tlacitko = Button.getButtons();  
            Delay.msDelay(100);  
        }  
        LCD.clear();  
        Delay.msDelay(1000);  
  
        switch (tlacitko) {  
        case Button.ID_UP:  
            LCD.drawString("Moment,", 0, 3);  
            LCD.drawString("prepocitavam...", 0, 5);  
            Delay.msDelay(1000);  
            hadejInterval(hadaneCislo, max);  
            break;  
        case Button.ID_DOWN:  
            LCD.drawString("Moment,", 0, 3);  
            LCD.drawString("prepocitavam...", 0, 5);  
            Delay.msDelay(1000);  
            hadejInterval(min, hadaneCislo);  
            break;  
        case Button.ID_ENTER:  
            hadejInterval(hadaneCislo, hadaneCislo);  
            break;  
        }  
    }  
}
```

ZÁVĚR

Lego Mindstorms je robotická stavebnice, která mě oslovila.

Tak komplexní stavebnici, jako je Lego Mindstorms, kde vše zapadá do sebe, nemusí se nic utahovat a šroubovat a téměř vše je se vším mechanicky kompatibilní, snad k sehnání ani není. Když přidáme elektroniku – senzory, motory, ovládací prvky a propojení se softwarem, který stačí jedním kliknutím nainstalovat, a téměř okamžitě můžeme stavebnici ovládat, je stavebnice naprosto vhodná pro použití k výuce v celé věkové kategorii 9-18 let.

Programy jsou provedeny ve dvojí verzi. Jedna je pro vývojové prostředí Lego Mindstorms EV3 Home Edition a druhá je napsána v programovacím jazyce Java. Výuku a vývoj v prostředí Lego Mindstorms EV3 Home Edition doporučuji pro mladší žáky 9-13 let, protože se nemusí tolik psát na klávesnici, všechno má svoji barvu a názorný obrázek. Naopak vývoj v jazyce Java a v prostředí LeJOS doporučuji starším žákům 14-18 let. Komplexnější kód se vyvíjí rychleji a je i přehlednější. Jazyk Java umožňuje vyvinout v podstatě jakékoli aplikace a v reálném prostředí se vyskytuje častěji.

Řešení úloh v Lego Mindstorms EV3 Home Edition a LeJOS se v některých částech liší. Chci tím tak demonstrovat, že ke správnému výsledku se dá přijít odlišnými cestami a bude záležet na žácích, jak se s tím poperou.

SEZNAM POUŽITÉ LITERATURY

- [1] *LEGO* [online]. 2015. [cit. 2015-05-15]. Dostupné z: [http://www.lego.com/cs-cz/
http://www.nxtprograms.com/](http://www.lego.com/cs-cz/http://www.nxtprograms.com/)
- [2] Lego Mindstorms. 2015. *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation [cit. 2015-05-15]. Dostupné z: http://cs.wikipedia.org/wiki/Lego_Mindstorms
- [3] Lego Mindstorms NXT. 2015. *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation [cit. 2015-05-15]. Dostupné z: http://en.wikipedia.org/wiki/Lego_Mindstorms_NXT
- [4] Lego Mindstorms EV3. 2015. *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation [cit. 2015-05-15]. Dostupné z: http://en.wikipedia.org/wiki/Lego_Mindstorms_EV3
- [5] *Techcapm* [online]. 2014. [cit. 2015-05-15]. Dostupné z: <http://www.techcamp.com.au/>
- [6] *LeJOS: Java for LEGO Mindstorms* [online]. 2009. [cit. 2015-05-15]. Dostupné z: <http://www.lejos.org/>
- [7] KLIMŠA, Petr. 2012. *Úvod do programování robotů 2* [online]. Obchodní akademie Orlová [cit. 2015-05-15]. Dostupné také z: <http://scholanova.obaka-orlova.cz/files/robot2.pdf>
- [8] Karel (programovací jazyk). 2015. *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation [cit. 2015-05-15]. Dostupné z: http://cs.wikipedia.org/wiki/Karel_%28programovac%C3%AD_jazyk%29
- [9] KLÍMA, Karel. 2007. *Karel* [online]. [cit. 2015-05-15]. Dostupné z: <http://karel.webz.cz/co-je-karel>
- [10] Raspberry Pi. 2015. *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation [cit. 2015-05-15]. Dostupné z: http://cs.wikipedia.org/wiki/Raspberry_Pi
- [11] *BeagleBone Black* [online]. 2015. [cit. 2015-05-15]. Dostupné z: <http://beagleboard.org/BLACK>
- [12] *Raspberry Pi* [online]. 2015. [cit. 2015-05-15]. Dostupné z: <https://www.raspberrypi.org>

- [13] Mouser Serving Up BeagleBone Black for Same-Day Shipping. 2015. *Electronic Products* [online]. [cit. 2015-05-15]. Dostupné z: http://www.electronicproducts.com/Board_Level_Products/Single_Board_Computer/Mouser_Serving_Up_BeagleBone_Black_for_Same-Day_Shipping.aspx
- [14] How to deploy, debug and profile Java on the Raspberry Pi. 2015. *JAXenter* [online]. [cit. 2015-05-15]. Dostupné z: <http://jaxenter.com/how-to-deploy-debug-and-profile-java-on-the-raspberry-pi-2-108008.html>
- [15] *Libbulldog: Java for embedded systems* [online]. 2015. [cit. 2015-05-15]. Dostupné z: <http://libbulldog.org/bulldog/>
- [16] *Eclipse* [online]. 2015. [cit. 2015-05-15]. Dostupné z: <http://www.eclipse.org/>
- [17] *EV3 Technology* [online]. 2013. [cit. 2015-05-15]. Dostupné také z: <http://www.depts.ttu.edu/coe/stem/gear/ev3/documents/EV3-Motors-Sensors.pdf>
- [18] RileyRover - EV3 Classroom robot design. 2015. KEE, Damien. *Damien Kee: technology in education* [online]. [cit. 2015-05-15]. Dostupné z: <http://www.damienkee.com/home/2013/8/2/rileyrover-ev3-classroom-robot-design.html>
- [19] *Robotshop* [online]. 2015. [cit. 2015-05-15]. Dostupné z: <http://www.robotshop.com/>
- [20] *Jameco: Robot Store* [online]. 2015. [cit. 2015-05-15]. Dostupné z: <http://www.jameco.com/Jameco/robot/robotic-kits.html>
- [21] *Robot Karel* [online]. 2013. [cit. 2015-05-15]. Dostupné z: <http://karel.oldium.net/>
- [22] *Robotstore.cz: open source hardware* [online]. 2015. [cit. 2015-05-15]. Dostupné z: <http://robotstore.cz/>
- [23] *Libbulldog: Java for embedded systems* [online]. 2015. [cit. 2015-05-15]. Dostupné z: <http://libbulldog.org/bulldog/>
- [24] *LEGO Mindstorms EV3* [online]. 2015. [cit. 2015-05-15]. Dostupné z: <http://www.lego.com/cs-cz/mindstorms>
- [25] *The Early Childhood Robotics Network* [online]. 2015. [cit. 2015-05-15]. Dostupné z: [1] <http://tkroboticsnetwork.ning.com/page/wedom>
- [26] *LDraw.org: Centralised LDraw Resources* [online]. 2015. [cit. 2015-05-15]. Dostupné z: <http://www.ldraw.org/>

- [27] *FreeCAD: An Open Source parametric 3D CAD modeler* [online]. 2015. [cit. 2015-05-15]. Dostupné z: <http://www.freecadweb.org/>
- [28] 3D LEGO pirate final. 2015. *Deviant Art* [online]. [cit. 2015-05-15]. Dostupné z: <http://ninjatoespapercraft.deviantart.com/art/3D-LEGO-pirate-final-196876724>
- [29] *Rhinoceros: Life is but A Span..* [online]. 2013. [cit. 2015-05-15]. Dostupné z: <http://rhinotoday.com/>

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

JRE Java runtime environment – prostředí, umožňující běh Java programů.

IDE Integrated Development Environment – vývojové prostředí.

SEZNAM OBRÁZKŮ

Obrázek 1: Prostředí výukového jazyka Karel. [22].....	11
Obrázek 2: Sada Lego WeDo Education [1]	12
Obrázek 3: Programové bloky vývojového prostředí pro lego WeDo [26].....	12
Obrázek 4: NXT Brick [4]	13
Obrázek 5: Robot postavený z Lego Mindstorms NXT [4].....	14
Obrázek 6: EV3 Brick [3]	14
Obrázek 7: Původní vývojové prostředí pro Lego Mindstorms. [6].....	15
Obrázek 8: Instalace LeJOS Plug-In přímo v Marketplace Eclipsu.	16
Obrázek 9: Internetový obchod robostore.cz [23]	16
Obrázek 10: Raspberry Pi (model B+) 2014 [11].....	17
Obrázek 11: Internetová stránka s manuály pro Rapsberry Pi [13].....	17
Obrázek 12: BeagleBone Black	18
Obrázek 13: Stránky knihovny libbulldog pro přímý přístup k GPIO. [24].....	18
Obrázek 14: Lego Digital Designer	19
Obrázek 15: MLCad program z rodiny LDraw.	20
Obrázek 16: FreeCAD [28].....	20
Obrázek 17: Program SketchUp [29].....	21
Obrázek 18: Program Rhinoceros [30]	21
Obrázek 19: Stránka pro stažení vývojového prostředí Lego Mindstorms EV3	24
Obrázek 20: První spuštění vývojového prostředí Lego Mindstorms EV3 Home Edition	24
Obrázek 21: Vytvoření nového projektu v Lego Mindstorms EV3 Home Edition	25
Obrázek 22: Základní programování v Lego Mindstorms EV3 Home Edition.....	25
Obrázek 23: Nastavení vlastnosti vstupu na Wired.	25
Obrázek 24: Úvodní program v Lego Mindstorms EV3 Home Edition.....	26
Obrázek 25: Projekt LeJOS na sourceforge.net.....	27
Obrázek 26: Určení JDK pro LeJOS.	28
Obrázek 27: Konec instalace, zaškrtnutím políčka Launch EV3SDCard utility zajistíme spuštění programu pro nastavení microSD karty	28
Obrázek 28: Program pro instalaci LeJOS na kartu microSD.	29
Obrázek 29: Sekce Download na http://www.eclipse.org/	29
Obrázek 30: Eclipse – v okně po spuštění nastavujeme pracovní adresář.....	30

Obrázek 31: V Eclipse Marketplace se nachází plugin LeJOS.	30
Obrázek 32: Eclipse Marketplace.	30
Obrázek 33: Po instalaci LeJOS pluginu se zobrazí v menu položka s názvem leJOS EV3.....	31
Obrázek 34: Eclipse preferences – nastavení leJOS, zadání IP adresy připojeného EV3 Bricku.	31
Obrázek 35: Zadávání hesla na wifi v EV3 Bricku.	32
Obrázek 36: Po připojení Bricku do sítě se IP adresa zobrazí na hlavní obrazovce...32	
Obrázek 37: Vytváření nového LeJOS projektu.	32
Obrázek 38: Vytváření nového LeJOS projektu.	33
Obrázek 39: Nový projekt se zobrazí v package exploreru.	33
Obrázek 40: Vytvoření třídy pro programování EV3 Bricku.	34
Obrázek 41: Základní seznámení s EV3 Brick35	
Obrázek 42: Echo telegraf37	
Obrázek 43: Program pro echo telegraf.38	
Obrázek 44: Část programu parkovacího asistentu.40	
Obrázek 45: Sekačka.42	
Obrázek 46: Program pro ovládání sekačky.43	
Obrázek 47: Jednoduchý 3-kolový robot.....45	
Obrázek 48: Program pro spuštění motorů vpřed na 3s.....46	
Obrázek 49: Nastavení rychlosti a směru v první vlastnosti motoru.46	
Obrázek 50: Program detekce překážek.47	
Obrázek 51: Vysokozdvížený vozík.....50	
Obrázek 52: První část kódu – nastavení radlic do spodní krajní polohy pomocí tlačítek.51	
Obrázek 53: Druhá část kódu – vypsání instrukcí na displej a příprava proměnné pro uchování úhlu otočení.51	
Obrázek 54: Třetí část programu – otočení radlic do vrchní krajní polohy.....51	
Obrázek 55: Konečná třetí část kódu vysokozdvížného vozíku pro převoz nákladu. 52	
Obrázek 56: Robot sběrač s robotickou rukou.....54	
Obrázek 57: Program robota sběrače.....55	
Obrázek 58: Sumo robot.....58	
Obrázek 59: Příklad hřiště.58	

Obrázek 60: Program sumo robota.	59
Obrázek 61: Auto s klasickým řízení.....	63
Obrázek 62: První část programu slouží pro nastavení předních kol na střed.....	64
Obrázek 63: Druhá část programu – část smyčky Loop, která se stará o zatáčení podle detekce bílé barvy na senzorech barvy.	65
Obrázek 64: Třetí část programu – konečná část smyčky Loop, která se stará o zatáčení podle detekce bílé na senzorech barvy.	65

SEZNAM PŘÍLOH

- Příloha 0: Demo ukázka
- Příloha 1: Základní seznámení
- Příloha 2: Echo telegraf
- Příloha 3: Ultrazvukový parkovací asistent
- Příloha 4: Sekačka
- Příloha 5: 3-kolový robot
- Příloha 6: Vysokozdvížený vozík
- Příloha 7: Robot sběrač
- Příloha 8: Sumo robot
- Příloha 9: Auto
- Příloha 10: Hádání čísel