VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INTELIGENTNÍCH SYSTÉMŮ
DEPARTMENT OF INTELLIGENT SYSTEMS

LEGO MINDSTORM EV3 VE VÝUCE PROGRAMOVÁNÍ A ROBOTIKY

LEGO MINDSTORM EV3 IN EDUCATION OF PROGRAMMING AND ROBOTICS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **JAROSLAV PÁRAL**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. FILIP ORSÁG, Ph.D.**
SUPERVISOR

BRNO 2017

Zadání bakalářské práce/19954/2016/xparal02

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav inteligentních systémů Akademický rok 2016/2017

Zadání bakalářské práce

Řešitel: **Páral Jaroslav**
Obor: Informační technologie
Téma: **Lego Mindstorm EV3 ve výuce programování a robotiky**
Lego Mindstorm EV3 in Education of Programming and Robotics
Kategorie: Vestavěné systémy

Pokyny:

1. Seznamte se s vývojovými platformami pro Lego Mindstorm EV3: ev3dev, LeJOS, ROS, Matlab, EV3RT, ROBOTC a proveďte stejnou sady testů na jednotlivých platformách, výsledky porovnejte.
2. Vyberte vývojovou platformu, která je vhodná pro začátečníky, a vytvořte pro ni sadu ukázkových aplikací.
3. Vytvořte materiály pro výuku programování vybrané platformy v češtině (příklady, české tutoriály a vývojové prostředí pro snadnou obsluhu a práci s danou platformou na systému Windows).
4. Celé řešení zhodnoťte a otestujte v praxi.
5. Výsledky shrňte a zhodnoťte přínosy řešení pro studenty (především středních škol).

Literatura:

- Yixiao Li, Takuya Ishikawa, Yutaka Matsubara, Hiroaki Takada. A Platform for LEGO Mindstorms EV3 Based on an RTOS with MMU Support. In: *Proceedings of the 10th Annual Workshop on Operating Systems Platforms for Embedded Real-Time Applications*. Madrid. 2014
- Brian Bagnall. Maximum LEGO EV3: Building Robots with Java Brains (LEGO Mindstorms EV3). 2014.Variant Press. pp. 464. ISBN 978-0986832291

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 a 2 zadání.

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

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

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

Vedoucí: **Orság Filip, Ing., Ph.D.**, UITS FIT VUT
Datum zadání: 1. listopadu 2016
Datum odevzdání: 17. května 2017

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

L.S.

doc. Dr. Ing. Petr Hanáček
vedoucí ústavu

Abstrakt

Bakalářská práce řeší problematiku vývoje softwaru pro LEGO MINDSTORMS EV3 *Brick*. Cílem je umožnit začínajícím programátorům vytvářet i složitější programy, které již nelze jednoduše navrhnout ve standardním prostředí. Práce porovnává dostupné platformy a na základě výskledků testů vybírá jako nejlepší systém EV3RT. Součástí je návrh objektového API, které snižuje obtížnost přechodu z originálního systému. Zároveň poskytuje vhodnou platformu uživatelům, kterým již standardní vývojové prostředí dodávané se stavebnicí nedostačuje a hledají výkonnější alternativu.

Abstract

In this bachelor thesis, we tackle the problem of software development for the Lego EV3 brick. We want to allow the beginner programmers to create complex programs that cannot be easily created using the original graphical programming interface. We provide comparison of several alternatives of the original system from performance and user point of view. Based on this comparison, we choose EV3RT as base for our work. We also present our object-oriented API built on top of the EV3RT system to ease the transfer from graphical to text programming. Therefore, we provide a suitable platform for the beginners, who already reached the limits of standard system and want to develop more complex programs demanding more computing power.

Klíčová slova

LEGO MINDSTORMS EV3, programovací platformy, ev3dev, Matlab, ROBOTC, ROS, EV3RT, ev3cxx

Keywords

LEGO MINDSTORMS EV3, programing platforms, ev3dev, Matlab, ROBOTC, ROS, EV3RT, ev3cxx

Citace

PÁRAL, Jaroslav. *Lego Mindstorm EV3 ve výuce programování a robotiky*. Brno, 2017. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Filip Orság, Ph.D.

Lego Mindstorm EV3 ve výuce programování a robotiky

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Filipa Orsága, Ph.D. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Jaroslav Páral
17.05.2017

Poděkování

Rád bych na tomto místě poděkoval lidem, díky kterým mohla tato práce vzniknout: Kamarádům Jakubu Streitovi a Janu Mrázkovi, s kterými jsem strávil stovky hodin konzultacemi a řešením problémů spojených s prací. Jiřímu Váchovi za podnětné připomínky a návrhy k zamyšlení a také za to, že právě on mne před lety přivedl k zájmu o robotiku. Lucii Karmové za obrovskou trpělivost s mými texty a pomoc při jazykové korektuře. Rodině, která mi vždy byla velkou oporou, a v posledních měsících obzvlášť. Svému vedoucímu Ing. Filipu Orságovi, Ph.D. za vedení této práce.

Obsah

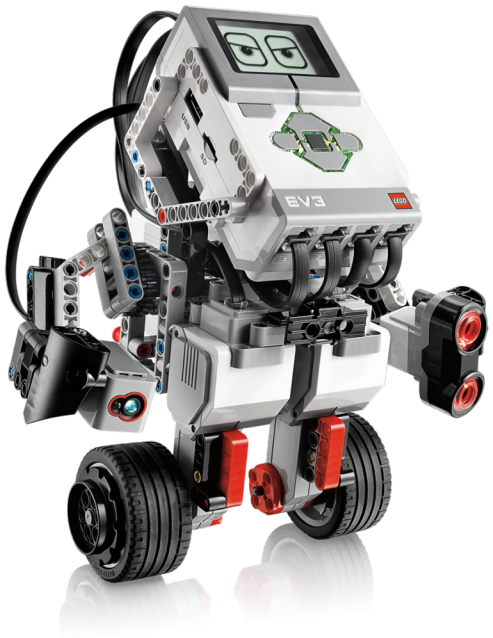
Seznam obrázků

Kapitola 1

Úvod

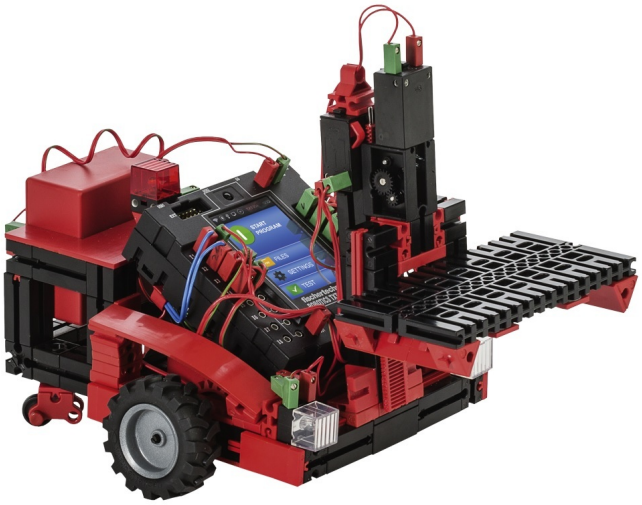
Stavebnice LEGO je jedna z nejznámějších a nejprodávanějších stavebnic na světě. V nabídce firmy LEGO je i robotický set s názvem LEGO MINDSTORMS. Dle oficiálních údajů se jedná historicky o nejprodávanější set z celé nabídky firmy [5]. To je také jeden z hlavních důvodů, proč se tato práce věnuje této stavebnici.

Z výše uvedených informací vyplývá, že jde pravděpodobně o nejdostupnější robotickou stavebnici na světě (pokud pomineme platformu Arduino, která je ovšem zaměřena na jinou skupinu lidí – lidí, kteří se nebojí elektroniky, elektronických obvodů, složitějších návrhů konstrukce, ...



Obrázek 1.1: LEGO MINDSTORMS EV3 – balancující robot[1]

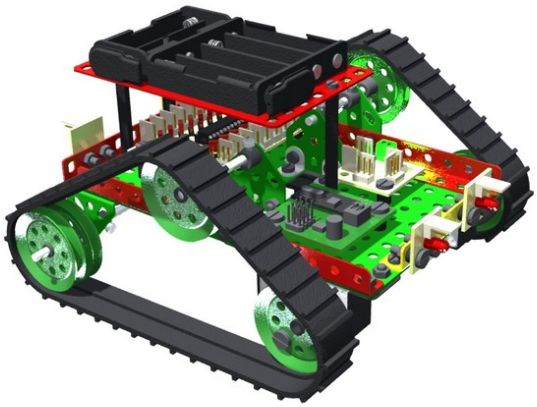
[1] Zdroj: https://www.bermotech.com/training/coding-for-teenagers-and-children/y-robotics-with-lego-mindstorm-ev3/



Obrázek 1.2: Fischertechnik – ROBO TX Explorer[2]

Existuje i mnoho podobných stavebnic [31]. Například *fischertechnik* prodává podobné robotické sety jako LEGO [25].

Uživateli nabízí rozsáhlejší pohled do elektroniky a fungování jednotlivých modulů. Umožňuje také relativně snadno přidat vlastní moduly. Zároveň má jednoduché grafické programovací prostředí podobně jako LEGO. *Fischertechnik* ovšem není tak rozšířen jako LEGO MINDSTORMS, protože ačkoliv nabízí v některých ohledech více funkcí, zároveň klade větší nároky na uživatele, je podstatně dražší (základní set [26] stojí zhruba dvojnásobek ceny LEGO MINDSTORMS EV3 Základní soupravy [40]) a nemá takový věhlas a značku.



Obrázek 1.3: MERKUR – Pásový podvozek 01 – ATMEL + RC[3]

[2]Zdroj: http://www.helago-cz.cz/eshop-519143-workstation-robo-tx-training-lab-tx-explorer-146560.html

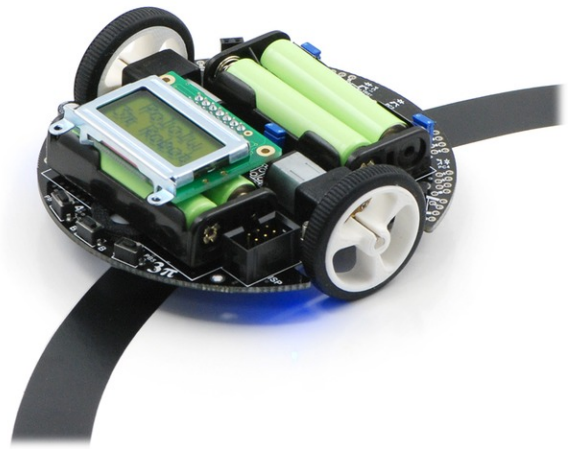
Další zajímavou stavebnicí jsou Robotické sety od Merkuru [47]. Nabídka jednotlivých setů je relativně široká a v porovnání s LEGO MINDSTORMS nebo *fischertechnik* nabízí ještě bližší kontakt s elektronikou a samotným hardwarem. Jako řídicí mikrokontroléry můžete využít PICAXE nebo AVR od firmy Microchip.

Vzhledem k použitým procesorům je možné tyto stroje programovat v C/C++, PICAXE BASIC nebo i v grafickém prostředí [52]. Robotické stavebnice od Merkuru mají podobné „problémy“ jako *fischertechnik*. Kladou na uživatele větší nároky, nemají tak zvučnou značku, mají omezenější základní sadu senzorů a neumožňují tak rychlou stavbu fungujícího robota.

Na závěr lze zmínit výukové roboty, kteří jsou primárně, na rozdíl od robotických stavebnic, zaměřeny na velmi úzkou oblast činností například jízdu po čáře, plnění jednoduchých sekvenčních úkolů, ... Tyto roboty jsou zajímavé z pohledu ceny, ale i jednoduššího využití ve výuce. Žáci nemusí sestavovat konstrukci a hardware je plně připraven k používání.

Naopak již neumožňují rozvoj kreativity studentů při stavbě a přizpůsobení robota pro různé soutěže (většinou je lze využívat jen v jedné soutěžní kategorii). Mezi takovéto roboty patří například Pololu 3pi [53] (primárně určen pro jízdu po čáře – zvládá jezdit až 1 m/s) nebo Edison [7] (umí sledovat čáru, lze jej programovat graficky i v Pythonu, má různé senzory, je možné jej kombinovat s LEGO kostkami).

Tyto roboty ovšem neumožňují takový rozsah činností jako LEGO MINDSTORMS.



Obrázek 1.4: Robot Pololu 3pi[4]

Obrázek 1.5: Robot Edison[5]

1.1 Motivace autora

Již čtvrtým rokem vedu robotický kroužek na své bývalé střední škole SPŠ a VOŠ Brno, Sokolská a zároveň organizuji tábory a další vzdělávací akce v oblasti techniky a robotiky na pobočce Robotárna, Dům děti a mládeže Brno, Helceletova. Už na střední škole jsem se v rámci těchto dvou organizací účastnil různých soutěží, jako například Robotický den v Praze, Mikrokontroléry letí na FEKT VUT Brno nebo Středoškolská odborná činnost

[3] Zdroj: http://www.merkurtoys.cz/vyrobky/pasovy-podvozek-merkur-s-elektronikou-rc
[4] Zdroj: https://www.pololu.com/product/975
[5] Zdroj: https://meetedison.com/meet-edison-v2-0/

(obory strojírenství a elektrotechnika). Z některých soutěží jsem si dovezl cenná umístění, ale i mnoho zkušeností, inspirace a podnětů na přemýšlení.

Za dobu, kterou se věnuji programování mikrokontrolérů a robotice, jsem již narazil na mnoho překážek a problémů, které bylo třeba překonat. Tyto překážky jsou ovšem výrazně náročnější pro studenty, kteří v těchto oblastech teprve začínají a seznamují se s nimi.

Často může nastat i situace, kdy je student se zájmem o robotiku odrazen její počáteční složitostí. Přitom by z něj mohl být v budoucnu perfektní programátor. Jen mu zrovna tento mikrokontrolér nedaří naprogramovat nebo mu koupený H-můstek nechce roztočit motor. Proto se svým studentům vždy snažím nachystat co nejlepší prostředí pro začátky s robotikou.

Již jsem zkoušel různé platformy (například využít k výuce robota Pololu 3pi) i způsoby výuky (kombinace elektroniky, programování na PC a programování mikrokontrolérů), ale bohužel vždy jsem se dostal do stejné situace:

Ačkoliv mi studenti chodili již rok do robotického kroužku, pořád je dělily minimálně dva roky od schopnosti postavit a naprogramovat složitějšího robota – to je: postavit si hardware (s tím, že by použili předpřipravenou elektroniku) a naprogramovat logiku robota (s použitím předpřipravených knihoven).

Nedávno jsme ale do kroužku zakoupili LEGO MINDSTORMS EV3. Stavebnici, která by měla umožnit postavit robota za den.

Prakticky stavebnice snů. No není to nádherná představa? A opravdu tomu tak bylo. Robota jezdícího po čáře jsme s manuálem zvládli poskládat a zprovoznit za den. Studenti nemuseli studovat a řešit zapojení jednotlivých komponentů do řídicí elektroniky.

Nebylo třeba trávit mnoho času vysvětlováním způsobu programování 8bitových mikrokontrolérů (omezení paměti a výkonu, menší datové typy – `uint_8t`, komunikace po sériové lince, ...). Z dílků LEGO si poskládali hardware a pomocí kabelů (které nelze otočit ani zapojit špatně a tím něco zničit) spojili jednotlivé moduly. A v grafickém prostředí si vytvořili svůj program.

Pak jsme se ale začali soustředit na zlepšování softwaru i hardwaru tak, abychom využili stavebnici na 100 procent a v ten moment jsme narazili.

Dokud jsme si s EV3 jen „hráli“ a nesoustředili se primárně na výkon a spolehlivost, bylo vše v pořádku. Jakmile jsme ale chtěli mít regulační smyčku PID regulace pro robota na sledování černé čáry s periodou 10 ms, zjistili jsme, že to není možné (a to máme v EV3 *Bricku* 300 MHz ARM). Přitom díky předešlým zkušenostem s 8bitovými mikrokontroléry Atmel AVR jsme věděli, že by to neměl být problém.

To samé platí o grafickém vývojovém prostředí dodávaném s EV3. Dokud máte na obrazovce několik programových bloků, vše funguje bez problémů. Když se však program rozroste na několik obrazovek a samostatných modulů, velmi rychle ztrácíte přehlednost a efektivitu programování. Zároveň vám již moc nepomohou ladicí nástroje, které jsou součástí prostředí, přestože při jednoduchém programu je lze relativně dobře využít. Ale při rozsáhlejších programech již nejsou k dispozici, nebo je jejich použití velmi komplikované.

Proto jsme začali hledat alternativní platformy, které by odstraňovaly zmíněné problémy, což vyústilo v tuto práci.

Kapitola 2

Historie LEGO MINDSTORMS

Historie firmy LEGO je velmi dlouhá a sahá až do roku 1932 [35]. V období, kdy se počítače stávají standardní součástí domácnosti, se dostávají í do LEGO a tím roku 1998 vzniká LEGO MINDSTORMS [33].

2.1 LEGO MINDSTORMS RCX

První verze byla označena jako LEGO MINDSTORMS RCX[1] Intelligent Brick and Robotics Invention System a obsahovala 8bitový mikrokontrolér Hitachi H8/3292 [29] s procesorem H8/300 taktovaným na 16 MHz a s 32 KB RAM [3].

Obrázek 2.1: LEGO MINDSTORMS RCX[2]

[1] RCX = Robotic Command eXplorers

Oficiálně bylo možné stavebnici programovat ve dvou prostředích. První prostředí ROBOLAB, založené na programu LabVIEW od firmy National Instruments, bylo pro výukové účely (do škol) a bylo součástí výukového setu. Běžní zákazníci (lidé, kteří si kupovali stavebnici domů) měli k dispozici RCX Code, které bylo jednodušší na obsluhu a používání, ale nemělo tak rozsáhlé možnosti programování. Zároveň vznikla i vývojová prostředí třetích stran, která umožňovala programování v mnoha běžně používaných jazycích (C++, Java, ...).

Již v prvním roce prodeje stavebnice vznikla soutěž *FIRST* LEGO League (FLL)[3]. Cílem soutěže je povzbudit a motivovat k navrhování, stavění a‘programování vlastních inteligentních systémů [27]. Jednotlivá kola probíhají po celém světě a lze postoupit až do celosvětového finále. Účastníci musí být ve věku od 10 do 16 let.

2.2 LEGO MINDSTORMS NXT

Následující verzi vydalo LEGO v roce 2006 [33]. Byl kompletně změněn způsob připojování jednotlivých modulů. Moduly se již nepřipojují pomocí speciálních LEGO kostek, ale využívají upravený konektor RJ-12 (kolík sloužící pro správné zapojení konektoru je oproti běžné telefonní RJ-12 umístěn mimo střed [34]).

Obrázek 2.2: LEGO MINDSTORMS NXT s komponenty, které lze používat[4]

Řídicí kostka (dále jen *Brick*) se již neprogramuje přes infraport, ale využívá se standardní USB kabel, případně integrovaný Bluetooth [36].

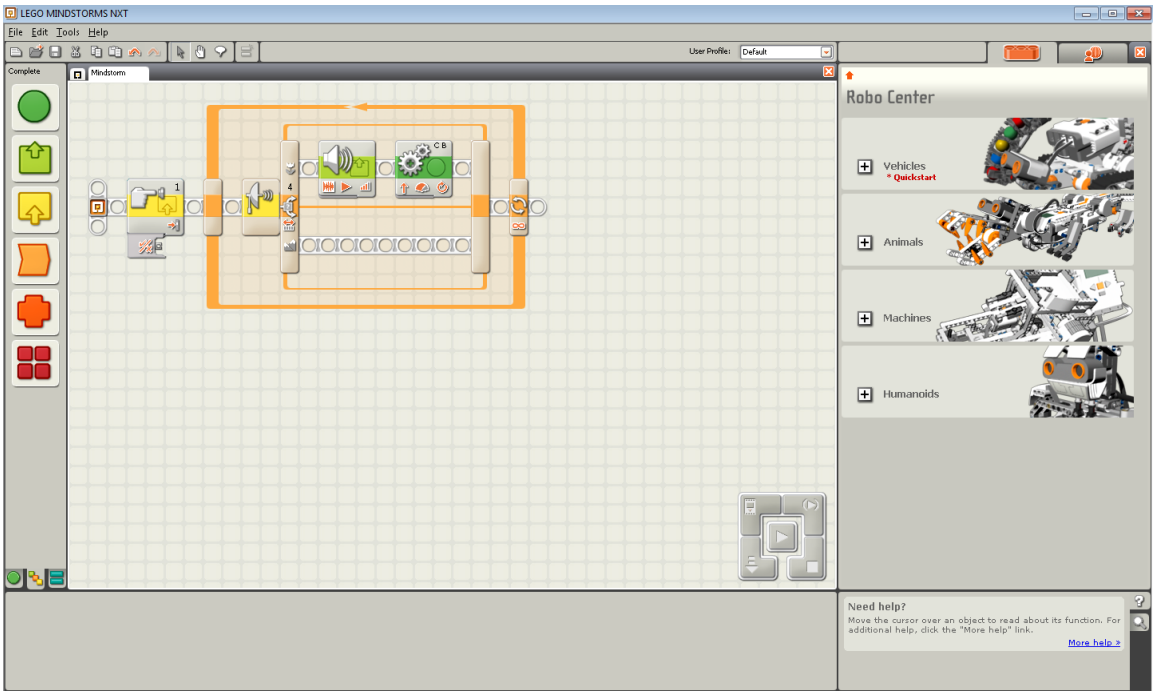
Po hardwarové stránce došlo ke značnému posunu. Původní 8bitový mikrokontrolér byl nahrazen 32bitovým ARM mikrokontrolérem AT91SAM7S256 (256 KB FLASH, 64 KB RAM,

[2]Zdroj: https://en.wikipedia.org/wiki/Lego_Mindstorms
[3]*FIRST* = For Inspiration and Recognition of Science and Technology

48 MHz) a k němu byl přidán 8bitový ko-procesor ATmega48 (4 KB FLASH, 512 Byte RAM, 8 MHz). Oba tyto procesory dodávala firma Atmel (nyní již Microchip) [36].

Brick lze programovat pomocí vývojového prostředí NXT-G[5] (viz obrázek 2.3), které LEGO vyvinulo opět ve spolupráci s National Instruments [60]. Zároveň National Instruments dodává toolbox do LabVIEW, ve kterém lze *Brick* také programovat.

LEGO MINDSTORMS NXT má ovšem bohatou nabídku alternativních programovacích prostředí, která lze využít k vytváření kódu pro *Brick*. Některá fungovala již na RCX a byla upravena i pro NXT. Jako příklad mohu uvést jedno z nejpoužívanějších prostředí BricxCC[6] s jazykem NXC[7]. NXC je vysokoúrovňový open-source jazyk podobný C [48].



Obrázek 2.3: LEGO MINDSTORMS NXT-G[8]

Existují ale i komerční varianty. Například ROBOTC [61] je univerzální programovací prostředí a jazyk umožňující programovat větší množství hardwaru (od LEGO MINDSTORMS RCX, NXT, EV3 až po Arduino nebo PIC). Jak již název napovídá, ROBOTC je založen na jazyku C. Uživatel ovšem může přecházet mezi grafickou a textovou formou programování, což může být pro začátečníky velmi vhodné. Také má možnost vybrat si úroveň abstrakce, na jaké bude v textovém módu pracovat.

ROBOTC je velmi zajímavé prostředí. Bohužel nikdy nebyla vydána objektová varianta, s kterou by mohlo být programování ještě jednodušší. Zároveň se jedná o placený produkt a jednotlivé licence [62] tvoří přibližně čtvrtinu současné ceny základní sady stavebnice LEGO MINDSTORMS EV3 [40], což už není zanedbatelná částka.

[4]Zdroj: http://www.itnetwork.cz/java/lego-nxt/seznameni-s-nxj-pro-lego-nxt
[5]NXT-G = NXT Graphic - grafické
[6]BricxCC = Bricx Command Center
[7]NXC = Not eXactly C
[8]Zdroj: http://lukedainton-robots.blogspot.cz/2012/07/control-systems-shooterbot-nxt-g.html

Pro NXT existuje celá řada dalších platforem a jazyků (Javy, C#, Lua, Ada, Python [51]), které lze použít. Jelikož je ale tato práce zaměřena na LEGO MINDSTORMS EV3 , nebudou tyto platformy dále řešeny.

2.3 LEGO MINDSTORMS EV3

Aktuálně nejnovější verzí stavebnice LEGO MINDSTORMS je verze EV3, která byla vydána v roce 2013 [33]. Proběhlo u ní mnoho inovací a změn, i tak ale byla zachována velmi dobrá kompatibilita s verzí NXT [73].

EV3 přináší zásadní změnu procesoru. Jedná se o 32-bitový ARM (ARM9), tentokrát ale od firmy Texas Instrument, s označením AM1808 (16 MB FLASH, 64 MB RAM, 300 MHz) [38]. Jelikož tento procesor není primárně určen pro programování bare-metal, LEGO dodává upravenou verzi Linuxu. To také velmi zjednoduše správu komplikovaných periferií, jako je USB nebo WiFi.

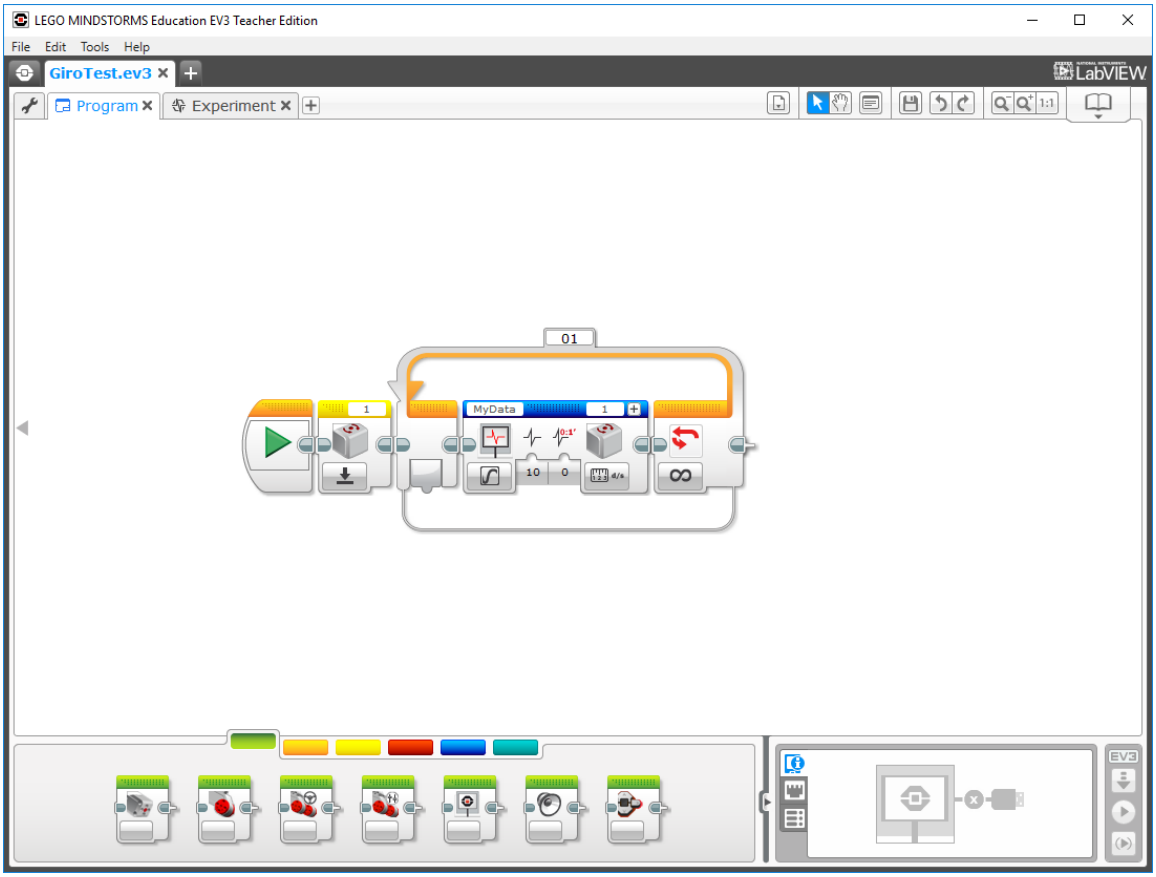
Obrázek 2.4: LEGO MINDSTORMS EV3 *Brick*[9]

Nová verze obsahuje také čtečku Micro SD karet, USB host interface a jeden výstupní port pro motory navíc (celkem 4 porty, NXT jen 3 porty). Dále obsahuje reproduktor (NXT jej obsahovalo také, ale kvůli použitému procesoru umělo jen pípat) a dvoubarevnou indikační LED [70].

[9] Zdroj: http://hackeducation.com/2015/04/10/mindstorms

Slot na SD karty umožňuje rozšířit paměť pro programy, ale lze jej hlavně využít pro spouštění alternativních operačních systémů.

Díky USB host interface lze k EV3 připojovat různé periferie, pro které je v systému a vývojovém prostředí připravená obsluha. Lze tak připojit Wifi modul nebo klávesnici. Zároveň můžete přes USB kabel spojit až čtyři EV3 *Bricky* a ovládat je jedním programem (z jednoho hlavního *Bricku*).



Obrázek 2.5: LEGO MINDSTORMS Education EV3 Software – vývojové prostředí

Programovací prostředí pro EV3 je znovu postaveno na LabVIEW a ačkoliv se design v porovnání s prostředím pro NXT výrazně změnil (můžete porovnat obrázky 2.3 a 2.5), neměli by mít uživatelé NXT s přechodem problém. Nové prostředí zvládne naprogramovat jak EV3, tak NXT *Brick*.

V rámci alternativních platforem máme opět velký výběr [43]. Vhledem k nutnosti používání operačního systému je většinou pro dané prostředí potřeba nahrát konkrétní systém na SD kartu a spustit *Brick* s touto kartou.

Jednotlivé alternativní platformy budou podrobněji probrány v následující kapitole.

Kapitola 3

Dostupné platformy

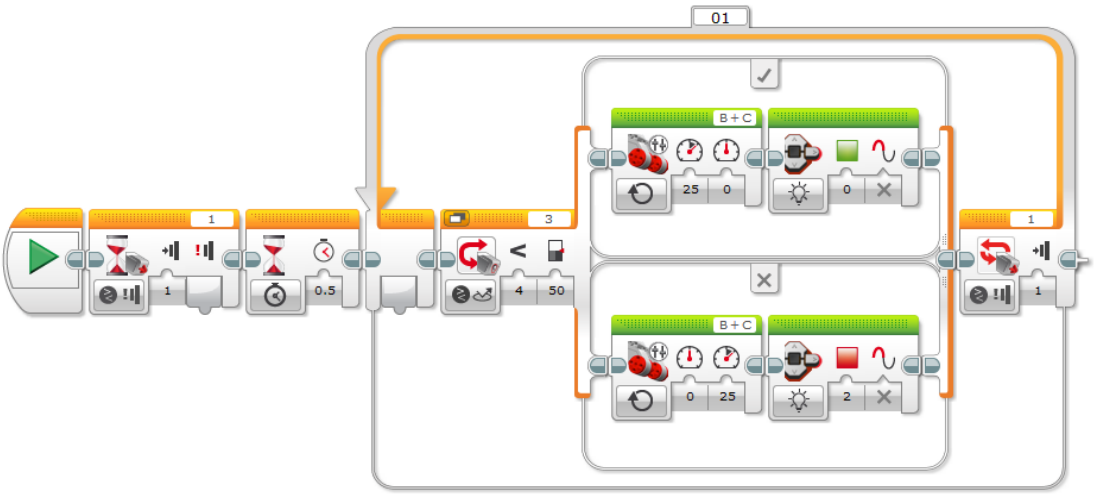
Pro LEGO MINDSTORMS EV3 existuje mnoho vývojových platforem. Většinou se jedná o specificky navržené programovací API ve spojení s vývojovým prostředím. Tyto platformy často fungují ve spolupráci s oficiálním systémem v EV3, ale jsou i platformy s vlastním operačním systémem. V tomto textu bude popsán jen výběr těch nejzajímavějších.

3.1 Originální prostředí od LEGO

V rámci této podkapitoly je rozebírán samotný operační systém na EV3 a také vývojové prostředí, určené pro tvorbu a ladění programů. LEGO si pro EV3 vytvořilo vlastní operační systém, postavený na linuxovém jádru [38]. Uvnitř systému běží virtuální stroj, který zpracovává byte-code uživatelské aplikace. Byte-code je vytvořen ve vývojovém prostředí na PC a pak je do *Bricku* odeslán po USB kabelu, Bluetooth nebo WiFi.

Celý systém s vývojářskými nástroji a dokumentací je volně k dispozici na webu [37]. Jelikož LEGO uvolnilo zdrojové kódy, mohly vzniknout alternativní systémy pro EV3 jako *ev3dev* (kapitola 3.2) nebo EV3RT (kapitola 3.7).

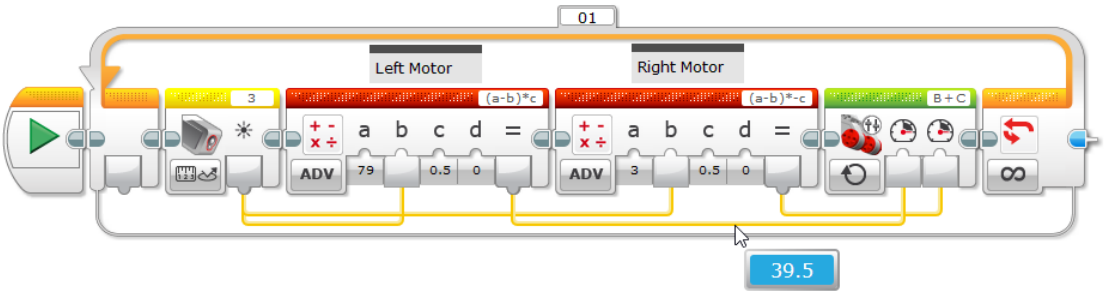
3.1.1 LEGO MINDSTORMS EV3 Software



Obrázek 3.1: Ukázka programovacích bloků v LEGO MINDSTORMS EV3 Software

LEGO MINDSTORMS EV3 Software je vývojářský program, který je k EV3 dostupný zdarma. Tento software vytvořilo LEGO ve spolupráci s firmou National Instruments. Je tudíž postavený na vývojovém prostředí LabVIEW. LEGO s National Instruments již spolupracovalo na předešlých vývojových prostředích. Program se snaží působit jako profesionální vývojové a laboratorní studio, což se mu celkem daří. Nabízí celou řadu funkcí od modulu pro programování přes tvorbu různých experimentů až po přípravy interaktivních návodů a průvodců programováním v tomto prostředí.

V režimu tvorby programu jsou hlavními stavebními kameny *programovací bloky*, které se ve formě diagramů skládají do podoby výsledného programu (obrázek 3.1). Bloky jsou rozděleny do několika skupin (akční, tok programu, senzory, datové operace, pokročilé a vlastní), které se liší svojí barvou a použitím. Pomocí těchto bloků lze řídit tok programu (podmínky, cykly, čekání a vlákna), pracovat s proměnnými, obsluhovat motory či senzory, provádět matematické operace anebo i využívat vlastní bloky.

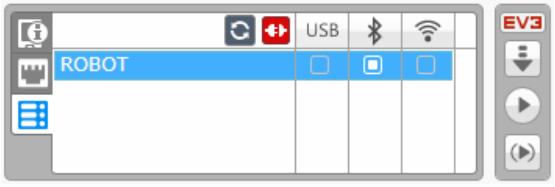


Obrázek 3.2: Ukázka debuggování za běhu programu

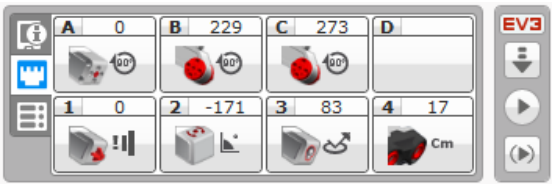
Vše je pěkně graficky zpracováno a působí jako jednotný celek. Při běhu programu lze sledovat jeho tok (kde v diagramu se program nachází) nebo si třeba zobrazit aktuální stav (hodnotu proměnné, výsledek matematické operace, vstupní data ze senzoru, ...). Ukázku debuggování si lze prohlédnout na obrázku 3.2, kde je vidět hodnota 39.5, odpovídající výsledku výpočtu výkonu pro levý motor v závislosti na naměřené hodnotě od barevného senzoru. Zároveň je podle šrafování vrchní lišty bloků vidět, že program aktuálně probíhá v daném cyklu.

Pro práci s *Brickem* je k dispozici modul *Správa hardwaru*, kde si lze zjistit informace jako jméno, stav baterie, verze firmwaru nebo obsazenost paměti. Umožňuje také jednoduše nahrávat, spouštět a ukončovat programy.

Nalezneme zde i správu připojení, v rámci které lze vyhledávat a připojovat se k dostupným *Brickům*. K dispozici jsou tři druhy připojení: USB kabel, Bluetooth a WiFi. Bluetooth je již integrován, ale pro použití WiFi je potřeba připojit WiFi modul.



Obrázek 3.3: Správa připojení k *Bricku*



Obrázek 3.4: Informační panel s porty

Nejzajímavějším panelem je správce portů (obrázek 3.4). Zobrazuje, na jakém portu je připojen který senzor nebo motor. Uživatel může vidět také aktuální hodnoty komponentů

a přepínat jejich režimy. Například u barevného senzoru si může vybrat, zda se bude zobrazovat barva, nebo odrazivost povrchu. U motorů lze naopak nastavit, jestli má být ujetá vzdálenost zobrazování v otáčkách, nebo ve stupních. Tento panel považuji ze jednu z nejpodstatnějších funkcí celého programu, jelikož dává uživateli dobrý přehled o aktuálním stavu hardwaru. Jednoduše lze tak ladit konstanty v programech (rozhodovací úroveň pro barevný senzor při detekci čáry, počet otáček motoru potřebných k dojetí na požadovanou pozici nebo vzdálenost od překážky naměřenou ultrazvukem).

V následujících odrážkách je shrnut výčet nevýhod a nedostatků originálního LEGO prostředí. Za nejdůležitější bod bych považoval nepřehlednost rozsáhlejších programů.

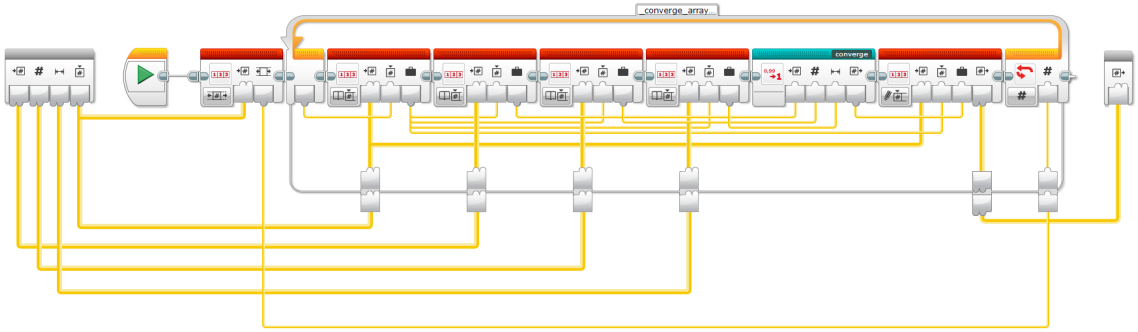
To je pravděpodobně hlavní důvod, proč uživatelé LEGO MINDSTORMS vyhledávají alternativní prostředí. Ukázku rozsáhlejších programů je možné vidět na obrázcích 3.5 a A.2.

Nevýhody LEGO MINDSTORMS EV3 Software:

- náročný na hardware počítače
- ve větších programech je obtížné se zorientovat anebo najít chybu
- nelze editovat proměnné ani rozhraní uživatelských bloků (jejich parametry)
- debuggovací režim nefunguje uvnitř uživatelských bloků
- při rozsáhlejších programech přestává prostředí fungovat (zamrzá a padá)
- není dostupný na Linuxu a není open-source

Nevýhody operačního systému v *Bricku*:

- dlouhé zapínání a vypínání
- značně pomalé zpracovávání některých operací (. . .)
- nepracuje v real-time režimu a časování jednotlivých operací tedy není garantováno
- podpora jen jednoho typu WiFi modulu



Obrázek 3.5: Ukázka nepřehlednosti rozsáhlejších programů - žluté dráhy značí předávání vstupních a výstupních parametrů mezi jednotlivými bloky - velmi špatně se zjišťuje a kontroluje správnost zapojení žlutých drah.

LEGO MINDSTORMS EV3 Software lze nainstalovat na operační systémy Windows a Mac. K dispozici jsou i aplikace s omezenými možnostmi programování pro tablety na platformě Android a iPady.

3.2 ev3dev

Platforma *ev3dev* [9] je pravděpodobně nejrozšířenější a nejrozsáhlejší alternativa k originálnímu LEGO prostředí. Jedná se o systém, drivery a různé wrappery pro jednotlivé programovací jazyky. Samotný systém je založený na linuxové distribuci Debian, který má v sobě naportovány originální LEGO drivery pro EV3 *Brick* a veškeré periferie.

Jelikož je *ev3dev* postaven na standardní linuxové distribuci, lze používat velkou škálu programovacích jazyků. K dispozici jsou nízkoúrovňové jazyky C a C++. Připraveny jsou i knihovny a wrappery pro Python, JavaScrip, Go [13]. Každý si zde může vybrat podle svých zkušeností nebo preferencí a v případě chuti naimplementovat i podporu pro další jazyky. Aktuálně jsou stále rozpracovány implementace pro Java, Lua, Ruby, ...

Pro práci se systémem je k dispozici SSH připojení, případně lze využít terminál dostupný na úvodní obrazovce a přes USB port si připojit externí klávesnici. Tento způsob ale z důvodu velikosti displeje na *Bricku* není moc komfortní. Systém *ev3dev*, na rozdíl od originálního LEGO prostředí, má lepší podporu ovladačů pro WiFi a funguje prakticky s jakýmkoliv USB modulem. Systém umožňuje nastavení automatického připojování na konkrétní WiFi. Pak se již lze k *Bricku* připojovat bezdrátově. K připojení je možné případně použít i USB Ethernet modul.

Co se týče podpory LEGO motorů a senzorů, *ev3dev* zvládá obsluhovat všechny typy určené pro EV3, NXT a některá zařízení z WeDo [14] [18]. Je implementována i široká podpora alternativních výrobců jako mindsensors.com nebo HiTechnic [2]. Mindsensors.com na svých webových stránkách u některých modelů zmiňují i podporu *ev3dev* [41].

Pro zajímavost lze dodat, že tento systém podporuje i alternativní *Bricky* postavené na jednodeskových počítačích Raspberry Pi nebo BeagleBone. Příkladem komerčně prodávaných *Bricků* jsou BrickPi od Dexter Industries [39] a PiStorms od mindsensors.com [42]. Obě tyto alternativy jsou postaveny na Raspberry Pi a umožňují připojit stejný počet standardních LEGO motorů a senzorů jako EV3. Jejich hlavní výhodou je přítomnost výrazně výkonnějšího procesoru v Raspberry Pi. Na klasickém *Bricku* je totiž *ev3dev* dosti pomalý. Například samotný start systému trvá přes 60 sekund [8]. Otevření SSH spojení přes WiFi, nahrání binárního souboru do *Bricku* a následné spuštění zabere až 40 sekund. K tomuto zdlouhavému procesu bohužel dochází při každé změně programu. Díky použití Raspberry Pi je také možné jednoduše připojit například kameru anebo přímo na GPIO piny přidat vlastní senzory (měření teploty, tlaku, vlhkosti, hodiny reálného času, ...). Mezi nevýhody můžeme počítat nemožnost účasti s těmito alternativními *Bricky* soutěží, které jsou striktně omezeny na použití jen LEGO komponent (v České republice: *FIRST* LEGO League, Robosoutěž ČVUT, Robotiáda).

Avšak i tento systém trápí podobné problémy jako originální LEGO prostředí. Hlavním nedostatkem je, že Linux nemá standardně real-time kernel. Není tak garantováno přesné přidělování výpočetního času pro uživatelské programy, což je velký problém pro běh různých regulačních smyček, které potřebují konstantní dobu mezi jednotlivými průběhy (například stabilizace samobalancujícího robota). V *ev3dev* dle měření dochází až k 20 ms prodlevám, kdy si kernel zabere výpočetní čas a uživatelská aplikace se nedostane k vykonávání programu [19]. Jedno z řešení je výrazně zpomalit regulační smyčku, ale ani to negarantuje, že se program dostane k vykonávání v požadovaný čas. V rámci *ev3dev* proběhl pokus o zakomponování real-time kernelu do separátní vývojové větve. [10]. Jelikož se ale tato větev potýkala s problémy ovladačů motorů, byl její vývoj ukončen [11].

```
if(motor.speed_regulation_enabled() == "off")
    motor.set_speed_regulation_enabled("on");
motor.set_speed_sp(speed);
motor.run_forever();
```

Obrázek 3.6: Ukázka nastavení rychlosti motoru v implementaci C++ na *ev3dev*

Co považuji za další problém, je roztříštěnost této platformy a náročnost pro méně zkušené uživatele. Jednotlivé implementace programovacích jazyků mají různorodou kvalitu dokumentace a většinou relativně složité rozhraní. Na obrázku 3.6 je ukázka kódu pro nastavení rychlosti jednoho motoru v C++ na *ev3dev*. Pro další inspiraci je možné se podívat na demo příklady k tomuto API [17]. Bohužel pro člověka, který doposud programoval pouze přes LEGO prostředí, je podle mě toto rozhraní velmi složité.

Příklad v C++ API jsem vybral z důvodu svých předešlých zkušenosti s tímto rozhraním a předpokladu, že by u C++ nemělo docházet k degradaci výkonu, což by mohlo nastávat u interpretovaných programovacích jazyků jako Python nebo JavaScript.

Ze svých dlouhodobých zkušeností se začátečníky mám dobrou představu, jakých chyb se nejčastěji dopouští a co jim dělá největší problém. Proto jsem se v rámci testování *ev3dev* podílel i na tvorbě knihovny [12], která byla vyvíjena se záměrem co nejvíce usnadnit uživateli programování robotů v *ev3dev*. Přestože tato knihovna mnohé neduhy programování na tomto systému odstraňuje, stále tu přetrvává celá řada problémů souvisejících s použitím linuxového systému (viz popis v úvodu této kapitoly).

Při zkoumání API pro Python [16] jsem došel k podobnému závěru. Je pravda, že k němu lze najít rozsáhlejší dokumentaci [15]. I tak není dle mého názoru pro programátora – začátečníka, který přechází z LEGO softwaru, toto API dostatečně jednoduché a intuitivní.

3.3 ROS

S LEGO MINDSTORMS EV3 je možné používat i ROS (Robot Operating System) [67].

ROS je platforma určená přímo pro robotiku a roboty. Jde o systém, který v sobě integruje nástroje pro komunikaci, loggování, zpracování obrazu, navigaci, simulace nebo ovládání zařízení. Uživateli značně usnadňuje obsluhu vlastního robota, protože má předpřipraveno již mnoho nástrojů.

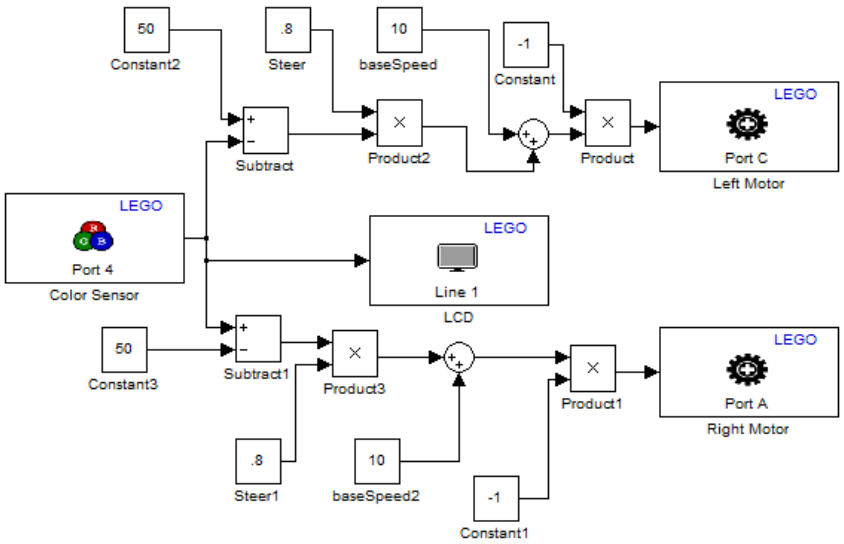
Na samotném EV3 kvůli nedostatečnému výkonu *Bricku* kompletní ROS neběží. Pro EV3 ale existuje utilita do systému *ev3dev*, přes kterou lze pomocí příkazů komunikovat s ROSem běžícím na standardním PC. Tím lze získávat data ze senzorů a posílat zpět instrukce pro motory. Komunikace mezi PC a EV3 *Brickem* může probíhat po WiFi nebo Ethernetu. Veškerá logika jako plánování trasy, herní strategie, případně zpracování dat je prováděna v rámci klasického PC. Hlavními důvody pro použití ROSu na EV3 je široká paleta balíčků, které řeší například práci s různými senzory jako kamera, Kinect, lidar, ale i ovládacích zařízení typu joystick či gamepad. Zároveň je v ROSu integrována transformace souřadnic, plánování trasy, herní strategie nebo i simulátor s fyzikálním enginem.

Při využívání ROSu společně se stavebnicí LEGO je ovšem problematická nutnost mít k *Bricku* připojené klasické PC. Jak bylo již uvedeno v kapitole 3.2, mnoho soutěží pro LEGO MINDSTORMS vyžaduje striktní použití jen LEGO komponent, a to běžné PC není. Zároveň i soutěže, které nemají tuto podmínku, například Robotický den v Praze nebo Istrobot v Bratislavě, zakazují spojení s jakýmkoliv externím zařízením. Tím pádem

je nutné, aby na sobě LEGO robot vozil klasické PC, což může značně zvýšit hmotnost a náročnost konstrukce. V dnešní době by to ovšem již neměl být takový problém (např. lze použít Raspberry Pi anebo zařízení založené na Intel Atom).

3.4 Balíček od MathWorks

MathWorks pro LEGO MINDSTORMS připravil doplňky do svých programů Matlab a Simulink. Cílem těchto doplňků je umožnit uživateli ovládat a programovat LEGO MINDSTORMS ve standardním MathWorks prostředí. V rámci balíčků je podporováno LEGO MINDSTORMS NXT [46] i EV3 [45]. Uživatel může ovládat *Brick* pomocí příkazů z Matlabu anebo lze program vytvořit formou diagramů v Simulinku (podobně jako v oficiálním LEGO prostředí). Příklad programu pro sledování čáry lze vidět na obrázku 3.7. Programy v Simulinku vypadají srozumitelně a přehledně, ale vzhledem k povaze samotného prostředí (laboratorní až vědecký simulační nástroj) není jeho ovládání a tvorba aplikací jednoduchá a pro nováčka občas až odrazující.



Obrázek 3.7: Program na sledování čáry v prostředí Simulink[1]

Doplňky ke svému fungování využívají standardní LEGO systém. Proto se při ovládání *Bricku* z Matlabu posílají příkazy, které LEGO připravilo pro vzdálené řízení. EV3 tyto příkazy běžně využívá v režimu Daisy-Chain, kdy lze USB kabely spojit do série až čtyř *Bricků* a řídit je jedním programem.

Při návrhu programu v Simulinku se vytváří ekvivalentní byte-code uživatelská aplikace, která se do *Bricku* nahrává stejně jako program ze standardního LEGO prostředí. Z toho vyplývá, že vlastnosti a chování programů vytvořených v Matlabu a Simulinku budou ekvivalentní se standardními programy z LEGO prostředí.

Problém balíčku od MathWorksu je i cena licencí za jejich software. Standardně se pohybuje v řádech deseti tisíců korun za jednu licenci [30]. To je pro běžného uživatele nepřijatelná částka. MathWorks nabízí i speciální licence pro školy za výrazně lepší ceny.

[1] Zdroj: http://blogs.mathworks.com/simulink/2012/03/05/running-simulink-models-on-lego-mindstorms-nxt/

K této licenci se ovšem běžný uživatel nedostane, a proto jsem toto prostředí dále nezkoumal. Doplňky jsou dostupné pro Windows, Mac i Linux.

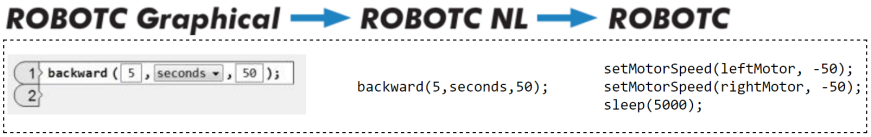
3.5 Balíček od National Instruments

I firma National Instruments nabízí modul pro LEGO MINDSTORMS do svého programu LabVIEW [50]. Funkcionalitou i vlastnostmi se velmi podobá doplňkům od MathWorks [28]. Taktéž využívá standardní LEGO systém a platí pro něj tedy vše, co je popsáno v kapitole 3.4 věnující se doplňkům pro Matlab a Simulink. Cena licencí je také obdobná. Modul je dostupný pro Windows a Mac.

3.6 ROBOTC

ROBOTC je programovací jazyk pro roboty založený na jazyku C. Zároveň se jedná o platformu podporující různé robotické systémy jako LEGO MINDSTORMS (RXC, NXT, EV3), Arduino, VEX Robotics, PIC a další [61]. Jde o komerční produkt, který je přímo zaměřen na hobby robotiku. Na trhu je již dlouhou dobu a v minulosti byl hojně využíván pro programování LEGO MINDSTORMS NXT. Nabízí zajímavou kombinaci grafického a textového programování. Grafické programy (ROBOTC Graphical) mají podobný vizuální styl jako Scratch [68] (jeden z nejrozšířenějších grafických programovacích jazyků určený pro výuku programování – oficiální webové stránky aktuálně navštíví každý měsíc kolem 17 miliónů uživatelů [69]). U textových programů lze volit míru abstrakce, nad kterou se pracuje (ROBOTC NL / ROBOTC). Mezi grafickým a textovým režimem se bohužel nelze přepínat.

Součástí prostředí je i simulátor pro testování programů bez potřeby hardwaru. Uživatel tak může jednoduše ladit program, aniž by měl u sebe fyzicky robota. Této možnosti mohou využít zvláště týmy připravující se na určitou soutěž. Část týmu může stavět nebo upravovat hardware, zatímco ostatní mohou ladit program ve virtuálním prostředí.



Obrázek 3.8: ROBOTC nabízí tyto varianty programování[2]

ROBOTC běží na standardním LEGO systému podobně jako doplňky od MathWorks nebo National Instruments. Programy z ROBOTC se na PC překládají do byte-code aplikací a následně se spouští na *Bricku* ve virtuálním stroji. Tudíž zde může docházet k podobným problémům, jako je popsáno v kapitole 3.1.1.

V rámci vývojového programu pro toto prostředí je k dispozici celá řada podpůrných nástrojů typu debugger, dataloger, vykreslování grafů nebo i ovládání robota přes joystick. Samotné prostředí je relativně propracované a obsahuje mnoho funkcí. Aktuální verze by si zasloužila grafické přepracování tak, aby byla intuitivnější. Je trochu škoda, že není k dispozici implementace jazyka ROBOTC v C++ stylu, ale vzhledem k dobré koncepci současného jazyka to asi není nutné.

Jak bylo uvedeno v úvodu, ROBOTC je komerční program, a s tím souvisí licenční politika. Ceny licencí s platností na jeden rok jsou přibližně poloviční v porovnání s časově

[2]Zdroj: Oficiální nápověda v programu ROBOTC – záložka ROBOTC Language Progression

neomezenými verzemi. Neomezená verze dle jednotlivých balíčků stojí: $79, $299, $599 [62]. V porovnání s produkty od MathWorks nebo National Instruments nejsou tyto sumy tak závratné a když zvážíme cenu samotné stavebnice LEGO MINDSTORMS EV3 (aktuálně 11 700 Kč [40], v přepočtu přibližně $470), zdá se mi tato suma přijatelná.

I tak může být cena za tento produkt pro mnohé uživatele, školy a vzdělávací instituce rozhodující. Zároveň kvůli uzavřenosti tohoto softwaru není možné jednoduše rozšiřovat funkcionalitu a přidávat další vylepšení do programu. Vzhledem k výše uvedeným důvodům jsem se rozhodl, že pro svou práci nebudu toto prostředí dále zvažovat.

3.7 EV3RT

Při hledání příčin problémů s časováním na EV3 [19] jsem narazil na zajímavý japonský projekt EV3RT [23]. Jedná se o port real-timového systému TOPPERS/HRP2 na LEGO MINDSTORMS EV3. Projekt EV3RT vznikl v rámci výzkumu na Nagoya University v Japonsku s cílem odstranění problémů originálního LEGO systému a prostředí. Veškeré zdrojové kódy jsou volně k dispozici na GitHubu [21].

Historie projektu TOPPERS[3] sahá do roku 2003, kdy vznikl jako společný projekt průmyslu, univerzit a vlády [71]. Cílem projektu bylo vytvoření kvalitního otevřeného real-time systému, využitelného v průmyslu pro embedded zařízení. Zároveň měly být vytvořeny výukové kurzy a materiály pro studenty. V Japonsku je tento systém využíván jak v průmyslu, tak i při výuce a prakticky tvoří průmyslový standard pro embedded zařízení.

TOPPERS je vyvíjen s celou řadou real-time kernelů. Pro EV3 byl portován kernel s označením HRP2[4]. Jedná se o statický RTOS[5] kernel s podporou ochrany paměti, který splňuje vysokou spolehlivost a bezpečnost aplikací [44]. V systému je možné vytvářet tasky (obdoba vláken v Linuxu) s různou prioritou. Jsou zde dostupné standardní prostředky pro komunikaci mezi tasky: semafory, mutexy, fronty.

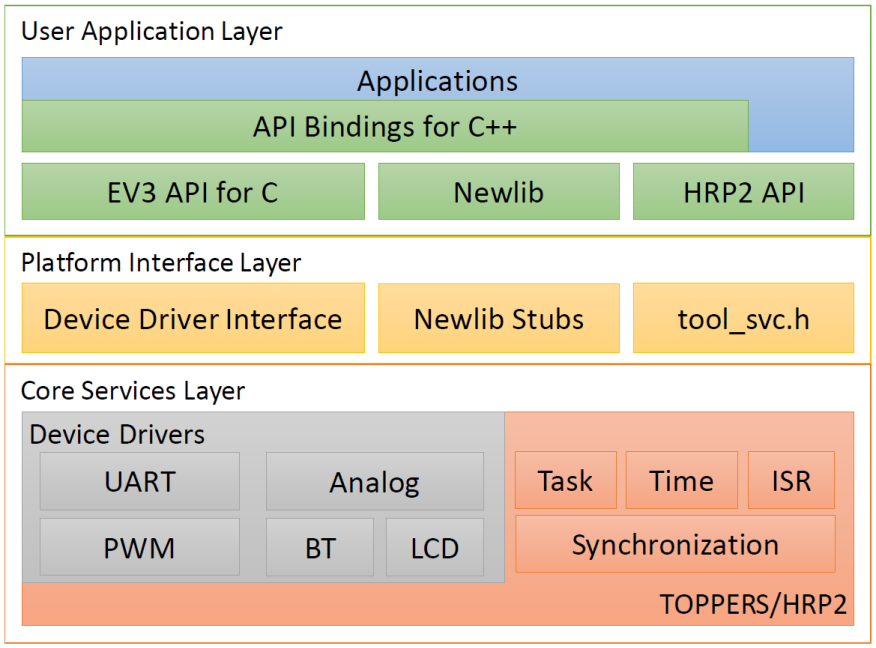
V rámci EV3RT jsou aktuálně podporovány všechny druhy senzorů a motorů ze stavebnice LEGO MINDSTORMS EV3 a fungují veškeré periferie na EV3 *Bricku* (displej, LED, tlačítka, Bluetooth a reproduktor). Zatím není odzkoušená podpora pro NXT komponenty ani alternativní senzory a motory.

EV3RT poskytuje C API pro přístup ke všem funkcím na EV3. Jednoduše lze vyčíst hodnoty ze senzoru nebo roztočit motor. Toto API není ale tak intuitivní, jak by mohlo být při použití C++. K dispozici je i implementace dynamické alokace paměti a lze tedy používat funkce `malloc()` i `free()`. Platforma nemá problém s překladem standardní C++ knihovny (STL). Uživatel tak může v projektech využívat například `std::string`, `std::vector`, `std::queue`, `...`

Výkonnostně by na tom měl být systém EV3RT nejlépe ze všech platforem pro EV3, které byly zmíněny v této kapitole. Hlavní příčinou je samotný RTOS, který má velmi nízkou režii na svůj chod a umožňuje rychlý přístup k hardwaru. Oproti standardnímu LEGO systému zde také neběží žádný virtuální stroj, který může výrazně degradovat výkon při některých operacích. Dle testů provedených tvůrci EV3RT je například provedení nastavení rychlosti (*motor_set_speed*) v průměru až 100krát rychlejší než v originálním systému [44].

Shrnutí hlavních vlastností EV3RT:

[3] TOPPERS = Toyohashi OPen Platform for Embedded Real-time Systems
[4] HRP2 = High Reliable system Profile version 2
[5] RTOS = Real-Time Operating System
[6] Zdroj: článku A Platform for LEGO Mindstorms EV3 Based on an RTOS with MMU Support [44]



Obrázek 3.9: Architektura systému EV3RT[6]

- malá velikost systému (do 5 MB) i uživatelských aplikací (do 1 MB)
- velmi rychlý start (do 5 sekund) a prakticky okamžité vypnutí
- jednodušší úprava a doprogramování vlastních funkcí do jádra systému
- snadné portování linuxových driverů
- podpora dynamické alokace paměti
- preemptivní multitasking s velmi rychlým přepínáním kontextu (do 8 μs)
- multiplatformní vývoj aplikací pro EV3

Nevýhodou TOPPERS a tedy i EV3RT je, že se jedná primárně o japonský projekt. Většina dokumentace k TOPPERS je tak v japonštině, což se týká i velké části zdrojového kódu v EV3RT. Části kódu, které byly převzaty z oficiálního LEGO systému nebo byly vytvářeny primárně pro EV3RT, jsou zdokumentovány v angličtině.

Například ale C API obsahuje pouze japonskou dokumentaci. Jedinou anglickou dokumentací k funkcím dostupných v TOPPERS/HRP2 je tak všeobecná RTOS specifikace μITRON z roku 2002 [32], kterou HRP2, a tím pádem i EV3RT, splňují. Naštěstí překlad z japonštiny do angličtiny zvládá Google Translate docela dobře, a tak je možné tyto komplikace překonat.

Ačkoliv má EV3RT určité nevýhody popsané v předchozím odstavci, rozhodl jsem se jej využít pro tvorbu programovacího prostředí pro LEGO MINDSTORMS EV3. Tento systém jako jediný nabízí opravdovou real-timovost. Netrpí degradací výkonu. Zvládne velmi rychle nastartovat a okamžitě se vypne. Nabízí relativně snadnou možnost zásahů do implementace (úpravu driverů pro EV3 – úprava konstant PID regulátoru, modifikace a přidávání funkcí do uživatelského rozhraní, ...).

Pro EV3RT lze vyvíjet na libovolné POSIX-kompatibilní platformě obsahující GCC překladač pro procesory ARM. Vývoj je tedy možný jak na Linuxu a Macu, tak i na Windows s použitím Cygwinu [22].

3.8 Srovnání dostupných platforem

V této kapitole shrnuji porovnání jednotlivých platforem mezi sebou. Vybral jsem několik bodů, které považuji za důležité u platformy určené pro programování stavebnice LEGO. Zároveň by tyto body měly zohledňovat možnosti budoucího rozvoje daného prostředí a snadné využití platformy ve výuce.

Platformy budu posuzovat v těchto bodech:

1. dostupné zdarma
2. open-source
3. intuitivní programovací rozhraní pro neprogramátory
4. nenáročnost na počáteční zprovoznění
5. real-timovost
6. nenáročné na hardware počítače
7. možnost zásahů do vývojového prostředí
8. podporované operační systémy

Prostředí:	1	2	3	4	5	6	7	8
Originální LEGO	✔	✗	✔	✔	✗	✗	✗	Windows, Mac
ev3dev	✔	✔	✗	✗	✗	✔	✔	Windows, Linux, Mac
ROS	✔	✔	✗	✗	✗	✗	✔	Linux
MathWorks doplňky	✗	✗	✗	✔	✗	✗	✗	Windows, Linux, Mac
National Instruments modul	✗	✗	✗	✔	✗	✗	✗	Windows, Mac
ROBOTC	✗	✗	✔	✔	✗	✔	✗	Windows
EV3RT	✔	✔	✗	✗	✔	✔	✔	Windows, Linux, Mac

Z výše uvedené tabulky i hlubšího zkoumání jednotlivých prostředí v rámci této kapitoly mi jako nejvhodnější volba pro programování stavebnice LEGO MINDSTORMS EV3 připadá prostředí EV3RT. Má některé nedostatky, které u jiných prostředí nejsou (složitější API, náročné na počáteční zprovoznění), ale právě tato problematická místa by mělo být možné díky otevřenému zdrojovému kódu odstranit. Proto se již budu v následujících kapitolách věnovat převážně tomuto systému.

Kapitola 4

Testování

Cílem testování v této kapitole je zjistit výkonnost jednotlivých platforem a porovnat je vůči sobě. Testy jsou zaměřeny primárně na měření časů průběhu hlavní smyčky při určitých činnostech tak, aby bylo možné srovnat výkonnost a stabilitu jednotlivých platforem. Stabilitou je v tomto případě myšlen rozdíl mezi průměrným a maximálním naměřeným časem průchodu smyčky v rámci opakovaných měření a jednotlivých průchodů.

Při předchozí práci s EV3 se systémem *ev3dev*(viz kapitola 3.2), že minimální perioda hlavní smyčky s pár příkazy na vyčtení dat ze senzoru a následného nastavení motorů dle těchto hodnot (klasický program na jízdu po čáře) není kratší než 10 ms. Vzhledem k výkonu procesoru (32bitový ARM na 300 MHz) v EV3 je tento čas velmi zarážející.

O to větší překvapení nastalo, když jsem srovnával průměrnou periodu s maximální. Ukázalo se, že ačkoliv je průměrná perioda přibližně 20 ms, maximální perioda může dosahovat až 100 ms. Takto velký rozptyl je pro potřeby regulace velmi problematický. Ve většině případů je potřeba mít konstantní čas průchodu hlavní smyčkou, aby rozdíl v časech periody neovlivňoval měření a následné řízení.

Pokud bude nutné zajistit stabilní periodu, je třeba softwarově zpomalit průběh hlavní smyčky na maximální naměřený čas. Jedině tak lze zajistit konstantní periodu. Když by se použil tento postup, dostali bychom se na frekvenci 10 Hz, což je z pohledu regulace již velmi nepříznivá hodnota.

Tyto testy byly provedeny na velmi jednoduchém programu. V případě rozsáhlejších projektů, kde by se pracovalo s více vlákny a periferiemi, mohou být výsledky ještě horší.

Proto jsem chtěl provést testování s ekvivalentními programy na jednotlivých platformách a zjistit, jestli tyto neduhy některá z nich neodstraňuje a za jakých situací k podobným problémům dochází.

4.1 Forma testování

Na počátku jsem zvažoval dvě formy testování. První počítala s rozsáhlejší metodikou, v rámci které bych testoval různé funkce EV3 *Bricku* a zjišťoval, kde mohou být nejproblémovější místa z pohledu výkonu.

V rámci testů jsem se chtěl zaměřit na několik oblastí:

- matematické operace
- čtení senzorů

- řízení motorů

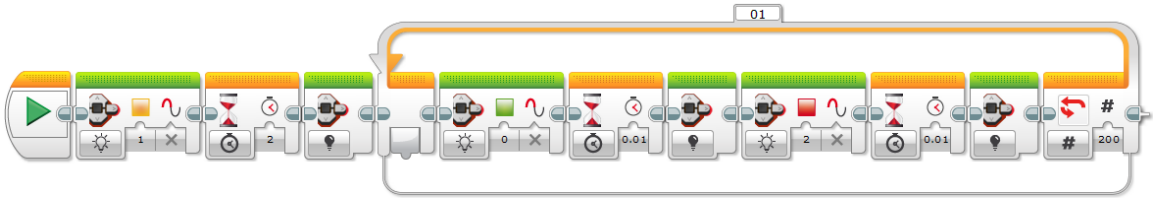
Tuto variantu jsem na počátku odložil, abych zkusil najít metodu jednoduší na implementaci a replikaci na jiné platformy. Následně v ní ale pokračuji v další části kapitoly.

Druhá varianta měla zajistit co možná nejstejnoměrnější podmínky měření času tak, aby jej neovlivňovalo fungování jednotlivých platforem. Proto mělo být k měření využito Arduino Uno a velmi jednoduchý program (příklad programu ve standardním prostředí pro EV3 na obrázku 4.1). Program by jen blikal s dvěma LED[1] (zelenou a červenou) umístěnými na přední straně pod ovládacími tlačítky na EV3 *Bricku* a čas svitu jednotlivých LED by měřilo Arduino.

Jednalo by se totiž o jeden z nejlepších způsobů, jak demonstrovat problémy s dobou trvání jednotlivých cyklů a práci s periferiemi. V optimálním případě by totiž měla kombinací těchto dvou barevných LED vzniknout oranžová.

Pokud by se tak nestalo, pozorovatel by měl hned optickou zpětnou vazbu a zároveň lze jednoduše měřit časové prodlevy mezi spínáním a vypínáním jednotlivých LED pomocí měřicí sestavy s Arduinem.

4.2 Měření pomocí LED



Obrázek 4.1: Ukázka jednoduchého programu na blikání zelenou a červenou LED

Ukázkový program je připraven tak, že každá LED se vždy rozsvítí na 10 ms ($T = 0.01\ s = 10\ ms$), následně zhasne a rozsvítí se druhá LED.

Celkový čas jednoho průchodu cyklem trvá 20 ms ($T = T_{LED1} + T_{LED2}$) a frekvence blikání by měla být 50 Hz ($f = \frac{1}{T} = \frac{1}{0.02\ s} = 50\ Hz$). Což je frekvence, kterou již lidské oko nezaznamená (filmy mají standardní frekvenci obnovování snímku 24 až 30 Hz). Celkový počet opakování hlavní smyčky je zvolen na 200. Blikání má tedy probíhat 4 sekundy ($t = T \cdot counter = 0.02\ ms \cdot 200 = 4\ s$).

Pro měřicí sestavu byl vybrán fototranzistor BPX 81[2] (má velmi široké rozpětí vstupních vlnových délek – zvládne viditelné i infra světlo) s Arduinem Uno (viz obrázek 4.2).

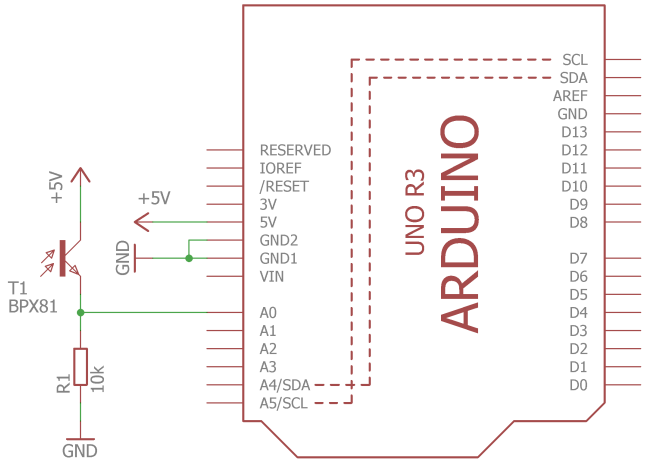
Tato kombinace mi přijde nejvhodnější z několika důvodů:

- velmi rozšířený hardware (Arduino Uno)
- rychlé sestavení
- možnost snadného naprogramování
- jednoduchá duplikovatelnost (zopakování experimentu)

[1] LED = Light-Emitting Diode — dioda emitující světlo

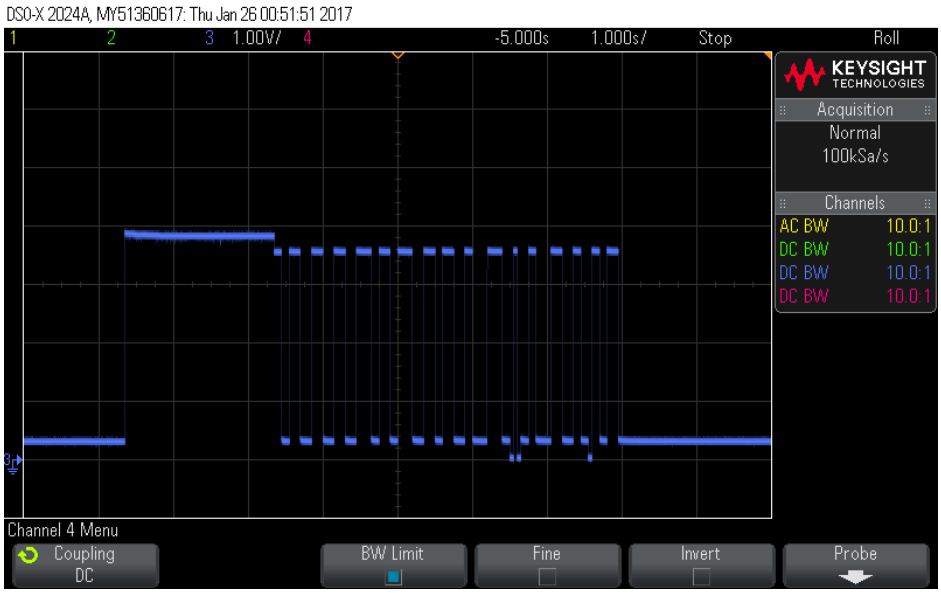
[2] http://cz.rs-online.com/web/p/fototranzistory/6655255/

Nejdůležitějším bodem pro tuto variantu byla jednoduchá duplikovatelnost, která umožňuje komukoliv provést stejná měření, ať už pro kontrolu v této práci prezentovaných výsledků, nebo pro otestování jiné platformy a porovnání s naměřenými daty z této práce.



Obrázek 4.2: Schéma zapojení měřicího systému

Jelikož se druhá varianta zdála vhodnější, otestoval jsem ji s programem, prezentovaným výše, s danou měřicí sestavou a osciloskopem.



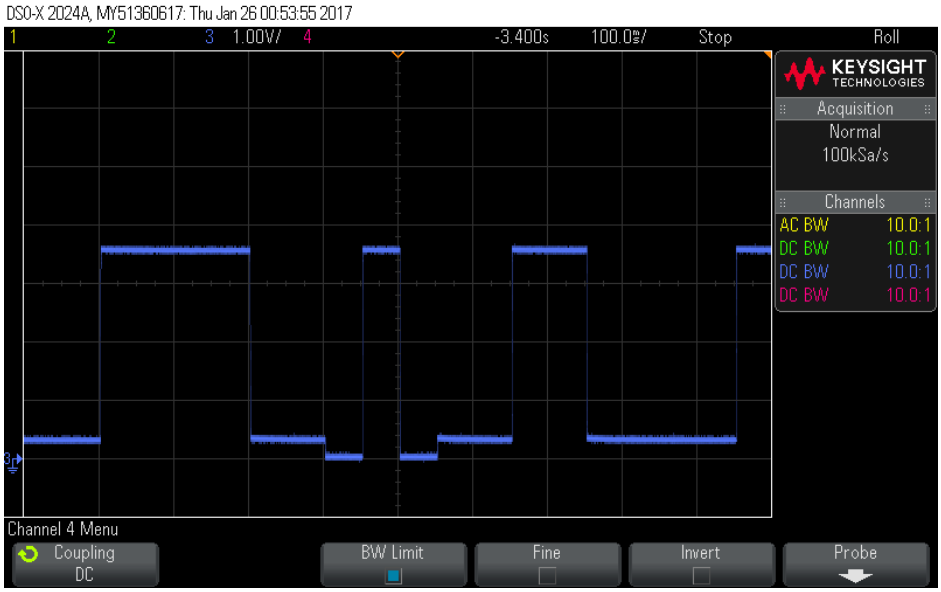
Obrázek 4.3: Záznam signálu z osciloskopu po provedení zkušebního programu

Výsledky byly velice překvapivé. Celkový počet pulzů červené LED (napětí na osciloskopu přibližně 3,7 V; pro zelenou 0,3 V; oranžová 3,9 V) byl 17 (viz obrázek 4.3). Přičemž dle programu mělo proběhnout 200 pulzů.

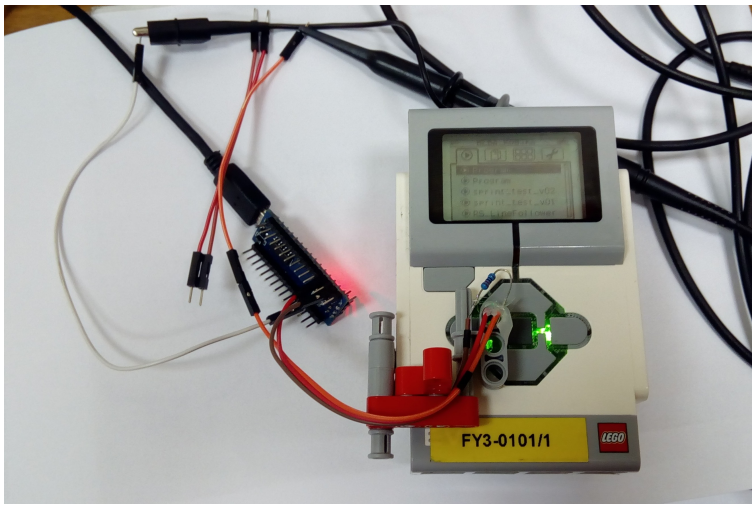
Při detailním zkoumání se ukázalo, že LED nedokáží blikat na 50 Hz, ale mohou svůj stav měnit vždy s frekvencí 20 Hz. Každých 50 ms docházelo ke kontrole stavových proměnných u LED a případné změně stavu (viz obrázek 4.4 – jeden dílek odpovídá 100 ms a 1 V).

Toto chování lze například vysvětlit tak, že v rámci operačního systému EV3 *Bricku* je v určité periodě (přibližně 50 ms) vyvoláno přerušení od časovače, které porovná stav

proměnných a registrů LED a dle výsledku buď rozsvítí, nebo zhasne. Proto v daných intervalech dochází k interferenci mezi časováním LED (20 ms) a interního přerušení (přibližně 50 ms) a podle toho, jak se tyto události sejdou, se jednotlivé LED přepínají. Tento časovač bude pravděpodobně fungovat jen pro LED, ale nebude dále ovlivňovat další chování systému.



Obrázek 4.4: Zoom na záznam z osciloskopu po provedení zkušebního programu



Obrázek 4.5: Foto měřicí sestavy

Z naměřených výsledků vyplývá, že tuto formu testování nelze použít.

4.3 Měření jednotlivých operací

Jelikož nebyli výsledky z prvního testování uspokojivé, rozhodl jsem se provést testy výkonosti jednotlivých operací na standardním LEGO systému a RTOS EV3RT (popis v kapitole

3.7), který dle předchozího zkoumání měl předpoklady k nejlepšímu výkonu.

V rámci testů byla vytvořeny dvě sady:

- matematických operací (sčítání, násobení, dělení, ...) a podmínka
- operace se senzory a motory

Jednotlivé měření se prováděli vícenásobným voláním daných metod (v rozmezí od 1 do 100 000 volání v závislosti na době trvání dané operace) tak, aby bylo možné změřit čas s rozumnou přesností. Každé měření pak bylo zopakováno 1000, aby bylo možné určit průměr a maxima. Zdrojové soubory k testům (jak LEGO, tak EV3RT) jsou k dispozici v repozitáři RB-ev3rt-hrp2-sdk [65].

Matematické operace, jako sčítání, násobení, dělení byly prováděny v plovoucí desetinné čárce (double).

Z výkonnostních testu (tabulka 4.1 a 4.2) je dobře vidět, že zpracování operací probíhá na systému EV3RT více jak o 2 řády rychlejší než u oficiálního systému EV3. Proto jsem vybral tento systém jako nejvhodnější variantu pro vytvoření vývojové prostředí.

Tabulka 4.1: Porovnání relativního výpočetního výkonu originálního LEGO systému a EV3RT – matematické operace a podmínka

Označení testu	AVG/MAX	LEGO	EV3RT	Porovnání
sčítání	AVG	563 us	0.586 us	960 ×
	MAX	680 us	0.602 us	1129 ×
násobení	AVG	388 us	0.506 us	766 ×
	MAX	510 us	0.526 us	969 ×
dělení	AVG	403 us	2.627 us	153 ×
	MAX	510 us	2.943 us	173 ×
mocnina	AVG	557 us	43.350 us	12 ×
	MAX	1480 us	47.860 us	30 ×
odmocnina	AVG	488 us	5.250 us	92 ×
	MAX	890 us	6.146 us	144 ×
sinus	AVG	411 us	12.520 us	32 ×
	MAX	590 us	15.780 us	37 ×
podmínka	AVG	419 us	0.126 us	3325 ×
	MAX	510 us	0.139 us	3669 ×

Tabulka 4.2: Porovnání relativního výpočetního výkonu originálního LEGO systému a EV3RT – *Brick* komponenty, motory a senzory

Označení testu	AVG/MAX	LEGO	EV3RT	Porovnání
brick button	AVG	966 us	0.350 us	2760 ×
	MAX	1100 us	0.362 us	3038 ×
brick led	AVG	787 us	2.866 us	274 ×
	MAX	900 us	3.247 us	277 ×
brick display	AVG	1449 us	396.080 us	3 ×
	MAX	1620 us	409.330 us	3 ×
motor power	AVG	603 us	8.770 us	68 ×
	MAX	720 us	11.630 us	61 ×
motor speed	AVG	612 us	8.840 us	69 ×
	MAX	760 us	11.670 us	65 ×
motors speed	AVG	1023 us	17.490 us	58 ×
	MAX	1160 us	21.120 us	54 ×
CS reflected	AVG	785 us	3.870 us	202 ×
	MAX	940 us	6.980 us	134 ×
CS ambient	AVG	830 us	3.860 us	215 ×
	MAX	990 us	7.100 us	139 ×
CS color	AVG	799 us	3.890 us	205 ×
	MAX	930 us	6.730 us	138 ×
UTS cm	AVG	811 us	3.841 us	211 ×
	MAX	960 us	4.212 us	227 ×
UTS inch	AVG	827 us	3.838 us	215 ×
	MAX	1030 us	4.197 us	245 ×
UTS listen	AVG	845 us	3.769 us	224 ×
	MAX	1220 us	4.115 us	296 ×
GYRO angle	AVG	851 us	3.910 us	217 ×
	MAX	1050 us	6.740 us	155 ×
GYRO rate	AVG	847 us	3.920 us	216 ×
	MAX	1250 us	7.190 us	173 ×
GYRO reset	AVG	25712 us	3.510 us	7325 ×
	MAX	34410 us	6.420 us	5359 ×

Kapitola 5

Vývojová platforma

Poté, co jsem si udělal přehled všech existujících platforem pro LEGO MINDSTORMS EV3 v rámci kapitoly 3, a provedl testování výkonu v kapitole 4 jsem pro svou práci vybral prostředí EV3RT. Nad touto platformou jsem se rozhodl vybudovat vývojové prostředí vhodné pro začátečníky a stanovil jsem si následující požadavky:

- C++ API blízké programovacím blokům v LEGO prostředí s anglickou dokumentací
- průvodce práce s vývojovou platformou v češtině
- modul pro správu EV3 portů (obdoba informačního panelu s porty v EV3 prostředí)
- předpřipravená obsluha komunikace a práce se soubory
- snadné ovládání základních periferií na EV3 (displej, LED, tlačítka)
- vývojářské IDE[1] s jednoduchým rozhraním a našeptáváním
- zautomatizovaný build proces a nahrávání programů do *Bricku*
- jednoduchá instalace u uživatele

V následujících podkapitolách se budu podrobněji věnovat jednotlivým bodům.

5.1 Vývojový proces při rozšiřování EV3RT

Při vývoji a rozšiřování systému EV3RT jsem chtěl využít všech běžných vývojářských nástrojů a služeb. Prioritou bylo verzování, bez kterého se dnes prakticky žádný dlouhodobější projekt neobejde. Díky tomu lze na projektech snadno spolupracovat s lidmi po celém světě. Jelikož EV3RT již mělo svůj repozitář na GitHubu [21], neměl jsem důvod přemýšlet nad jinou službou. Celý projekt vyvíjím jako forky původních repozitářů. Mohu tak jednoduše vyvíjet pro EV3RT a zasílat pull-requesty autorům s případnými opravami nebo vylepšeními. Repozitář RB-ev3rt-hrp2 [64] obsahuje kompletní systém EV3RT (RTOS TOPPERS/HRP2, drivery pro *Brick*, ...).

V repozitáři RB-ev3rt-hrp2-sdk [65] je workspace pro aplikace a knihovny s API pro EV3 (nachází se zde kompletní implementace C++ API). RB-ev3rt-hrp2-sdk je v podobě subrepozitáře distribuován i v RB-ev3rt-hrp2. Repozitáře jsou na GitHubu uloženy pod

[1] IDE = Integrated Development Environment

komunitou RobotikaBrno [63], na jejímž chodu se ve volném čase podílím. Proto také mají i všechny repozitáře předponu RB [66]. RobotikaBrno má za cíl podporu robotiky v Brně a České republice.

V rámci testovaní a integrace se systémem EV3RT jsem použil CI[2] na serveru Travis [72]. Tato služba má velmi dobré propojení s GitHubem. CI po každém commitu spouští kompilaci nad jednotlivými testovacími projekty. Aktuálně tak provádím základní testování přeložitelnosti kódu [57]. Díky tomu mohu zachytit nekonzistentnost na různých místech API, ověřit kompatibilitu s EV3RT a zkontrolovat zkompilovatelnost pro všechny vzorové projekty. Jelikož je celý systém EV3RT hardwarově závislý na LEGO MINDSTORMS EV3, finální testování provádím spouštěním předpřipravených testovacích projektů na reálném hardwaru.

Když jsem přemýšlel nad tím, jak by nejlépe mělo vypadat programovací API pro LEGO, došel jsem po různých pokusech a diskuzích se studenty k závěru, že pro začátečníka bude nejvhodnější objektově orientované API. Pro začínajícího programátora je dle mého mínění objektová abstrakce přirozená z reálného světa a proto se mu s objekty bude lépe pracovat. Zároveň zapouzdřenost objektů odstiňuje uživatele od implementace a tím jej méně rozptyluje od jeho záměru a cíle.

5.1.1 EV3RT C API

Předtím, než jsem začal tvořit vlastní C++ API, jsem si důkladně nastudoval a vyzkoušel EV3RT C API, abych měl jasnou představu, jak vše funguje. Dokumentace byla tvořena v Doxygenu [6], a tak jsem se rozhodl ji rozšířit za pomocí Google Translate a vlastní intuice o anglický překlad. Doxygen je jeden z nejpoužívanějších nástrojů na tvorbu dokumentace k softwarovým projektům. Zvládá generovat jak webové stránky, tak PDF dokumenty pro offline použití.

Po přeložení všech knihoven jsem přes GitHub Pages vytvořil online verzi dokumentace [56]. Zároveň jsem zaslal pull-request autorům EV3RT, kteří jej přijali a na svých webových stránkách odkazují na mnou vytvořenou dokumentaci [20].

5.1.2 EV3RT C++ API

Pro EV3RT již existovalo základní C++ API [20], vytvořené v rámci projektu ETrobocon/etroboEV3 [24]. Z následujících důvodů jsem se jej rozhodl nevyužít:

- nebylo součástí oficiálních EV3RT repozitářů
- autoři EV3RT jej nepovažují za oficiální API a nejedná se o jejich výtvor
- implementovány jen některé funkce a dokumentace v japonštině
- rozdílná představa o implementaci některých části API

5.1.3 C++ API

Při procházení C API v EV3RT jsem nebyl spokojen s jeho použitím pro začátečníky. API sice umožňuje přímý přístup k hardwaru, tím ale uživateli poskytuje větší prostor k chybám a tomu je potřeba předcházet. V C++ API těmto věcem předcházím pomocí

[2] CI = Continuous Integration

různých prostředků: konstrukcí jazyka (např. `enum class`, defaultní parametry funkcí, přetěžováním konstruktorů), názvy funkcí nebo pojmenováním parametrů či výčtových typů.

Pro srovnání jsem připravil ukázku C API (obrázek 5.2) a svého C++ API (obrázek 5.4). Na ukázkách je program pro objetí trojúhelníku. V C API není k dispozici funkce, která by umožňovala řídit oba motory naráz, proto se implementace lehce liší.

```
ev3_motor_rotate(motor_port_t port, int degrees,
                 uint32_t speed_abs, bool_t blocking)
```

Obrázek 5.1: Prototypy C API funkce pro nastaveni ujeté vzdálenosti na jednom motoru

```
1  void main_task(intptr_t unused) {
2      ev3_motor_rotate(EV3_PORT_B, 360, 50, false);
3      ev3_motor_rotate(EV3_PORT_C, 360, 50, true);
4      ev3_motor_rotate(EV3_PORT_B, 520, 50, true);
5
6      ev3_motor_rotate(EV3_PORT_B, 360, 50, false);
7      ev3_motor_rotate(EV3_PORT_C, 360, 50, true);
8      ev3_motor_rotate(EV3_PORT_B, 520, 50, true);
9
10     ev3_motor_rotate(EV3_PORT_B, 360, 50, false);
11     ev3_motor_rotate(EV3_PORT_C, 360, 50, true);
12     ev3_motor_rotate(EV3_PORT_B, 520, 50, true);
13 }
```

Obrázek 5.2: Ukázka kódu pro objetí trojúhelníku v C API
Uživatel musí použít několikrát příkaz pro nastavení otočení motorů se správně zadaným portem (EV3_PORT_B/EV3_PORT_C) a parametry. Může zde jednoduše dojít k přepsání se u označení portu, požadovaného úhlu otočení nebo u parametru blokování – vykonávání programu je pozastaveno do doby, než motor dojede na požadovanou pozici.

```
MotorTank::onForDegrees(int left_speed = 50, int right_speed = 50,
                        int degrees = 360, bool_t brake = true,
                        bool_t blocking = true,
                        unsigned int wait_after_ms = 60);
```

Obrázek 5.3: Prototypy C++ API funkce pro nastaveni ujeté vzdálenosti v režimu MotorTank (spojení dvou motory)

```
1  void main_task(intptr_t unused) {
2      ev3cxx::MotorTank motors(ev3cxx::MotorPort::B, ev3cxx::MotorPort::C);
3
4      motors.onForDegrees(50, 50, 360);
5      motors.onForDegrees(50,  0, 520);
6
7      motors.onForDegrees(50, 50, 360);
8      motors.onForDegrees(50,  0, 520);
9
10     motors.onForDegrees(50, 50, 360);
11     motors.onForDegrees(50,  0, 520);
12 }
```

Obrázek 5.4: Ukázka kódu pro objetí trojúhelníku v mnou navrženém C++ API
Uživatel si nejprve vytvoří instanci objektu `MotorTank` a v konstruktoru nadefinuje porty, na kterých má motory připojeny. Následně již stačí volat funkci `onForRotations()` nad instancí `motors` pro dojetí na konkrétní pozice.

5.1.4 Implementace

V mém C++ API jsem implementoval obsluhu všech komponentů, které EV3RT podporuje (EV3 motory a senzory, tlačítka, LED, displej a reproduktor na *Bricku*). Zároveň jsem chtěl usnadnit uživateli práci se soubory, zápis na displej a komunikace přes Bluetooth, a proto jsem pro tyto činnosti nachystal jednotné rozhraní.

API jsem tvořil v podobě C++ knihoven. Zpočátku jsem se domníval, že rozdělení souborů na zdrojové (`.cpp`) a hlavičkové (`.h`) povede k lepší přehlednosti a rychlejšímu překladu.

Když jsem ale začal používat šablony a část implementace byla ve zdrojových a část v hlavičkových souborech, začal jsem nad výhodností rozdělení pochybovat. Při každé změně API bylo nutné udržovat konzistenci obou souborů a tato činnost zpomalovala vývoj. Proto jsem se rozhodl používat jen hlavičkové soubory. Knihovna je relativně malá a předpokládám, že se to na čase překladu výrazně neprojeví. Dle pozorování to opravdu nemá žádný podstatný vliv.

Jeden zdrojový soubor (`ev3cxx.cpp`) jsem si ale ponechal. Vytvářím v něm globální instance tříd, které by měl mít uživatel k dispozici. Nechci tedy, aby si uživatel tyto instance vytvářel sám lokálně.

Snažím se tím předejít vícenásobnému vytvoření daných objektů a následného nežádoucího chování. Toto řešení jsem zvolil například u třídy pro obsluhu displeje. Každá instance displeje si udržuje informace o aktuální poloze kurzoru na displeji a v případě vytvoření více instancí by docházelo k nežádoucímu chování v podobě přepisování textů z jedné instance tou druhou. Zvažoval jsem i použití návrhového vzoru singleton [4], ale nakonec jsem jej zavrhl, protože by uživatel musel používat konstrukci `getInstance()`, čímž jsem nechtěl začínající uživatele zatěžovat.

Pro snadné používání v uživatelských projektech jsem se rozhodl jeden hlavičkový soubor (`ev3cxx.h`) označit jako hlavní a includovat v něm všechny ostatní hlavičkové soubory s implementacemi jednotlivých tříd. Uživatel tak nemusí řešit jaký soubor includovat, když chce využít danou třídu. Vždy si jen na začátku každého projektu includne jeden hlavičkový soubor. Obzvláště začátečníci to podle mě ocení.

Pro jednoznačné oddělení objektů v mém API jsem vše vložil do namespacu `ev3cxx`. Zároveň ve všech příkladech píši plně kvalifikovaná jména, aby si uživatel tuto zásadu zažil. Podle mého názoru to dobře ukazuje, co je součástí API, a co ne. V rámci knihoven je ještě použit namespace `detail`, který skrývá implementace některých částí kódu před uživateli. V tomto namespace je ukryta např. implementace třídy `display`. Tím jsem docílil, že se uživatel ani při použití našeptávače o daných třídách nedozví, a pokud ano, v příručce mu bude vysvětleno, že s těmito objekty a funkcemi nemá pracovat.

5.2 Vývojářská IDE

Pro programování je vždy nutné mít k dispozici nějaký editor, ať už se jedná o jednoduchý textový editor v podobě Notepadu či VIMu, nebo rozsáhlá vývojová studia typu Eclipse, Microsoft Visual Studio nebo produkty od JetBrains (CLion, PyCharm, PhpStorm, ...).

Ze svých zkušeností z robotických kroužků a kurzů mohu říct, že čím složitější vývojové studio, tím více se v něm začínající programátoři ztrácejí. Na druhou stranu ani kombinace terminálu a VIMu není dle mého názoru pro začátečníky vhodná.

Proto jsem se rozhodl udělat si rozsáhlejší průzkum několika vývojových prostředí z různých kategorií:

A) základní textové editory

B) editory s doplňky, rozšiřujícímy moduly a podporou skriptování

C) velká vývojový IDE s integrovanými překladači a debuggery

V následujících podkapitolách bych chtěl některá IDE rozebrat podrobněji a vysvětlit, co je na nich dle mého názoru dobré a co špatné pro použití s začínajícími programátory a pro výuku na školách.

5.2.1 Notepad

Většina lidí se s ním již někdy setkala a jde o zástupce z kategorie A. Je standardní součástí Windows (ostatní systémy mají podobné alternativy) a umožňuje jen základní úpravu textu. Pro rychlé zapsání krátké poznámky do souboru většinou postačí. Editor ale nenabízí funkce jako syntax highlighting, automatické odsazování dle kontextu, našeptávání nebo spouštění skriptů. Pro potřeby vývoje je tedy nevhodný.

5.2.2 PSPad

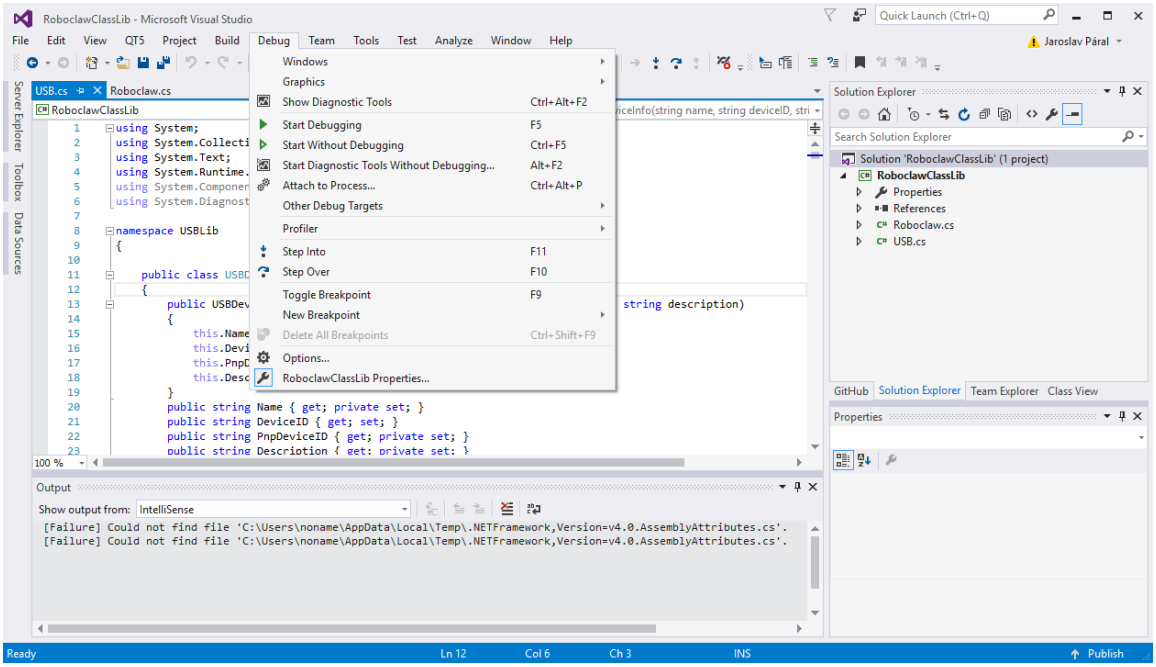
PSPad je jeden ze zástupců kategorie B. Jedná se o editor s velkou nabídkou funkcí ať už pro práci se samotným textem (zobrazování netisknutelných znaků, volba typu konce řádku, nastavení znakových sad, porovnání rozdílů v textech), tak i pro správu celých projektů (FTP připojení, spouštění skriptů). V minulosti byl hojně využívám pro tvorbu webových stránek nebo i programování.

Nabízí základní slovník a předpřipravené šablony pro různé jazyky jako HTML, PHP, C/C++, TeX a mnoho dalších. Ve svých schopnostech už ale ztrácí, co se týká inteligentního našeptávání například u dostupných funkcí v C/C++ knihovnách. Zároveň možnost spouštění skriptů není nijak moc zaintegrována do prostředí a nepůsobí konzistentně. Proto tento editor není vhodný pro rozsáhlejší vývoj.

5.2.3 Microsoft Visual Studio, Eclipse, JetBrain produkty

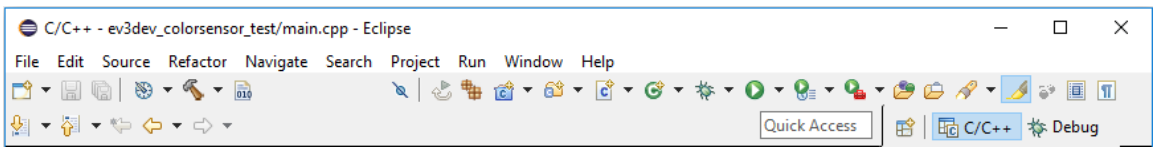
Vývojová IDE jako Microsoft Visual Studio, Eclipse, JetBrain produkty (kategorie C) jsem se rozhodl shrnout do jedné kapitoly, protože ačkoliv to možná není na první podhled patrné, z pohledu UI jsou na tom velmi podobně. Nabízí obrovskou paletu funkcí, které jsou uživateli k dispozici. Relativně jednoduše lze vytvářet celé projekty s předpřipravenými build procesy. Obsahují jedny z nejlepších našeptávačů, které na trhu existují. Jsou zde k dispozici debuggery, pomocí kterých lze lépe najít chybu v programech. A to je výčet jen několika zajímavých funkcí.

Ovšem podle mých zkušeností většina začínajících uživatelů využije sotva 5 procent z těchto prostředí. To znamená, že zbylých 95 procent funkcí je pro tyto uživatele přebytečných a matoucích. Pro příklad uvádím obrázky 5.5 a 5.6, na kterých lze vidět rozsáhlost UI a složitost jednotlivých položek v menu. Proto jsem se rozhodl, že tato IDE nejsou vhodná pro začátečníky.



Obrázek 5.5: Ukázka prostředí v Microsoft Visual Studio Community 2015

Aby uživatel spustil program, musí vybrat volbu `Start Without Debugging`, což je v pořadí až druhá volba a začátečník ani nemusí tušit, co znamená slovo debugging.



Obrázek 5.6: Ukázka panelu nástrojů v prostředí Eclipse

Zkuste vysvětlit skupině 15 začátečníků, kterou ikonku mají zmáčknout pro překlad, a kterou pro spuštění. S velkou pravděpodobností to polovina udělá poprvé špatně.

5.2.4 Arduino IDE a Processing

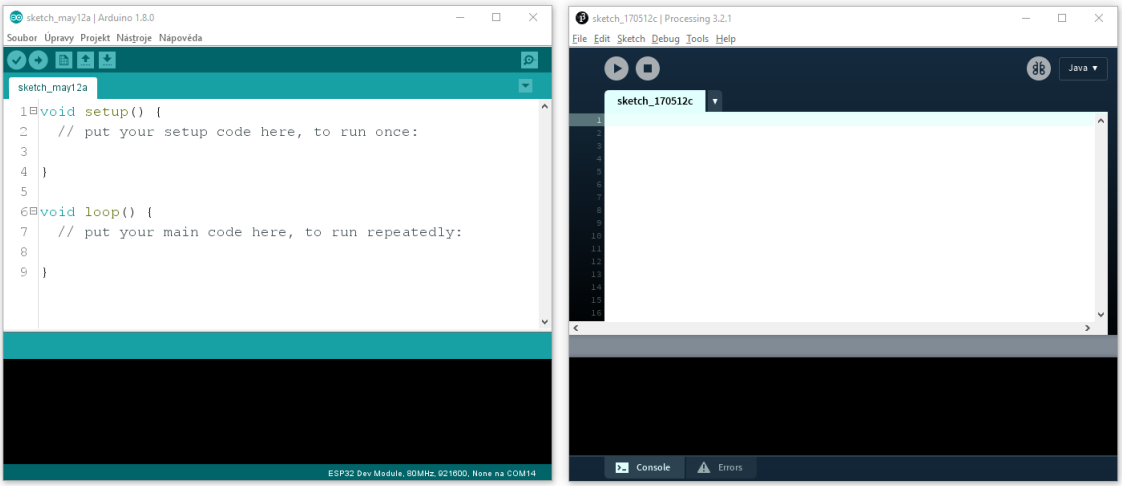
Na opačné straně vývojářských nástrojů stojí programy Arduino IDE [1] a Processing [55] (stále kategorie C).

Arduino IDE je vývojářský nástroj pro open-source elektronickou platformu Arduino, s kterou lze jednoduše sestavovat hardware a programovat software. To vše bez potřeby drahého vybavení a hlubších znalostí programování a mikroprocesorů. Platforma využívá převážně mikrokontroléry AVR od firmy Atmel. Jako programovací jazyk se používá Wiring [74], který vychází z jazyka C.

Processing je taktéž open-source vývojářský program, ale i jazyk, určený pro snadnou tvorbu vizuálně zajímavé a interaktivní grafiky. Vznikl v rámci MIT Media Lab [54] jako nástroj pro umělce a lidi, kteří chtějí tvořit elektronické umění, ale nejsou programátoři. Proto bylo hlavním cílem vytvořit jednoduché prostředí, které jim to umožní. Jazyk Processing je postaven nad Javou.

Arduino na Processing navázalo a snaží se tuto myšlenku rozšířit na hardware. Zároveň použilo vývojářský program od Processingu a upravilo jej pro své potřeby. V mnoha oblastech se tyto dva projekty doplňují. Například lze pomocí Processingu graficky vizualizovat naměřené hodnoty ze senzorů připojených k Arduinu.

Podle mého názoru nabízejí začátečníkovi přesně to co potřebuje: jednoduché UI, v kterém se rychle zorientuje a hned najde základní komponenty potřebné k práci (hlavně tlačítka pro překlad a spuštění programu). Vše si lze prohlédnout na obrázcích 5.7 a 5.8.



Obrázek 5.7: Ukázka jednoduchých prostředí Arduino IDE a Proccessing
Základní ovládací prvky potřebné při vývoji programů (tlačítka pro přeložení a spuštění programu) jsou jasně a zřetelně na očích a neztrácejí se v mnoha dalších ikonkách.

Obrázek 5.8: Panel s nástroji v Arduino IDE – uživatel zde hned najde vše, co potřebuje

I tyto programy mají ale své problémy. Arduino IDE neumí našeptávat nebo zobrazovat nápovědu k funkcím. Zároveň špatně zobrazuje chybové hlášky objevené při překladu.

Processing je na tom lépe. Má integrovaný základní našeptávač. Dokonce zvládá ukazovat datové typy parametrů. Nezobrazuje však už jejich název, tudíž většinou nelze určit, co za parametr se očekává. Lépe je v editoru zapracováno i vypisování chyb a je možné zapnout i kontrolu správnosti kódu v průběhu psaní.

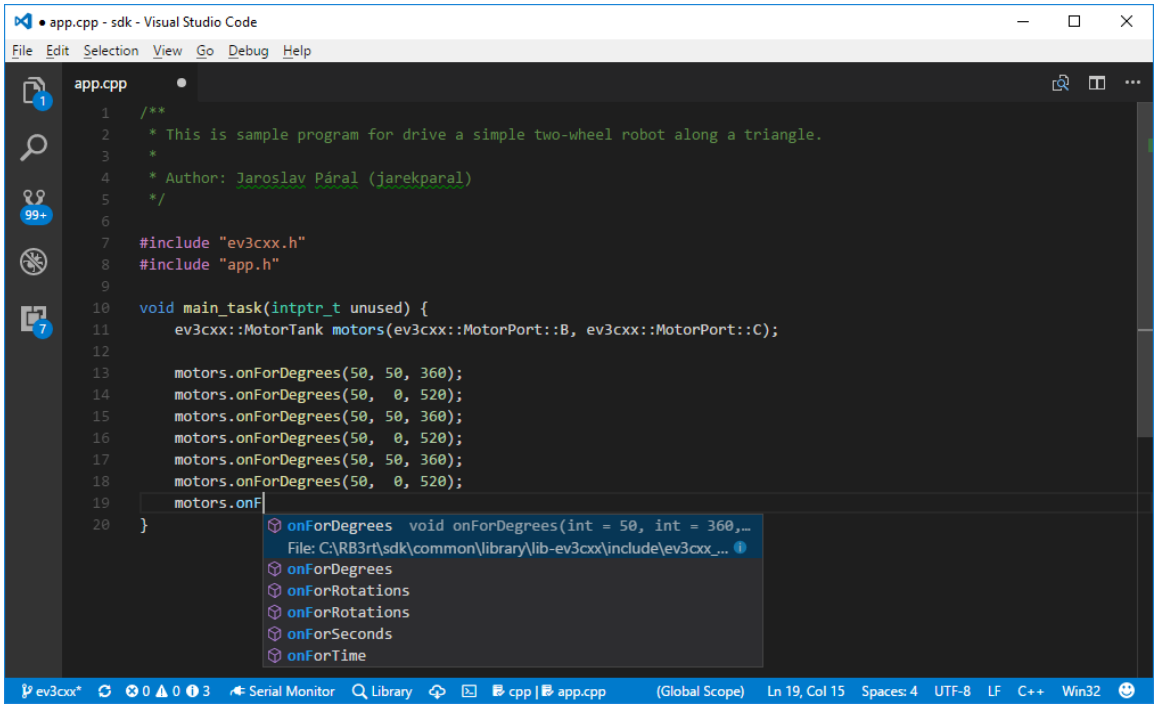
Bohužel tato IDE jsou šitá na míru daným platformám a proto je nelze v rámci mé práce využít. Poslouží ale jako dobrá inspirace pro výběr vhodného editoru.

5.2.5 Visual Studio Code

Po vyzkoušení výše zmíněných IDE jsem se rozhodl vrátit ke kategorii B a vyzkoušet nový editor Visual Studio Code. Jedná se o relativně čerstvý počin Microsoftu, který tento editor představil v rámci BUILDu 2015 [49].

Visual Studio Code (dále jen VS Code) je zdarma dostupný multiplatformní (Windows, Linux, Mac) open-source editor s integrovaným našeptáváním, Gitem a podporou doplňků a skriptování. Probíhá u něj velmi rychlý vývoj a aktuálně jej lze považovat za jeden z nejlepších editorů v této kategorii. V mnoha ohledech již předčil své konkurenty (editor Atom a Sublime). Například ve srovnání s Atomem je výrazně rychlejší.

Oproti vývojovým IDE z kategorie C má dle mého názoru značně jednoduší UI, při zachování rozsáhlých možností rozšiřování a pokročilých funkcí. Základní funkce jsou dostupné přes nabídku v horní a boční liště. Všechny ostatní funkce lze najít přes speciální příkazovou řádků, která se spouští klávesovou zkratkou `Ctrl+Shift+P`. Tato příkazová řádka je ale běžnému uživateli skrytá a tak jej neruší při používání. Díky tomu nabízí VS Code velké množství funkci při zachování jednoduchého uživatelského rozhraní.



Obrázek 5.9: Ukázka našeptávání ve Visual Studio Code – doplňování názvu funkce

Do VS Code je možné nainstalovat celou řadu rozšíření. Tím nejzajímavější je doplněk `C/C++ for Visual Studio Code` [75] přímo od Microsoftu, který nabízí velmi dobré našeptávání, automatické formátování nebo vyhledávání definic a symbolů. Tato funkcionalita

například velmi chybí v Arduino IDE, na což si většina mých studentů v kurzech stěžuje, když zjistí, že ji jiný nástroj umí.

Dalšími důležitými funkcemi jsou integrovaný terminál a možnost jednoduchého spouštění skriptů. VS Code také nabízí snadné nadefinování klávesových zkratek, s jejichž pomocí lze volat vlastní skripty. Touto cestou jsem se rozhodl implementovat překlad uživatelských programů a následné zasílání programu do EV3. K veškeré činnosti uživateli stačí jen tři klávesové zkratky: přeložení programu (`F5`), odeslání programu (`F6`) a kombinace předchozích dvou příkazů v rámci jedné klávesy (`F7`). Zároveň jsem předpřipravil zkratku `F8` pro spuštění aplikace Lorris s integrovaným terminálem pro komunikaci přes Bluetooth a `F9` pro otevření webových stránek s dokumentací.

5.3 Zprovoznění prostředí

Vývojové prostředí se skládá z editoru, kompilátoru, EV3RT systému, knihoven, skriptů a bootovatelné image systému, určené ke spouštění na *Bricku*. V následujících dvou podkapitolách bude probrán postup na zprovoznění celého prostředí.

5.3.1 Instalace editoru, překladače a nahrání programu do *Bricku*

Pro překlad a nahrání programu do systému EV3RT na Windows je potřeba nainstalovat následující software:

- Cygwin (s balíčky make, perl, diffutil, gcc-core, gcc-g++, curl a libboost-devel)
- GCC překladač pro procesory ARM – gcc-arm-none-eabi
- Visual Studio Code s doplňkem C/C++ for Visual Studio Code

Abych uživateli a případně učitelům usnadnil instalaci a konfiguraci celého prostředí, připravil jsem batch skript, který všechen potřebný software stáhne a postupně nainstaluje. Skript vyžaduje administrátorská práva a občasnou interakci uživatel. Součástí skriptu je i stažení systému EV3RT s C++ API a šablona se vzorovým projektem pro uživatele. Aktuálně je vyžadováno fixní umístění softwaru Cygwin a prostředí pro EV3RT do kořenového adresáře disku C.

Po instalaci si již může uživatel nakopírovat vzorový projekt, který je uložený ve složce `C:/RB3rt-projekt`, kam uzná za vhodné. Otevřít jej ve VS Code, nastavit klávesové zkratky a začít programovat. Pokud chce svůj kód přeložit, stačí zmáčknout klávesu `F5`. Tím se zavolá přepřipravený skript, který spustí kompilaci přes Makefile v Cygwinu. Jelikož jsem musel vyřešit propojení kompilace v Cygwinu s VS Code, kopíruji vždy uživatelův projekt do adresáře se systémem EV3RT, kde následně probíhá samotný překlad.

Celý průběh překladu vidí uživatel v terminálu, který je součástí VS Code. Pokud se vyskytne jakákoliv chyba, je o ní uživatel informován přes zmíněný terminál. Při úspěšném přeložení lze následně zkompilovanou aplikaci nahrát na EV3 *Brick*. První možností, jak nahrát aplikaci do systému je za pomocí USB. *Brick* s EV3RT se při propojení s PC přes USB kabel chová jako USB Mass Storage zařízení připojit k počítači a lze s ním pracovat jako se standardní SD kartou. Další možností, většinou komfortnější, je využít integrovaný Bluetooth. Přes něj lze zasílat aplikace do systému a zároveň komunikovat s programem.

EV3RT nabízí dva módy nahrávání přes Bluetooth: SPP[3], standardní sériový port a PAN[4], v kterém se EV3RT chová jako síťové zařízení.

Z pohledu používání je lepší volba PAN módu, protože nekoliduje s SPP módem v uživatelských aplikacích. V rámci obsluhy Bluetooth, při programování vlastních aplikací, má uživatel dostupný jen SPP mód. Proto pro odesílání aplikací využívám právě PAN a uživatel ve VS Code spustí odeslání pomocí klávesy `F6` (případně pokud chce program přeložit a hned odeslat, může použít klávesu `F7`). Před tím, než může ale uživatel nahrát program do EV3RT, je potřeba se na *Bricku*, v menu systému, přepnout do režimu příjmu aplikace přes PAN.

5.3.2 Nahrání systému EV3RT do EV3

Pro provozování EV3RT na *Bricku* je potřeba nahrát image systému na micro SDHC kartu (minimální velikost musí být 4 GB – EV3RT podporuje jen SDHC karty). EV3 *Brick* umožňuje spouštění systémů i z SD karty. Tento režim lze přirovnat k dual-bootu na PC. Výhodou je, že tím nehrozí zničení originálního systému a je možné velfbrmi rychle přecházet mezi EV3RT a LEGO systémem. Musíte jen vložit SD kartu s EV3RT a nastartovat jej nebo naopak vypnout EV3RT, vyjmout kartu a spustit původní systém.

Image systému se automaticky stáhla v rámci instalace vývojové prostředí, popsané v předchozí kapitole, do složky `C:/RB3rt-image`. Uživatel jen musí veškerý obsah daného adresáře nakopírovat na micro SDHC kartu a následně ji vložit do *Bricku*.

5.4 Dokumentace

Dokumentace k vývojovému prostředí lze rozdělit na dvě varianty. První varianta jsou komentáře ve zdrojovém kódu se syntaxí Doxygenu. Tuto dokumentaci jsem tvořil pro pokročilejší uživatele, kteří by například chtěli rozšiřovat funkčnost C++ API. Zároveň je tvořená v angličtině a tudíž může posloužit lidem z celého světa. Dokumentace je automaticky generovaná a je součástí repozitáře s API [56].

Druhou variantou, která je primárně určená pro české uživatele tvořím za pomocí značkovacího jazyku RestructuredText (reSt) a webové službu Read The Docs [58]. Dokumentace je díky této službě dostupná jak online [59], tak ji lze stáhnout v PDF formátu nebo HTML pro offline používání. Pomocí obrázků programových bloku z LEGO softwaru ukazuji spojitost se svým objektovým API a LEGO bloky. Součástí textů je i krátký tutoriál, který uživatele seznámí v několika lekcích se základními komponentami z LEGO MINDSTORMS EV3 a jejich ekvivalenty v C++ API. V rámci jednotlivých úkolů si je uživatel prakticky vyzkouší. Ukázku dokumentace lze najít v příloze.

5.5 Dokumentace

Dokumentace k vývojovému prostředí lze rozdělit na dvě varianty. První varianta jsou komentáře ve zdrojovém kódu se syntaxí Doxygenu. Tuto dokumentaci jsem tvořil pro pokročilejší uživatele, kteří by například chtěli rozšiřovat funkčnost C++ API. Zároveň je tvořená v angličtině a tudíž může posloužit lidem z celého světa. Dokumentace je automaticky generovaná a je součástí repozitáře s API [56].

[3] SPP = Serial Port Profile

[4] PAN = Personal Area Network

Druhou variantou, která je primárně určená pro české uživatele, tvořím za pomocí značkovacího jazyku RestructuredText (reSt) a webové službu Read The Docs [58]. Dokumentace je díky této službě dostupná jak online [59], tak ji lze stáhnout v PDF formátu nebo HTML pro offline používání. Pomocí obrázků programových bloku z LEGO softwaru ukazuji souvislost mezi objektovým C++ API a LEGO bloky. Součástí textů je i krátký tutoriál, který uživatele seznámí v několika lekcích se základními komponentami z LEGO MINDSTORMS EV3 a jejich ekvivalenty v C++ API. V rámci jednotlivých úkolů si je uživatel prakticky vyzkouší.

5.6 Testování se studenty

V rámci testování celého prostředí jsem začal zkoušet používat objektové API se studenty na SPŠ a VOŠ Brno, Sokolská, kde vedu robotický kroužek. Studenti mají k dispozici již přes rok stavebnice LEGO MINDSTORMS EV3 a za tuto dobu jsem s nimi již absolvoval několik soutěží. Někteří z nich se s programováním setkali poprvé až v LEGO softwaru. Část ale měla již předchozí zkušenosti v jiných programovacích jazycích (C/C++, Java).

Testování probíhalo tak, že studenti dostali půl hodiny na prostudovaní dokumentace k C++ API na Read The Docs [58]. Během té doby pokládali dotazy, s tím co jim nebylo jasné. Většinou se jednalo o nedostatečně vysvětlenou oblast v API.

Následně jsem studentům postupně zadával řešení jednotlivých úkolů, které jsou zpracovány v rámci Robotutoriálu na Read The Docs [58]. Studenti tak zkoušeli základní činnosti jako objet s robotem trojúhelník, nakreslit s ním domeček, dojet k překážce a odvézt ji. Finálním úkolem byla jízda po čáře. Ke každému příkladu bylo předpřipraveno i vzorové řešení, jak v LEGO blocích, tak C++ API. Pokud si studenti nevěděli rady, mohli se podívat na řešení. Dle pozorování ovšem skoro nikdo vzorová řešení nevyužil a zvládali požadované úkoly jen s dokumentaci k jednotlivým třídám a metodám. Prakticky to brali jako výzvu, kdo první zvládne naprogramovat daný úkol aniž by se díval na řešení.

Při těchto jednoduchých úkolech nedělalo studentům problém s C++ API pracovat a bylo vidět, že jim podobnost s LEGO bloky velmi pomáhá. V rámci rozsáhlosti testovacích úkolů (v optimálním případě byl zdrojový kód vždy do 15 řádků v hlavní funkci) jsme nemohli otestovat přehlednost a srozumitelnost kódu při rozsáhlejších projektech v porovnání s LEGO bloky. Aktuálně ale mají studenti API k dispozici a plánují jej využít k programování robota na soutěž Ketchup House v rámci Robotického dne 2017 v Praze. Během této doby budu se studenty dále pokračovat na rozvoji API, tak aby jim co nejlépe vyhovovalo. Již teď si ale myslím, že celé prostředí lze běžně využívat k programování LEGO MINDSTORMS EV3.

Kapitola 6

Závěr

Práce zpracovává rešerši na dostupné vývojové platformy v rámci výuky programování na LEGO MINDSTORMS EV3. Z výsledků rešerše vyplynulo, že nejvhodnější systém pro EV3 je RTOS EV3RT. Hlavními důvody jsou výkonost, real-timovost, otevřený zdrojový kód, snadná modifikovatelnost a dostupnost pro uživatele na všech běžných operačních systémech (Windows, Mac, Linux).

RTOS EV3RT byl výkonnostně porovnán se standardním systémem v EV3 *Bricku* rozsáhlou sadou testů. Výsledky ukazují výrazný výkonnostní rozdíl při provádění jednotlivých činností. EV3RT překonává LEGO systém ve většině oblastí v rozmezí sto až tisíci násobku. Z těchto důvodů byl RTOS EV3RT vybrán jako vhodný systém pro výuku programování.

Práce se dále věnuje návrhu vývojového prostředí. Vybírá požadavky, které budou u daného prostředí prioritou. Pro EV3RT práce zpracovává objektově orientované C++ API s anglickou dokumentací, které pokrývá kompletní funkcionality LEGO MINDSTORMS EV3. API bylo vyvinuto s důrazem na jednoduchý přechod z obrázkového LEGO Softwaru a proto by pro uživatele mělo být snadné začít v něm programovat. To se také potvrdilo během testování se studenty.

Byl proveden i průzkum několika vývojářských editorů. Za nejvhodnější pro začátečníky v programování byl zvolen Visual Studio Code, kvůli svému jednoduchému UI, které zároveň zachovává rozsáhlé možností přizpůsobení, velkou palety doplňků a kvalitní našeptávání.

K celému prostředí je připravená česká dokumentace. Pro uživatele je připraven popis C++ API i vývojářského editoru. Pro každou oblast používání je zpracováno přehledné přirovnání jednotlivých funkcí s LEGO Softwarem. Žáci SPŠ a VOŠ Brno, Sokolská již postupně toto prostředí začínají využívat k programování. Během testování s nimi byli odhaleny některé nedostatky v dokumentaci a pojmenování metod v API. Jinak si ale studenti prostředí i API pochvalovali. Za jednu hodinu si zvládli projít dokumentaci a začít programovat. Díky podobnosti API s LEGO Softwarem jim nedělalo problém psát ekvivalentní programy. Aktuálně probíhají průběžné úpravy dle zpětné vazbu studentů.

Celé prostředí je momentálně připraveno k plnému využívání s LEGO MINDSTORMS EV3. I tak je tu mnoho oblastí, ve kterých jej lze dále rozvíjet. Rozpracovány jsou tutoriály se složitějšími úlohami pro studenty, na kterých by se mohli naučit další činnosti (programování více souběžných tasku, vytváření PID regulátorů, rozsáhlejší loggování do souborů). V budoucnu je i plánu přidání další vrstvy v API pro jednoduché zakomponování činností jako práce se souřadnicemi, plánování trasy nebo herní logika.

Literatura

[1] *Arduino – Home*. Arduino, [Online; navštíveno 11.05.2017].
URL https://www.arduino.cc/

[2] Baichtal, J.: *Hacking your Lego Mindstorms EV3 kit*. Indianapolis, Indiana: Que, 2016, ISBN 978-0-7897-5538-4.

[3] Caprani, O.: *RCX Manual*. University of Aarhus, Department of Computer Science„ Únor 2006, [Online; navštíveno 21.01.2017].
URL http://www.legolab.daimi.au.dk/CSaEA/RCX/Manual.dir/RCXManual.html

[4] *Design Patterns Singleton Pattern*. tutorialspoint.com, [Online; navštíveno 11.05.2017].
URL https://www.tutorialspoint.com/design_pattern/singleton_pattern.htm

[5] Diaz, J.: *Everything You Always Wanted to Know About Lego*. Gizmodo Media Group, Červen 2008, [Online; navštíveno 19.01.2017].
URL http://gizmodo.com/5019797/everything-you-always-wanted-to-know-about-lego

[6] *Doxygen: Main Page*. Doxygen, [Online; navštíveno 11.05.2017].
URL https://www.doxygen.org/

[7] *Robot Edison*. [Online; navštíveno 21.01.2017].
URL https://meetedison.com/

[8] EV3, A.-O.: *Lego Mindstorms EV3 Boot to ev3dev Operating System*. [Online; navštíveno 13.05.2017].
URL https://www.youtube.com/watch?v=0DGYAJDQa0g

[9] *ev3dev Home*. ev3dev.org, [Online; navštíveno 23.01.2017].
URL http://www.ev3dev.org/

[10] *ev3dev – Kernel Release Cycle 10*. ev3dev.org, [Online; navštíveno 26.01.2017].
URL http://www.ev3dev.org/news/2016/04/11/Kernel-Release-Cycle-10/

[11] *ev3dev – Kernel Release Cycle 18*. ev3dev.org, [Online; navštíveno 26.01.2017].
URL http://www.ev3dev.org/news/2017/01/25/kernel-release-cycle-18/

[12] *ev3dev-lang-cpp/drive-test.cpp at master · ddemidov/ev3dev-lang-cpp*. GitHub, Inc., [Online; navštíveno 09.05.2017].
URL https://github.com/RoboticsBrno/RB-ev3dev-cpp-lib

[13] *ev3dev – Programming Languages*. ev3dev.org, [Online; navštíveno 23.01.2017].
URL http://www.ev3dev.org/docs/programming-languages/

[14] *Motors / Output Devices – ev3dev-jessie Linux kernel drivers 19 documentation*. ev3dev.org, [Online; navštíveno 10.04.2017].
URL http://docs.ev3dev.org/projects/lego-linux-drivers/en/ev3dev-jessie/motors.html

[15] *Python language bindings for ev3dev — python-ev3dev 0.8.1.post13 documentation*. Ralph Hempel et al., [Online; navštíveno 26.01.2017].
URL http://python-ev3dev.readthedocs.io/en/latest/index.html

[16] *rhempel/ev3dev-lang-python: Pure python bindings for ev3dev*. GitHub, Inc., [Online; navštíveno 11.04.2017].
URL https://github.com/rhempel/ev3dev-lang-python

[17] *RoboticsBrno/RB-ev3dev-cpp-lib*. GitHub, Inc., [Online; navštíveno 11.04.2017].
URL https://github.com/ddemidov/ev3dev-lang-cpp/blob/master/demos/drive-test.cpp

[18] *Sensors / Input Devices – ev3dev-jessie Linux kernel drivers 19 documentation*. ev3dev.org, [Online; navštíveno 10.04.2017].
URL http://docs.ev3dev.org/projects/lego-linux-drivers/en/ev3dev-jessie/sensors.html

[19] *Tips to get a more constant loop time · Issue #324 · ev3dev/ev3dev*. GitHub, Inc., [Online; navštíveno 27.01.2017]].
URL https://github.com/ev3dev/ev3dev/issues/324

[20] *Documents · EV3RT*. EV3RT Project, [Online; navštíveno 11.05.2017].
URL http://ev3rt-git.github.io/documents/

[21] *EV3RT*. GitHub, Inc., [Online; navštíveno 27.01.2017].
URL https://github.com/ev3rt-git

[22] *Get started · EV3RT*. EV3RT Project, [Online; navštíveno 10.05.2017].
URL http://ev3rt-git.github.io/get_started/

[23] *What is EV3RT? · EV3RT*. EV3RT Project, [Online; navštíveno 27.01.2017].
URL ev3rt-git.github.io

[24] *ETrobocon/etroboEV3*. GitHub, Inc., [Online; navštíveno 11.05.2017].
URL https://github.com/ETrobocon/etroboEV3

[25] *fischertechnik – ROBOTICS*. fischertechnik GmbH, [Online; navštíveno 20.01.2017].
URL http://www.fischertechnik.de/en/Home/products/computing.aspx

[26] *HELAGO eshop – fischertechnik ROBOTICS TXT COMPETITION SET*. HELAGO-CZ s. r. o., [Online; navštíveno 20.01.2017].
URL http://www.helago-cz.cz/eshop-kategorie-fischer-technik-2459.html

[27] *WHAT IS FIRST LEGO LEAGUE?* FIRST, [Online; navštíveno 23.01.2017].
URL http://www.firstlegoleague.org/about-fll

[28] Garber, G.: *Learning LEGO Mindstorms EV3 : build and create interactive, sensor-based robots using your LEGO Mindstorms EV3 kit*. Birmingham, UK: Packt Publishing, 2015, ISBN 978-1-78398-502-9.

[29] *Hitachi Single-Chip Microcomputer H8/3297 Series H8/3297 HD6473297, HD64333297 H8/3296 HD6433296 H8/3294 HD6473294, HD6433294 H8/3292 HD6433292*. HITACHI, [Online; navštíveno 21.01.2017].
URL http://www.legolab.daimi.au.dk/CSaEA/RCX/Manual.dir/H8Hardware.pdf

[30] HUMUSOFT: *Ceník Matlab*. [Online; navštíveno 25.01.2017].
URL http://www.humusoft.cz/DOCS/matlab.pdf

[31] *Best Alternatives to Lego Mindstorms Kits*. Into Robotics, Prosinec 2016, [Online; navštíveno 20.01.2017].
URL
https://www.intorobotics.com/best-alternatives-lego-mindstorms-kits/

[32] ITRON Committee, T. A.: *µITRON4.0 Specification Ver. 4.00.00*. TRON ASSOCIATION, JAPAN, 2002, [Online; navštíveno 27.01.2017].
URL http://www.ertl.jp/ITRON/SPEC/FILE/mitron-400e.pdf

[33] *History of LEGO Robotics*. The LEGO Group, [Online; navštíveno 21.01.2017].
URL https://www.lego.com/en-us/mindstorms/history

[34] *LEGO MINDSTORMS – RJ-12 connector*. [Online; navštíveno 23.01.2017].
URL http://www.philohome.com/nxtplug/cable02.jpg

[35] *The LEGO History 1930's – vznik společnosti LEGO*. The LEGO Group, [Online; navštíveno 19.01.2017].
URL https://www.lego.com/en-us/aboutus/lego-group/the_lego_history/1930

[36] *What's Inside the NXT Brick?* mindstormsnxt.blogspot.cz, Srpen 2006, [Online; navštíveno 23.01.2017].
URL
http://mindstormsnxt.blogspot.cz/2006/08/whats-inside-nxt-brick.html

[37] *Ke stažení - Mindstorms LEGO.com*. The LEGO Group, 2013, [Online; navštíveno 12.02.2017].
URL https://www.lego.com/cs-cz/mindstorms/downloads

[38] *LEGO MINDSTORMS EV3 Firmware Developer Kit*. The LEGO Group, 2013, [Online; navštíveno 23.01.2017].
URL https://le-www-live-s.legocdn.com/sc/media/files/ev3-developer-kit/lego%20mindstorms%20ev3%20firmware%20developer%20kit-7be073548547d99f7df59ddfd57c0088.pdf?la=en-us

[39] *BrickPi - Dexter Industries*. Dexter Industries, [Online; navštíveno 05.09.2016].
URL https://www.dexterindustries.com/BrickPi/

[40] *Eduxe.cz eshop – 45544 EV3 Základní souprava*. EDUXE s.r.o., [Online; navštíveno 20.01.2017].
URL http://www.eduxe.cz/product/45544-ev3-zakladni-souprava-710/

[41] *Gyro, MultiSensitivity Accelerometer and Compass for NXT or EV3*. mindsensors.com, [Online; navštíveno 10.05.2017].
URL http://www.mindsensors.com/ev3-and-nxt/15-gyro-multisensitivity-accelerometer-and-compass-for-nxt-or-ev3

[42] *PiStorms LEGO Interface – mindsensors.com*. mindsensors.com, [Online; navštíveno 05.09.2016].
URL http://www.mindsensors.com/content/78-pistorms-lego-interface

[43] *Lego Mindstorms – Programming languages*. Wikipedia.org, [Online; navštíveno 23.01.2017].
URL
https://en.wikipedia.org/wiki/Lego_Mindstorms#Programming_languages_2

[44] Li, Y.; Ishikawa, T.; Matsubara, Y.; aj.: A Platform for LEGO Mindstorms EV3 Based on an RTOS with MMU Support. In *OSPERT 2014: the 10th Annual Workshop on Operating Systems Platforms for Embedded Real-Time Applications; July 8th, 2014 in Madrid, Spain*, editace B. B. Brandenburg; S. Kato, MPI-SWS, 2014, s. 51–59, [Online; navštíveno 24.01.2017].
URL
https://people.mpi-sws.org/~bbb/proceedings/ospert14-proceedings.pdf

[45] *LEGO MINDSTORMS EV3 Support from MATLAB – Hardware Support – MATLAB & Simulink*. The MathWorks, Inc., [Online; navštíveno 25.01.2017].
URL http://www.humusoft.cz/DOCS/matlab.pdf

[46] *LEGO MINDSTORMS NXT Support from MATLAB – Hardware Support – MATLAB & Simulink*. The MathWorks, Inc., [Online; navštíveno 25.01.2017].
URL
https://www.mathworks.com/hardware-support/lego-mindstorms-matlab.html

[47] *MerkurToys eshop – Robotické sety MERKUR*. MERKURTOYS s. r. o, [Online; navštíveno 20.01.2017].
URL http://www.merkurtoys.cz/vyrobky/roboticke-a-mechatronicke-sety

[48] *Lego Mindstorms NXT – BricxCC, Next Byte Codes, Not eXactly C*. Wikipedia.org, [Online; navštíveno 23.01.2017].
URL https://en.wikipedia.org/wiki/Lego_Mindstorms_NXT#BricxCC.2C_Next_Byte_Codes.2C_Not_eXactly_C

[49] Montgomery, John: *BUILD 2015 News: Visual Studio Code, Visual Studio 2015 RC, Team Foundation Server 2015 RC, Visual Studio 2013 Update 5*. Microsoft Corporation, [Online; navštíveno 11.05.2017].
URL https://blogs.msdn.microsoft.com/visualstudio/2015/04/29/build-2015-news-visual-studio-code-visual-studio-2015-rc-team-foundation-server-2015-rc-visual-studio-2013-update-5/

[50] *NI LabVIEW Module for LEGO® MINDSTORMS® – National Instruments*. National Instruments Corporation, [Online; navštíveno 25.01.2017].
URL http://sine.ni.com/nips/cds/view/p/lang/cs/nid/212785

[51] *Lego Mindstorms NXT – Programming*. Wikipedia.org, [Online; navštíveno 23.01.2017].
URL https://en.wikipedia.org/wiki/Lego_Mindstorms_NXT#Programming

[52] *Blockly pro PICAXE....* PICAXE.CZ, Září 2015, [Online; navštíveno 20.01.2017].
URL http://www.picaxe.cz/blockly-pro-picaxe/

[53] *Robot Pololu 3pi Robot*. Pololu Corporation, [Online; navštíveno 21.01.2017].
URL https://www.pololu.com/product/975

[54] *Overview Processing.org*. Processing Foundation, [Online; navštíveno 11.05.2017].
URL https://processing.org/overview/

[55] *Processing.org*. Processing Foundation, [Online; navštíveno 11.05.2017].
URL https://processing.org/

[56] Páral, Jaroslav: *EV3RT API Reference*. [Online; navštíveno 11.05.2017].
URL https://roboticsbrno.github.io/RB-ev3rt-hrp2-sdk/

[57] *RoboticsBrno/RB-ev3rt-hrp2-sdk – Travis CI*. Travis CI, GmbH, [Online; navštíveno 11.05.2017].
URL https://travis-ci.org/RoboticsBrno/RB-ev3rt-hrp2-sdk

[58] *Home | Read The Docs*. Read The Docs, [Online; navštíveno 15.05.2017].
URL https://readthedocs.org/

[59] *Dokumentace k EV3RT C++ API!* Jaroslav Páral, [Online; navštíveno 15.05.2017].
URL http://rb3rt.readthedocs.io/cs/latest/

[60] *NXT-G*. robosoutez.cz, [Online; navštíveno 23.01.2017].
URL http://www.robosoutez.cz/index.php?sekce=home&id=nxtg

[61] *ROBOTC a C Programing Language for Robotics*. Robomatter, Inc, [Online; navštíveno 24.01.2017].
URL http://www.robotc.net/

[62] *ROBOTC for LEGO MINDSTORMS 4.x – price*. Robomatter, Inc, [Online; navštíveno 24.01.2017].
URL http://www.robotc.net/purchase/lego/#annual

[63] *Home | Robotika Brno*. Robotika Brno, [Online; navštíveno 11.05.2017].
URL http://robotikabrno.cz/

[64] *RoboticsBrno/RB-ev3rt-hrp2*. GitHub, Inc., [Online; navštíveno 11.05.2017].
URL https://github.com/RoboticsBrno/RB-ev3rt-hrp2

[65] *RoboticsBrno/RB-ev3rt-hrp2-sdk*. GitHub, Inc., [Online; navštíveno 11.05.2017].
URL https://github.com/RoboticsBrno/RB-ev3rt-hrp2-sdk

[66] *RoboticsBrno (RobotikaBrno)*. GitHub, Inc., [Online; navštíveno 11.05.2017].
URL https://github.com/RoboticsBrno

[67] *Robots/EV3 - ROS Wiki*. ROS.org, [Online; navštíveno 26.01.2017].
URL http://wiki.ros.org/Robots/EV3

[68] *Scratch - Imagine, Program, Share*. MIT Media Laboratory, [Online; navštíveno 11.05.2017].
URL https://scratch.mit.edu/about/

[69] *Scratch Statistics - Imagine, Program, Share*. MIT Media Laboratory, [Online; navštíveno 11.05.2017].
URL https://scratch.mit.edu/statistics/

[70] Soldaat, X.: *Comparing the NXT and EV3 bricks*. botbench.com, Leden 2013, [Online; navštíveno 23.01.2017].
URL http://robotsquare.com/2013/07/16/ev3-nxt-compatibility/

[71] *TOPPERS Project / Project Overview*. TOPPERS Project, Inc., [Online; navštíveno 27.01.2017].
URL http://www.toppers.jp/project.html

[72] *Travis CI – Test and Deploy with Confidence*. Travis CI, GmbH, [Online; navštíveno 11.05.2017].
URL https://travis-ci.com/

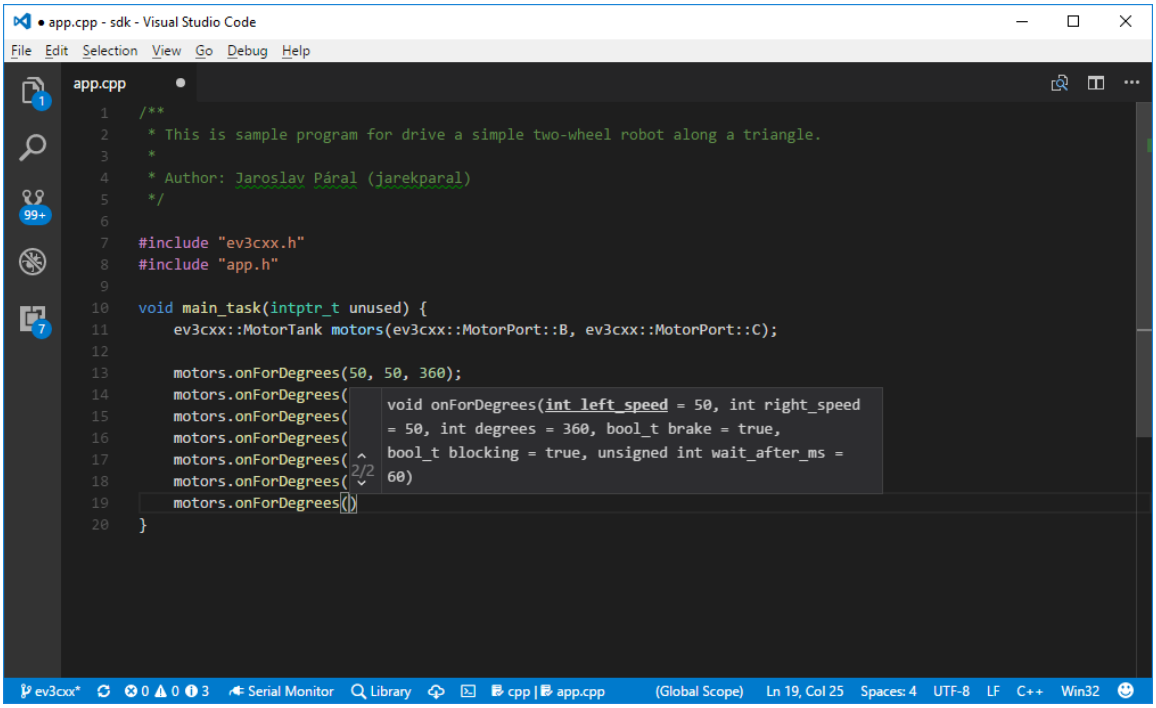
[73] Valk, L.: *EV3 and NXT: Difference and Compatibility*. ROBOTSQUARE, 2013, [Online; navštíveno 23.01.2017].
URL http://robotsquare.com/2013/07/16/ev3-nxt-compatibility/

[74] Voda, Zbyšek: *Programujeme Arduino | Arduino.cz*. ARDUINO.CZ, Říjen 2014, [Online; navštíveno 11.05.2017].
URL https://arduino.cz/programujeme-arduino/

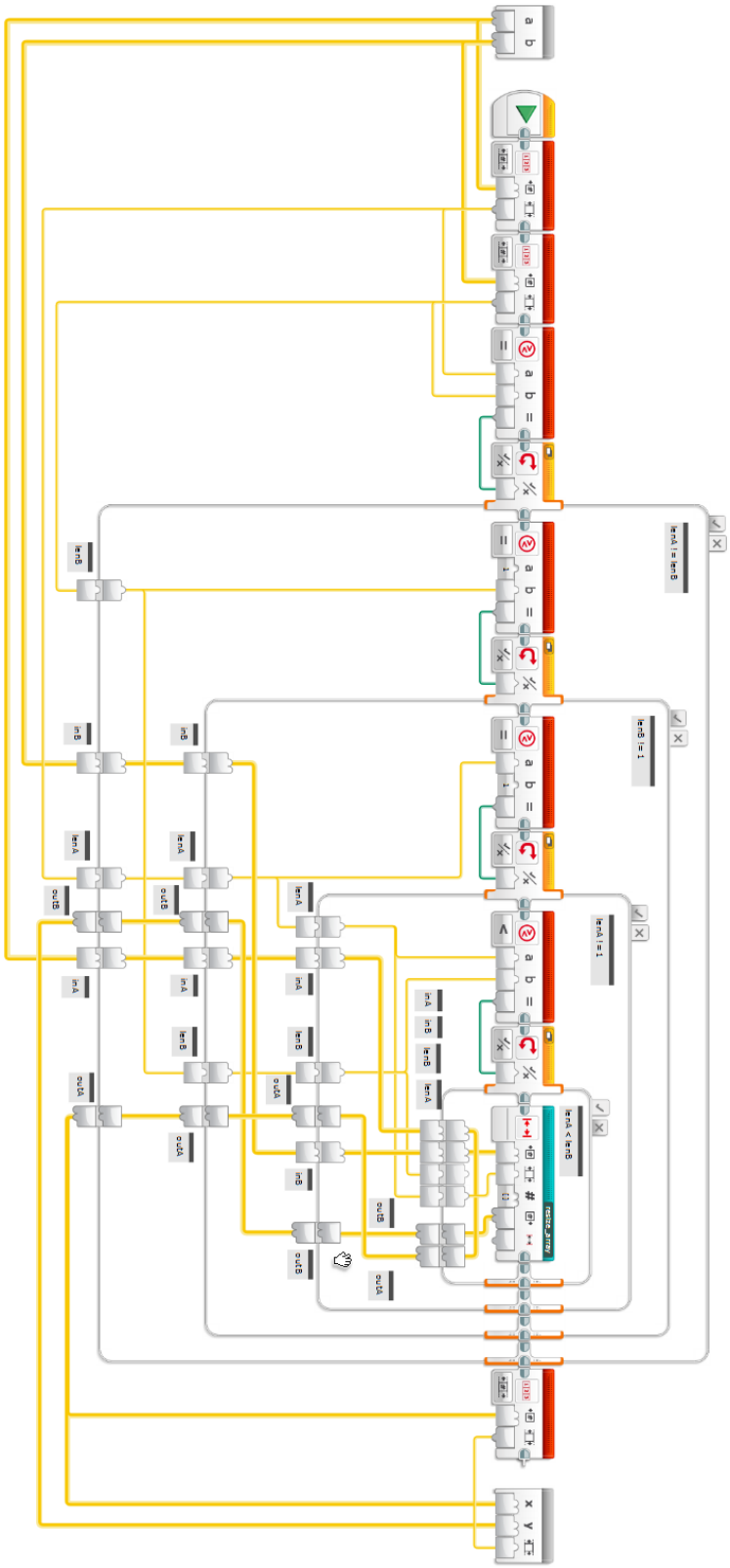
[75] *C/C++ - Visual Studio Marketplace*. Microsoft, [Online; navštíveno 11.05.2017].
URL
https://marketplace.visualstudio.com/items?itemName=ms-vscode.cpptools

Příloha A

Obrázky



Obrázek A.1: Ukázka našeptávání ve Visual Studio Code – zobrazování parametrů funkce



Obrázek A.2: Předávání hodnot v LEGO MINDSTORMS EV3 Software

Příloha B

Ukázka návodu k C++ API

Výběr 1 až 3 kapitoly z EV3RT C++ API dokumentace. Kompletní verze k dispozici na http://rb3rt.readthedocs.io/.

KAPITOLA 1

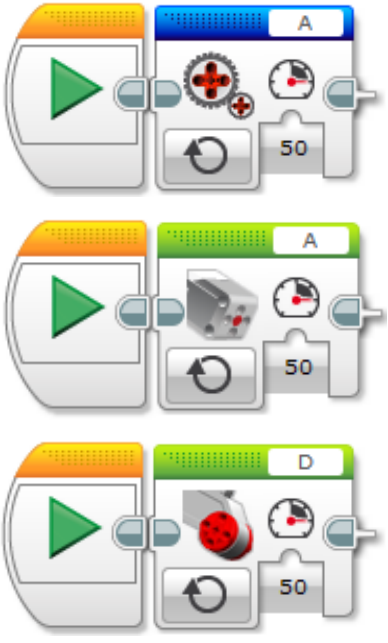
Motory

Motory jsou jednou ze základních komponent robota a proto s nimi začneme. Nejprve je potřeba vytvořit si instanci motorů:

```
ev3cxx::Motor motor(ev3cxx::MotorPort::A, ev3cxx::MotorType::LARGE);
```

Vytvořili jsme objekt `motor`, který je nastaven na port `A` a typ `LARGE`.

K dispozici máme všechny motorové porty na *Bricku* : `A`, `B`, `C` a `D`. U typů máme 3 volby, které odpovídají stejným blokům v originálním LEGO Softwaru: `UNREGULATED`, `MEDIUM` a `LARGE`.



- neregulované motory (`UNREGULATED`): u motorů se nastavuje jen výkon, změny zatížení (jízda do kopce) budou značně ovlivňovat rychlost

- regulované motory střední a velké (`MEDIUM` a `LARGE`): u motorů se nastavuje rychlost a motor se tuto rychlost snaží udržovat, upravuje tak výkon v závislosti na okolním prostředí (nerovnosti, překážky, atd.)

Při inicializaci je potřeba se rozhodnout v jakém režimu budete chtít s motorem pracovat.

Poznámka:

Pokud nebude řečeno jinak:

- při zadání parametru mimo rozsah se automaticky nastavuje maximální/minimální povolená hodnota.
- výchozí hodnoty metod odpovídají standardním hodnotám v LEGO Softwaru.

Příklad: Rozsah povolených hodnot je v rozmezí od -100 do 100. Při zadání hodnoty -101, dojde k ořezání na hodnotu -100. Při zadání hodnoty 101, dojde k ořezání na hodnotu 100.

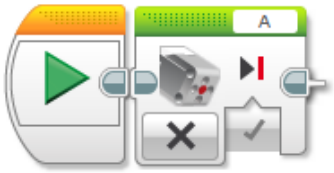
Výkon a rychlost

Poznámka: Parametry při nastavování rychlosti a výkonu.

- `speed`: rychlost motoru při jízdě; rozsah od -100 do 100
- `brake`: brzdění; `true` - motor brzdí, `false` - motor lze volně protáčet

---

off()

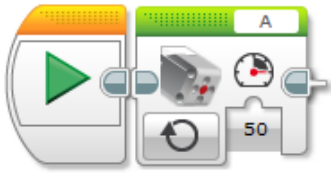



```
void off(bool brake = true)
```

Metoda `off()` zastavuje motor. Nastavuje rychlost nebo výkon (v závislosti na daném režimu) na 0. Jako parametr se předává zda má motor zároveň brzdit (`true`) nebo se volně protáčet (`false`). Ve výchozím stavu brzdí (`true`).

Použití: `motor.off();`

on()




```
void on(int power = 50)
```

Metoda `on()` nastavuje rychlost motoru. Jako parametr se předává požadovaná rychlost v rozsahu -100 až 100. Ve výchozím stavu je hodnota 50.

Použití: `motor.on(50);`

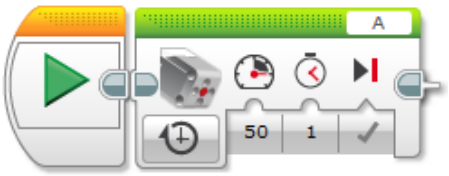
Čas a otáčky

Poznámka: Nové parametry při nastavování otáček.

- `speed`: rychlost motoru při běhu; rozsah od -100 do 100
- `time_ms`: čas v milisekundách, po který se bude motor točit;
- `degrees`: počet stupňů, o které se má motor otočit; lze otáčet i o více než +- 360 stupňů
- `rotations`: počet otáček, které má motor udělat; lze zadávat i desetinná čísla
- `brake`: brzdění po otočení o daný počet stupňů; `true` - motor po dotočení brzdí, `false` - motor lze volně protáčet
- `blocking`: když `true` - metoda blokuje další provádění programu, dokud nedokončí svůj úkol
- `wait_after_ms`: parametr, který nastavuje čekání po ukončení dané akce (jen v případě `blocking = true`); nechte výchozí hodnotu

---

onForSeconds()




```
void onForSeconds(int speed = 50,
                  unsigned int time_ms = 1000,
                  bool brake = true)
```

Metoda `onForSeconds()` nastavuje čas, jak dlouho se má motor točit. Jako parametry se předávají: `speed`, `time_ms`, `brake`.

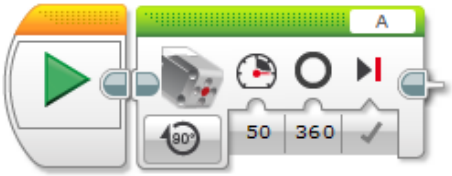
Použití: `motor.onForSeconds(50, 1000);`

Poznámka: LEGO pracuje se sekundami a desetinnými čísly, EV3CXX používá milisekundy a celá čísla

---

Varování: Metoda je vždy blokující. Další příkazy v programu se začnou vykonávat až metoda skončí.

onForDegrees()

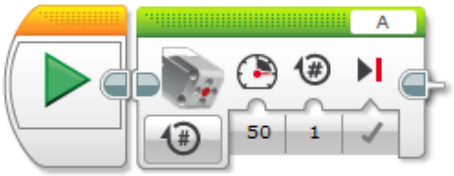



```
void onForDegrees(int speed = 50,
                  int degrees = 360,
                  bool brake = true,
                  bool blocking = true,
                  unsigned int wait_after_ms = 60)
```

Metoda `onForDegrees()` nastavuje počet stupňů, o které se má motor otočit. Jedna otáčka motoru odpovídá 360 stupňům. Jako parametry se předávají: `speed`, `degrees`, `brake`, `blocking`, `wait_after_ms`.

Použití: `motor.onForDegrees(50, 360);`

onForRotations()



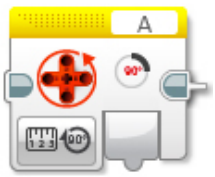

```
void onForRotations(int speed = 50,
                    float rotations = 1,
                    bool brake = true,
                    bool blocking = true,
                    unsigned int wait_after_ms = 60)
```

Metoda `onForRotations()` nastavuje počet otáček, o které se má motor otočit. Jako parametry se předávají: `speed`, `rotations`, `brake`, `blocking`, `wait_after_ms`.

Použití: `motor.onForRotations(50, 1);`

Čtení polohy a rychlosti

degrees()

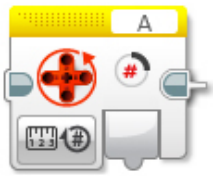



```
int degrees()
```

Metoda `degrees()` vrací polohu motoru ve stupních.

Použití: `motor.degrees();`

rotations()



```
float rotations()
```

Metoda `rotations()` vrací polohu motoru v otáčkách (`float` = desetinné číslo).

Použití: `motor.rotations();`

resetPosition()

```
void resetPosition()
```

Metoda `resetPosition()` vyresetuje pozici motoru (ovlivní metodu `degrees()` a `rotations()`).

Použití: `motor.resetPosition();`

currentPower()

```
int currentPower()
```

Metoda `currentPower()` vrací aktuální rychlost motoru.

Použití: `motor.currentPower();`

Dostupné metody

Po vytvoření objektu `motor` lze na něm volat metody:

- `off()` - vypne motory a začne brzdit
- `on()` - nastaví rychlost na motorech
- `onForSeconds()` - jede po zadanou dobu
- `onForDegrees()` - otočí se o daný počet stupňů
- `onForRotations()` - otočí se o daný počet otáček
- `degrees()` - vrátí aktuální počet stupňů na motoru
- `rotations()` - vrátí aktuální počet otáček na motoru
- `resetPosition()` - vyresetuje pozici motoru (ovlivní metodu `degrees()` a `rotations()`)
- `currentPower()` - vrátí aktuální rychlost motoru
- `getType()` - vrátí aktuálně nastavený typ motoru na daném portu v systému EV3RT

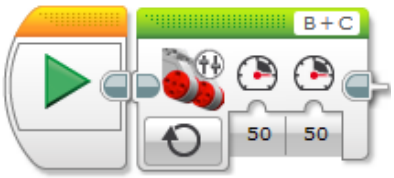
KAPITOLA 2

Tank motory

Pokud chceš řídit robota jako pásové vozidlo, můžeš využít třídu MotorTank

```
ev3cxx::MotorTank motors(ev3cxx::MotorPort::B, ev3cxx::MotorPort::C);
```

Vytvořili jsme si objekt `motors`, kde levý motor je nastaven na port `B` a pravý na port `C`. Tankový mód aktuálně podporuje jen `LARGE` motory. Ty se tedy nastavuji automaticky.



Poznámka:

Pokud nebude řečeno jinak:

- při zadání parametru mimo rozsah se automaticky nastavuje maximální/minimální povolená hodnota.
- výchozí hodnoty metod odpovídají standardním hodnotám v LEGO Softwaru.

Příklad: Rozsah povolených hodnot je v rozmezí od -100 do 100. Při zadání hodnoty -101, dojde k ořezání na hodnotu -100 Při zadání hodnoty 101, dojde k ořezání na hodnotu 100.

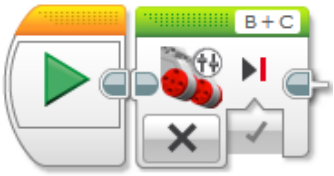
Výkon a rychlost

Poznámka: Parametry při nastavování rychlosti a výkonu.

- `left_speed`: rychlost levého motoru při jízdě; rozsah od -100 do 100
- `right_speed`: rychlost pravého motoru při jízdě; rozsah od -100 do 100

- `brake`: brzdění; `true` - motor brzdí, `false` - motor lze volně protáčet

off()

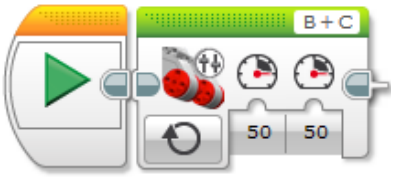



```
void off(bool brake = true)
```

Metoda `off()` zastavuje motory. Nastavuje rychlost na 0. Jako parametr se předává zda mají být motory zabrzděny (`true`) nebo se mají volně protáčet (`false`). Ve výchozím stavu brzdí (`true`).

Použití: `motors.off();`

on()




```
void on(int left_speed = 50, int right_speed = 50)
```

Metoda `on()` nastavuje rychlost motorů. Jako parametry se předávají rychlosti motorů v rozsahu -100 až 100. Ve výchozím stavu jsou `left_speed` a `right_speed` rovny hodnotě 50.

Použití: `motors.on(50, 50);`

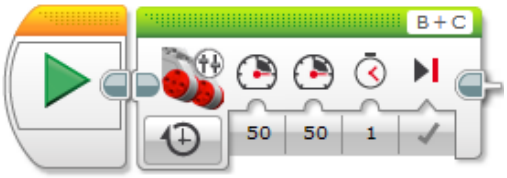
Čas a otáčky

Poznámka: Nové parametry při nastavování otáček.

- `left_speed`: rychlost levého motoru při jízdě; rozsah od -100 do 100
- `right_speed`: rychlost pravého motoru při jízdě; rozsah od -100 do 100
- `time_ms`: čas v milisekundách, po který se budou motory točit;
- `degrees`: počet stupňů, o které se má motor otočit; lze otáčet i o více než +- 360 stupňů
- `rotations`: počet otáček, které má motor udělat; lze zadávat i desetinná čísla
- `brake`: brzdění po otočení o daný počet stupňů; `true` - motor po dotočení brzdí, `false` - motor lze volně protáčet
- `blocking`: když `true` - metoda blokuje další provádění programu, dokud nedokončí svůj úkol

- `wait_after_ms`: parametr, který nastavuje čekání po před zahájením dané akce (jen v případě `blocking = true`); nechte výchozí hodnotu

onForSeconds()




```
void onForSeconds(int left_speed = 50,
                  int right_speed = 50,
                  unsigned int time_ms = 1000,
                  bool brake = true)
```

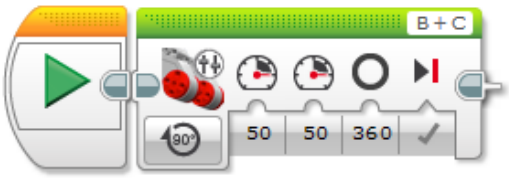
Metoda `onForSeconds()` nastavuje čas, jak dlouho se mají motory točit. Jako parametry se předávají: `left_speed`, `right_speed`, `time_ms`, `brake`.

Použití: `motors.onForSeconds(50, 50, 1000);`

Poznámka: LEGO Software pracuje se sekundami a desetinnými čísly, EV3CXX používá milisekundy a celá čísla

Varování: Metoda je vždy blokující. Další příkazy v programu se začnou vykonávat až metoda skončí.

onForDegrees()

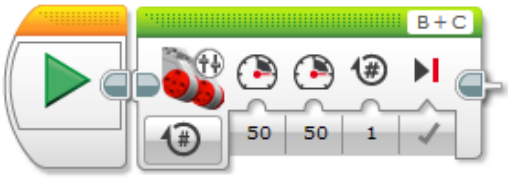



```
void onForDegrees(int left_speed = 50,
                  int right_speed = 50,
                  int degrees = 360,
                  bool brake = true,
                  bool blocking = true,
                  unsigned int wait_after_ms = 60)
```

Metoda `onForDegrees()` nastavuje počet stupňů, o které se má rychlejší motor otočit. Jedna otáčka motoru odpovídá 360 stupňům. Jako parametry se předávají: `left_speed`, `right_speed`, `degrees`, `brake`, `blocking`, `wait_after_ms`.

Použití: `motors.onForDegrees(50, 50, 360);`

onForRotations()




```
void onForRotations(int left_speed = 50,
                    int right_speed = 50
                    float rotations = 1,
                    bool brake = true,
                    bool blocking = true,
                    unsigned int wait_after_ms = 60)
```

Metoda `onForRotations()` nastavuje počet otáček, o které se má rychlejší motor otočit. Jako parametry se předávají: `left_speed`, `right_speed`, `rotations`, `brake`, `blocking`, `wait_after_ms`.

Použití: `motors.onForDegrees(50, 50, 1);`

leftMotor() a rightMotor()

```
Motor& rightMotor();
```

Přes tyto metody, lze ovládat jen jeden motor z páru. Nemusíte si tedy vytvářet nový objekt, pokud budete chtít v určitých situacích ovládat jen jeden motor. Metoda `leftMotor()` vrací instanci motoru, který byl při vytvoření objektu předán jako první, `rightMotor()` vrací druhý motor v pořadí.

Metody vrací instanci daného motoru a následně nad ní lze volat všechny metody dostupné ve třídě `Motor`.

Použití: `motors.rightMotor().onForDegrees(50, 1);`

Dostupné metody

Po vytvoření objektu `motor` lze na něm volat metody:

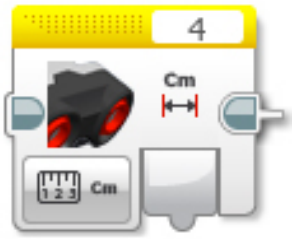
- `off()` - vypne motory a začne brzdit
- `on()` - nastaví rychlost na motorech
- `onForSeconds()` - jede po zadanou dobu
- `onForDegrees()` - otočí se o daný počet stupňů
- `onForRotations()` - otočí se o daný počet otáček
- `leftMotor()` - vrátí instanci levého motoru
- `rightMotor()` - vrátí instanci pravého motoru

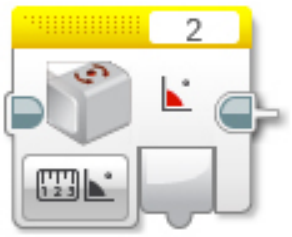
KAPITOLA 3

Senzory

Když už umíme ovládat motory, měli bychom se naučit pracovat se senzory. Pomocí senzorů můžeme získávat informace z okolí a reagovat na ně. Lze tak třeba řídit rychlost motorů, podle pozice robota na čáře nebo zastavit robota před překážkou. V EV3CXX jsou k dispozici všechny základní senzory z LEGO MINDSTORMS EV3.

- `TouchSensor` - dotykový senzor (detekce nárazu, překážky, STOP tlačítko)
- `ColorSensor` - barevný senzor (jízda po čáře, třídění dle barvy)
- `UltrasonicSensor` - ultrazvukový senzor (měření vzdálenosti od překážky nebo mantinelu)
- `GyroSensor` - gyro senzor (určení o kolik stupňů se robot otočil - jízda rovně)





Inicializace

Všechny senzory se inicializují:

```
//ev3cxx::nazev_tridy_senzoru nazev_objektu(ev3cxx::SensorPort::cislo_portu);
ev3cxx::TouchSensor touchS(ev3cxx::SensorPort::1);
```

Vytvořili jsme tedy objekt `touchS`, která je nastavena na port číslo `1`.

Na *Bricku* můžeme využít všechny porty pro senzory: `1`, `2`, `3` a `4`.

TouchSensor

Metody dostupné ve třídě `TouchSensor`:

- `isPressed()` - vrací `true` pokud je senzor zmáčklý
- `waitForPress()` - čekání, dokud se senzor nezmáčkne
- `waitForRelease()` - čekání, dokud se senzor neuvolní
- `waitForClick()` - čekání na zmáčknutí a uvolnění senzoru

isPressed()


```
int isPressed();
```

Vrací `true` v případě, že je dotykový senzor zmáčklý, jinak `false`.

waitForPress()


```
void waitForPress();
```

Program je pozastaven, dokud nebude dotykový senzor zmáčknut.

waitForRelease()


```
void waitForRelease();
```

Program je pozastaven, dokud nebude dotykový senzor uvolněn.

> **Varování:** Nezapomínejte, že v běžném stavu může být dotykový senzor uvolněn. Volání této metody program pozastaví pouze pokud je v daný okamžik dotykový senzor zmáčknutý.

waitForClick()


```
void waitForClick();
```

Program je pozastaven, dokud neproběhne zmáčknutí a uvolnění dotykového senzoru.

ColorSensor

Barevný senzor může pracovat v několika režimech:

- `reflected()` - vrací naměřenou intenzitu odrazu
- `reflectedRawRgb()` - vrací naměřenou intenzitu odrazu pro jednotlivé barevné složky (RGB - červená, zelená, modrá)

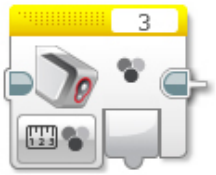
ambient()

```
int ambient();
```

Vrací naměřenou intenzitu světla dopadajícího na senzor, ale **bez přisvětlení vlastními světly**. Vhodné např. pro kalibraci senzoru pro různá osvětlení. Více informací v poznámce u metody `reflected()`.

Rozsah výstupních hodnot je od 0 do 100.

color()



```
colorid_t color();
```

Vrací rozpoznanou barvu povrchu z výčtového typu `enum colorid_t`.

Hodnoty v typu `colorid_t`:

- `COLOR_NONE` - barva nerozpoznána
- `COLOR_BLACK` - černá barva
- `COLOR_BLUE` - modrá barva
- `COLOR_GREEN` - zelená barva
- `COLOR_YELLOW` - žlutá barva
- `COLOR_RED` - červená barva
- `COLOR_WHITE` - bílá barva
- `COLOR_BROWN` - hnědá barva

Příklad:

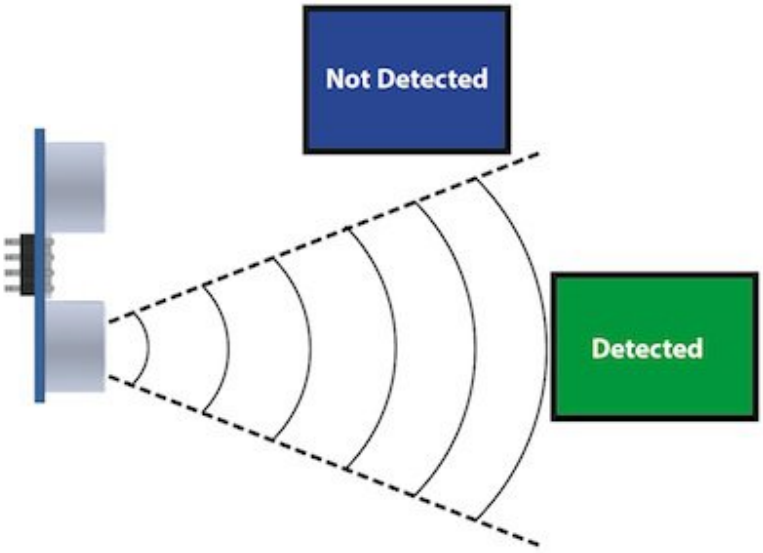
```
colorid_t color_value;
color_value = colorS.color();

if (color_value == COLOR_BLACK)
{
     // sensor on black color
}
```

UltrasonicSensor

Ultrazvukový senzor je primárně určen na měření vzdálenosti. Můžeme jej využít pro detekci překážky, určení vzdálenosti od mantinelu nebo i pro korekci jízdy.

Poznámka: Šíření ultrazvukových vln v prostoru



Obr. 3.1: Ultrazvukové vlny se od vysílače šíří v kuželu. To znamená, že s rostoucí vzdáleností od senzoru pokrývají větší plochu. Zároveň s tím ale klesá rozlišovací schopnost, proto senzor při větších vzdálenostech nedokáže zachytit předměty, které na blízko zachytí. To je podstatný rozdíl v porovnání s infra senzorem, jehož paprsky se šíří prakticky přímo (s mnohem menším rozptylem do stran).

Zdroj obrázku: http://arcbotics.com/products/sparki/parts/ultrasonic-range-finder/

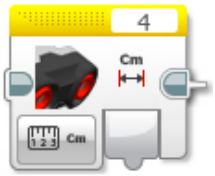
Ultrazvuk v EV3CXX poskytuje tyto metody:

- `centimeters()` - vrací naměřenou vzdálenost v centimetrech
- `millimeters()` - vrací naměřenou vzdálenost v milimetrech
- `inches()` - vrací naměřenou vzdálenost v palcích
- `inchesLine()` - vrací naměřenou vzdálenost v line (1/12 palce)
- `listen()` - vrací zda přijímá signál z jiného ultrazvukového vysílače

Varování: Ultrazvuk v EV3 umí měřit v rozsahu od 3 do 255 centimetrů. Pokud se budete pohybovat na hranici 3 centimetrů, může se stát, že ultrazvuk nedokáže danou vzdálenost změřit a místo hodnoty blízké 3 cm vrátí hodnotu rovnou maximální vzdálenosti => 255 cm.

Pamatujte na tuto vlastnosti při návrhu a programování vašich robotů. Nejbezpečnějším řešením je umístit ultrazvuk tak, aby samotná konstrukce nedovolila menší vzdálenost než 4 a více centimetrů.

centimeters()




```
int centimeters();
```

Vrací naměřenou vzdálenost v centimetrech.

Rozsah měření je od 3 do 255.

Varování: Na rozdíl od LEGO Softwaru, v EV3CXX tato metoda pracuje v celých číslech. Pokud chcete vyšší přesnost použijte metodu `millimeters()`.

millimeters()

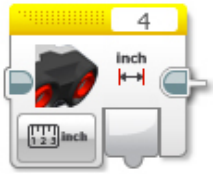
```
int millimeters();
```

Vrací naměřenou vzdálenost v milimetrech.

Rozsah měření je od 30 do 2550.

Poznámka: Tato metoda nemá odpovídající blok v LEGO Softwaru. Jelikož ultrazvuk v EV3 má rozlišení na milimetry a v LEGO Softwaru to řeší pomocí desetinných čísel, je v EV3CXX implementována tato metoda.

inches()




```
int inches();
```

Vrací naměřenou vzdálenost v palcích (1 palec = 2,54 cm).

Rozsah měření je od 1 do 100.

Varování: Na rozdíl od LEGO Softwaru, v EV3CXX tato metoda pracuje v celých číslech. Pokud chcete vyšší přesnost použijte metodu `inchesLine()`.

inchesLine()

```
int inchesLine();
```

Vrací naměřenou vzdálenost v linech (1 line = 1/12 palce).

Rozsah měření je od 10 do 1200.

Poznámka: Tato metoda nemá odpovídající blok v LEGO Softwaru. Jelikož ultrazvuk v EV3 má rozlišení na milimetry a v LEGO Softwaru to řeší pomocí desetinných čísel, je v EV3CXX implementována tato metoda.

---

listen()


```
int listen();
```

Senzor poslouchá a pokud zachytí ultrazvukový signál, od jiného vysílače, vrací `true`, jinak `false`.

GyroSensor

Gyroskop umožňuje změřit o kolik stupňů se robot otočil nebo jak rychle se otáčí. EV3 obsahuje jednoosý gyroskop a tak si při stavbě musíte vybrat v jaké rovině chcete měřit.

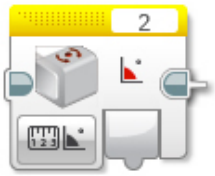
Gyroskop v EV3CXX poskytuje tyto metody:

- `angle()` - vrací aktuální úhel natočení ve stupních
- `rate()` - vrací aktuální rychlost otáčení ve stupních za vteřinu
- `reset()` - nastavuje počáteční polohu gyroskopu
- `resetHard()` - provádí úplný restart senzoru

Varování: Gyroskop v EV3 se občas zasekne a začne měnit aktuální úhel (ujíždět), i když se robot nehýbe. Většinou nepomůže standardní `reset()` a proto je v těchto případech potřeba provést `resetHard()`.

Při každém vytváření objektu ze třídy GyroSensor se provádí `resetHard()`. V tento moment se nesmí gyroskop pohybovat, jinak bude měřit špatně.

angle()



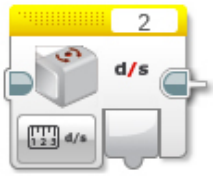
```
int angle();
```

Vrací aktuální úhel natočení vůči počáteční pozici (odchylku od počáteční pozice). Počáteční pozice se nastavuje při vytváření objektu nebo pomocí metody `reset()`.

Rozsah měření je od -32768 do 32767. Při překročení maximální nebo minimální hodnoty (např. 32767 + 1) začne gyroskop vracet hodnotu z druhého konce rozsahu (=> -32768).

Varování: Při použití metody `rate()` dochází k restartu počáteční polohy u metody `angle()`. Ta pak ukazuje opět od nuly.

rate()



```
int rate();
```

Vrací aktuální rychlost změny polohy ve stupních za sekundu.

reset()

```
void reset();
```

Nastavuje počáteční polohu gyroskopu pro metodu `angle()` a také kalibruje senzor. Při volání metody `reset()` by se Gyro senzor neměl vůbec hýbat. Jinak bude špatně měřit. Dejte pozor na vibrace a dojezdy setrvačností.

Metoda může v některých případech odstranit *ujíždění* aktuálního úhlu gyroskopu pro metodu `angle()`, ale ne vždy funguje.

resetHard()

```
void resetHard();
```

Provádí úplný restart senzoru. Měl by odstranit problém s ujížděním, kdy ačkoliv se Gyro senzor vůbec nehýbe, jeho poloha má konstantní přírůstek. V tento moment gyroskop nelze využívat a je potřeba jej restartovat.

Poznámka: Tato metoda nemá odpovídající blok v LEGO Softwaru.

Varování: Počítejte s tím, že `resetHard()` může trvat i několik sekund a po tuto dobu bude zastaven běh programu. Je tedy potřeba provádět úplný reset jen v nutných případech a na místech v programu, kde tato prodleva nebude vadit.

Během restartu se nesmi gyroskop pohybovat, jinak nebude měřit správně.

Poznámka: Implementace úplného restartu je v celku jednoduchá. Aby došlo k restartu, je potřeba přepnout gyroskop mezi režimy v jakých pracuje. První režim je měří úhel natočení (`angle()`) a druhý režim měří rychlost otáčení (`rate()`). Při přepínání mezi těmito režimy dochází k úplnému restartu gyroskopu. Pro stoprocentní funkčnost se v metodě `resetHard()` provádí vícenásobné přepínání (`angle()` => `reset()` => `rate()` => `reset()` => `angle()`).

Zdroj 1: https://bricks.stackexchange.com/questions/7115/how-can-ev3-gyro-sensor-drift-be-handled

Zdroj 2: https://www.us.lego.com/en-us/mindstorms/community/robot?projectid=96894a3a-45db-48f9-9544-abf66f481b32
