Středoškolská technika 2015

Setkání a prezentace prací středoškolských studentů na ČVUT

AUTONOMNÍ BAGR

Radek Singer

Střední průmyslová škola elektrotechnická
V Úžlabině 320, Praha 10

Anotace

Cílem této práce je představit, porovnat stavebnice lega, naprogramovat a sestavit autonomního bagru, který je vytvořen ze stavebnice EV3 a vlastnoručně vyrobených dílů. V bagru je nahrán program, jenž obsahuje proporcionálně derivační regulátor, který řídí sledování černé čáry a při načtení 95-100 % odrazu pozná, že má před sebou materiál k naložení, nebo že má materiál z radlice vysypat. To je zajištěno pomocí funkcí, které obsahují instrukce pro motory a senzor barvy.

Annotation

The goal of my work is introduce, compare construction sets of Lego, programme and assemble autonomous digger which is built from Lego EV3 and personally made parts. In the digger there is a program which contains proportional derivative regulator that controls tracing a black line and when reading data 95-100 % data, it is able to recognize a digger material to pick up or unload from blade. This is provided functions which comprise instructions for motors and light sensors.

Obsah

Úvod

Práci jsem se rozhodl udělat, abych dokázal, že i dnešní hračky pro děti zvládnou lehkou automatizaci. Má práce by se dala využít ve větších dolech, kde je potřeba převážet sypký materiál z bodu A do bodu B bez jakéhokoliv lidského faktoru. Tato práce se dá použít také při výuce programování. Vybral jsem si tuto práci, protože mně zajímají stavebnice Lego a programovací prostředí pro tuto stavebnici. Lego se většinou využívá jako učební pomůcka, i já ho využívám na svém kroužku pro děti v Domu dětí a mládeže Praha 9.

1 Porovnání stavebnic

1.1 NXT

Díky dlouhodobému výskytu na trhu je pro tuto verzi robota k dispozici velké množství programovacích jazyků a vývojových prostředí. Použití některých jazyků ale vyžaduje přehrání systému robota. Toto se týká hlavně interpretovaných jazyků. C++ je výjimka. V neoficiálních jazycích není moc snadné program zkompilovat, protože to vyžaduje vyšší interakci robota a počítače. V případě robota musí mít daný jazyk zabudovanou podporu pro ladění v runtime.

Obr. 1 Robot NXT

1.1.1 NXT Programovací jazyky

Obr. 2 Inteligentní kostka NXT

1.1.1.1 NXT-G 2.0

Jedná se o grafický jazyk. V praxi se jedná o slabě staticky typovaný jazyk s automatickým managementem paměti. Implementuje procedurální paradigma patřící do rodiny imperativních paradigmat. To znamená, že lze tvořit funkce. Jádrem jazyka jsou za sebou spojené kostky, které realizují funkční volání. NXT-G má dobrou a intuitivní podporu pro tvorbu paralelních programů. NXT-G s sebou dále nese kompletní IDE s podporou pro sledování běžícího programu. To lze využít k omezenému kompilovaní programu. IDE

obsahuje velké množství návodů na roboty. Dokumentace jazyka a API robota je pro tento jazyk velmi dobrá. Komunita je taktéž velká, takže není problém získat pomoc.

1.1.1.2 C#

C# je klasický textový a interpretovaný jazyk. Silně staticky typovaný s podporou OOP paradigmatu. Podporu zajišťuje Microsoft => k dispozici IDE i dokumentace. Komunita je taktéž dostatečně velká, takže není problém se sehnáním pomoci.

1.1.1.3 NXC

Textový jazyk se slabým statickým typováním. Jedná se o upravený jazyk C realizující procedurální paradigma. NXC má podporu pro paralelismus a automatický paměťový management. Přímý přístup do paměti přes pointery byl ale odstraněn z důvodu odstranění zdroje potencionálních problémů pro nezkušené vývojáře. Dokumentace je k tomuto jazyku velmi dobrá, stejně tak komunita je velmi velká.

1.1.1.4 NXJ

NXJ je modifikovaná Java a minimalistický VM upravený pro použití v robotu. Java je staticky typovaný jazyk s automatickým paměťovým managementem. Ve všech případech se jedná o interpretovaný jazyk, takže je nutné přehrát firmware robota z důvodu nutnosti instalace interpretu. Realizuje OOP paradigma. NXJ má vynikající dokumentaci a komunitu, takže se jedná o dobrou volbu.

1.1.1.5 NXT-Python

Python je interpretovaný, dynamicky typovaný jazyk s automatickou správou paměti. Realizuje OOP paradigma. Protože vyžaduje interpret, je nutno ho do robota nahrát anebo se vydat cestou této implementace. NXT-Python překládá kód Pythonu do NXC, které je již možno nahrát do robota. Dokumentace není ani třeba, protože NXT-Python sdílí API s NXC. Komunita je bohužel velmi malá a závislost na kompatibilitě NXC je ~~bohužel~~ slabé místo této technologie.

1.1.1.6 LabVIEW

Jedná se o IDE s grafickým programovacím jazykem. Jedná se o slabě staticky typovaný jazyk s automatickým managementem paměti a dobrou dokumentací. K dispozici je vynikající platforma pro kompilování díky silným vizualizačním dovednostem LabVIEW a také díky tomu, že robot může být programem řízen „nepřímo“ přes připojení. Kompilace a nahrání programu do robota jsou taktéž k dispozici. LabVIEW realizuje procedurální paradigma.

1.1.2 Hardware NXT

Display	100x64 pixelů jednobarevný LCD
Procesor	Atmel AT91SAM7S256 (ARM7TDMI core) @48 MHz
Paměť	64KB=RAM 256KB=Flash
Port USB	NE
Wi-Fi	NE
Bluetooth	ANO

Obr. 3 – NXT Senzory

1.2 EV3

Obr. 4 Inteligentní kostka EV3

Obr. 5 Robot EV3

1.2.1 Programovací jazyky

Z důvodu krátkého času na trhu není pro tento model robota moc kompatibilních jazyků. Stejná situace je i na poli vývojových prostředí. Výše popsané problémy s některými jazyky jsou stejné jak pro NXT, tak i pro EV3. Jazyky, které nejsou uvedeny pod tímto modelem, ale najdeme je pod NXT, jsou buď nepodporované či velmi špatně zdokumentované, nebo jsem nedokázal potvrdit kompatibilitu s EV3.

1.2.1.1 NXT-G 3.0

Třetí verze NXT-G se liší především graficky, nikoli po funkční stránce. Samozřejmě API se rozšířilo o nové dovednosti aktuální verze robota. Grafické změny vedly především ke zpřehlednění rozhraní. Všechny parametry kostek se nastavují na nich samých, nikoli na dvou místech jako ve druhé verzi jazyka (v panelu dole a na kostkách). Bohužel s novým grafickým kabátem vznikly problémy při přetahování kostek, takže tvoření s novou verzí je zpomaleno kvůli špatně se napojujícím kostkám.

1.2.1.2 C#

Díky oficiální podpoře od Microsoftu je k dispozici jazyk C# i s kompletním IDE. Kromě rozšířeného API o další dovednosti robota EV3 se nekoná žádná další změna.

1.2.1.3 NXJ

Pouze rozšířené API a dále nebylo přineseno nic nového.

1.2.1.4 LabVIEW

Kromě rozšířeného API to samé, co pro NXT.

1.2.2 Hardware EV3

Display	178x128 pixelů jednobarevný LCD
Procesor	TI Sitara AM1808 (ARM926EJ-S core) @300 MHz
Paměť	64 MB RAM 16 MB Flash microSDHC Slot
Port USB	ANO
Wi-Fi	ANO- nutno přikoupit
Bluetooth	ANO

Obr. 6 - EV3

1.3 Tetrix

Je rozšiřující sada k NXT. Rozšiřující sada je to z důvodu, že zde potřebujeme NXT inteligentní kostku, do níž připojujeme senzory a servomotory přes patice. Do patic pasují konektory NXT a z patic vedeme samostatné kabely pro jednotlivé motory, popř. senzory. Proto Tetrix můžeme programovat pouze v LabVIEW.

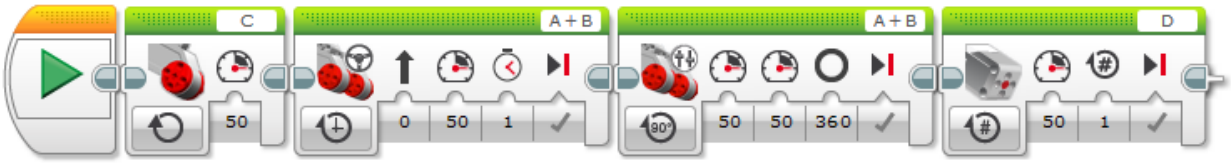
Hlavní výhodu nacházíme v kovových součástkách, přirovnal bych jej k období neelektrického Merkuru. To nám nabízí sestrojení mnohem přesnějšího a robustnějšího robota než ze stavebnic EV3 a NXT.

Tetrix se doporučuje pro vysoké školy z důvodu zapojování do patic a ovládání pouze přes LabVIEW. Cena je bezmála 15 000.

1.4 Konkrétní porovnání programovacích prostředí

Pro porovnání u každého prostředí vytvoříme program pro pohyby s motory a výpis vzdálenosti na display.

1.4.1 NXT-G 3.0



Obr. 7 - NXT-G 3.0 Motory

Pro programování motorů máme hned několik „kostek“, které mají různé funkce.

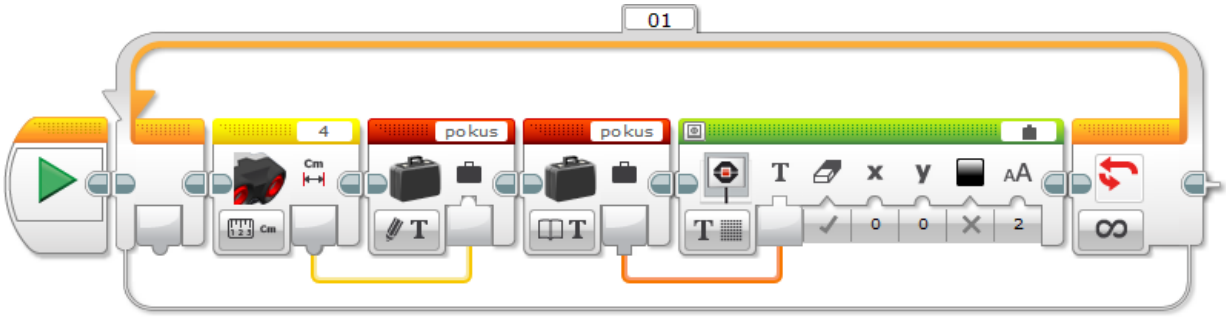
Obecně se používá pro všechny bloky, které řídí akční členy (motory). V horní části při poklepání nastavíme port, kde máme řízený motor. Při poklepnutí na zacyklené kolečko se nám objeví nabídka, zda motor: **vypnout** – **zapnout** a nastavit pouze rychlost – **zapnout** na určitý čas— **zapnout** a nastavit stupně - **zapnout** a nastavit, kolikrát se má motor otočit.

První „kostka“ je pro jeden motor, který můžeme řídit. V této nabídce je nastavená první možnost, motor tedy neustále běží a my nastavujeme pouze rychlost. Pro vypnutí motoru bychom museli zařadit další kostku, kde nastavíme vypnout.

Druhá „kostka“ nám umožňuje regulovat dva motory. Výrobci ji označují volantem, tudíž řízení nastavujeme pod šipečkou číslem. To nám určí, o kolik procent bude robot zatáčet. Poté nastavujeme rychlost a čas.

Třetí v pořadí reguluje dva motory a označení je „tankové řízení“, a to z důvodu, že zde nastavujeme pro každý motor jednotlivou rychlost. Pak nastavujeme stupně.

Poslední v řadě je řízení středních motorů. Lze řídit pouze jeden. Zde jsem zvolil nastavení rychlosti a počtu otáček.

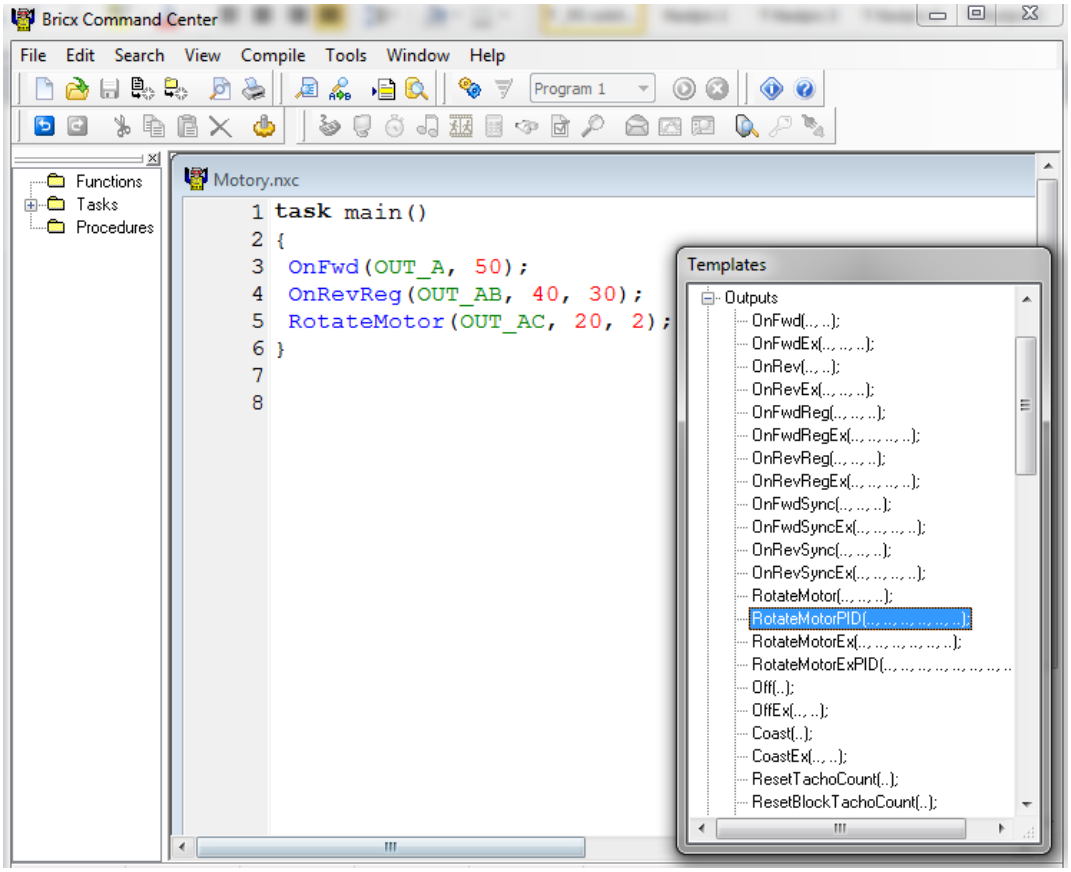


Obr. 8 - NXT-G 3.0 Výpis vzdálenosti na display

Prvním krokem pro vytvoření výpisu vzdálenosti je vložení nekonečného cyklu. Do nekonečného cyklu vložíme senzor, kde nastavíme výstupní veličinu - centimetry, tlačítko leží vlevo dole. Nesmíme zapomenout nastavit port v horní části. Další operací je nahrání aktuální hodnoty do proměnné pokus, která nám ji převede do stringové podoby. Pro správné nahrání nastavíme vlevo dole zápis string a nahoře do prádného pole při poklepání nazveme svou proměnnou. Poté znovu zavoláme proměnnou, nastavíme čtení string a přetáhneme na display. Na display je třeba nastavit, zobrazovaní textu, to se provede tlačítkem vlevo dole, nahoře nastavíme podivný obrazec spojování (umožní stringový vstup z programu).

1.4.2 Brixc Command Center

Pro programování EV3 je potřeba nainstalovat doplňující program CSLITE. Ten doplní API a drivery pro EV3.

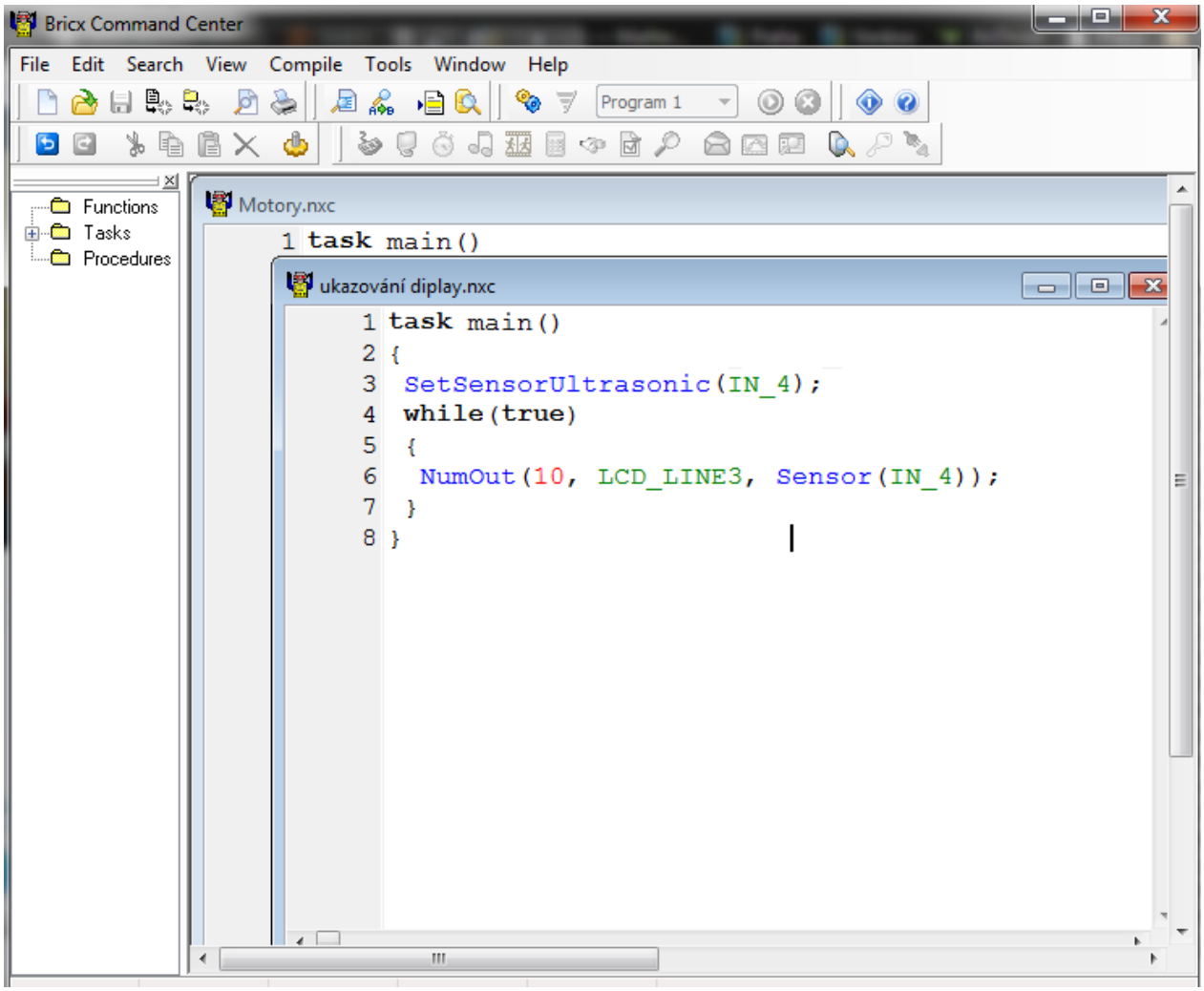


Obr. 9 - Bricx Command Center Motory

První příkaz zapne motor napořád. OUT_A značí, jaký port má pustit, a číslo určuje rychlost, jakou motor bude poháněn.

Druhý příkaz zapne motory na portech A a B, pojede rychlostí 40 a udělá pohyb 30°.

Poslední zapne motory na 20 a ty udělají dvě otáčky.



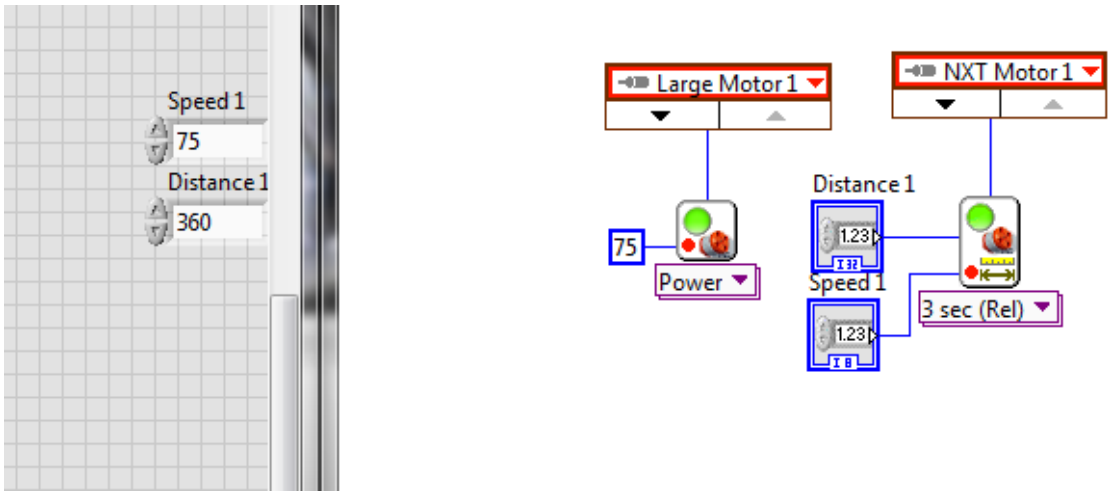
Obr. 10 - Bricx Command Center Výpis vzdálenosti

Pro zobrazení vzdálenosti ze senzoru na display stačí pouze nadefinovat ultrazvukový senzor. Následně vytvoříme nekonečný cyklus pomocí while a zadáme funkci výpisu čísla na display. Prvním číslem nastavíme velikost, následně nastavíme pozici na display a posledním číslem, nastavujeme vypisování čísla.

1.4.3 LabVIEW

Program LabVIEW je rozdělen do dvou oken: „přední“ (obr. vlevo) a „zadní“(obr. vpravo). „Přední“ okno je zadávací pro uživatele a „zadní“ slouží vývojářům pro vytváření samotného algoritmu, popř. pohybů s akčními členy.

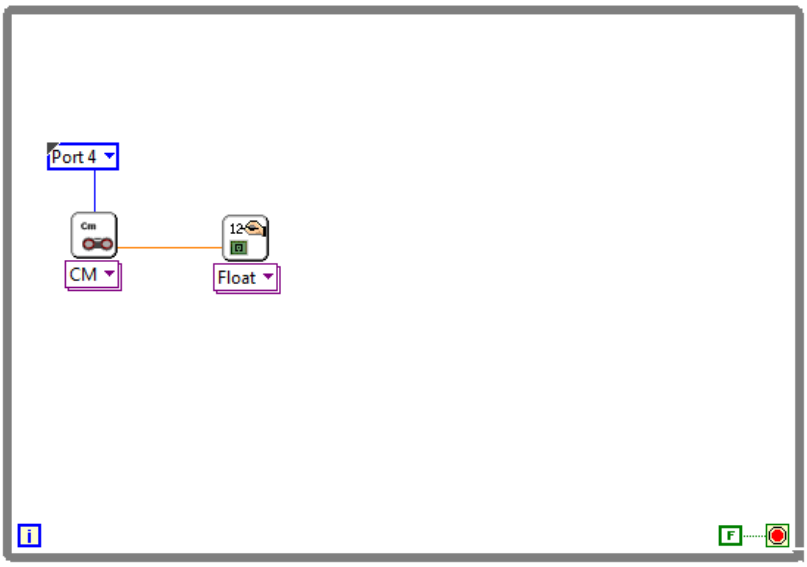
Pro vytvoření programu v LabVIEW je potřeba naistalovat rozšiřující balíček Lego Mindstorms. Poté co máme nainstalováno, musíme vytvořit Minstorms projekt a nadefinovat, na jakém portu máme motory, popř. senzory.



Obr. 11 - LabVIEW Motory

První obrázek motoru je nastaven na konstatní hodnotu rychlosti 75, tuto hodnotu nadefinujeme najetím na ikonu konstanty. Pro nastavení portu, kde se motor nachází, vytvoříme zase najetím na konektor a zvolíme konstantu portů.

Druhý obrázek motoru je řízen nastavenými hodnotami, které zadá uživatel. Jelikož je zadávání určitého otočení pouze ve stupních, je to o něco složitější, nebo si můžeme vytvořit funkci na přepočet a můžeme zadávat například v otočení motoru. Samozřejmě i zde musíme umístit konstantu, kde motor leží.



Obr. 12 - LabVIEW Výpis vzdálenosti na display

Tento program zobrazuje hodnoty vzdálenosti na display. Celý rámeček znamená nekonečný cyklus (while). LabVIEW vyžaduje, abychom vpravo dole nastavili konstantu pravdy/nepravdy. Samozřejmě zde můžeme zavést vyskočení z cyklu. Obrázek s CM je ultrazvukový senzor, v horní části musíme najet na konektor a vytvořit konstantu, která nám určí, kde máme senzor. Následně musíme vytvořit ikonku display, jen přetáhneme z nabídky a on se automaticky přetypuje dle toho, co na něj připojíme. Není třeba dělat změnu datových typů jako u NXT-G 3.0.

2 Sestrojení příslušenství k robotu

Prvním krokem k výrobě příslušenství bylo vytvoření technických výkresů. K tvorbě výkresů jsem použil AutoCAD 2014-2015, v něm jsem vytvořil hrubé výkresy. Z důvodu mé nepřesné výroby jsem následně po výrobě musel některé hodnoty upravit. Tudíž jsem překreslil výkresy se správnými hodnotami.

2.1 Radlice

Viz Příloha 1

Výroba probíhala narýsováním radlice na 0,8mm pozinkovaný plech. Dále jsem na hydraulických nůžkách na plech ostříhal čtverec, poté jsem úhlovou bruskou vyřízl trojúhelníčky. Pokračoval jsem vyvrtáním děr na uchycení radlice. Na ohýbačce plechu jsem naohýbal radlici. V předposledním kroku jsem sletoval horní hrany radlice. Poté jsem radlici nastříkal zinkovým sprejem.

2.2 Pásový dopravník

Viz Příloha 2

Prvním krokem bylo nařezání pásoviny o rozměrech 16x5mm. Nařezal jsem si všechny díly z pásoviny, poté jsem odřízl čtyři kusy kulatiny o průměru 10mm a délce 30mm, dva z nich jsem na soustruhu provrtal skrz, abych sem mohl vložit 5mm kulatinu jako osičky pro pohon dopravníku. Osu s kulatinou jsem bodově svařil, aby se neprotáčela, následně jsem složil pásoviny a kulatiny dle výkresu a svařil vše dohromady pomocí CO_2. Posledním krokem bylo nastříkat celý dopravník zinkovým sprejem a po zaschnutí natáhnout gumu na pás.

2.3 Násypka

Viz Příloha 3

Násypku jsem nakreslil ve dvou dílech na pozinkovaný plech, poté pomocí hydraulických nůžek jsem nařezal hrany, které bylo možné odříznout, aniž bych poškodil potřebný materiál. Zbytek jsem dořezal úhlovou bruskou. Tyto dva díly se ohýbaly na ohýbačce, aby vytvořily tzv. trychtýř - do rohů byly navařeny pomocí CO_2 nařezané kvadrátky, které byly nařezány na pásové pile. Ze spodní strany násypky jsem udělal 4 bodové sváry CO_2, aby se mi lépe letovalo z druhé strany. Poletovaný výrobek jsem celý nastříkal zinkovým sprejem.

2.4 Dok pro zachycení materiálu

Viz Příloha 4

Dok jsem nakreslil na pozinkovaný plech, možné díly ustřihl na hydraulických nůžkách a zbytek dořezal úhlovou bruskou. Následně jsem dok naohýbal na ohýbačce dle výkresu. Pak jsem sletoval překrývající plechy, aby držely bočnice. Kvůli změně materiálu jsem přidělal silikonové zarážky na dok, aby proso nevylétávalo ven.

3 Sestrojení robota

Robota jsem sestrojil ze školní stavebnice EV3. Z důvodu nedostatku součástek jsem použil některé součástky ze sady NXT.

Jako předlohu jsem použil stavební návod Tank bot. Postupoval jsem dle návodu, pokud chyběly díly, což bylo cca v 50 % případů, improvizoval jsem a vylepšil předlohu, a tím i svého reálného robota. Po sestavení rádoby tank bota (do tank bota velmi daleko) jsem vytvořil uchycení pro radlici, dále jsem vytvořil samotnou radlici, kde kvůli střednímu motoru a jeho nepřesnosti jsem byl nucen udělat převodovku. Vytvořil jsem převodovku pro rameno bagru, zde je velká zátěž, tudíž je třeba několik převodů. Kvůli převodům vznikly vůle, které nešly na pevno stanovit (pokud zatížím moc, bude jiná, při jiném zatížení též), proto zde je umístěno tlačítko, které pomáhá při programování nabírání a vysýpání materiálu.

Po odzkoušení jsem robota pouze designově zdokonaloval a optimalizoval.

4 Vytvoření řídicího programu

Řídicí program se skládá ze sledovače čáry a funkcí, které obsahují nakládání materiálu a vykládání materiálu. Sledovač čáry je sestrojen z proporcionálně derivačního regulátoru, kam je vložena podmínka, pokud bude hodnota odrazu mezi 95-100 %, vyvolají se funkce nakládání/vykládání.

Jelikož v každém programovacím prostředí se program vytvoří jinak, tak je třeba, abyste si otevřeli jednotlivé programovací prostředí a shlédli konkrétní programy. Pokud nemáte prostředí, kde byste program otevřeli, můžete se podívat na přílohy 5, 6 a 7.

Závěr

Práci bych označil za zdařilou a velmi úspěšnou. Vlastní díly, které jsem vyráběl, jsou přesné a kvalitní. Robot sleduje čáru a funguje za určitých podmínek, podmínky jsou dány kvalitou senzorů a osvětlením v dané místnosti. Pokud je v místnosti moc anebo naopak málo světla, vznikají jiné odchylky a musí se přenastavit hodnoty v programu. Po přenastavení robot funguje správně. Nabírání a vykládání funguje, bohužel občas se projevuje, že je bagr sestrojen z hračky a motory nedokážou udělat přesné pohyby, což způsobí kolaps. Díky této práci jsem se naučil programovat v mnoha programovacích prostředích a zdokonalil se v obrábění kovů.

Seznam použité literatury a zdrojů

Dostupné z www: http://en.wikipedia.org/wiki/Lego_Mindstorms_EV3

Dostupné z www: http://en.wikipedia.org/wiki/Lego_Mindstorms_NXT

Dostupné z www: http://www.joe.org/joe/2014october/tt9.php

Obrázek 1: www.brighthub.com

Obrázek 2: www.levnicek.cz

Obrázek 3 a 6: www.kostkamodelcentrum.cz

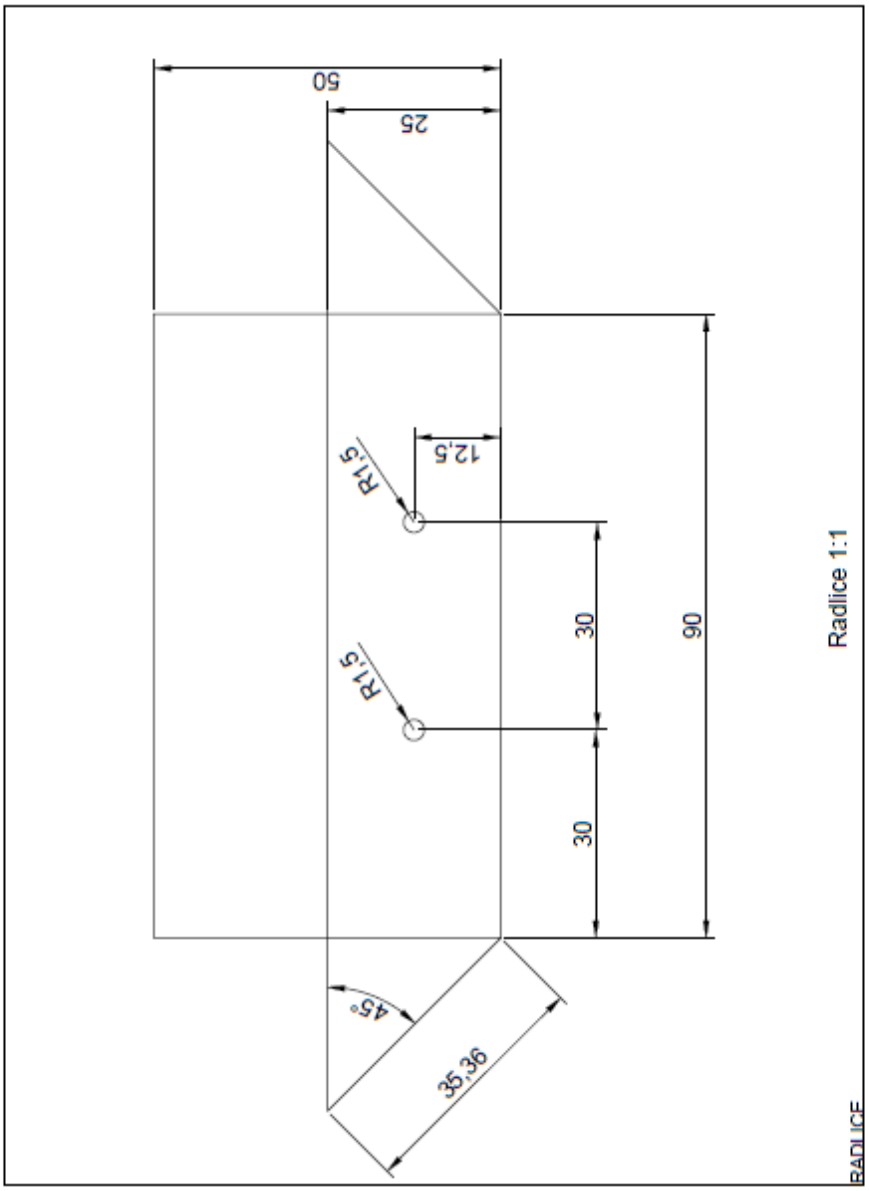
Obrázek 4 a 5: www.robodoupe.cz

Seznam použitého software:

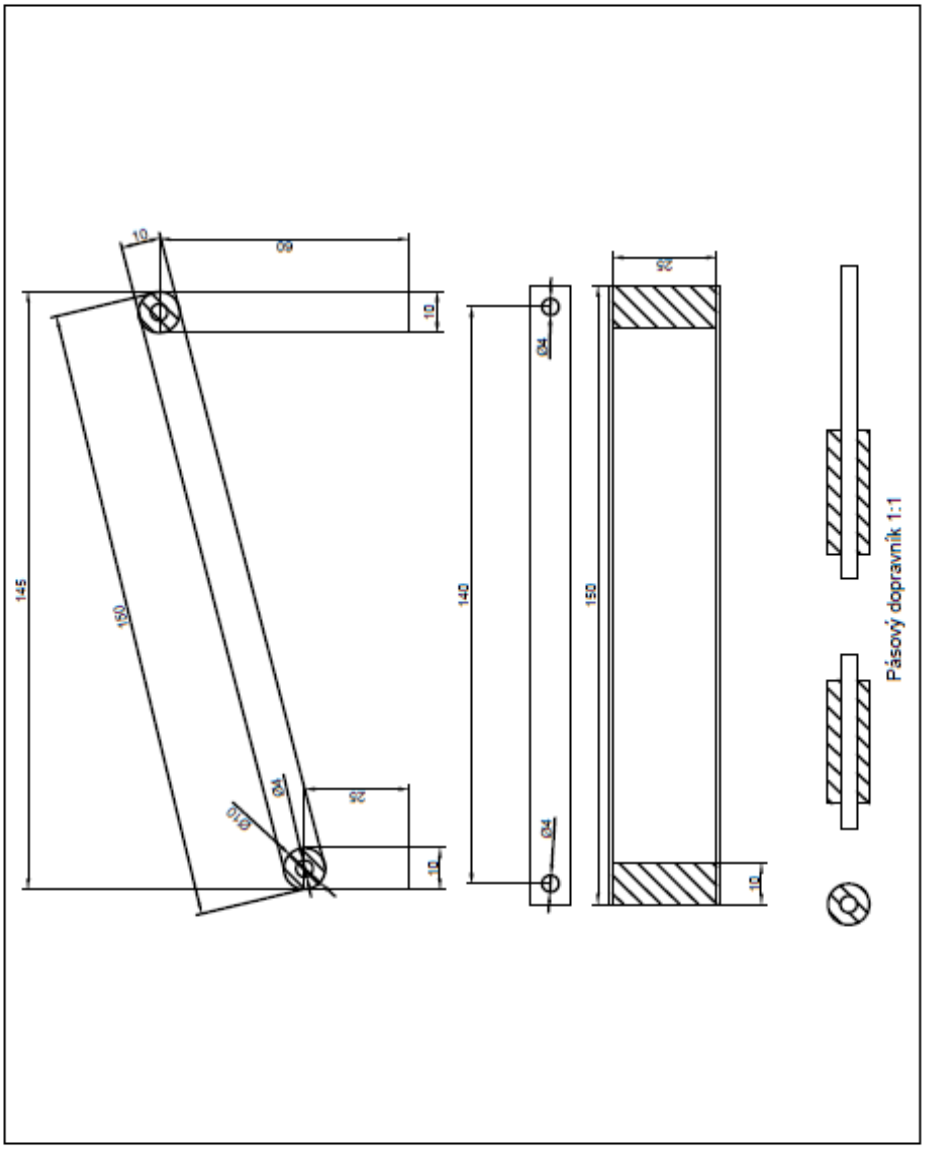
- AutoCAD 2014-2015
- Lego Mindstorms Education EV3
- Bricx Command Center
- LabVIEW 2014

Seznam příloh

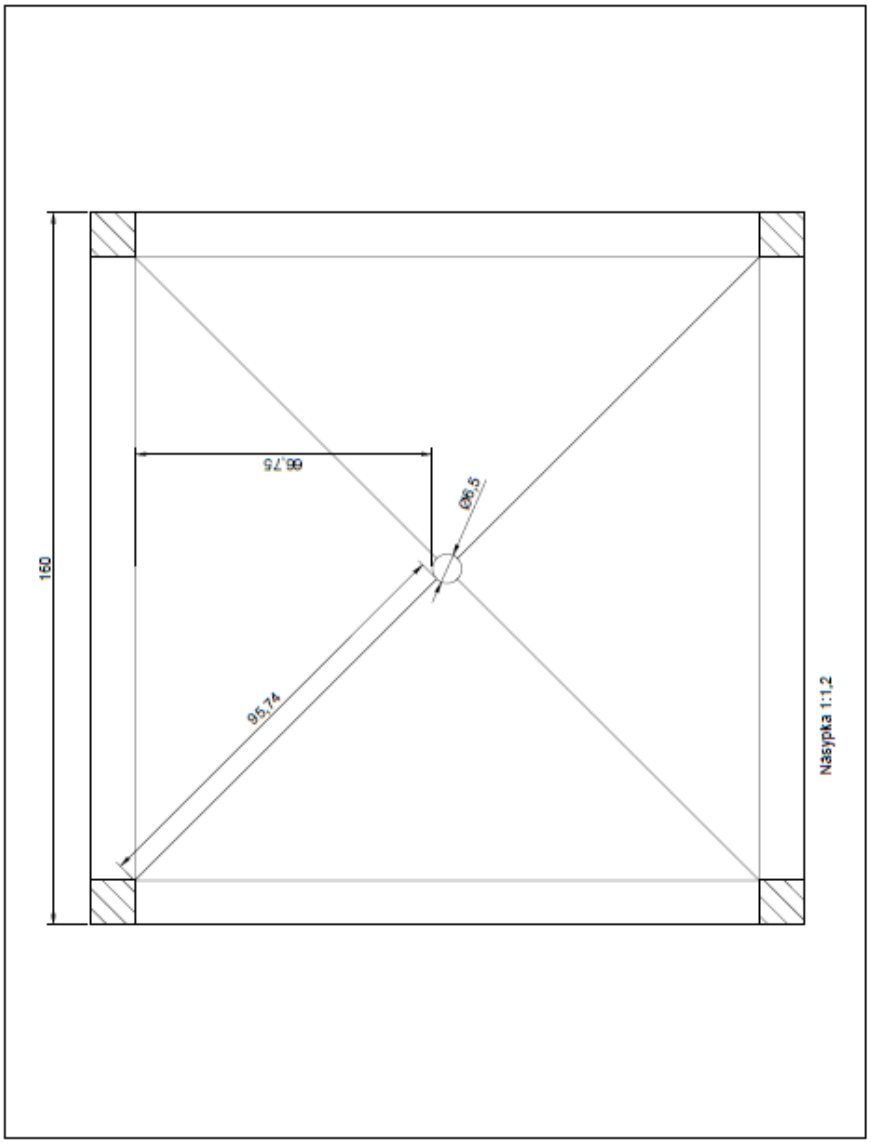
Příloha 1 Technický výkres Radlice

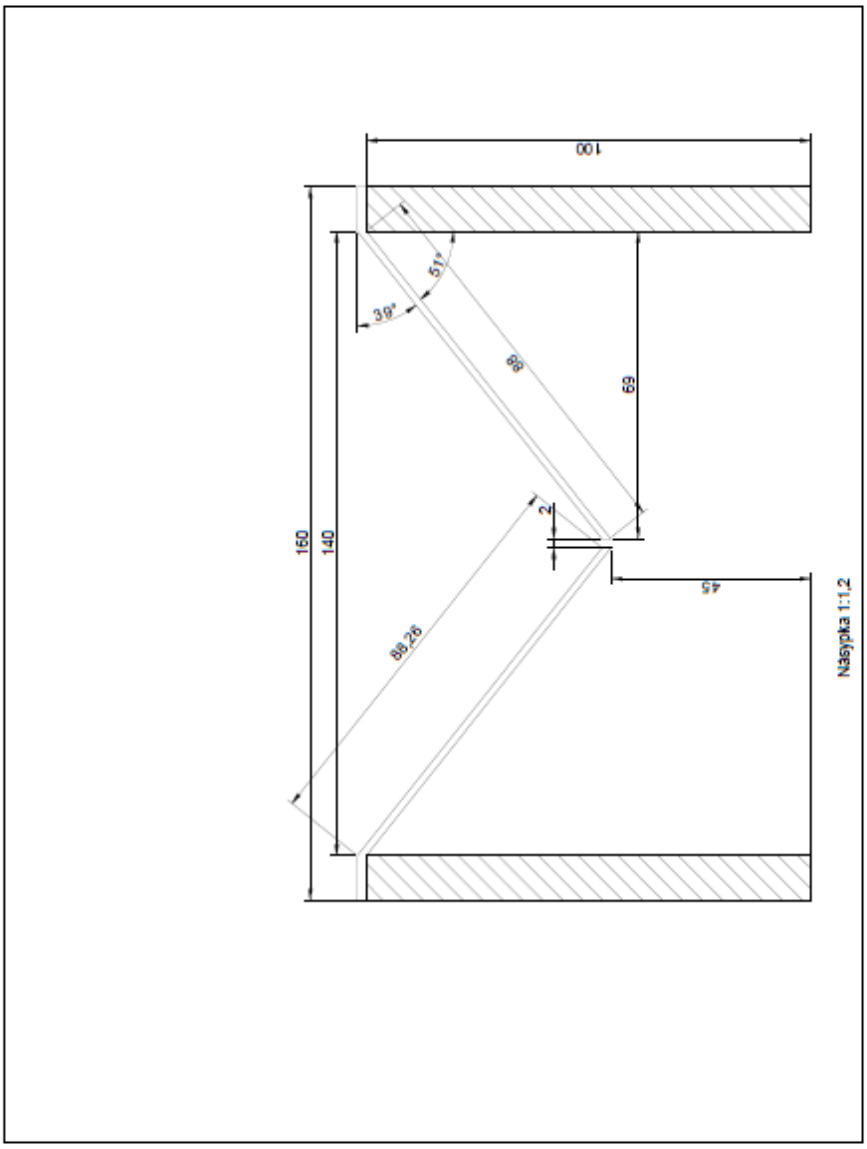


Příloha 2 Technický výkres Pásového dopravníku

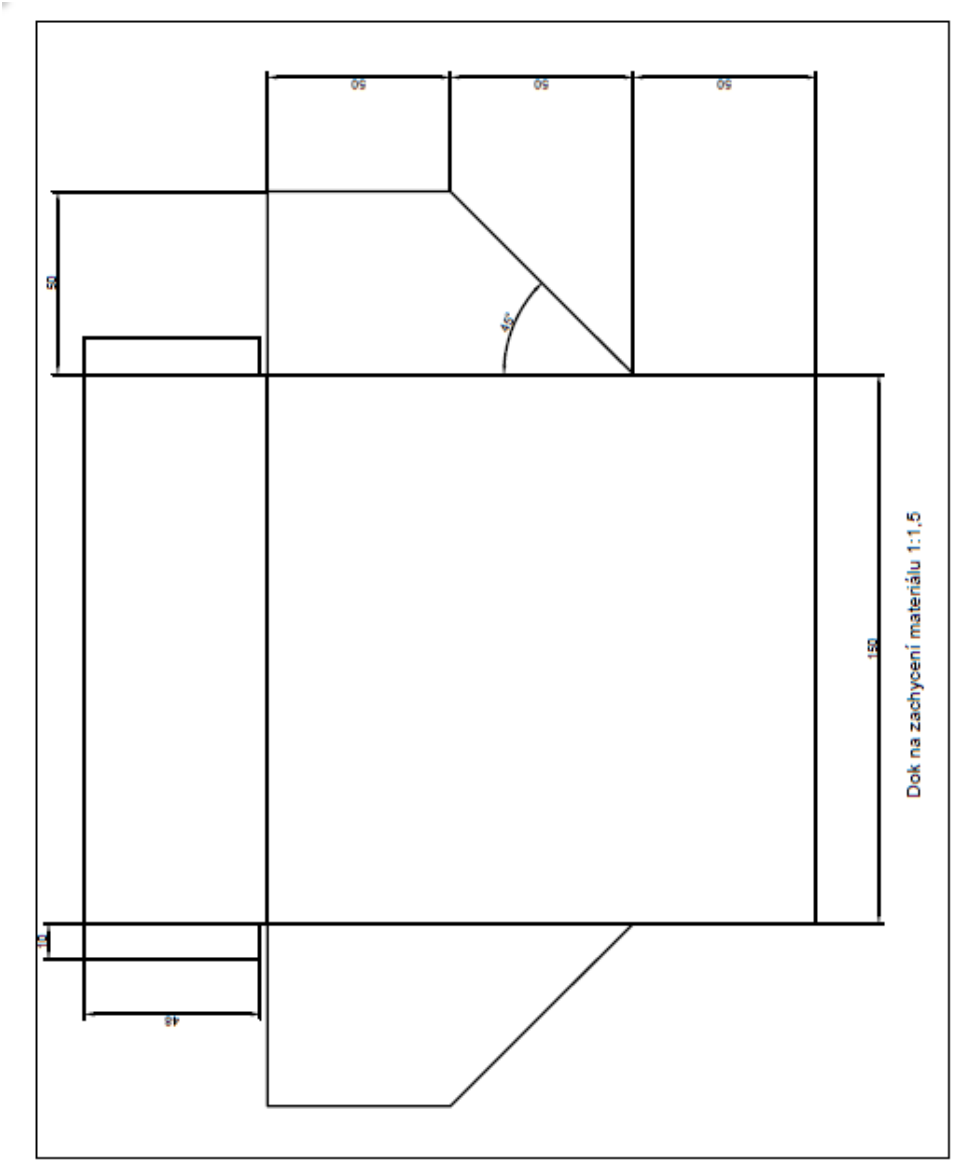


Příloha 3 Technický výkres Násypky

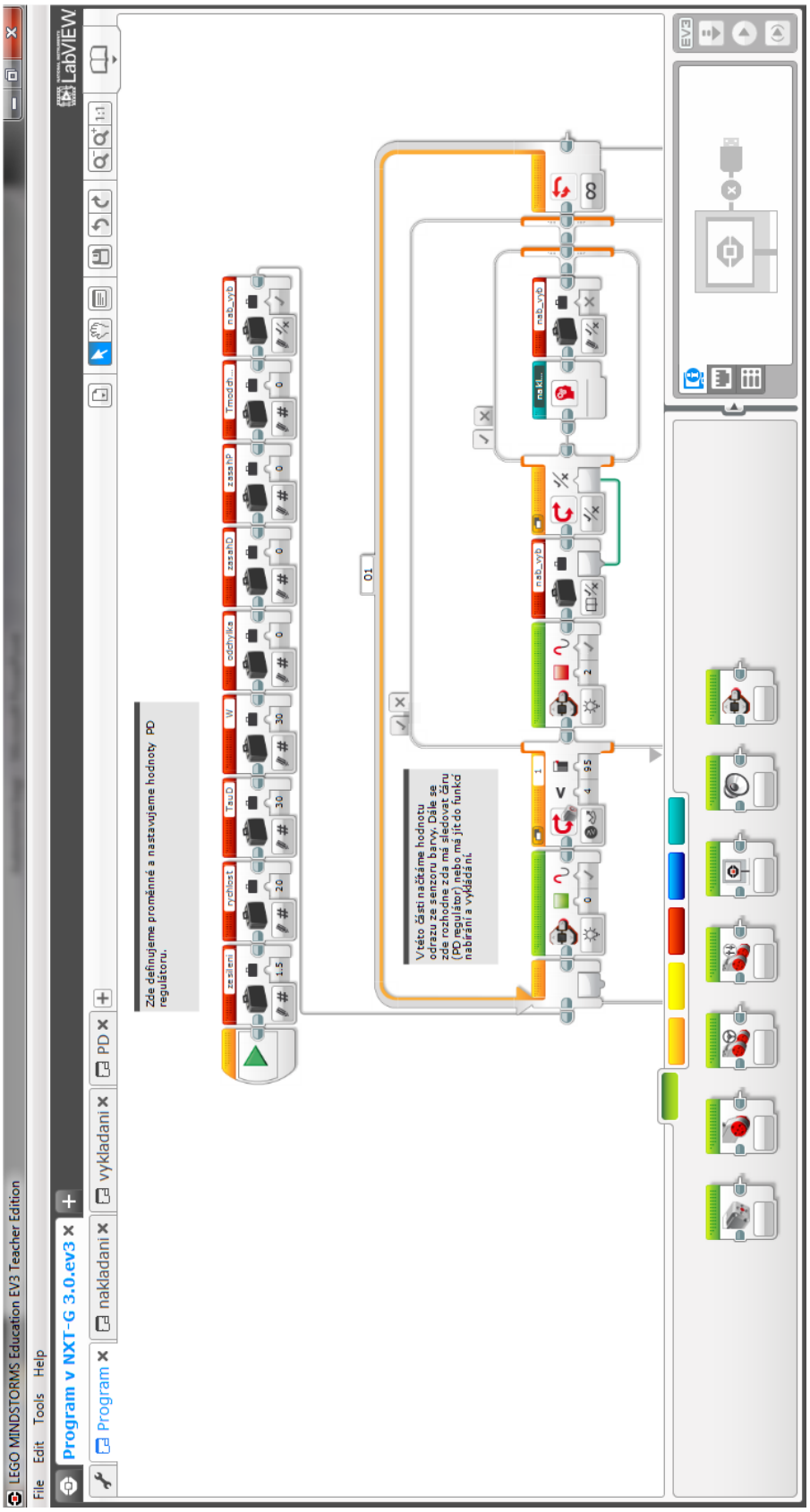



100
51°
39°
88
69
160
140
2
45
88,26
Násypka 1:1,2

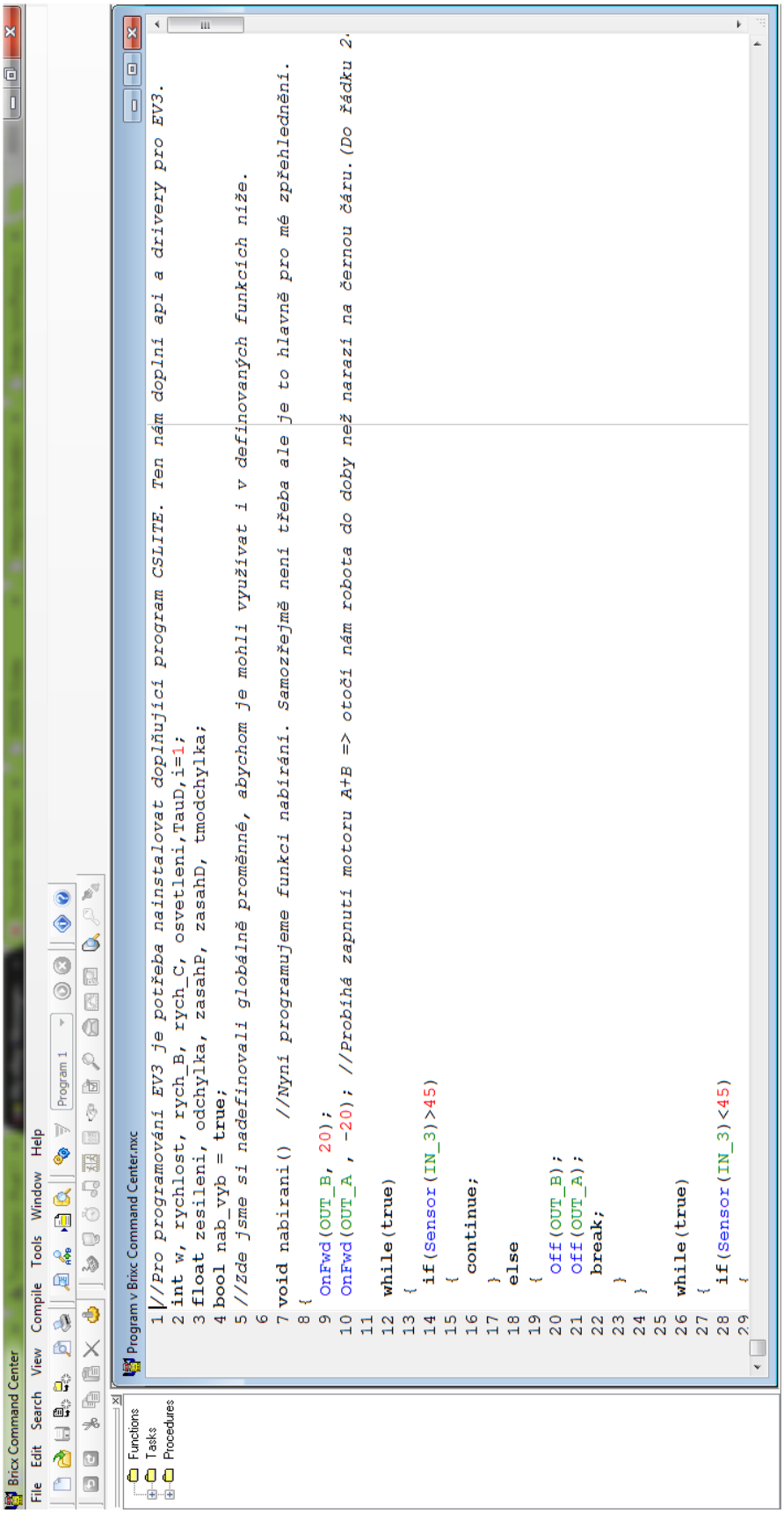
Příloha 4 Technický výkres Doku pro zachycení materiálu



Příloha 5 Lego Mindstorms education EV3



Příloha 6 Brixc Command Center



Příloha 7 LabVIEW