VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ
FACULTY OF MECHANICAL ENGINEERING

ÚSTAV AUTOMATIZACE A INFORMATIKY
INSTITUTE OF AUTOMATION AND COMPUTER SCIENCE

ARDUINO: PROGRAMOVÁNÍ V PROSTŘEDÍ MATLAB/SIMULINK

ARDUINO: PROGRAMMING BY MEANS OF MATLAB/SIMULINK

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — Tomáš Zimek
AUTHOR

VEDOUCÍ PRÁCE — doc. Ing. Radomil Matoušek, Ph.D.
SUPERVISOR

BRNO 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA STROJNÍHO INŽENÝRSTVÍ

Zadání bakalářské práce

Ústav:	Ústav automatizace a informatiky
Student:	**Tomáš Zimek**
Studijní program:	Strojírenství
Studijní obor:	Aplikovaná informatika a řízení
Vedoucí práce:	**doc. Ing. Radomil Matoušek, Ph.D.**
Akademický rok:	2016/17

Ředitel ústavu Vám v souladu se zákonem č. 111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma bakalářské práce:

Arduino: programování v prostředí Matlab/Simulink

Stručná charakteristika problematiky úkolu:

S využitím prostředí Matlab/Simulink (MATLAB® Support Package for Arduino® Hardware) budou vytvořeny minimálně tři demonstrační úlohy pro vývojový kit Arduino. Platforma Arduino s příslušenstvím i sw jsou k dispozici.

Cíle bakalářské práce:

1) Seznámení s problematikou programování platformy Arduino.
2) Seznamení s MATLAB® Support Package for Arduino® Hardware.
3) Po konzultaci se školitelem navrhnout minimálně tři demonstrační úlohy, pro které rovněž vytvořen popis.
4) Realizace úloh.

Seznam literatury:

VODA, Z., tým HW Kitchen. Průvodce světem Arduina. Martin Stříž, Bučovice, 2015. 240 s. ISBN: 978-80-87106-90-7

Termín odevzdání bakalářské práce je stanoven časovým plánem akademického roku 2016/17.

V Brně, dne 7. 11. 2016

L. S.

doc. Ing. Radomil Matoušek, Ph.D.
ředitel ústavu

doc. Ing. Jaroslav Katolický, Ph.D.
děkan fakulty

ABSTRAKT

Práce se zabývá plaformou Arduino, problematikou jejího programování v prostředí Matlab/Simulink a efektivní aplikací na třech demonstračních úkolech. První se představuje rozpoznání barev, když Arduino je využito jako vstupně/výstupní jednotka. V druhém úkolu deska za kompatibility s rozšířením Motor Shield jako sestavené vozítko sleduje čáru. V třetím úkolu se demonstruje pomocí manipulátoru přemístění objektů na základě odlišné barvy.

ABSTRACT

The thesis deals with the Arduino platform, its programming in Matlab/Simulink and effective application on three tasks. The first deals with color recognition, when the Arduino is is used as an input/output device. In the second task the Arduino board with Motor Shield compatibility follows a line as a vehicle. The third task demonstrates using a manipulator to move objects based on its color.

KLÍČOVÁ SLOVA

Snímání barev, rozpoznání barev, sledování čáry, praktická aplikace platformy Arduina

KEYWORDS

Color sensing, color recognition, line following, practical application of the Arduino platform

BIBLIOGRAFICKÁ CITACE

ZIMEK, T. *Arduino: programování v prostředí Matlab/Simulink.* Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2017. 77 s. Vedoucí bakalářské práce doc. Ing. Radomil Matoušek, Ph.D.

PODĚKOVÁNÍ

Děkuji doc. Ing. Radomilovi Matouškovi, Ph.D. za odborné vedení mé práce, ochotu, cenné rady a inspiraci pro realizaci praktických úkolů. Za podporu chci poděkovat svým rodičům.

ČESTNÉ PROHLÁŠENÍ

Prohlašuji, že tato práce je mým původním dílem, zpracoval jsem ji samostatně pod vedením doc. Ing. Radomila Matouška, Ph.D a s použitím literatury uvedené v seznamu literatury.

V Brně dne 26. 5. 2017 ……

Tomáš Zimek

OBSAH

1 ÚVOD

Pro potřebu realizace mnoha typů technických úloh se využívají zařízení označovaná jako tzv. vestavěné systémy (embedded systems). Tyto platformy se vyznačují integritou hardwarového řešení, nízkým příkonem, efektivností implementace, dobrým poměrem cena/výkon apod. Ke známým platformám patří Basic Stamp od společnosti Parallax, NetXia BX-24, Phidgets, BeagleBoard a další. Pro tuto práci byla požadována realizace úkolů pomocí open-source vývojové platformy Arduino. Ta se vyznačuje vysokou flexibilitou při použití různých externích modulů, kdy jako vestavěná platforma v tomto kompatibilním spojení umožňuje rychle vytvářet jednoduchá zařízení. Nachází uplatnění v senzorice, kde pro relativně velké množství vstupů a výstupů s ní lze zacházet jako se vstupně/výstupním zařízením. V současnosti s rychle se rozvíjejícím Internetem věcí (IoT) nachází uplatnění i v tomto odvětví, což platforma umožňuje za podpory shieldů (rozšíření).

Cílem práce je seznámit čtenáře s vývojovou platformou Arduino a na demonstračních příkladech ukázat její efektivní využití prostřednictvím standardního výpočetního prostředí pro technické výpočty Matlab, který tuto možnost implementoval. Vlastní práce je rozdělena na rešeršní a praktickou část. Do rešeršní části patří druhá a třetí kapitola. Kapitola 2 seznamuje s platformou Arduino, popisuje její problematiku programování a konkrétně rozebírá technické možnosti v práci použitých desek Arduino Uno a Arduino Due. V kapitole 3 se práce zaměřuje na MATLAB® Support Package for Arduino® hardware. Popisuje problematiku od instalace produktu až po vlastnosti jednotlivých funkcí pro psaní kódu přímo v Matlabu. V jejím závěru se práce orientuje na seznámení se s funkčními bloky v knihovně Simulink. První zvolenou úlohou je v kapitole 4 demonstrace rozpoznání barev na vytvořeném prototypu. Tento úkol poskytuje návod na jeho provedení od návrhu schématu zapojení součástí, sestavení senzoru až po objasnění stěžejních částí kódu programu psaném v prostředí Matlab. Arduino je přitom ovládáno v externím módu a slouží právě jako vstupně/výstupní jednotka. V 5. kapitole se řeší úkol, jehož účelem je zhotovit model pro sledování černé čáry docílené sestavením vozítka s pohonem dvou DC motorků. Program je tvořen v prostředí knihovny Simulink a po kompilaci nahrán do zařízení Arduino, které funguje v prototypu vozítka jako samostatná řídící jednotka. 6. kapitola představuje úlohu, jejímž cílem je sestavení manipulátoru pro přemístění předmětu ze tří výchozích pozic do dvou koncových. Tyto jsou odlišeny na základě barvy přenášeného předmětu. Manipulátor se skládá ze čtyř servo motorů a je ovládán externím režimem Arduina, které komunikuje s programem Matlab.

2 ARDUINO

Arduino je elektronická platforma zahrnující jak hardware, tak i software v podobě vývojvého prostředí a programovacího jazyka. Jedná se o open-source projekt, tedy takový, že pod veřejnými licencemi (CC a LPGL) jsou k dispozici hardwarová schémata [1]. Vyráběné desky spadají do skupiny mikrořadičů neboli jednodeskových počítačů založených na harvardské architektuře. Jsou typické původem svého procesoru od firmy Atmel a logem viditelným na obr. 1.

Obr. 1: Logo platformy Arduino [1]

Existuje mnoho dalších podobných mikrokontrolérů a mikroprocesorových platforem, například Basic Stamp od firmy Parallax, BX-24 vyvinuté ve společnosti Netmedia nebo Phidgets. Arduino se odlišuje od podobných platforem několika výhodami a vlastnostmi. Desky jsou poměrně levné, jednoduché typy lze sestavit i ručně na nepájivém poli. IDE běží na operačních systémech Linux, Windows a Macintosh OSX, kdežto jiné platformy jsou omezeny jen na Windows. Díky otevřenosti softwaru lze programovací jazyk rozšířit pomocí knihoven vytvořených v jazyku C++. Schémata desek jsou vydávány pod veřejnou licencí Creative Commons, díky čemuž lze vytvářet vlastní verze a tyto rozšířit nebo vylepšit.

Arduino pochází z názvu baru v městečku Ivrea, které leží na severozápadě Itálie. Tento je byl pojmenován podle Arduina z Ivrea, který v letech 1002 až 1014 kraloval Itálii. Spoluzakladatel projektu Massimo Banzi si tento bar tak oblíbil, že na jeho počest vybral pro pojmenování platformy právě jeho název. Banzi v roce 2002 pracoval v Institutu interaktivního designu v městu Ivrea, kde se snažil učit studenty jak vytvářet elektroniku rychle a levně. Využíval k tomu mikrokontrolér BASIC Stamp. Postupem času se setkával s problémem, kdy výpočetní možnosti tohoto jednočipového počítače byly nedostatečné k uskutečnění projektů, které jeho studenti chtěli provést. Navíc výše zmíněný mikrokontrolér vybavený základním příslušenstvím byl relativně drahý, neboť jeho cena pohybovala okolo sta amerických dolarů. Potřeboval něco, co by mohlo běžet na počítačích Macintosh, které měli na institutu k dispozici. Banziho kolega z MIT (Massachusetts Institute of Technology) vyvinul programovací jazyk Processing, který byl velice snadno použitelný pro začínající programátory. Bylo to totiž integrované

vývojové prostředí s vizualizací dat. Banzi si přál vytvořit podobný softwarový nástroj pro kódování jeho zamýšleného mikropočítače.

Jeho student Hernando Barragán vytvořil platformu Wiring. Ta zahrnovala malou obvodovou desku k použití s vývojovým prostředím IDE (Integrated Development Environment). Banzimu se podařilo tuto platformu ještě zdokonalit a první prototypová deska byla vyrobená v roce 2005. Snaha o zpřístupnění desky studentům se projevila na ceně, kdy deska byla dostupná asi za třicet amerických dolarů. Důležitá podmínka při vývoji platformy byla učinit desku typu plug-and-play, tedy takovou, kterou po rozbalení uživatel propojí do počítače a prakticky ihned může začít využívat [2, 3].

Existují dvě společnosti nesoucí Arduino ve svém názvu. Massimo Banzi je spojen s Arduino LLC, které se prezentuje webem arduino.cc. Tahle firma se stará o kód a vytváří aktualizace pro vývojové prostředí IDE. Díky partnerství s výrobci jako je např. firma Adafruit Industries sídlící v New Yorku se Arduino LLC stále více snaží vyrábět vlastní desky. Druhá firma, Arduino SRL (Small Projects) dostupná na webu arduino.org, se zaměřuje na výrobu samotných desek a dále je vývíjí. Byla založena v roce 2014 Gianlucou Martinem, nyní je ve vlastnictví Frederica Musta. Mezi těmito společnostmi byl v minulých letech soudní boj o značku Arduino. V roce 2016 došlo k urovnání sporů a následně začala spolupráce mezi zmíněnými firmami [4].

V současnosti je Arduino rozšířené ve světě mezi studenty, designery, inženýry a velkými společnostmi. Vytvořila se obrovská komunita lidí, která si díky otevřenosti systému vzájemně pomáhá přes internet s laděním kódu, vytvářením projektů a předáváním zkušeností spojených s platformou. Arduino dokáže interagovat s LED součástkou, spínačem, GPS jednotkou, motorem, kamerou, internetem nebo dokonce i s televizí a chytrým telefonem.

2.1 Arduino produkty

Existuje celá řada oficiálních výrobků, které lze dělit z hlediska jejich možností nebo výkonnosti. Produkty spadající do vstupní úrovně (Entry level) jsou snadno použitelné pro začátečnické projekty a jsou doporučené k seznámení se s elektronikou a problematikou jejího programování. Vylepšené produkty (Enhanced features) se vyznačují pokročilými funkcemi, rychlejšími výkony a většími možnostmi. Jiné lze využít pro IoT (Internet of Things), protože jsou navržené za účelem efektivního propojení s jinými zařízeními. Výrobky se dělí do tří hlavních skupin, o kterých je pojednáno níže [1, 3].

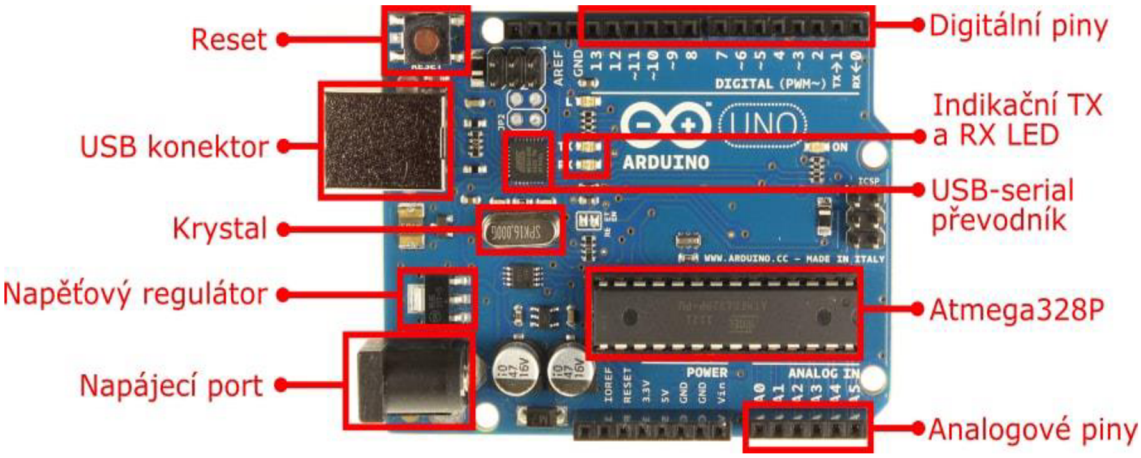
2.1.1 Desky

Jak už bylo zmíněno, Arduino běží na procesoru od firmy Atmel. Tento je propojen s dalšímy elektronickými součástkami. Zpravidla se jedná o převodník pro komunikaci s PC pomocí USB umožňující programování desky. Často jsou v desce zasazeny vstupní a výstupní piny typu samice. Tyto bývají digitální nebo analogové a slouží ke komunikaci s jinými zařízeními nebo dále popsanými moduly a shieldy. Další součástí bývá

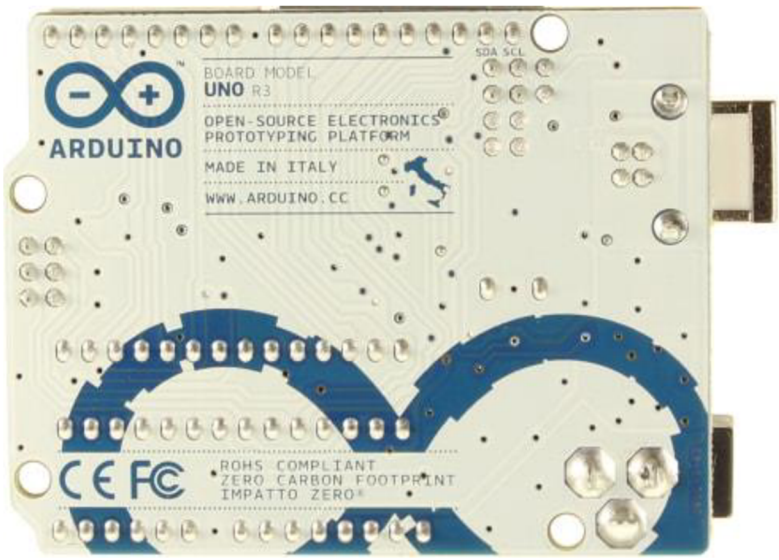
5V lineární regulátor napětí. Výsledné propojení vytváří celek s typickým grafickým designem v modré barvě. Některé typy mají více verzí, označované za svým názvem přidáním např. R3. Níže jsou kromě přehledu různých desek podrobněji rozebrány právě dvě, které byly použity pro účely praktické části této práce. Jedná se o Arduino Uno a Due.

Arduino Uno

Označované také jako Genuino Uno funguje na 8bitovém procesoru ATmega328P. Obsahuje 6 input pinů, 14 digitálních I/O (Input/Output) pinů, z nichž 6 lze použít jako PWM (Pulse Width Modulation) výstupy. Je vybaveno také konektory ICSP (In-Circuit Serial Programming). Dále pro komunikaci s PC má zabudovaný USB B konektor. Vedle něho je umístěn napájecí konektor a reset tlačítko. Krystal taktuje procesor na frekvenci 16 MHz. Rozložení komponentů je zřetelné z obr. 2, zadní strana desky je zachycena na obr. 3.



Obr. 2: Arduino Uno R3



Obr. 3: Arduino Uno R3 zadní strana

Tab. 1 popisuje technické vlastnosti desky. Procesor je předprogramovaný pomocí bootloaderu, který umožňuje nahrát nový kód bez použití externího hardwarového programátoru. Tomuto se lze vyhnout při použití ICSP konektoru [1].

Mikrořadič	Atmega328P
Provozní napětí	5 V
Vstupní napětí (doporučené)	7–12 V
Vstupní napětí (limit)	6–20 V
Digitální I/O piny	14 (z toho 6 PWM)
PWM digitální I/O piny	6
Analogové vstupní piny	6
Stejnosměrný proud na jeden I/O pin	20 mA
Stejnosměrný proud pro 3,3V pin	50 mA
Paměť Flash	32 KB (Atmega328P) z nichž 0,5 KB používá bootloader
SRAM	2 KB (Atmega328P)
EEPROM	1 KB (Atmega328P)
Krystal	16 MHz
Zabudovaná LED	Pin 13
Délka	68,6 mm
Šířka	53,4 mm
Hmotnost	25 g

Tab. 1: Technické specifikace pro Arduino Uno

Deska obsahuje vratnou pojistku (resetable polyfuse), která chrání USB port v PC před zkratem nebo proudovým přetížením. Tato pojistka přeruší spojení po dobu zkartu nebo v případě, že je proud větší než 500 mA.

Napájení může pocházet z USB spojení, nebo z externího zdroje. Ten je automaticky vybrán. Externí napájení může být přes AC/DC adaptér připojeným na 2,1 mm konektor, nebo baterií zapojenou do GND (Ground) a Vin pinu. Deska poskytuje skrz 5V pin regulované napětí, stejně tak přes 3.3V pin, kde produkuje napětí o velikosti 3,3 V a o maximálním proudu 50 mA. IOREF pin poskytuje referenční napětí na kterém operuje mikrokontrolér.

Každý ze 14 digitálních pinů může poskytnout nebo přijmout proud o velikosti 20 mA a funguje při napětí 5 V. Některé piny mají speciální funkce. Sériové RX (0) se používá k přijímání a TX (1) k vysílání sériových TTL (Transistor–transistor logic) dat. PWM piny (3, 5, 6, 9, 10 a 11) poskytují 8bitový výstup. Zabudovaná LED je zapnutá v případě, kdy pin 13 má hodnotu 1 (high), naopak se vypne při hodnotě 0 (low). Analogové vstupy A0 až A5 poskytují 10bitové rozlišení (1024 různých hodnot). Arduino Uno je nejčastěji používaný typ, je pokračovatelem původního Arduina, které mělo místo USB sériový port [1, 3].

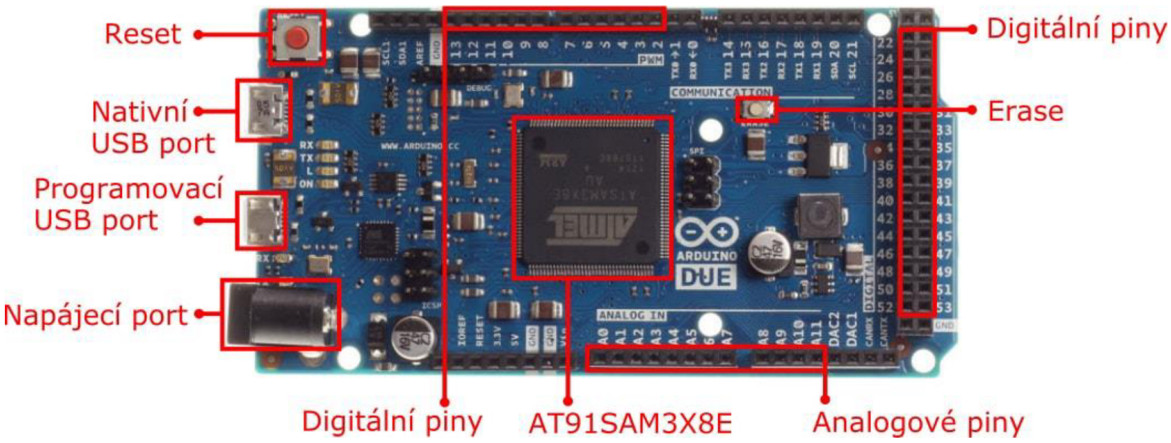
Arduino Due

Deska podobná Arduinu Uno, považovaná v hierarchii výše, je pokračovatelem Arduina Mega s podstatným rozdílem ve výkonnosti čipu. Je jím SAM3X8E ARM Cortex-M3 s 32-bitovým jádrem, procesor má taktovací frekvenci 84 MHz. Deska komunikuje přes dva micro-USB konektory, jeden (Programming port) slouží k programování mikrokontroléru, druhý (Native port) umožňuje připojení periferních zařízení jako jsou myš, klávesnice nebo chytrý telefon. Komunikaci umožňuje 54 digitálních pinů, 12 analogových vstupních pinů, 2 DAC (Digital to Analog Converter) výstupy. Součástí jsou i tlačítka RESET a ERASE. Narozdíl od ostatních desek je Due specifické v tom, že operuje na napětí 3,3 V. Tudíž přívod větší velikosti na I/O piny může nenávratně desku poškodit. Specifikace Arduina Due ukazuje tab. 2 [6].

Mikrořadič	AT91SAM3X8E
Provozní napětí	3,3 V
Vstupní napětí (doporučené)	7–12 V
Vstupní napětí (limit)	6–16 V
Digitální I/O piny	54 (z toho 12 PWM)
Analogové vstupní piny	12
Analogové výstupní piny	2 (DAC)
Celkový stejnosměrný proud na všechny I/O	130 mA
Stejnosměrný proud pro 5V pin	800 mA
Stejnosměrný proud pro 3,3V pin	800 mA
Paměť Flash	512 KB
SRAM	96 KB (64 KB a 32 KB)
Krystal	84 MHz
Délka	101,52 mm
Šířka	53,3 mm
Hmotnost	36 g

Tab. 2: Technické specifikace pro Arduino Due

Mikrokontrolér SAM3X8E má 512 KB (dva bloky po 256 KB) flash paměti vyhrazené k uložení kódu. Tuto paměť je možné vymazat stisknutím tlačítka ERASE po dobu několika sekund. SPI (Serial Peripheral Interface) konektory slouží ke komunikaci jen se zařízeními také podporující SPI sběrnici. Analogové vstupy mají 12bitové rozlišení, ale jsou nastaveny do defaultně na rozlišení 10bitové, z důvodu kompatibility s ostatními typy desek. Oba USB porty mohou být použity k programování, ale výrobce doporučuje používat jen Programming port. Arduino Due je navrženo tak, že je kompatibilní s shieldy navrženými pro Uno, Diecimila a Duemilanove. Rozložení součástí je zachyceno na obr. 4.

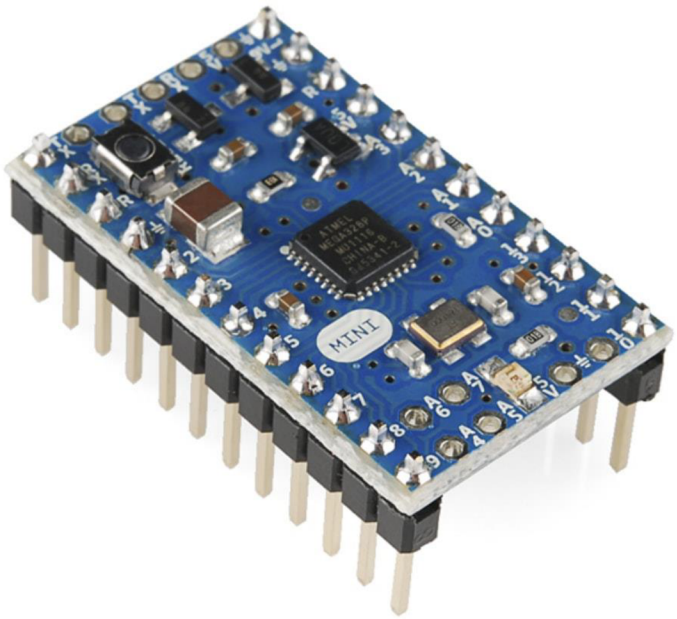


Obr. 4: Arduino Due R3

Mezi desky se dále řadí: **Leonardo**, které se od Una odlišuje čipem ATmega32u4. Je ideální pro projekty, které vyžadují od zařízení, aby reprezentovala USB HI (Human Interface) uživatelské rozhraní. **Arduino 101** obsahuje akcelerometr a gyroskop, využití nalezne např. při rozpoznávání gest. Komunikace je možná přes Bluetooth připojení. **Arduino Robot** je set se dvěma mikrokontroléry na dvou spojených deskách. Jeden je pro řízení motorů, druhý pro sběr dat ze senzorů a vykonávání uloženého kódu. **Mega 2560** je navrženo pro složitější projekty, hodně prostoru pro své využití umožňuje svými 54 digitálními I/O piny a 16 analogovými vstupy. **Arduino Yún** nachází uplatnění pro projekty využívané pro IoT. To vedle čipu ATmega32u4 obsahuje také mikroprocesor Atheros AR9331 podporující linuxový operační systém Linino OS. Deska obsahuje zabudovaný Ethernet port a WiFi podporu pro připojení k síti, což je žádoucí u využití v IoT např. při sběru dat a jejich následném posílání na server. **Arduino Ethernet** je téměř kopií Una, rozdíl je v připojení přes Ethernet konektor. **LilyPad Arduino** je specifický typ, který se může našít na oděv [3].

2.1.2 Moduly

Jsou to menší obnože klasických desek. Na obr. 5 je **Arduino Mini** vyznačující se svými relativně malými rozměry, kdy při zachování výkonu šetří místo v prostoru. Postrádá USB port pro sériovou komunikaci, pro programování je nutné použít externí USB sériový převodník. **Arduino Nano** je kompaktní zmenšená verze podobná k Unu. Je vybaveno USB portem a převodníkem. K modulům se řadí také **Arduino Micro** a **Yun mini.**



Obr. 5: Arduino Mini

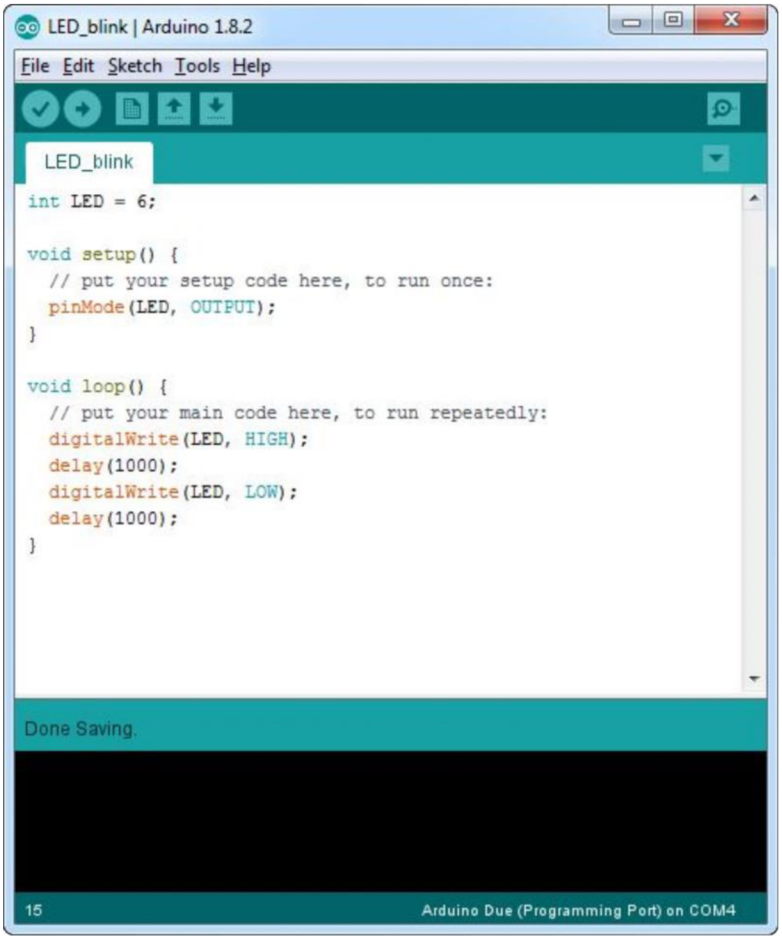
2.1.3 Shieldy

Jsou to rozšíření, které se svými vystupujícími nožičkami pinů zasunou do odpovídajících pinů desky, kterou vylepšují. Poté umožňují provádět funkce, kterým původní deska nebyla schopná. Uplatnění nacházejí **Motor Shield** k řízení dvou stejnosměrných DC motorů, nebo jednoho krokového. **USB Host Shield** dává možnost připojení USB zařízení k Arduino desce. **4 Relays Shield** je řešení pro řízení vysokého zatížení, které nelze ovládat digitálními I/O piny kvůli omezením proudu a napětí regulátorem. Shield je vybaven čtyřmi relé, každé má dva přepínací kontakty (NO a NC). Čtyři LED indikují stav zapnutí nebo vypnutí každého relé. Využití v IoT nacházejí shieldy **WiFi**, **Wireless SD** a **Ethernet Shield 2**.

2.2 Uživatelské rozhraní Arduino IDE

Na webu arduino.cc je nyní dostupná verze 1.8.2. Rozhraní lze spustit ve webovém prohlížeči, nebo si jej stáhnout na PC. Běží pod operačnímy systémy Windows, Linux a Mac OS X. Po instalaci na Windows a spuštění programu se objeví okno s vývojovým prostředím jak lze vidět na obr. 6. Některé desky vyžadují doinstalování přídavného jádra. V našem případě se jedná o Arduino Due, kdy pro možnost naprogramování této desky je nutné nainstalovat SAM jádro pro ARM/SAM mikrokontrolér desky Due. Cesta k tomuto v IDE je přes kliknutí postupně na *Tools* > *Board* > *Boards Manager*. Vybere se *Arduino SAM Boards* a po úspěchu se zavře *Boards Manager*. Po připojení Arduina, v našem případě Due, se toto nastaví přes *Tools* > *Board*, kde se vybere *Arduino Due (Programming port)*. Předem napsaný kód se může nahrát na desku. Vlevo nahoře je ikona Verify, díky které IDE zkontroluje kód na přítomnost chyb. Vedle je ikona Upload, která umožní nahrát kód na právě konfigurovanou desku.

Arduino je možné programovat v jazyce C, kdy se používá komplexní knihovna Wiring. Na obr. 6 lze vidět ukázkový kód pro blikání LED součástky. Lze si všimnout, že program se skládá ze dvou hlavních oddílů, které jsou podmínkou správného fungování. První část, funkce `setup()`, obsahuje kód, který se provede jen jednou na začátku programu. Slouží k nastavení módu pinů a inicializaci proměnných. Spustí se vždy při začátku napájení Arduina, nebo znovu po stisku tlačítka RESET. Druhá funkce, `loop()`, obsahuje tu část kódu, která se bude opakovat až do odpojení napájení.



Obr. 6: Rozhraní Arduino IDE

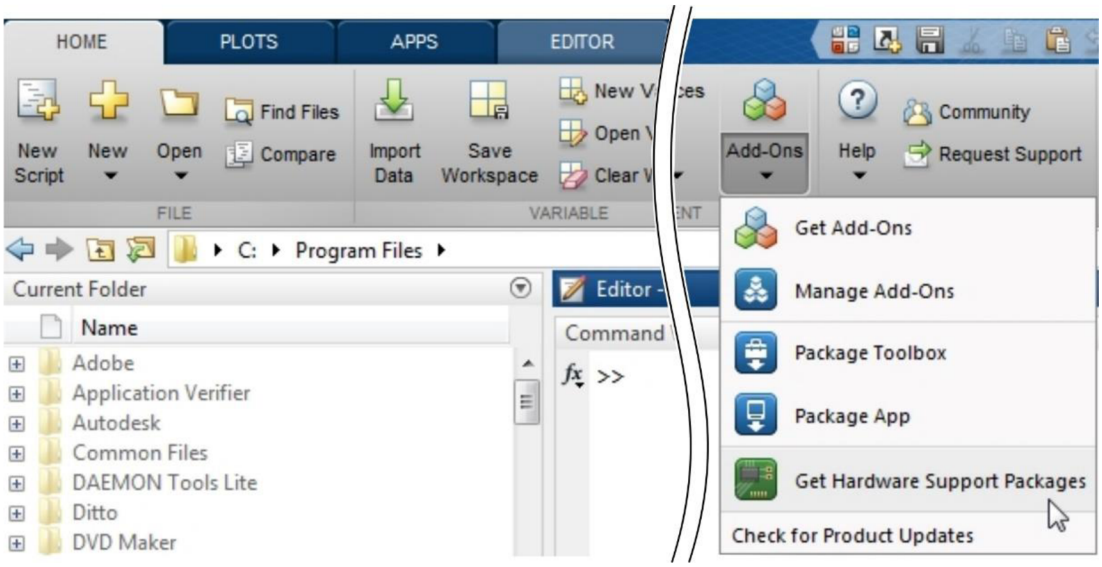
Je možné využít dalších programovacích jazyků pro Arduino. **Snap4Arduino** je vizuální jazyk, který poskytuje možnosti těm, kteří nepreferují textové kódování. Pro úplné začátečníky nebo pro ty, kteří preferují vizuální logické programování je vhodný jazyk **ArduBlock**. Sériová komunikace je umožněna v jazyce **C#** pomocí knihovny **CmdMessenger**. Další možností sériové komunikace s Arduinem je využití **Python** jazyka s jeho rozšířením **pySerial**. V následující kapitole se práce zaměří na programování Arduina v softwaru **Matlab** a v jeho nástroji pro grafické blokové programování **Simulink**.

3 MATLAB® SUPPORT PACKAGE FOR ARDUINO® HARDWARE

V praktické části této práce je pro realizaci úloh používán software Matlab verze R2015b (8.6.0.267246) běžící na 64-bitovém operačním systému Windows 7 Home Premium. Předchůdce rozšíření Matlabu pro Arduino, Legacy Matlab and Simulink Support for Arduino, je možné použít jen pro Matlab od verze R2011a až po R2013b. Nyní dostupná podpora Arduina, která je použita pro účely této práce je kompatibilní s verzemi od R2015a do R2017a. Kód napsaný v Matlabu pomocí tohoto rozšíření umožňuje přes sériové spojení ovládat vstupy a výstupy Arduina, používat ho jako vstupně výstupní zařízení, přičemž rozhodovací proces probíhá v Matlabu. Pro Simulink je kromě této možnosti ještě výhoda, že vytvořený program se může zkompilovat a nahrát na danou desku, která potom funguje jako samostatná řídící jednotka bez nutnosti připojení k Matlabu.

3.1 Instalace

Po spuštění Matlabu se k rozšířením dá dostat postupně kliknutím v horní části lišty na *HOME* > *Add-ons* > *Get Hardware Support Packages* jak je vidět z obr. 7. Zobrazí se okno, ve kterém se zvolí *Install from Internet* > *Next*. V seznamu na levé straně se vybere *Arduino*, v pravé části okna se potom objeví balíčky pro Matlab a pro Simulink. Oba je potřeba zatrhnout k akci *Install* a pak pokračovat *Next*. Zde je nutnost se přihlásit ke svému účtu na MathWorks, pokud ještě žádný založený není, musí se vytvořit. Po přihlášení začne instalace a po jejím dokončení se tahle rozšíření můžou začít funkčně používat.



Obr. 7: Otevření rozšíření Matlabu pro Arduino hardware

3.2 Matlab Support Package for Arduino Hardware

Jak už bylo výše zmíněno Matlab umožňuje externí řízení Arduina sériovou komunikací. Pro spojení s arduinem je k dispozici funkce, jejíže syntaxe je `a = arduino()`. Tato vrací parametry připojení do proměnné `a`. Na ni se další funkce určené k ovládání Arduina odkazují. Provedení tohoto příkazu je vidět na obr. 8. Vlastnosti ukazují, na který port je připojeno Arduino, o jaký typ desky se jedná, které I/O piny je možno ovládat a jaké typy funkcí jsou z rozšiřující knihovny použity.

```
Command Window
>> a = arduino()

a =

  arduino with properties:

                    Port: 'COM8'
                   Board: 'Uno'
          AvailablePins: {'D2-D13', 'A0-A5'}
              Libraries: {'I2C', 'SPI', 'Servo'}

fx >>
```

Obr. 8: Připojení se k Arduinu v Matlabu

Funkce pro I/O piny

`configurePin(a, pin, mode)` nastaví vybraný pin na specifikovaný typ módu. Ten může mít hodnotu z několika možností, z nichž je vhodné zmínit `DigitalInput` pro nastavení pinu ke sběru digitálních informací, `DigitalOutput` naopak pro jejich vysílaní. Mód `PWM` slouží k používání pulsně šířkové modulace. `Servo` umožňuje řídit servo motor.

Pro získání informací o hodnotě digitálního vstupu 0 (low) nebo 1 (high) je k dispozici funkce `readDigitalPin(a, pin)`, která jednu z těchto hodnot vrací. Obdobně pracuje funkce `writeDigitalPin(a, pin)`, pro vyslání digitální informace.

Funkce `writePWMVoltage(a, pin, voltage)` nastavuje hodnotu napětí pro PWM pin. Většina desek umožňuje 0–5 V, Arduino Due jen 0–3,3 V. Související funkcí je `writePWMDutyCycle(a, pin, dutyCycle)`, která přijímá hodnoty od 0,00 (0 %) do 1,00 (100 %) pro nastavení PWM cyklu.

Informaci o přijatém napětí na analogovém pinu umožňuje získat `readVoltage(a, pin)`, která vrací hodnotu ve Voltech.

Řízení bzučáku umožňuje `playTone(a, pin, frequency, duration)`, jenž si bere za parametr frekvenci tónu v hodnotách 0–32767 Hz, dalším parametrem je doba tohoto tónu v sekundách, platné hodnoty jsou 0–30 [7].

Funkce pro řízení Serv

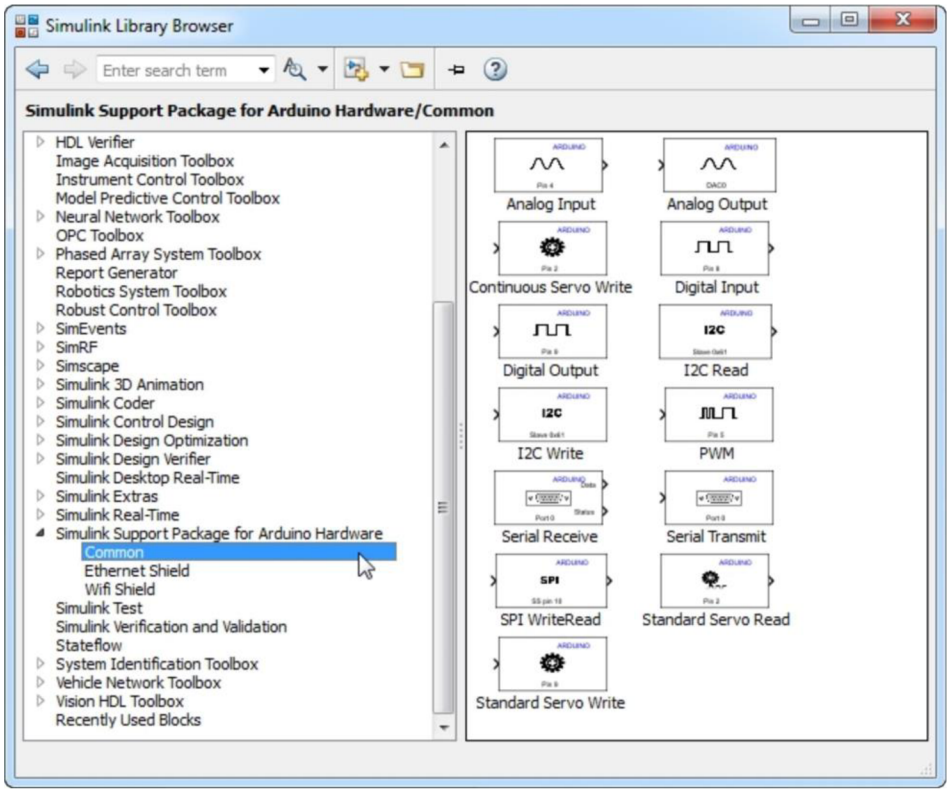
Vytvoření spojení s motorkem je realizováno funkcí, která vrací objekt, jenž je potřeba uložit do proměnné. Syntaxe `s = servo(a, pin)` přiřadí proměnné `s` objekt servo motoru konfigurovaný na vybraný pin arduina, jehož objekt je v proměnné `a` uložen.

Určení pozice servo motoru umožňuje `readPosition(s)`, funkce vrací hodnotu v rozmezí 0,00–1,00. Tato pak reprezentuje úhel natočení serva vzhledem k jeho maximálnímu možnému úhlu (tomu odpovídá hodnota 1,00). Analogicky pracuje `writePosition(s, position)`, která vydává příkaz k natočení serva pomocí parametru `position`, hodnota 0,50 představuje prostřední pozici.

K dispozici jsou ještě další typy funkcí, které ale pro tvorbu této práce nebyly využity. Jsou to funkce pro komunikaci s I2C, SPI zařízeními, posuvnými registry nebo jiné pro řízení motorů připojených k Motor Shield V2 od společnosti Adafruit [7].

3.3 Simulink Support Package for Arduino Hardware

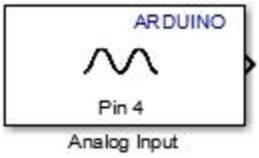
Prohlížeč knihovny funkčních bloků Simulinku je zpřístupněný v Matlabu přes *HOME* > *Simulink Library*. Po jeho otevření a vybrání z listu možností bloků je zde sekce podpory pro Arduino. Jak je patrné z obr. 9 jsou zde tři skupiny bloků. Common (běžné bloky), jenž tahle práce využívá, obsahuje bloky podobné funkcím pro Matlab zmíněné v předchozí podkapitole. Další dvě skupiny obsahují bloky pro řízení a ovládání Ethernet a Wifi Shieldu, které ovšem práce nevyužívá, proto o nich dále nebude pojednáno.



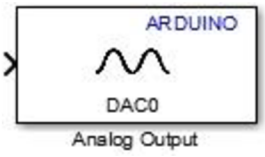
Obr. 9: Prohlížeč knihovny Simulink

Pro funkční vytvoření modelu v Simulinku je vhodné nejprve nakonfigurovat typ desky, s kterou se bude komunikovat a na které bude daný model spuštěn. V okně modelu se to nastaví stisknutím ikony ozubeného kola následovně: *Model Configuration Parametres* > *Hardware Implementation* > *Hardware Board*, kde se vybere daný typ desky.

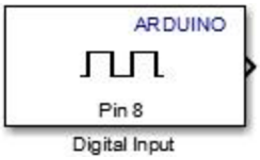
V následující části budou stručně charakterizovány funkční bloky skupiny Common, kam se řadí také bloky pro spojení s I2C a SPI zařízeními a pro sériovou komunikaci. Z důvodu nevyužití těchto bloků pro potřeby práce nebudou dále popsány.



Pomocí bloku navlevo se na daném pinu měří napětí relativní vůči referenčnímu napětí dané desky. Pokud je rovno GND (ground) napětímu desky, výstupní hodnota odpovídá 0, naopak napětí rovno referenčnímu je naměřeno jako maximální hodnota 1023 (10bitová). Při seřízení parametrů tohoto bloku je možné nastavit vzorkovací čas (Sample time) v sekundách, minimální hodnota je 0,000001 sekundy.



Analog Output generuje napětí na specifikovaném 12bitovém DAC pinu (DAC0 nebo DAC1). Blok sice přijímá hodnoty datového typu unit16, ale počítá jen s prvními 12 nejvýznamějšími bity. Rozsah hodnot je 0–4095.



Jestli je hodnota vstupu na daném pinu low, tento blok generuje hodnotu 0, v případě high je blokem produkována 1, datový typ výstupu je boolean. Stejně jako je zmíněno výše u bloku Analog Input i zde se nastavuje vzorkovací čas.

Digital Output funguje analogicky jako předešlý blok. Vstupem hodnoty 1 se nastaví specifikovaný digitální pin na hodnotu napětí 5 V, respektive 3,3 V u Arduina Due. Naopak hodnota 0 odpovídá 0 V (low).

Blok pro generování PWM signálu o předem stanoveném cyklu. Vstupní hodnoty v rozmezí 0–255 odpovídají 0–100 %. Frekvence výstupní vlny je konstantní okolo 490 Hz.

Následující blok pomocí PWM signálu vysílá řídící informaci o pozici natočení standardního serva. Přijímá hodnoty 0–180 odpovídající úhlovým stupňům. Pro tento blok a dva následující bloky k řízení servo motorů je dosti nepříjemné omezení, kdy dokumentace k rozšíření Simulinku pro Arduino nedoporučuje používat tyto bloky v externím režimu provádění programu [8], což je zmíněno v praktické části práce v kapitole 6.

Tento blok poskytuje hodnotu pozice natočení servo motoru ve stupních s rozsahem 0–180. Jak už bylo zmíněno u jiných bloků i pro tento se nastavuje vzorkovací frekvence.

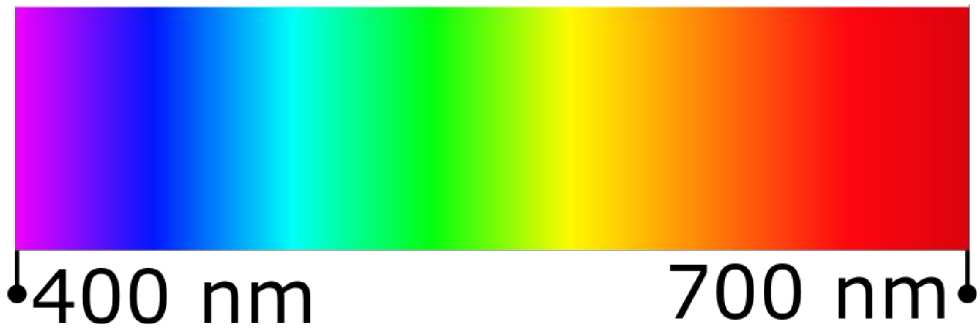
Continuous Servo Write určuje rychlost rotace hřídelky rotačního serva. Hodnota -90 znamená maximální rychlost rotace v jednom směru, 90 v druhém směru. Nula na vstupu do bloku zapříčiní zastavení rotace serva [9].

4 ROZPOZNÁNÍ BAREV

Zadání úlohy: Sestavit jednoduchý senzor, na kterém bude demonstrování rozpoznání barev. Program bude implementován jako M-file v Matlabu.

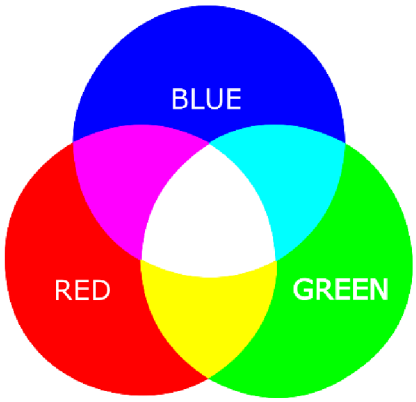
4.1 Rozbor

Barva je charakteristika lidského vizuálního cítění reprezentována kategoriemi (modrá, červená, fialová, žlutá, atd.). Tyto barvy jsou díky lidskému smyslu zraku přiřazeny objektům, od kterých se odráží světlo. Viditelné spektrum světla, jak lze vidět z obr. 10, je část elektromagnetického záření o vlnových délkách od 400 nm do 700 nm. Mozek interpretuje barvu objektu podle toho, které složky světla (vlnové délky odpovídají různým barvám) se od objektu odrazí a jsou následně zpracovány sítnicí v lidském oku pomocí světločivných buněk (čípků). Tyto zvláštní buňky se dělí na tři funkční typy, které jsou specializované ke vnímání tří barev světla, a to červené (red), modré (blue) a zelené (green). Konečná barva je výsledkem celkového vnímání intenzity těchto tří barev. Čípky v oku tedy plní funkci senzoru barvy, mozek následně pracuje z dodávanými informacemi, které vyhodnotí [10].



Obr. 10: Viditelné barevné spektrum

Rozlišuje se několik základních barevných modelů, z nichž aditivní RGB (obr. 11) je inspirován lidskou fyziologií. Tento model výslednou barvu určuje z toho jaké je množství (intenzita) třech základních barev. Jeho numerický model je takový, kdy jsou hodnoty intenzity jedné složky barvy uloženy jako celé číslo v rozsahu 0–255, který představuje jeden byte. Každá barva může být vyjádřena jako třísložkový vektor, kdy první složka odpovídá červené části, druhá složka zelené a poslední představuje modrou. Ve zmíněném modelu se bílá barva může psát jako WHITE = [255, 255, 255], protože všechny tři její složky se vyznačují maximální intenzitou. Naopak černá barva světlo pohlcuje, odráží jen malé množství, její vyjádření je BLACK = [0, 0, 0]. [11, 12] Podobnou schopnost vnímání barev jako lidské oko má také fotorezistor (LDR, light-dependent resistor), který mění odpor na základě intenzity dopadajícího záření.



Obr. 11: RGB aditivní model

4.2 Návrh provedení

Realizace je provedena sestavením RGB senzoru, který detekuje barevné kartičky. Ten je propojený s Arduino Uno, jenž má ve spojení s PC přes USB kabel funkci vstupně výstupní jednotky. Po zpracování dat ze senozoru vyhodnocuje program napsaný v Matlabu barvu. Vyšle řídící signál pro určené natočení serva, to na barevném ciferníku ukáže svým směrovníkem na rozpoznanou barvu.

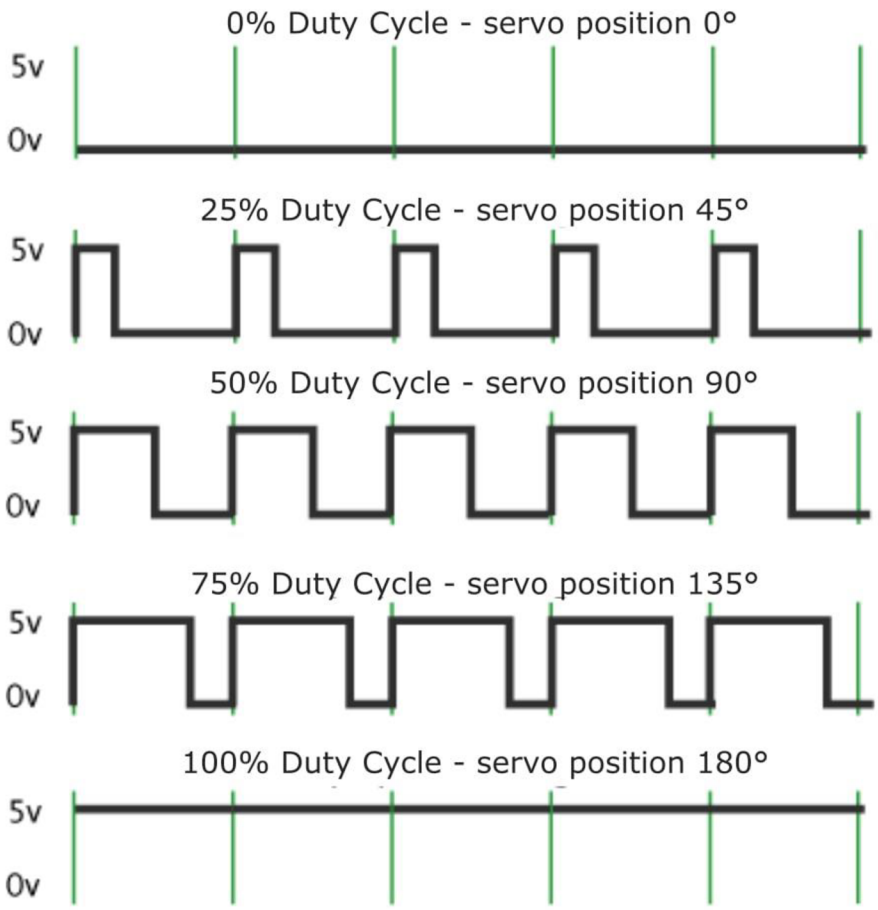
RBG senzor

Sestává se z **fotorezistoru 5mm GL5539** a **tříbarevné RGB diody se společnou katodou** jak lze vidět z obr. 16. Princip fungování je založen následovně: dioda vysílá postupně červené, zelené a modré světlo na snímanou kartičku. Od ní se světlo odráží a dopadá na fotorezistor. Ten podle intenzity dopadajícího světla mění svůj odpor, jenž je nepřímo přes napětí na tomto fotorezistoru měřen v analogovém vstupu desky Arduino. Po přepočítaní napětí na 8bitové hodnoty se takto získají tři barevné složky detekované kartičky.

GL5539 fotorezistor je tvořen polovodičovým materiálem s vysokým odporem. V tmavém prostření dosahuje rezistance až 1 MΩ, pokud je osvětlení intenzivní, odpor klesá až na několik ohmů. Tříbarevná LED dioda produkuje červené světlo vlnové délky v rozsahu 635–645 nm, zelené 520–530 nm a modré 465–475 nm.

Servo motor

Pro vizuální ukazatel barvy je použito servo **GO-13MG**. Může být napájeno 4,8–6 V, jeho rozsah natočení je 0–180 stupňů. Je řízeno PWM (Pulse Width Modulation) signálem, který určuje pozici serva. Tahle diskrétní modulace nahrazuje analogový signál digitální cestou. Posílá se obdélníková vlna, která má periodu složenou ze dvou hodnot, signál je zapnutý a signál je vypnutý. Signál nese informaci o hodnotě napětí jako poměr mezi stavy zapnuto/vypnuto. Princip v souvislosti s natočením serva ukazuje obr. 12.



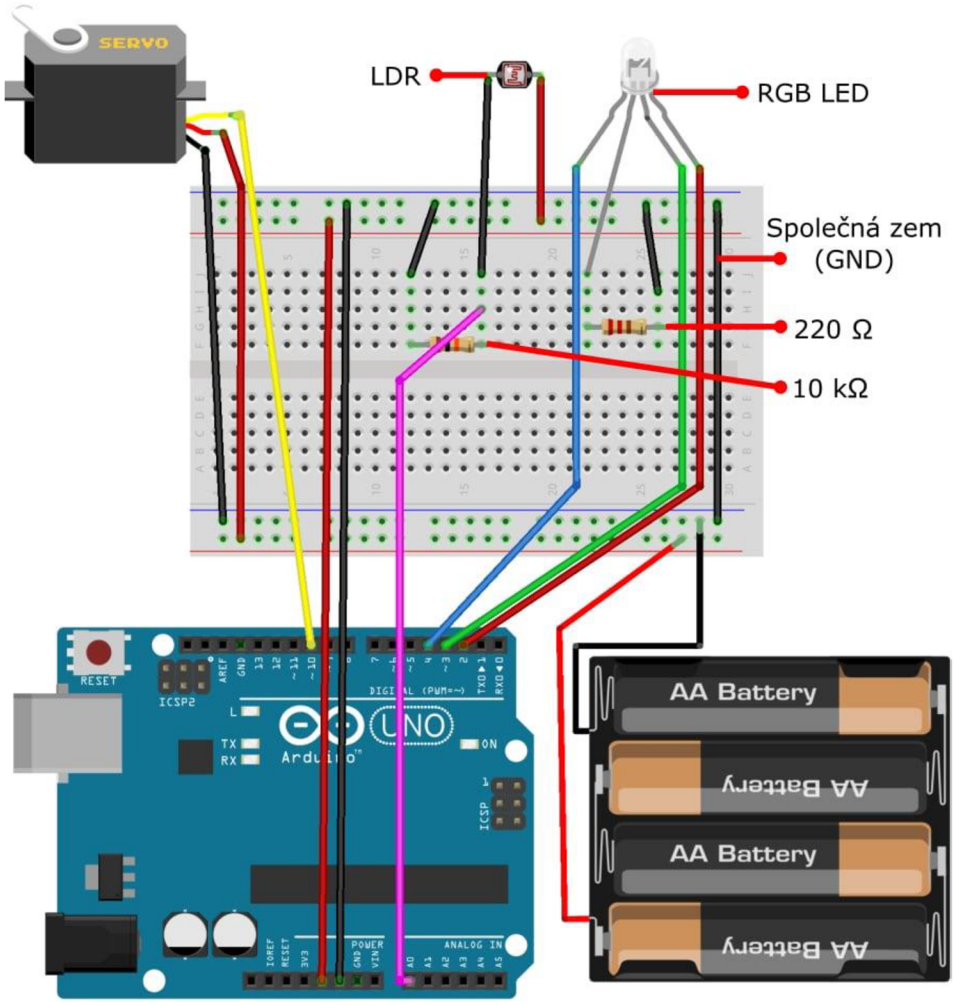
Obr. 12: Řízení úhlu natočení serva pomocí PWM [13]

Schéma zapojení

Obr. 13 poskytuje návod na zapojení součástí potřebných k vykonání úkolu. Jedná se o desku Arduino Uno (obr. 2), nepájivé kontaktní pole, konektorové kabely, čtyři AA baterie sériově spojené v pouzdře, tříbarevná LED se společnou katodou, fotorezistor GL5539 (LDR), uhlíkový rezistor RU 220R Ohm, rezistor RU 10k Ohm a analogové servo GO-13MG.

Zdroj elektrické energie pro motorek je z důvodu vyšších nároků serva na proud přiváděn na nepájivé pole bateriemi, které poskytují napětí 6 V. Řídící signál pro servo je veden kabelem napojeným na pin D10 Arduina. Důležité pro správnou funkčnost je vyrovnat uzemění z baterií a Arduina, což je realizováno společnou zemí (GND) na nepájivém poli jak je patrné z obr. 13.

Napětí 5 V je poskytováno z Arduina pro fotorezistor na jeho vstupní nožičku, výstupní je přes rezistor 10 kΩ spojena ke GND. Mezi výstupem LDR a rezistorem je připojen spoj (růžový kabel na obr. 13) pro vedení informace o velikosti napětí na fotorezistoru. Společná katoda RGB diody je spojena na rezistor 220 Ω a dále ke GND. Deska Arduina pomocí kabelu USB A-B komunikuje s programem v Matlabu [14].



Obr. 13: Schéma zapojení RGB senzoru a serva k [15]

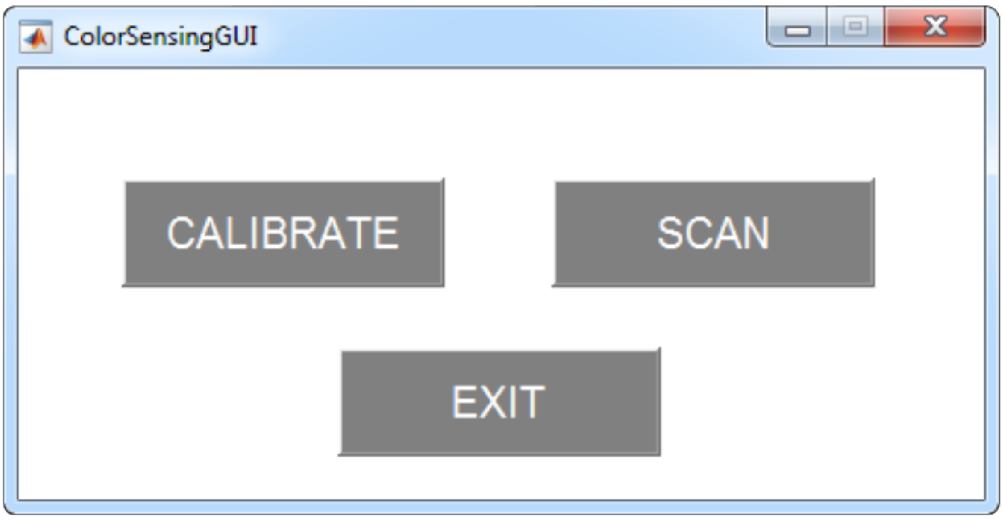
4.3 Implementace kódu

Program je psaný v M-file s názvem ColorSensing. Celý tento kód je k dispozici v příloze 1. Je strukturovaný tak, že využívá jednoduché GUI (Graphical User Interface) a čtyři vytvořené funkce, tj. `Read()`, `Calibrate()`, `CheckColor()`, `AssignColor()`. Pro správý běh RGB senzoru je potřeba tento před detekcí barevných kartiček kalibrovat, k tomu slouží právě zmíněné GUI.

Program se po spuštění připojí k Arduinu, nakonfiguruje potřebné vstupní a výstupní piny. Vytvoří čtyři trojsložkové vektory (**WHITE**, **BLACK**, **DIFF**, **COLOR**) pro ukládání informací o daných barvách. Každá složka určuje barvu hodnotou z rozmezí 0–255.

Následuje nekonečná smyčka, jenž spouší rozhraní (obr. 14), které dává uživateli na výběr ze tří možností. Toto je výchozí pozice sestaveného modelu a je zachycena na obr. 16. Pokud je stiknuto tlačítko EXIT, program se ukončí. Před skenováním kartiček se začíná kalibrací (CALIBRATE). Po stisknutí tlačítka se GUI zavře. Aby toto bylo provedeno, program volá dostupné funkce Matlabu

`delete(handles.figureColorSensingGUI)`, kde jako parametr se předává jmenovka (tag) okna GUI.



Obr. 14: Rozhraní ColorSensingGUI

Současně se předává informace k volání funkce `Calibrate()`, která přebírá jako parametr objekt Arduina `a`, vrací dva vektory, do kterých uloží informace o složkách bílé a černé barvy, které se skenovaly. Nejprve čte pomocí `Read()` postupně hodnoty třech složek bílé barvy, následuje pauza 5 sekund na výměnu bílé kartičky za černou. Stejně se snímají složky černé barvy. Předpoklad je takový, že po kalibraci by vektory měly mít následující parametry, tj. **WHITE** = [255, 255, 255] a **BLACK** = [0, 0, 0]. Z důvodu závislosti senzoru na intenzitě osvětlení v místnosti se toto nestává, je nutné barvy rozpoznávat relativně vůči naměřeným hodnotám složek bílé a černé, které pro ostatní barvy představují horní (hodnotou 255) a spodní (hodnotou 0) limit, o čemž bude pojednáno dále. Kalibraci zachycuje obr. 17.

Program tedy po kalibraci pokračuje zobrazením GUI, které pomocí tlačítka SCAN umožňuje detekovat barvu snímané kartičky. Spouští se funkce `CheckColor()`, která si v argumentu bere objekt Arduina a vrací vektor **COLOR**, do kterého uloží hodnoty složek detekované barvy. Tato se přepočte relativně vůči černé a bílé pomocí příkazu `COLOR = ((COLOR - BLACK) ./ DIFF) .* 255`, kde vektor **DIFF** byl získán příkazem `DIFF = WHITE - BLACK` [14]. Průběh tohoto je zdokumentován v obr. 18.

Funkce `AssignColor()` přebírá v argumentu vektor **COLOR**, podle jehož složek určí barvu a předá výsledek programu, který následně řídí pozici serva na správnou část ciferníku (obr. 19). Na obr. 15 je funkce `Read()`, kterou využívají funkce `Calibrate()` a `CheckColor()`. Jak lze z obrázku vyčíst, získává informace o napětí na právě osvětlovaném fotorezistoru jako průměr z deseti čtení. LDR je napájen 5V, výstup z něj může být v rozmezí téměř 0–5 V. Pro přepočítání na 8bitové rozmezí 0–255 je nutno čtené napětí vynásobit koeficientem `koef = 51`.

```
function avgR = Read(a)

INPUT = 'A0';
voltage = 0;
value = 0;
avgR = 0;
koef = 51;

for i = 1:1:10
    voltage = readVoltage(a, INPUT);
    % transform to 8-bit number 0-255
    value = koef * voltage;
    avgR = avgR + value;
    pause(0.1);
end

avgR = ceil (avgR / 10);
```

Obr. 15: Funkce `Read()`

4.4 Zhodnocení

Numerické RBG modely a jejich složky barev pro snímané kartičky jsou následující. ŽLUTÁ = [255, 255, 0], ČERVENÁ = [255, 0, 0], ZELENÁ = [0, 255, 0] a MODRÁ = [0, 0, 255]. Ovšem při detekci barvy kartiček byla čísla dosti odlišná od tohoto modelu. Proto bylo provedeno měření každé kartičky po 20 opakováních a podle limitních hodnot pro každou kartičku byly určeny hranice pro logiku použitou ve funkci `AssignColor()`. Výsledný přehled měření je shrnut v tab. 3.

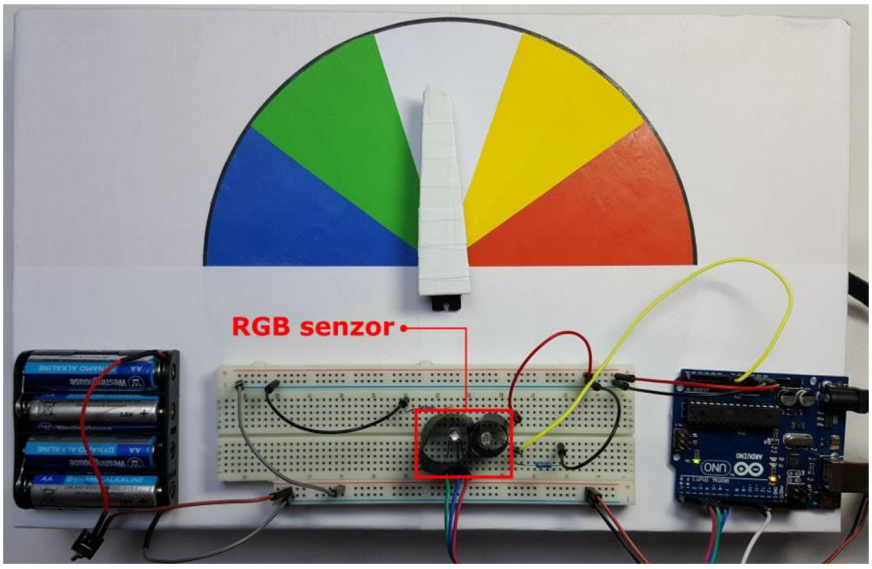
		R	G	B
Červená karta	**Min**	191	-4	-10
	Max	278	126	128
	Průměr	243	80	82
Zelená karta	**Min**	12	23	19
	Max	105	157	109
	Průměr	86	109	31
Modrá karta	**Min**	-41	-59	55
	Max	70	204	270
	Průměr	21	86	134
Žlutá karta	**Min**	236	203	128
	Max	314	391	476
	Průměr	265	279	293

Tab. 3: Výsledky opakovaného měření kartiček

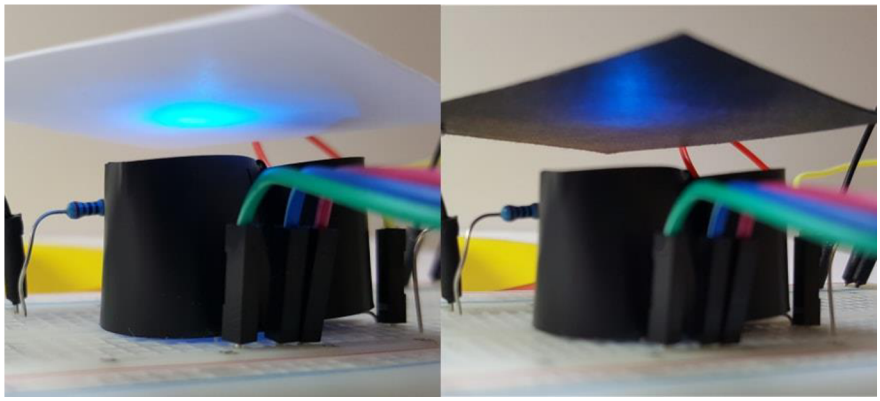
V souhrnu jde vidět, že hodnoty složek jsou větší než 255 nebo menší než 0. To je způsobeno tím, že daná barva vyslaná RGB diodou se odrazí od snímané kartičky více, než od kalibrované bílé (v případě vyšší hodnoty), nebo se právě odráží méně než od černé (v případě záporné hodnoty). Jedena z příčin nesladěnosti průměrných hodnot RGB složek z měření vůči modelovým hodnotám může být, že barvy použitých kartiček pravděpodobně neodpovídají modelovým barvám. Senzor při kalibraci s dosti velkou úspěšností zvládal rozpoznávat barvy čtyř karet, větší množství a celková různorodost by mohla při plnění úkolu dělat větší potíže se splněním. S ohledem na kvalitu použitých součástí, které tento senzor tvořily je i tak výsledek zčásti překvapivým. Celková finanční bilance součástí pro zhotovení tohoto úkolu je přehledně zachycena v tab. 4.

Součást	**Cena**
Arduino Uno	408
USB A-B kabel	110
Servo	229
Pouzdro pro baterie	30
Baterie	52
Nepájivé pole	144
RGB LED	30
Fotorezistor	3
2x rezistor	5
Přepínač	30
Propojovací kabely	40
Barevné papíry	30
Celkem	**1 111,– Kč**

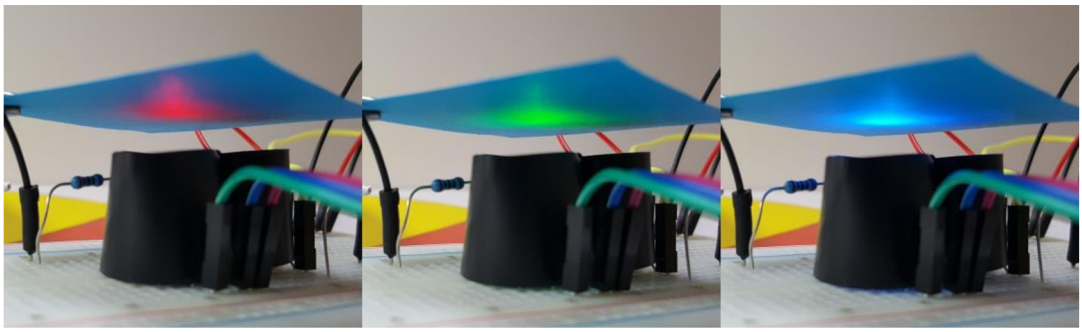
Tab. 4: Finanční bilance součástí úkolu k rozpoznání barev



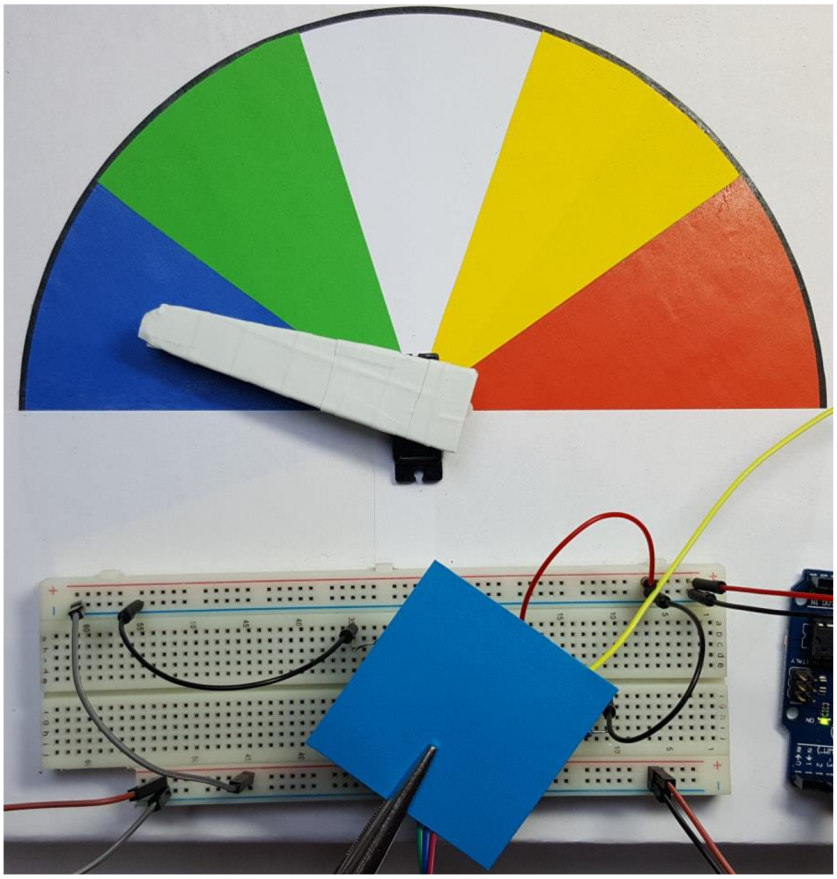
Obr. 16: Výchozí pozice



Obr. 17: Kalibrace



Obr. 18: Detekce kartičky



Obr. 19: Určení barvy snímané kartičky

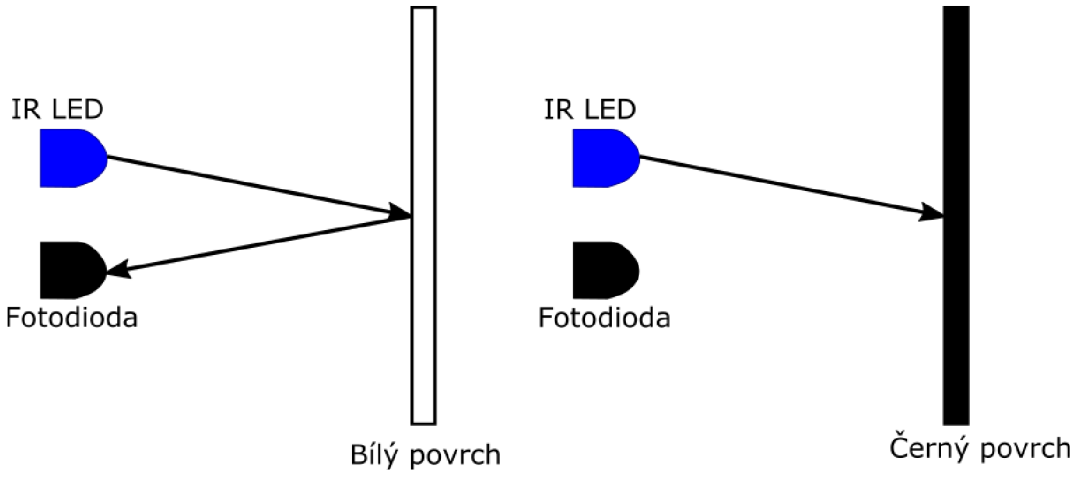
5 SLEDOVÁNÍ ČÁRY

Zadání úlohy: Zkonstruovat vozítko, které bude naprogramováno jako samostatná řídící jednotka a bude následovat čáru.

5.1 Rozbor

Sledovač (vozítko) pomocí senozoru rozpoznává přítomnost čáry. Ta je součástí podkladu pro pohyb vozítka. Jako tento slouží dřevotřísková deska s bílým povrchem (sololit). K tomu jako kontrast je vhodná černá čára, která se je vytvořena z izolační pásky. Pohyb je realizován dvěma koly, která jsou nasazena na hřídelky DC (Direct Current) motorů (stejnosměrných). Přední kolo je balanční. DC motor je napájený stejnosměrným proudem. Pro jeho řízení se využívá skutečnosti, že jeho otáčky jsou přímo úměrné napájecímu napětí.

IR (Infra Red) Senzor spolehlivě rozpoznává černou barvu od ostatních. Pomocí IR LED vysílá svazek záření, který se na bílém povrchu odrazí a dopadá zpět na detekční část senozoru (fotodiodu). Pokud je paprsek záření vyslán na černý povrch (čára), dojde k absorpci světla, na detekční část senzoru se světlo nedostane (obr. 20) [16].



Obr. 20: Princip IR senzoru

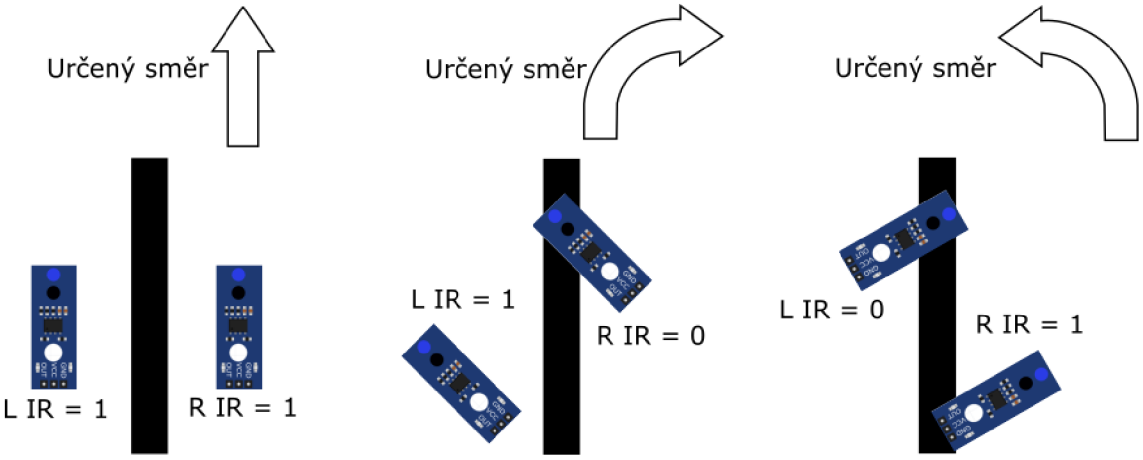
5.2 Návrh provedení

Aby mohlo Arduino fungovat jako samostatná řídící jednotka, kód je vytvořen jako model v Simulinku. Finální verze se zkompiluje a nahraje do paměti desky v modelu pomocí tlačítka Deploy to Hardware. Sledovač se pohybuje po sololitu, jakožto jeho dráze. Pro sledování trasy má vozítko dva IR senzory, které poskytují informaci o jeho poloze vůči čáře. Arduino podle polohy vyhodnotí směr, kterým se má vozítko ubírat v nadcházejícím kroku a pomocí shieldu k řízení DC motorů vysílá řídící signály.

Celá soustava je napájena baterií, vozítko sleduje čáru od zapnutí napájení Arduina bateriemi až do vypnutí.

IR senzor

Pro realizaci určení polohy jsou použity dva **TCRT5000 IR sensory** pro sledování čáry. Tento typ snímače pomocí výstupního napětí o hodnotě 0 (low) předává infrmaci o přítomnosti černé čáry, kdežto jinou čáru vyhodnotí jako hodnotu 1 (high). Provozní napětí snímače je 3,3 V nebo 5 V, detekční dosah 1–15 mm. Určení polohy vozítka vůči čáře a následném rozhodnutí o určení směru vyplývá z tab. 5 a obr. 21 [16].



Obr. 21: Určení směru jízdy

Levý senzor	Pravý senzor	Směr
0	0	Rovně
0	1	Doleva
1	0	Doprava
1	1	Rovně

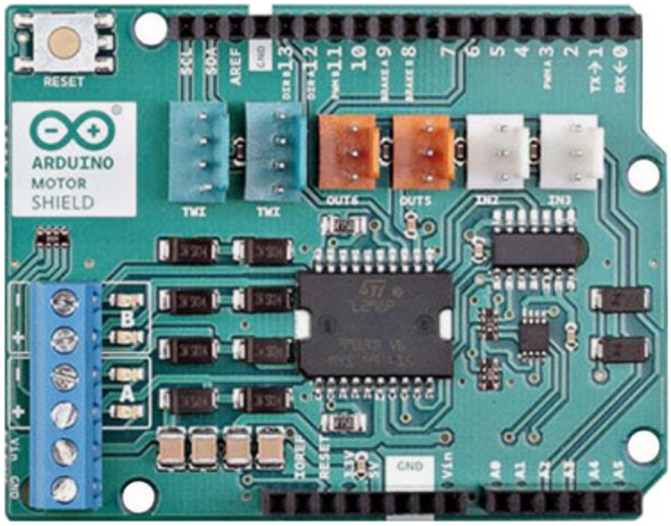
Tab. 5: Určení směru jízdy

DC motor

Dva stejnosměrné motorky jsou řízeny pomocí Arduino Motor Shieldu, který je popsán níže. Jsou zpřevodované 1:64, schopné vyvinout bez zátěže rychlost až 120 otáček za minutu. Provozní napětí motoru je 3–6 V.

Arduino Motor Shield

Je použita verze **Motor Shield R3** (obr. 22). Tento umožňuje ovládání směru a rychlosti dvou DC motorků nebo jednoho krokového motoru. Je založen na řídícím obvodu L298P, což je vysoce napěťový a vysoce proudový duální full-bridge driver. Umožňuje nezávislé řízení motorů. Provozní napětí shieldu je v rozmezí 5–12 V, maximální proud 2 A na jeden kanál, celkově tedy 4 A.



Obr. 22: Arduino Motor Shield R3

Shield je připevněn svými spodními vystupujícími piny do odpovídajících pinů vrchní strany desky Arduina. Musí být napájen externě (ne z USB), protože proudové nároky motorů přesahují maximální proud dodávaný z USB. Toto napájení může pocházet z AC/DC adaptéru připojeného do desky Arduina. Další možností je přivést napětí na svorkovnici do terminálů VIN a GND, což je varianta, která byla použita ke splnění tohoto úkolu. Pro řízení DC motorů má shield na svorkovnici dva kanály A a B, které využívají každý po čtyrech pinech desky Arduina. Díky tomu lze ovládat motory, důsledek tohoto je, že dané piny jsou tedy nedostupné k jinému použití. Funkce směr a brzda se řídí digitálně pomocí hodnot low a high. High značí směr dopředu a také zapnutí brzdy. Rychlost je ovládáná pomocí PWM signálu. Dané funkční vlastnosti těchto pinů jsou v tab. 6 [17].

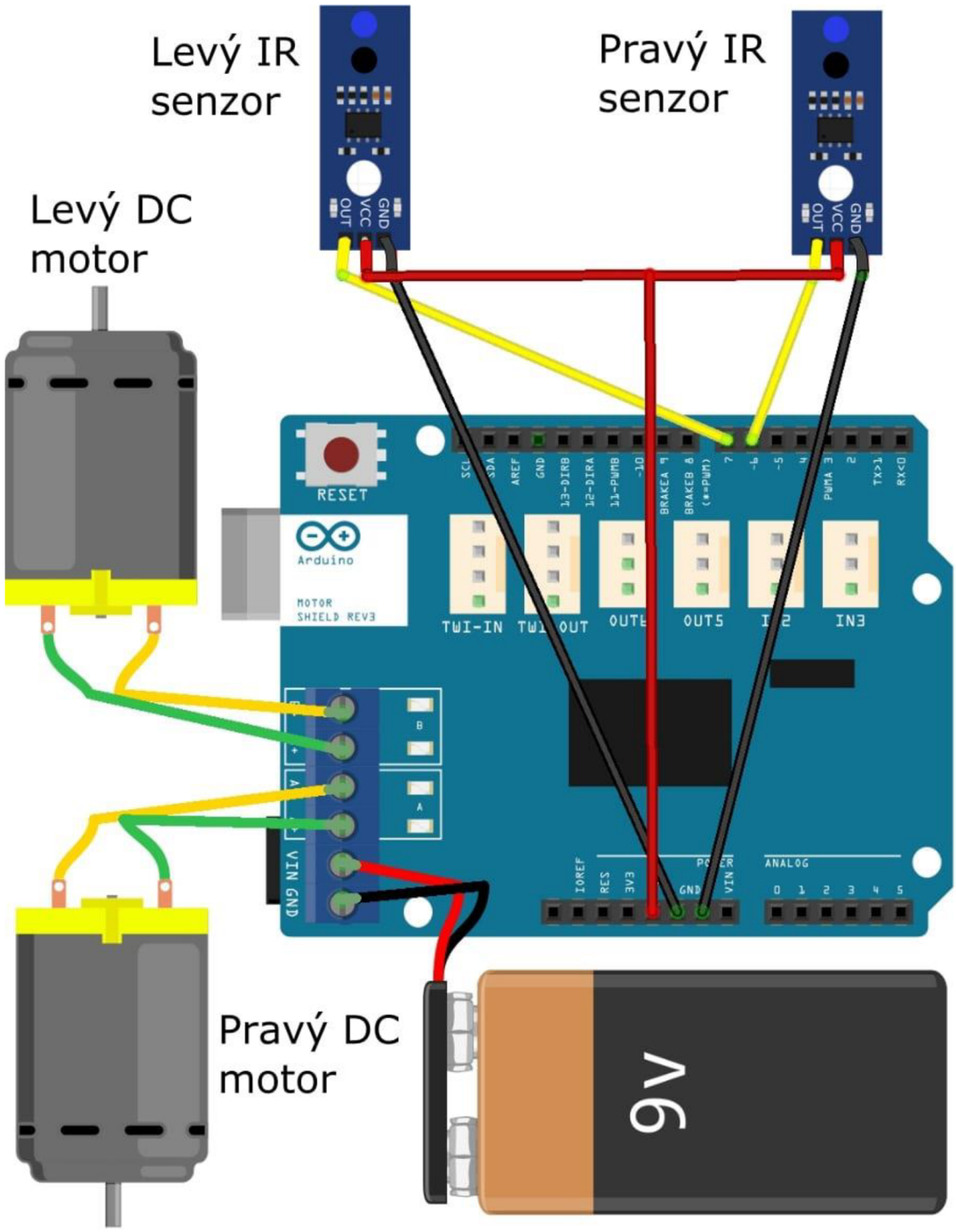
Funkce	Piny pro kanál A	Piny pro kanál B
Směr	D12	D13
Rychlost	D3	D11
Brzda	D9	D8
Snímání proudu	A0	A1

Tab. 6: Funkce pinů vstupujících do Motor Shieldu

Schéma zapojení

Součásti pro vytvoření obvodu z obr. 23 jsou: deska Arduino Uno (obr. 2), Motor Shield (obr. 22), konektorové kabely, 9V baterie v pouzdře, dva TCRT5000 IR senzory a dva DC motorky.

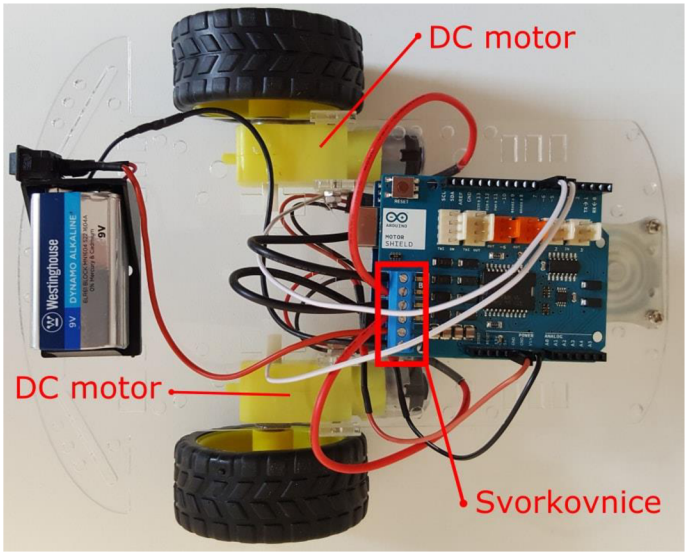
Elektrická energie je přiváděna z pinu 5V a GND k oběma IR senzorům. Ty poskytují informaci svými výstupy propojenými do digitálních vstupních pinů. Levý IR senzor pro pin D7, D6 pro pravý IR senzor. Napájení DC motorů je realizováno ze svorkovnice, kanál A je spojen k pravému motoru, B k levému motoru.



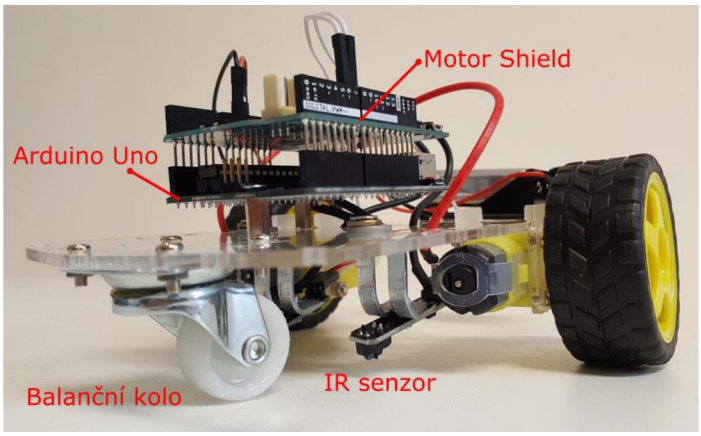
Obr. 23: Schéma zapojení vozítka pro sledování čáry [15]

Konstrukce

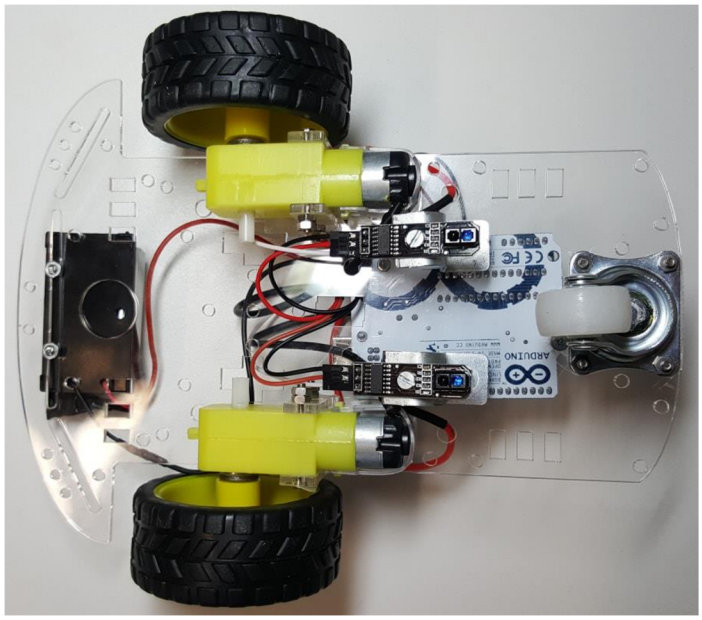
Základ tvoří podvozek z čirého plexiskla. Na jeho spodní stranu jsou připevněny DC motory, jenž na svých hřidelích mají gumová kola. V předu spodní strany podvozku je balanční kolo s ložiskem, které je menší, než gumová kola, tudíž povozek naklání vozítko dopředu. Ve spodní straně jsou ještě připevněny dva IR senzory. Horní strana podvozku má vzadu připěvněné pouzdro s 9V baterií a přepínačem. V přední části je na distančních sloupcích přišroubováno Arduino Uno, do kterého je zvrchu zasunut Motor Shield. Výsledek je na obr.24, 25 a 26.



Obr. 24: Vozítko horní pohled



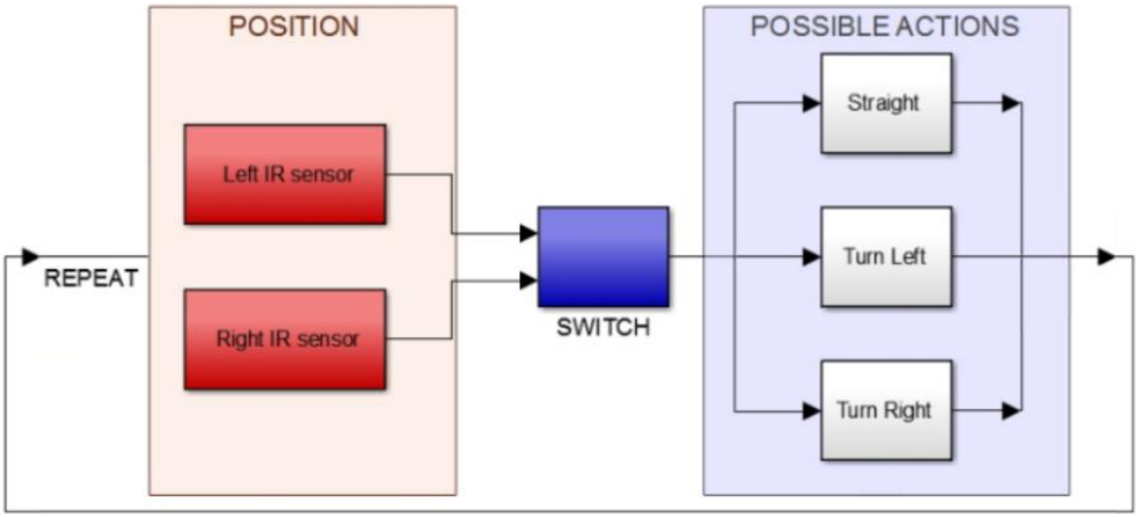
Obr. 25: Vozítko boční pohled



Obr. 26: Vozítko spodní pohled

5.3 Implementace kódu

Program je vytvořený jako model v Simulinku. Pro správnou funkčnost a nahrání zkompilovaného kódu do Arduina je potřeba provést následující kroky. Jak je uvedeno v úvodu strany 28 v kapitole 3.3 je nutnost nakonfigurovat typ desky. V horní části okna modelu je textové pole Simulation stop time, do kterého se zadává simulační doba programu v sekundách, v našem případě je vyplněna hodnota *inf* (nekonečná simulace opakující model ve smyčce). Vedle tohoto pole je výběrový seznam Simulation mode, pro nutnost kompilace a následné nahrání programu do desky Arduino jako samostatně výkonné jenotky je vybrán mód *Normal*.

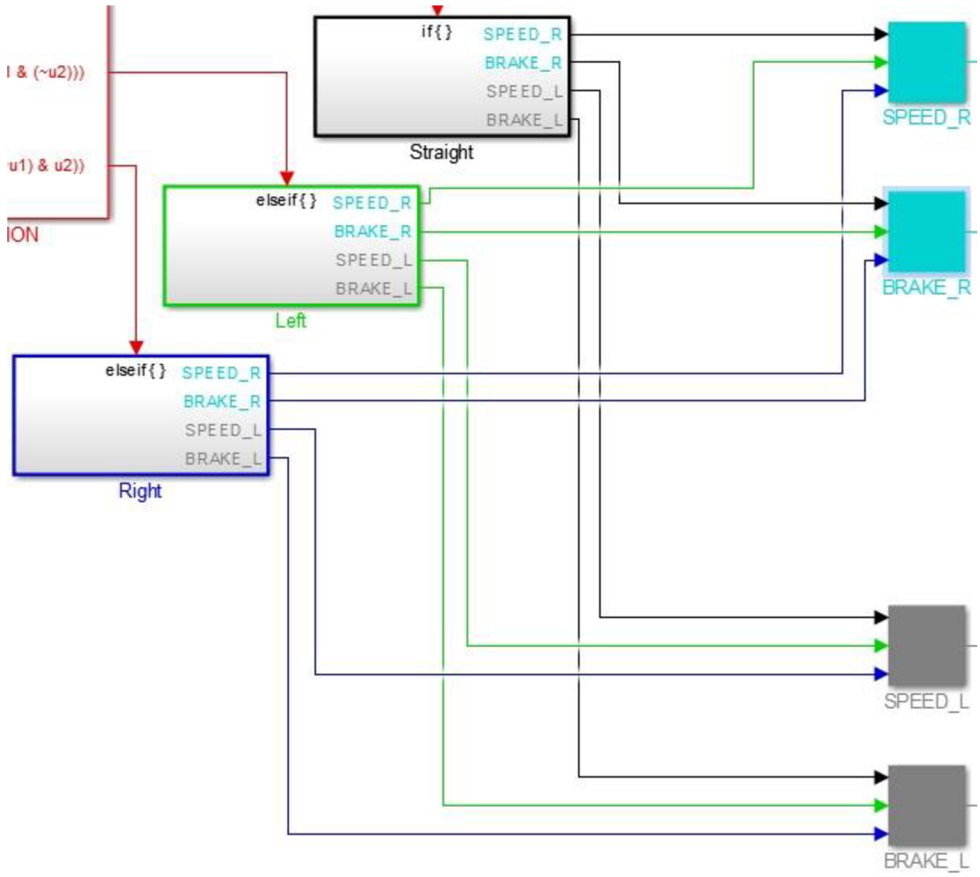


Obr. 27: Zjednodušené schéma modelu LineFollower

Program je k dispozici v příloze 2, zjednodušené schéma na obr. 27. Dva úvodní bloky jako digitální vstupy senozorů poskytují informaci o poloze vozítka vůči čáře. Jejich parametry jsou nastaveny jako pin D7 pro levý senzor a D6 pro pravý. U obou je vzorkovací frekvence zadána nejnižší možnou hodnotou a to 0,000001 s. Z každého bloku vede výstup (`u1` z pravého senzoru, `u2` z levého) do jednoho `POSITION` (typ If Block), který na základě polohy volá jednu ze tří řídících funkcí. Jak vyplývá z tab. 5, tohoto je dosaženo na základě implementace následující logiky do parametrů bloku `POSITION`. Funkce `Straight()` je volána, pokud je vyhodnocena první podmínka jako pravda: `(u1 & u2) | ((~u1) & (~u2))`. V případě pravdivého vyhodnocení druhé podmínky `(u1 & (~u2))` se volá funkce `Left()`. Logická pravda u třetí podmínky `((~u1) & u2)` volá funkci `Right()`.

Na blok `POSITION` navazují tři větve, které představují akční bloky `if{}` a dva `elseif{}`. Uvnitř těchto se volají výše zmíněné funkce. Motor shield potřebuje pro každý motor tři řídící signály (celkem tedy 6), jeden pro směr, druhý pro rychlost a třetí pro stav brzdy. Směr je u ubou motorů konstantní (rovně), tudíž je nutno aby každá jedna funkce předala dva signály pro každý motor, celkem mají tyto funkce čtyři výstupy. Signály z těchto výstupů vstupují do bloků určených pro piny na Arduinu. Aby se v jeden

diskrétní čas (vzorkování) považovaly za platné signály jen ty z právě použité funkce, je nutno odpovídající si výstupy ze všech tří funkcí platně rozlišit v blocích typu Merge. Tyto jsou na obr. 28 vidět jako zelené a šedé čtverce. Následně už putuje každý signál do patřičného bloku pro výstupní pin Arduina a simulace začíná od začátku do přerušení desky od napájení.

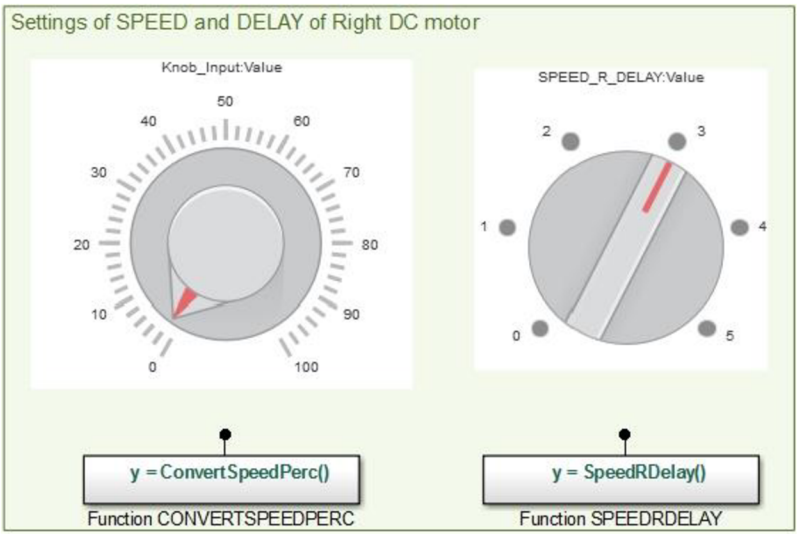


Obr. 28: Větvení a následné sjednocení signálů pro výstupy Arduina

5.4 Zhodnocení

Model je upravitelný vloženým uživatelským rozhraním, což jsou prvky z knihovny ve skupině Dashboard viditelné z obr. 29. Jedná se o nastavitelný kruhový ciferník (Knob) a rotační přepínač (Rotary Switch). Při testování vozítka bylo zjištěno, že levý motorek je při stejném řízení rychlostí mírně pomalejší, proto lze na pravém DC motoru vyvolat zpoždění z výběru hodnot 1–5, aby se zamezilo případu, že by vozítko nejelo rovně. Po vybrání některé z těchto hodnot na rotačním přepínači tento volá vytvořenou funkci `SpeedRDelay()`, která toto zpoždění předává do tří výše zmíněných funkcí. Taktéž druhý prvek, kruhový ciferník, po nastavení rychlosti motorů (rozmezí 0–100 odpovídá procentům) volá funkci `ConvertSpeedPerc()`, která je taktéž volána v již zmiňovaných funkcích nesoucí signály k řízení motorů. Další poznatek objevený při testování je, že rychlost vyjádřená v 8-bitové hodnotě od nuly po 77 není dostatečná

k rozběhnutí DC motoru. Jen při větších hodnotách se motory rozjedou. Proto je tato skutečnost taky zaimplementována do funkce `ConvertSpeedPerc()`, která hodnotu 1 z rotačního ciferníku převede pro PWM signál jako hodnotu 79, hodnotu 100 převede analogicky na hodnotu taky maximální a to 255. Postup je v zobrazen v příloze 2.

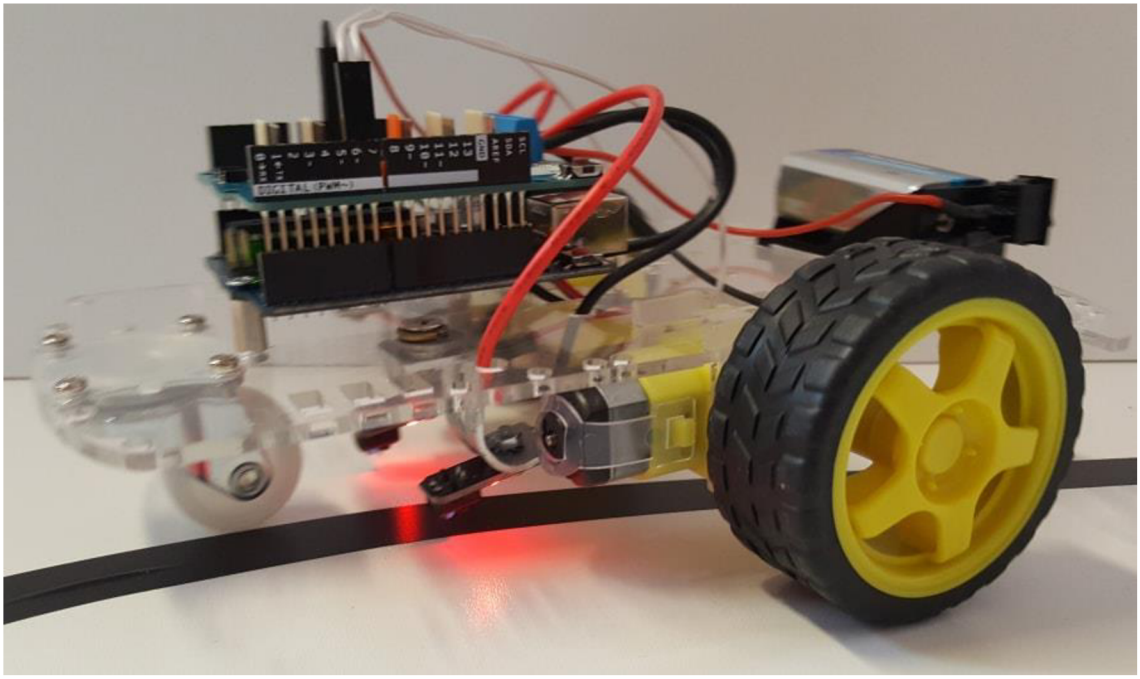


Obr. 29: Uživatelské rozhraní

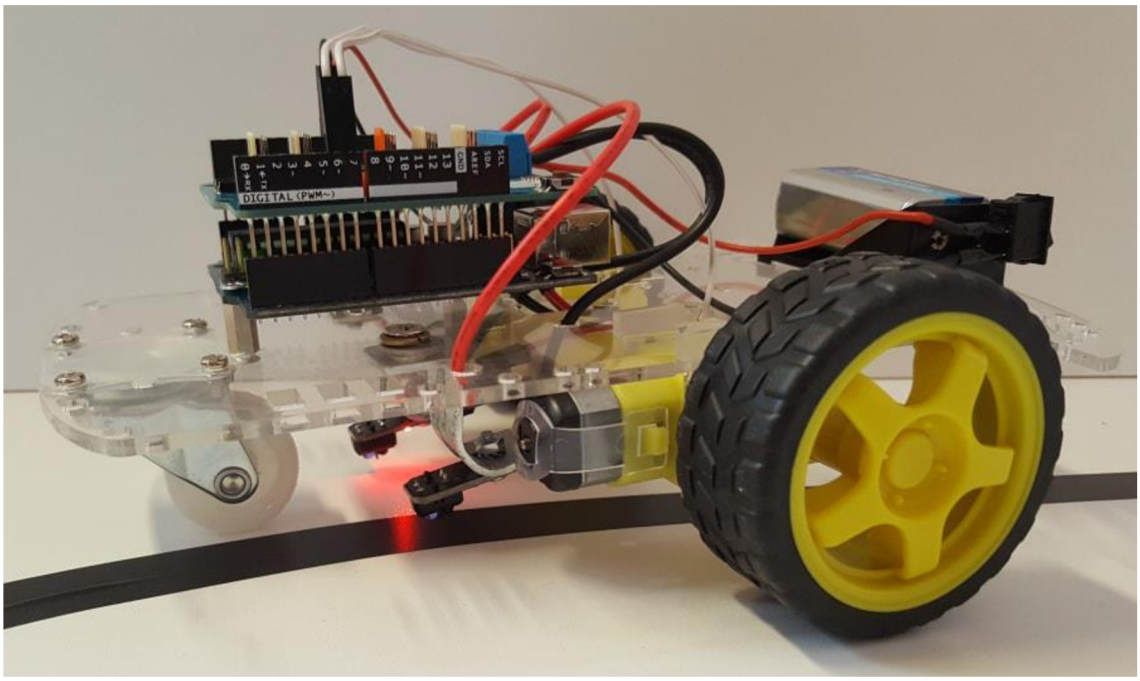
Při testování bylo dále zjištěno, že odezva ze snímačů není dostatečně rychlá na to, aby stíhala relativně vysokým otáčkám motorů a rychlosti vozítka. Z tohoto důvodu při ysších rychlostech vozítko v zatáčce nestíhá registrovat čáru a pokračuje rovně, i když by mělo zatáčet. Do jisté míry to lze korigovat větší šířkou sledované čáry, kdy při menších rychlostech se sledování daří. Sledovač je napájen 9V baterií, která ale v dosti omezeném čase zásoboje energií sledovač. Za zvážení jednoznačně stojí tento energetický zdroj nahradit kvalitnějším, leč dražším např. lipolovým akumulátorem. Na závěr je v tab. 7 vypsán finanční přehled nákladů komponent k zhotovení modelu pro tento úkol a také v obr. 30 a 31 je ukázka snímání polohy vozítka vůči čáře.

Součást	**Cena**
Arduino Uno	408
Arduino Motor Shield	690
Podvozek a 3 kola	220
2 DC motorky	300
2 IR senzory	180
9V baterie	50
Pozdro pro baterii	70
Přepínač	30
Propojovací kabely	30
Celkem	**1 978,– Kč**

Tab. 7: Finanční bilance součástí úkolu k sledování čáry



Obr. 30: Vozítko je v poloze uprostřed čáry, oba indikátory odrazu u senzorů svítí



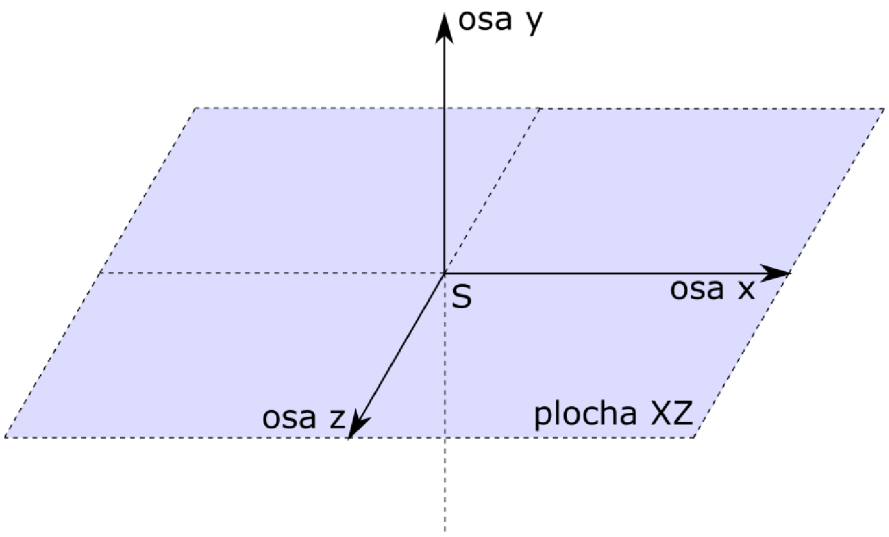
Obr. 31: Vozítko je v poloze napravo od čáry, indikátor odrazu svítí jen u pravého senzoru, levý je nad černou čárou (bez odrazu)

6 MANIPULÁTOR

Zadání úlohy: Bude zkonstruován manipulátor pomocí čtyř servo motorů. Ten ze tří startovních pozic přenáší bílý a černý předmět, podle barvy jej umístí na odpovídající výstupní pozici. Startovní pozice jsou vybírány pomocí uživatelského rozraní.

6.1 Rozbor

Servo motory manipulátoru plní funkci kloubů s možností svého natočení v rozsahu 0–180 stupňů. Polohu koncového bodu v prostoru určují tři serva, čtvrté je zodpovědné za otevírání, nebo zavírání koncové části (chapadla), která má funkci držet přemisťovaný předmět. Po umístění manipulátoru do trojrozměrného kartézského systému s osami *x, y, z* (obr. 32) je možné si manipulátor představit následovně. K pracovní ploše je připevněno první servo, které na své otočné části nese konstrukci. Otočnou částí prochází osa *y*. V důsledku je tedy úhel natočení prvního serva zodpovědný za úhel natočení manipulátoru v rovině plochy XZ.

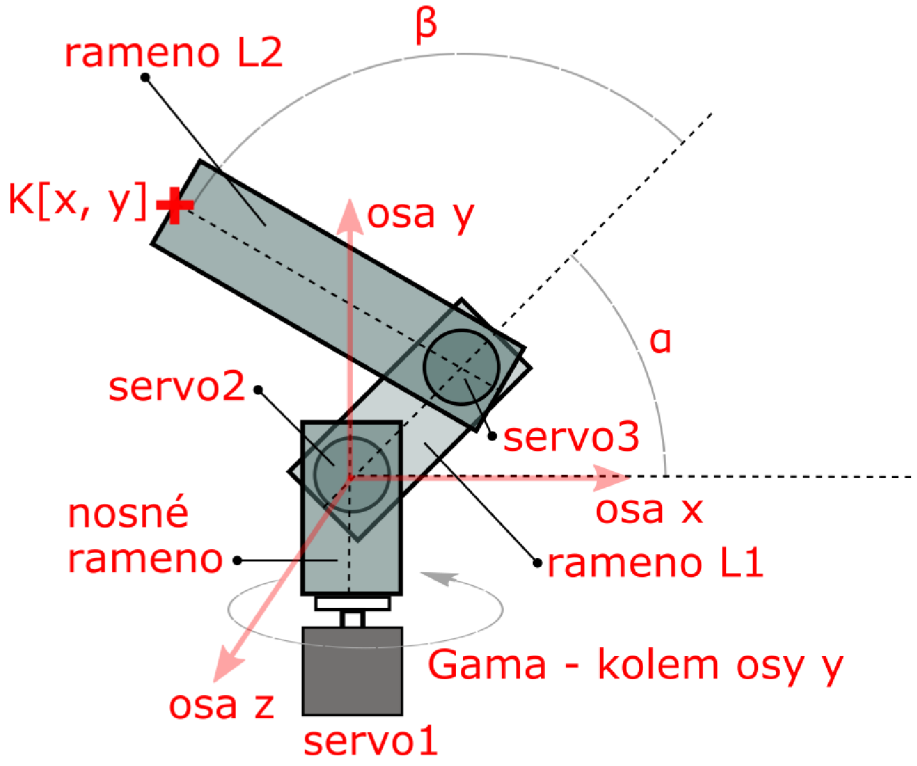


Obr. 32:

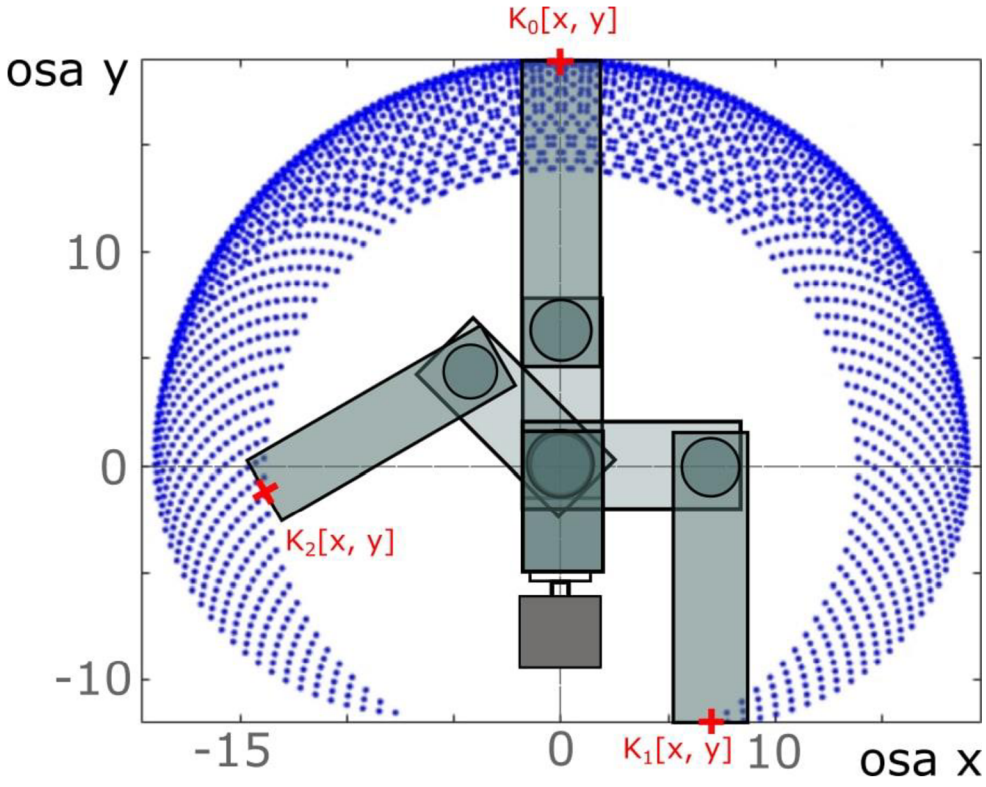
Za pozici koncové části v rovině XY tedy zodpovídají úhly natočení druhého a třetí serva, model je patrný z obr. 33. Konstrukce se skládá ze 3 ramen, kdy posun manipulátoru v rovině XZ umožňují svými pohyby ramena L1 a L2. První (nosné) rameno je umístěno kolmo k pracovní ploše (plocha XZ) a otáčí se jen kolem osy y. Tímto lze vysvětlit možné polohy manipulátoru v rovině XY (obr. 34). Pro získání všech možností v trojrozměrném prostoru se bere do úvahy úhel natočení v rovině XZ. Následující rovnice 1 a 2 vyjadřují získání souřadnic K[x, y] možných koncových bodů manipulátoru z obr. 34 [18].

$$x = L1 \cdot \cos(alfa) + L2 \cdot \cos(alfa + beta) \quad (1)$$

$$y = L1 \cdot \sin(alfa) + L2 \cdot \sin(alfa + beta) \quad (2)$$



Obr. 33: Model manipulátoru v rovině XY



Obr. 34: Možné koncové polohy manipulátoru v rovině XY

6.2 Návrh provedení

Výkonnými jednotkami manipulátoru jsou servo motorky, které jsou řízeny signály zprostředkované deskou Arduino Due. To má funkci vstupně/výstupní jednotky, komunikuje s programem běžícím v Matlabu přes USB kabel. Po spuštění programu se manipulátor řídícími signály nastaví do kolmé polohy k pracovní ploše. Spustí se uživatelské rozhraní, ve kterém následně uživatel vybírá ze tří pozic tu, na které je umístěn kvádr. Rozhraní se zavře a manupilátor si kvádr vyzvedne. Následuje přesun k senzoru. Ten pošle do programu přes Arduino informaci o barvě. Podle toho, jaká barva je zaznamenaná se manipulátor přesune na určené výstupní místo, kde kvádr uvolní a sám se znovu přemístí do kolmé polohy k pracovní ploše. Je tedy znovu připraven pomocí rozhraní (které se opět otvírá) reagovat na další zvolení vstupní pozice.

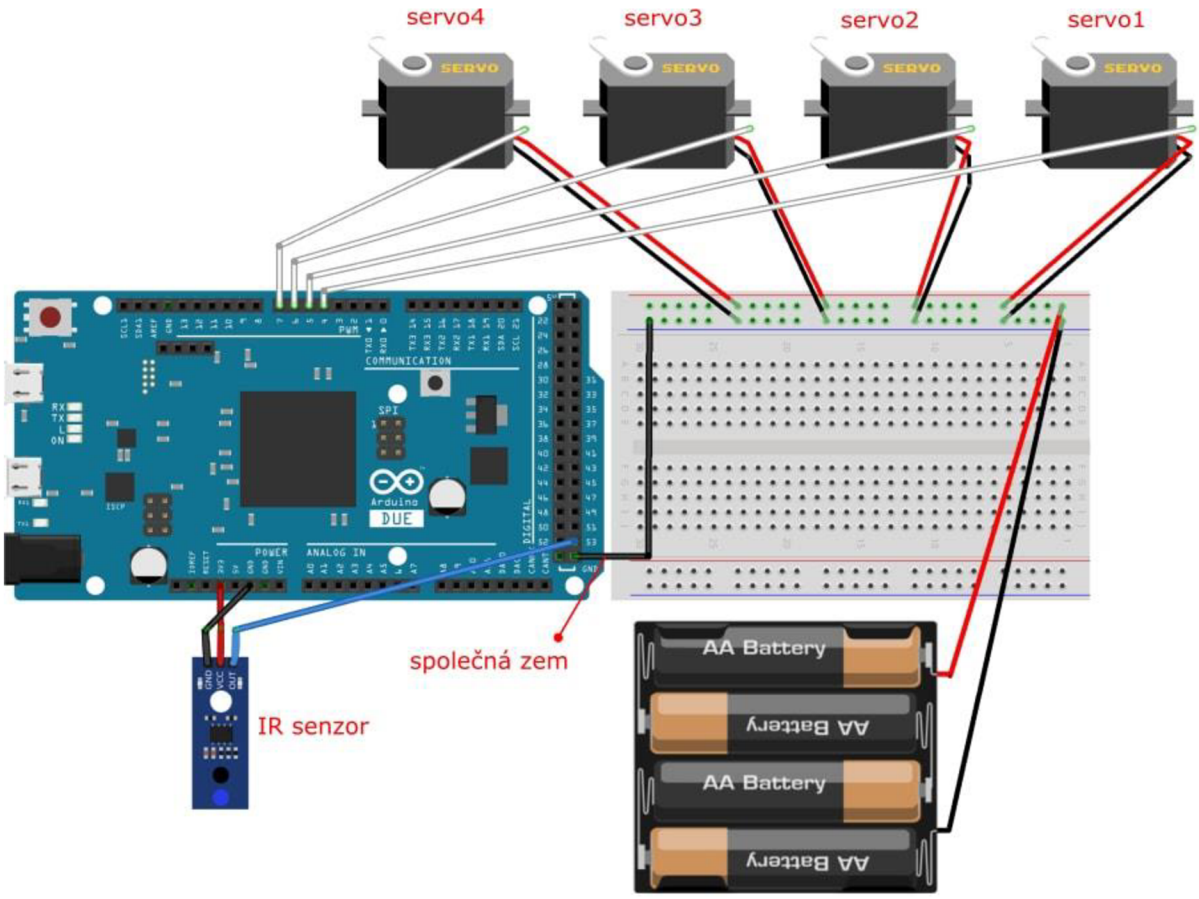
Serva

Jako základ je použito první servo v pořadí, a to standardní digitální **Hi-Torque Digital DS821**. Jeho hmotnost 45 g se neuplatňuje do zátěže manipulátoru na nároky tohoto serva a to díky konstrukci, kdy je toto servo připevněno nehybně k podstavci a teprve na své horní otočné části nese váhu manipulátoru. Při napájení 6 V dokáže vyvinout točivý moment o velikosti 6,3 kg.cm. Má nylonové převody. Další tři použitá serva jsou analogová micro **GO-09** vyznačující se váhou 9 g a při napájení 6 V točivým momentem 1,3 kg.cm. Tento typ má převody plastové. Všechna serva použitá v práci mohou být napájena pod napětím 4,8–6 V. Servo motory se vyznačují přesným polohováním a rychlostí přechodu z jedné polohy do druhé. Oba typy zachycuje obr. 35.

Obr. 35: Micro servo GO-09 a standardní servo DS821

Schéma zapojení

Elektrický obvod je tvořen servo motory, deskou Arduino Due, TCRT5000 IR senzorem, nepájivým polem a pouzdrem se čtyřmi 1,5V AA bateriemi. Schéma zobrazuje obr. 36.

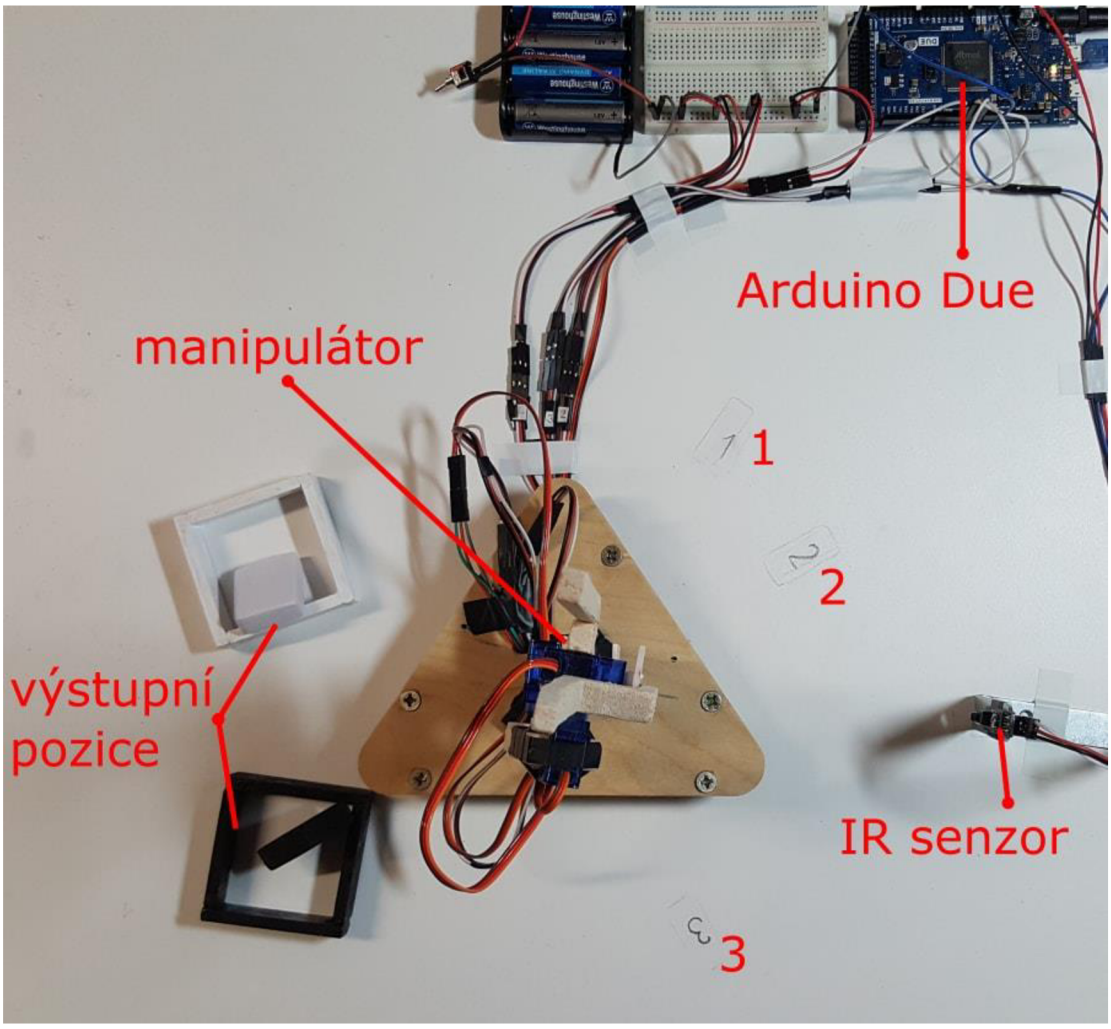


Obr. 36: Schéma zapojení manipulátoru [15]

Jelikož Arduino Due operuje na 3,3 V (viz popis Arduino Due v kapitole 2.1.1), je IR senzor k rozpoznání černého a bílého kvádru napájen z pinu 3.3V, tím poskytuje logickou 1 (high) pro pin D53 v hodnotě do 3,3 V, vyšší by mohla desku nebo přinejmenším daný pin poškodit. Arduino je přes USB kabel spojeno s PC, přes pin GND vytváří společné uzemění pro externí napájení servo motorků, obdobně jako v provedení úkolu v kapitole 4. Tady rovněž je zdroj elektrické energie z důvodu vyšších nároků serv na proud přiváděn na nepájivé pole bateriemi, které poskytují napětí 6 V. Řídící PWM signály pro serva jsou vedeny kabely pinů D4, D5, D6 a D7 postupně pro `servo1`, `servo2`, `servo3` a `servo4`.

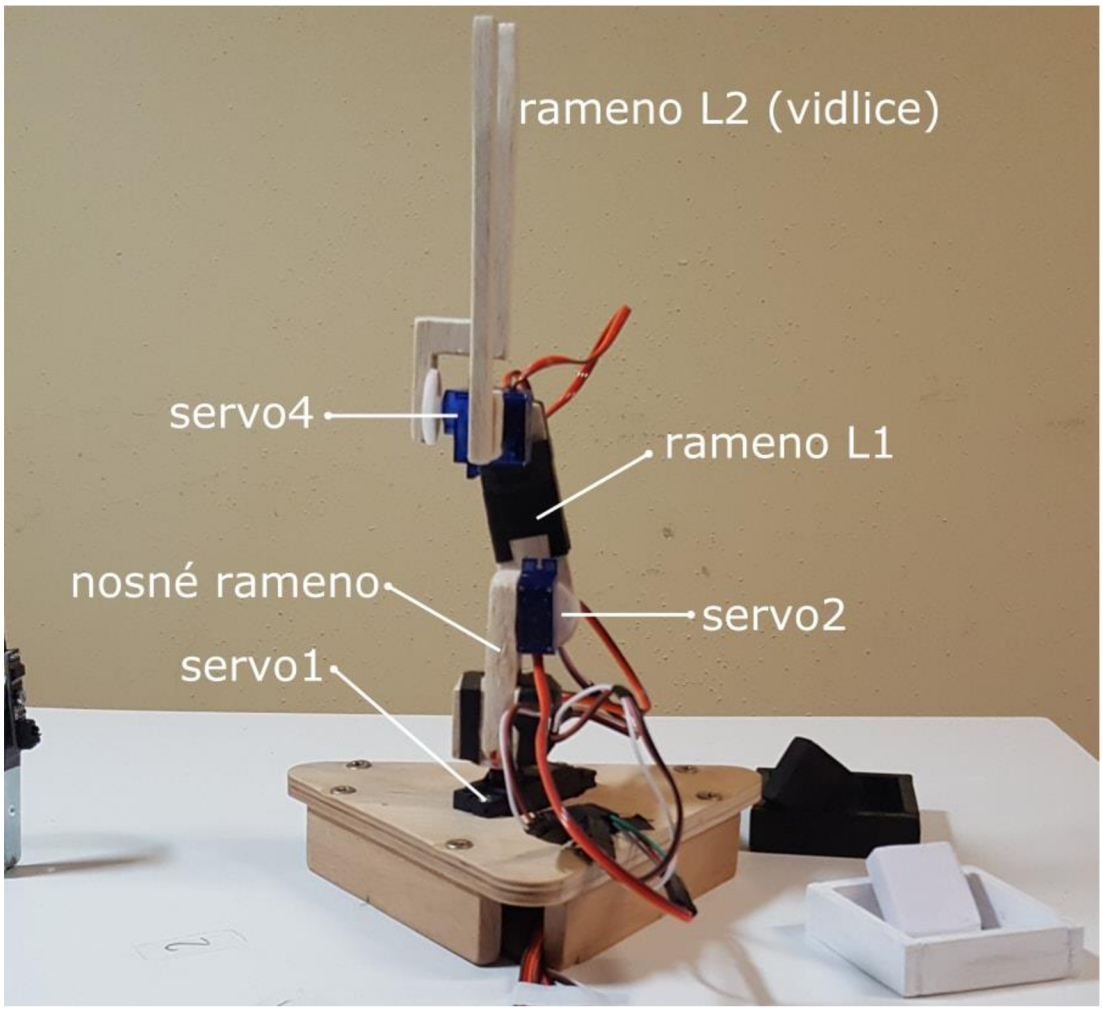
Konstrukce

Manipulátor je umístěn v centru pracovní plochy na vyvýšeném podstavci. Na pracovní ploše jsou označeny tři výchozí polohy, dvě koncová místa a připevněný IR senzor. Rozmístění ilustruje obr. 37.



Obr. 37: Vrchní pohled na pracovní plochu manipulátoru

Manipulátor je sestaven z balsového dřeva vyznačující se malou hustotou a s tím plynoucí malou hmotností, což byl jeden z důvodů k použití při tvorbě konstrukce. Celkem tři ramena z balsy jsou spojena do celku vytvářející manipulátor, jak je patrné z obr. 38. Základnu tvoří digitání servo, které je připevněno k podstavci, dále označováno jako servo1. Toto servo nese na své otočné části první nosné rameno délky 6 cm, uchyceno pomocí kyanoakrylátu (sekundové lepidlo). Na jeho horní část je stejně připevněno servo2, které představuje kloub k polohování druhého ramena o délce 7 cm, jenž je k tomuto servu uchyceno (výše v popisu modelu značeno jako rameno L1). Toto druhé rameno na své koncové části má zasazeno servo3, které podobně plní funkci kloubu a polohuje třetí rameno (L2). Třetí rameno je složeno ze dvou částí, které tvoří vidlici o délce 12 cm k uchopení předmětu. Tato vidlice začíná na čtvrtém servu, jenž je spojeno s otočnou částí třetího serva. Čtvrté servo polohuje otevření nebo uzavření vidlice.



Obr. 38: Popis částí manipulátoru

Použité analogové servo motory se vyznačují tím, že mají rozsah 0–180 stupňů, jejich výchozí poloha při připojení napájení je v 90 stupních. Proto je konstrukce sestavena tak, že rameno je v poloze kolmé k pracovní ploše. Dosah na pracovní ploše omezuje servo1, které je z výroby pevně naprogramované na rozsah 25–155 stupňů, kdy do krajních poloh 0 a 180 se nedostane. Toho lze u digitálních serv dosáhnout použitím speciálního programátoru.

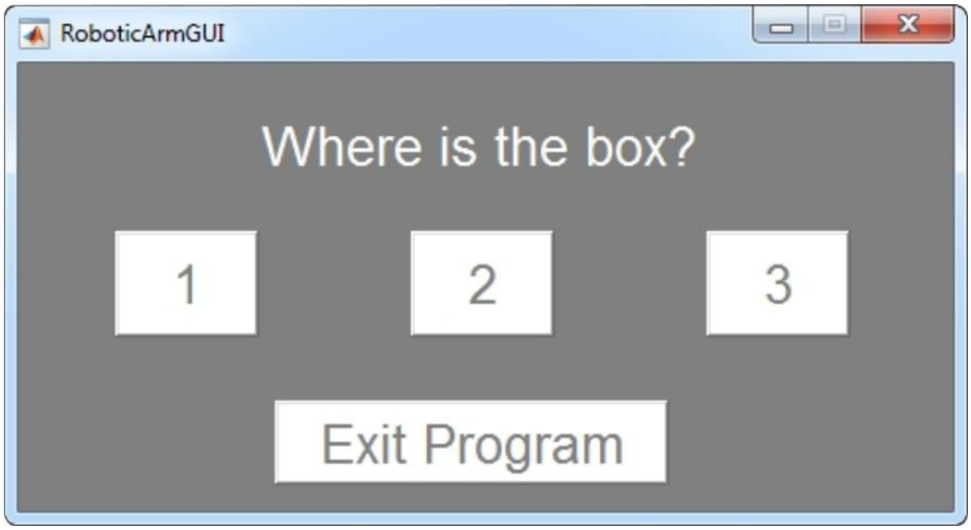
6.3 Implementace kódu

Jak už bylo uvedeno v kapitole 3.3 u popisků bloků pro řízení servo motorů, program využívající tyto bloky v modelu Simulinku není doporučený pro externí mód. Z tohoto důvodu je program psaný v Matlabu a celý je k dispozici v příloze 3. Využívá jednoduché GUI pro zvolení výchozí pozice a pět vytvořených funkcí.

Po spuštění si program v paměti vytváří čtyřsložkové vektory pro uložení hodnot natočení jednotlivých servomotorů. Tyto vektory slouží k uchování informace o odpovídající pozici manupulátoru při výkonu jednotlivých kroků. Jako příklad lze uvést pozici manipulátoru při přemístění k senzoru, kdy je tato pozice uložena právě v natočení

serv v příkazu `sensor = [0.50, 0.20, 0.88, 0.40]`. Dále postupující program se spojí s deskou Arduino, nakonfiguruje vstupní pin D53 pro senzor a výstupní piny pro PWM signály k řízení servo motorků. Tato konfigurace je uložena do vektoru následovně: `servo = [servo(a, 'D4'), servo(a, 'D5'), servo(a, 'D6'), servo(a, 'D7')]`.

Následně program vstupuje do nekonečné smyčky. Na jejím počátku se spustí jedoduché GUI (obr. 39), obdobně jako v úkolu kapitoly 4. Ze smyčky se lze dostat právě tlačítkem Exit Program. Po zvolení pozice uživatelem GUI zmizí a informace o výběru startovní pozice kvádru se přenáší do funkce `ToPosition()`, která přebírá jako první parametr vektor s konfigurací servo motorů a vektor s danou výchozí pozicí. Tato funkce volá čtyřikrát další funkci `ServoPosition()` pro nastavení polohy jednotlivých motorků. Manipulátor se nachází v pozici `ready = [0.50, 0.50, 0.50, 0.50]`. Například pro přesun k prvnímu výchozímu místu `position1 = [0.14, 0.14, 0.88, 0.40]` je třeba zvýšit natočení třetího serva směrem nahoru z 0,50 na 0,88, ale ostatní serva se musí natočit opačným způsobem. Tohoto funkce `ServoPosition()` docílí rozhodovacím procesem, kdy na základě aktuální polohy volá buď funkci `ServoPositionUp()` v případě nutnosti zvýšení hodnoty pro natočení motorku, nebo zavolá funkci `ServoPositionDown()` v případě opačném. K zjištění aktuální polohy serva se používá příkaz `position = readPosition(servo)`. Jakmile se manipulátor přesune k dané pozici a uchopí kvádr, dojde k zavolání funkce `PositionToSensor()`, která narozdíl od funkce `ToPosition()` nenastavuje natočení čtvrtého serva, jelikož toto je nyní v pozici, která musí držet vyzvednutý kvádr. U senzoru se čte informace o barvě předmětu a na základě výstupní informace je volána funkce `ToPosition()` s odpovídajícím druhým parametrem jakožto výstupní pozicí. Po přesunu manipulátoru na tuto pozici zde dojde k uvolnění předmětu. Za dalšího volání funkce `ToPosition()` se manipulátor přesunuje do své výchozí polohy a smyčka začíná znovu voláním GUI.



Obr. 39: Rozhraní RoboticArmGUI

6.4 Zhodnocení

Při postupném konstruování bylo vytvořeno několik verzí manipulátoru, kdy konečná verze je funkční a zdokumentovaná na vložených obrázcích. Původně měl manipulátor místo vidlice třetí rameno se čtvrtým servo motorem na své koncové části, kde servo ovládalo relativně těžké chapadlo z akrylových materiálů. Toto v koncečném důsledku přetížilo první a druhé serva natolik, že nebyly schopné reakce na řídící signály a neustále hledaly požadovanou polohu, manipulátor se stával neovladatelným. Toto chapadlo se tedy nahradilo vydlicí z balsy a servo které řídí její otevírací úhel je umístěno na začátek tohoto třetího ramena. Toto odlehčení ještě problém nevyřešilo a původní analogové servo GO-13MG s kovovými převody na pozici prvního serva bylo nahrazeno zmíněným standardním digitálním **Hi-Torque Digital DS821**. Díky tomuhle se konstrukce při pohybu stala jistější a umožnila tím funkční splnění úkolu.

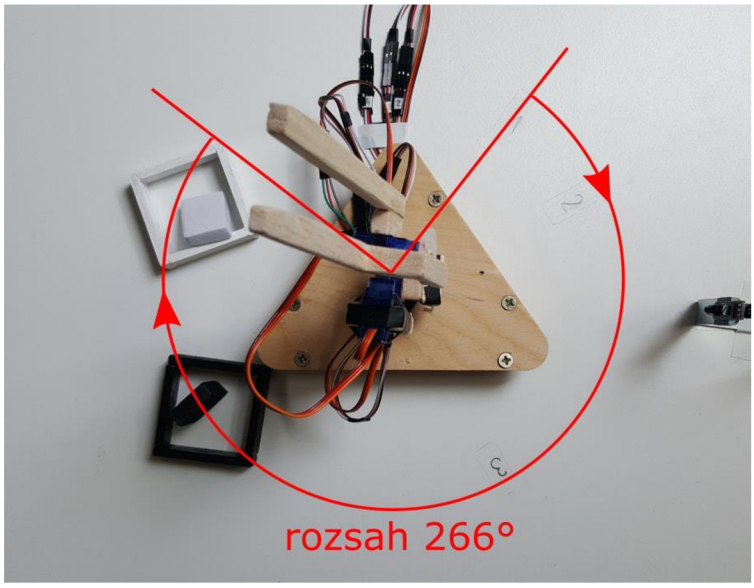
Vlastností této konstrukce je omezení hlavní přednosti servo motorů, tj. rychlosti a přesnosti polohování. Kostra z balsy způsobuje po delším provádění úkonů, že koncová pozice manipulátoru při stejných hodnotách natočení serv se nepatrně může změnit, což zapříčiňuje nutnost kalibrace používaných pozic a jejich upravení v kódu programu. Další vlastností je, že při využití plné rychlosti polohování serv se manipulátor značně chvěje a je nutnost tuto rychlost polohování snížit a docílit plynulost přechodu mezi polohami. Toto je zaimplementováno ve funkcích `ServoPositionUp()` a `ServoPositionDown()`. Jak je vidět z obr. 40 je toho dosaženo rozložením pohybu na mezikroky, kdy každý posun o jednu setinu natočení z rozmezí 0,00–1,00 je přerušen pauzou 0,01 sekundy.

ServoPositionUp.m | RoboticArm.m | ToPosition.m | PositionToSensor.

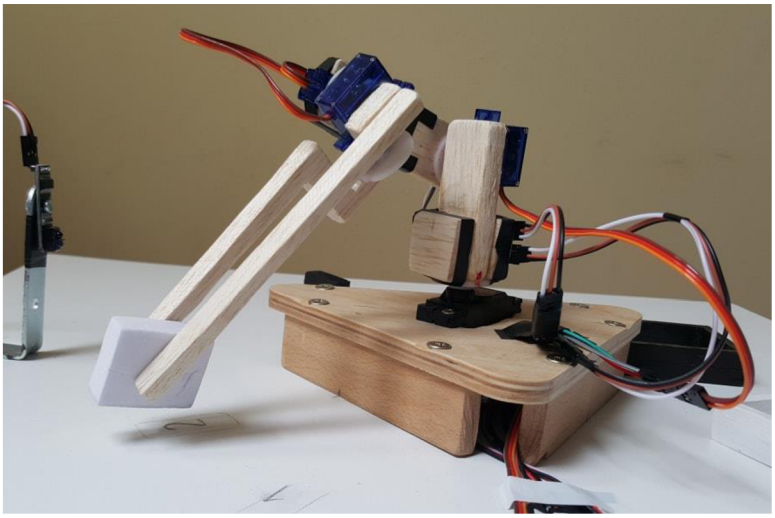
```
1   function ServoPositionUp(servo, oldPosition, newPosition)
2
3   for angle = oldPosition : 0.01 : newPosition
4       writePosition(servo, angle);
5       pause(0.01);
6   end
7   pause(1.5);
```

Obr. 40: Implementace funkce ServoPositionUp

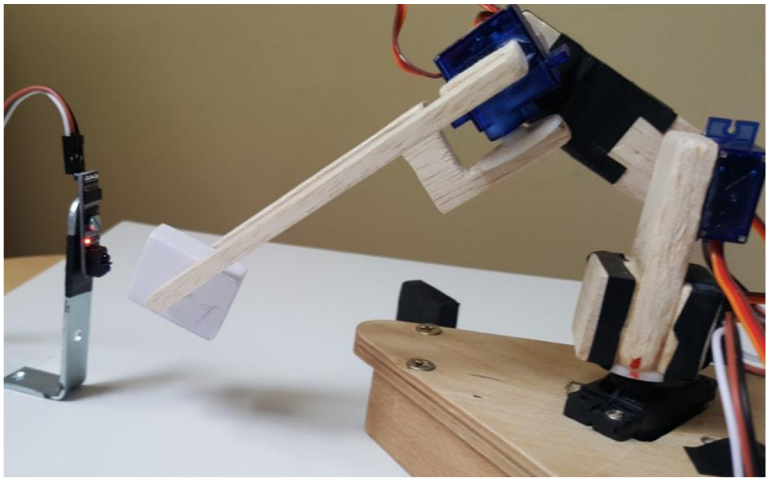
Jak už bylo zmíněno první servo pracuje v rozsahu 133 stupňů, konkrétně při implementaci v Matlabu narozdíl od ostatních použitých serv nereaguje na polohování pozic od 0,00 do 0,13 při spodní hranici a od 0,89 do 1,00 v horní hranici, což se snaží ilustrovat obr. 41. Obrázky 42, 43 a 44 dokumentují postupně vyzvednutí předmětu, rozpoznání jeho barvy u senzoru a doručení na výstupní místo. Součásti byly použity i s vědomím výše popsaných vlastností a to z důvodu dostupnosti možných prostředků. Finanční bilanci pro provedení úkolu zachycuje tab 7.



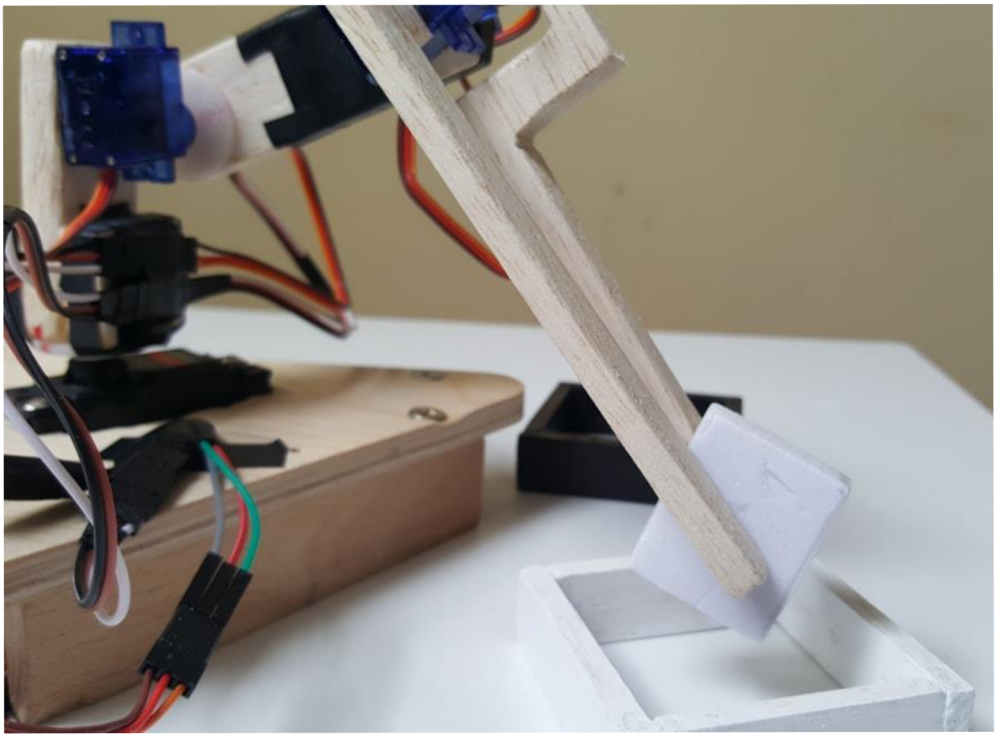
Obr. 41: Pracovní rozsah manipulátoru



Obr. 42: Vyzvednutí kvádru



Obr. 43: Rozpoznání barvy



Obr. 44: Umístění do výstupní pozice

Součást	**Cena**
Arduino Due	920
Digitální servo	370
3 analogová serva	375
IR senzor	90
Kostra z balsy	20
Dřevo na podstavec a ohrádky pro výstupní pozice	50
Pracovní plocha ze sololitu	80
Pouzdro pro baterie	30
Baterie	52
Propojovací kabely	114
Nepájivé pole	60
Spojovací materiál	140
Přepínač	30
Celkem	**2 331,– Kč**

Tab. 8: Finanční bilance součástí úkolu manipulátoru

7 ZÁVĚR

Autor práce se seznámil s platformou Arduino, jejími možnostmi pro fyzické výpočty a problematikou jejího programování. Prohloubil si také znalosti z programování této platformy v prostředí Matlab a jeho knihovny Simulink. Byly vytvořeny tři praktické úkoly, pro které jsou zkonstruovány a naprogramovány funkční prototypy. V prvním praktickém úkolu byl sestaven senzor pro rozpoznání barevných kartiček a spolu s deskou Arduino Uno a servo motorkem tvoří celek, na němž lze demonstrovat fungování platoformy. Druhý úkol sledování čáry umožnil využít přednost výpočetního standardu Matlab a jeho knihovny Simulink, kdy řízení vozítka a jeho dvou DC motorků lze intuitivně pomocí funkčních bloků modelovat. Ukázala se tady také možnost využít rozšiřující modul (shield) platfromy Arduino, kdy tento shield zajišťuje pohodlné programování a napájení motorů. V třetím úkolu po neúspěšných verzích konstrukce a časově náročných pokusech s původní konstrukcí manipulátoru se podařilo jeho kritické části odlehčit a jeho pohyb uvést do plynulého chodu, což vzhledem k dostupným prostředkům autor této práce považuje za podstatný úspěch. Práce autora obohatila a inspirovala v oboru aplikace mikrokontrolérů. Autor věří, že poskytuje návod, nebo recenzi pro čtenáře, kteří by se chtěli v oboru s touto problematikou seznámit.

8 SEZNAM POUŽITÉ LITERATURY

[1] *Arduino* [online]. 2017 [cit. 2017-05-24]. Dostupné z: https://www.arduino.cc/.

[2] KUSHNER, David. The Making of Arduino: How five friends engineered a small circuit board that's taking the DIY world by storm. *IEEE Spectrum* [online]. 2011 [cit. 2017-05-24]. Dostupné z: http://spectrum.ieee.org/geek-life/hands-on/the-making-of-arduino.

[3] VODA, Zbyšek a tým HW Kitchen. *Průvodce světem Arduina*. Bučovice: Martin Stříž, 2015. ISBN 978-80-87106-90-7.

[4] ARDUINO TEAM. *Two Arduinos become one* [online]. 2016 [cit. 2017-05-24]. Dostupné z: https://blog.arduino.cc/2016/10/01/two-arduinos-become-one-2/.

[6] *Arduino - Open Source Products for Electronic Projects* [online]. 2017 [cit. 2017-05-24]. Dostupné z: http://www.arduino.org/.

[7] MATLAB Support Package for Arduino Hardware. *MathWorks* [online]. Natick (Massachusetts), 2017 [cit. 2017-05-24]. Dostupné z: https://www.mathworks.com/help/supportpkg/arduinoio/index.html.

[8] Set shaft position of standard servo motor. *MathWorks* [online]. Natick (Massachusetts), 2017 [cit. 2017-05-24]. Dostupné z: https://www.mathworks.com/help/supportpkg/arduino/ref/standardservowrite.html.

[9] Simulink Support Package for Arduino Hardware. *MathWorks* [online]. Natick (Massachusetts), 2017 [cit. 2017-05-24]. Dostupné z: https://www.mathworks.com/help/supportpkg/arduino/.

[10] KONSTANTINOS N. PLATANIOTIS a ANASTASIOS N. VENETSANOPOULOS. *Color Image Processing and Applications*. Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN 978-366-2041-864.

[11] EVANS, Brian. *Beginning Arduino programming*. New York: Apress, 2011. ISBN 978-143-0237-785.

[12] RGB color model. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2017 [cit. 2017-05-24]. Dostupné z: https://en.wikipedia.org/wiki/RGB_color_model.

[13] Pulse Width Modulation. In: *Arduino* [online]. [cit. 2017-05-24]. Dostupné z: https://www.arduino.cc/en/Tutorial/PWM.

[14] FJORDCARVER. Using an RGB LED to Detect Colours. In: *Instructables* [online]. [cit. 2017-05-24]. Dostupné z: http://www.instructables.com/id/Using-an-RGB-LED-to-Detect-Colours/.

[15] Vytvořeno v software *Fritzing* [online]. 2017 [cit. 2017-05-24]. Dostupné z: http://fritzing.org.

[16] WARREN, John-David., Josh. ADAMS a Harald MOLLE. *Arduino robotics*. New York, NY: Apress, c2011. Technology in action series. ISBN 978-143-0231-844.

[17] Arduino Motor Shield. *Arduino* [online]. 2017 [cit. 2017-05-24]. Dostupné z: https://www.arduino.cc/en/Main/ArduinoMotorShieldR3.

[18] MATOUŠEK, Radomil. *Informatika - 1IN (Matlab): Cvičení s počítačovou podporou* [online]. Brno, 2012 [cit. 2017-05-24]. Dostupné z: https://mvm.fme.vutbr.cz/download.php.

9 SEZNAM ZKRATEK

AC (Alternating Current)/DC (Direct Current) – střídavý a stejnosměrný proud

CC (Creative Commons) – organizace, která si za cíl klade rozšířit množství autorských děl dostupných veřejnosti k legálnímu využívání a sdílení

DAC (Digital to Analog Converter) – digitálně analogový převodník

EEPROM (Electrically Erasable Programmable Read-Only Memory) – elektronicky vymazatelná paměť pouze pro čtení typu ROM-RAM

ICSP (In-Circuit Serial Programming) – schopnost jednočipů být programovány bez nutnosti jejich vyjmutí a vložení do zvláštního, k tomuto účelu sloužícímu zařízení

IDE (Integrated Development Environment) – vývojové prostředí

I/O (Input/Output) – vstup/výstup

LDR (Light-Dependent Resistor) – fotorezistor

LED (Light Emitting Diode) – polovodičová elektronická součástka, jejíž vlastností je schopnost vyzařovat světlo

LPGL (GNU Lesser General Public License) – licence svobodného softwaru, publikovaná Free Software Foundation

PWM (Pulse Width Modulation) – puslně šířková modulace

RX (receive) – příjmač, TX (transmit) – odesílatel, přenášet

SPI (Serial Peripheral Interface) – sériové periferní rozhraní

SRAM (Static Random Access Memory) – polovodičová pamět typu RAM realizovaná bistabilním klopným obvodem

TTL (Transistor–transistor logic) – tranzistorově-tranzistorová logika je standardem používaným pro implementaci digitálních integrovaných obvodů

10 SEZNAM PŘÍLOH

Příloha 1: Kód pro úkol Rozpoznání barev
Příloha 2: Model pro úkol Sledování čáry
Příloha 3: Kód pro úkol Manipulátor
Příloha 4: CD (video pro Sledování čáry, video pro Manipulátor, elektronická verze bakalářské práce, zdrojové soubory pro provedení programů)

PŘÍLOHY

Příloha 1: Kód pro úkol Rozpoznání barev

ColorSensing.m

```
clear all;clc;

% connect to Arduino board
a = arduino();

% definition of outputs and intputs
RED = 'D2';
GREEN = 'D3';
BLUE = 'D4';
PWM = 'D10';
INPUT = 'A0';

% vectors for measured data
WHITE = [0, 0, 0];
BLACK = [0, 0, 0];
DIFF = [0, 0, 0];
COLOR = [0, 0, 0];

Cali = 0;
Scan = 0;
Exit = 0;

configurePin(a, RED, 'DigitalOutput');
configurePin(a, GREEN, 'DigitalOutput');
configurePin(a, BLUE, 'DigitalOutput');
configurePin(a, INPUT, 'AnalogInput');

servo1 = servo(a, PWM);

% set servo to default position
writePosition(servo1, 0.50);

while ~Exit

    % show GUI
    ColorSensingGUI;
    uiwait(ColorSensingGUI);

    % EXIT program
    if Exit
       break;
    end
```

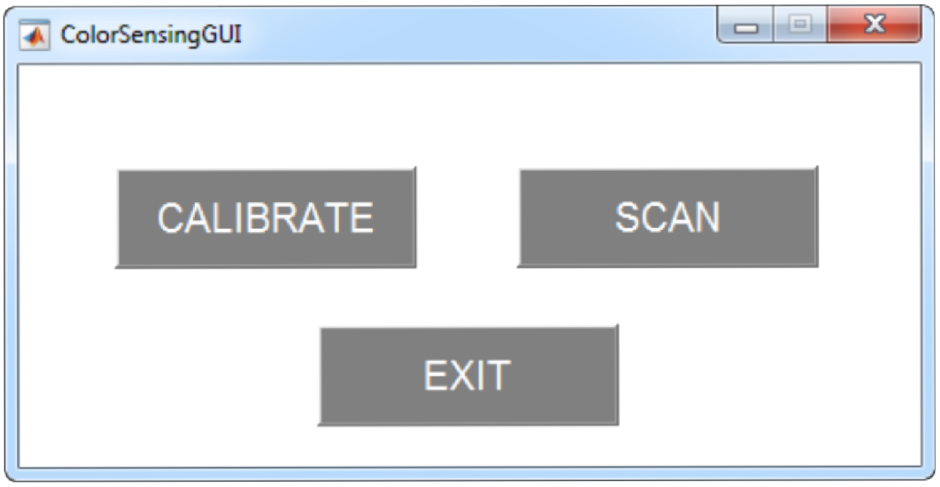
```
    % if selected, Calibrate sensor to WHITE (upper limit)
    % and BLACK (lower limit) color
    if Cali == 1
       [WHITE, BLACK] = Calibrate(a);
       DIFF = WHITE - BLACK;
    end %end if

    % if selected, Scan COLOR
    if Scan == 1
       COLOR = CheckColor(a);
       COLOR = ((COLOR - BLACK) ./ DIFF) .* 255 ;
       % round up to integer
       COLOR = ceil(COLOR);

       result = AssignColor(COLOR);

       %MOVE with SERVO
       if result == 1
         % disp('RED');
           writePosition(servo1, 0.05);
       elseif result == 2
         % disp('GREEN');
           writePosition(servo1, 0.75);
       elseif result == 3
         % disp('BLUE');
           writePosition(servo1, 0.95);
       elseif result == 4
         % disp('YELLOW');
           writePosition(servo1, 0.25);
       elseif result == 5
         % disp('NOT RECOGNIZED');
           writePosition(servo1, 0.50);
       end % end if

    end %end if

    Cali = 0;
    Scan = 0;

 end %end while

 writePosition(servo1, 0.50);
 clear all; clc;
```

ColorSensingGUI.m

```
function varargout = ColorSensingGUI(varargin)
% Last Modified by GUIDE v2.5 10-May-2017 22:32:43
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
```

```
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn',
@ColorSensingGUI_OpeningFcn, ...
                   'gui_OutputFcn',
@ColorSensingGUI_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State,
varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before ColorSensingGUI is made visible.
function ColorSensingGUI_OpeningFcn(hObject, eventdata,
handles, varargin)

% Choose default command line output for ColorSensingGUI
handles.output = hObject;
% Update handles structure
guidata(hObject, handles);

% UIWAIT makes ColorSensingGUI wait for user response (see
UIRESUME)
% --- Outputs from this function are returned to the
command line.
function varargout = ColorSensingGUI_OutputFcn(hObject,
eventdata, handles)
% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes on button press in Calibrate.
function Calibrate_Callback(hObject, eventdata, handles)
assignin('base', 'Cali', 1);
delete(handles.figureColorSensingGUI);

% --- Executes on button press in Scan.
function Scan_Callback(hObject, eventdata, handles)
assignin('base', 'Scan', 1);
delete(handles.figureColorSensingGUI);

% --- Executes on button press in Exit.
function Exit_Callback(hObject, eventdata, handles)
assignin('base', 'Exit', 1);
delete(handles.figureColorSensingGUI);
```

ColorSensingGUI.fig



Calibrate.m

```
function [WHITE, BLACK] = Calibrate(a)

avgR = 0;
RED = 'D2';
GREEN = 'D3';
BLUE = 'D4';

%set WHITE
pause(5);
    %RED part
     writeDigitalPin(a, RED, 1);
     pause(0.1);
     avgR = Read(a);
     WHITE(1) = avgR;
     writeDigitalPin(a, RED, 0);
     pause(0.1);
    %GREEN part
     writeDigitalPin(a, GREEN, 1);
     pause(0.1);
     avgR = Read(a);
     WHITE(2) = avgR;
     writeDigitalPin(a, GREEN, 0);
     pause(0.1);
    %BLUE part
     writeDigitalPin(a, BLUE, 1);
     pause(0.1);
     avgR = Read(a);
     WHITE(3) = avgR;
     writeDigitalPin(a, BLUE, 0);
     pause(0.1);
%set BLACK
pause(5);
    %RED part
     writeDigitalPin(a, RED, 1);
     pause(0.1);
```

```
     avgR = Read(a);
     BLACK(1) = avgR;
     writeDigitalPin(a, RED, 0);
     pause(0.1);
    %GREEN part
     writeDigitalPin(a, GREEN, 1);
     pause(0.1);
     avgR = Read(a);
     BLACK(2) = avgR;
     writeDigitalPin(a, GREEN, 0);
     pause(0.1);
    %BLUE part
     writeDigitalPin(a, BLUE, 1);
     pause(0.1);
     avgR = Read(a);
     BLACK(3) = avgR;
     writeDigitalPin(a, BLUE, 0);
     pause(0.1);
 pause(5);
```

CheckColor.m

```
function COLOR = CheckColor(a)
avgR = 0;
% definition of outputs
RED = 'D2';
GREEN = 'D3';
BLUE = 'D4';

pause(5);
    %RED part
     writeDigitalPin(a, RED, 1);
     pause(0.1);
     avgR = Read(a);
     COLOR(1) = avgR;
     writeDigitalPin(a, RED, 0);
     pause(0.1);
    %GREEN part
     writeDigitalPin(a, GREEN, 1);
     pause(0.1);
     avgR = Read(a);
     COLOR(2) = avgR;
     writeDigitalPin(a, GREEN, 0);
     pause(0.1);
    %BLUE part
     writeDigitalPin(a, BLUE, 1);
     pause(0.1);
     avgR = Read(a);
     COLOR(3) = avgR;
     writeDigitalPin(a, BLUE, 0);
     pause(0.1);
pause(5);
```

AssignColor.m

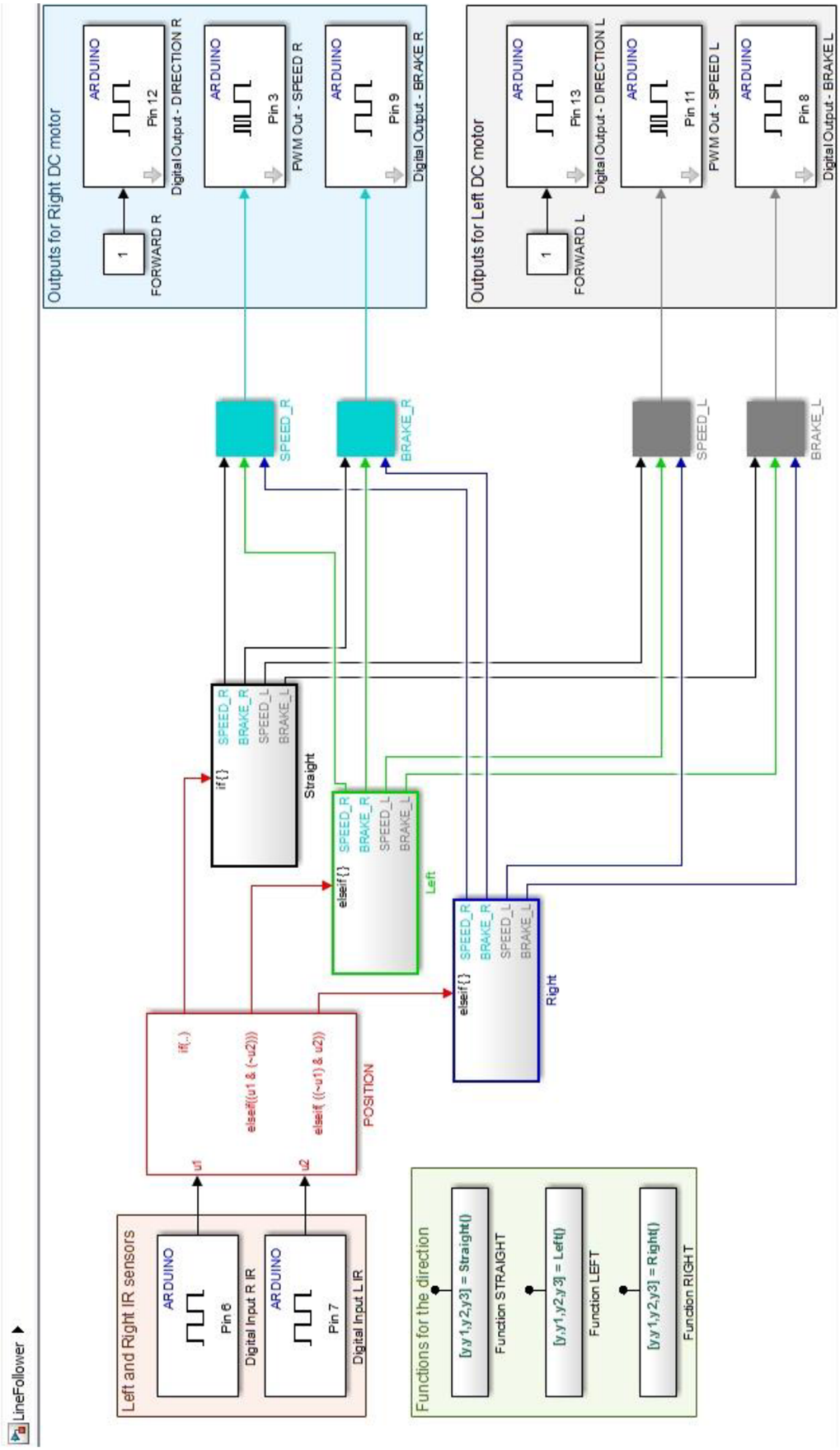
```
function result = AssignColor(COLOR)

% RED 1
if (COLOR(1) > 180) && (COLOR(1) < 300) && (COLOR(2) > -50)
&& (COLOR(2) < 140) && (COLOR(3) > -30) && (COLOR(3) < 145)
    result = 1;
% GREEN 2
elseif (COLOR(1) > -10) && (COLOR(1) < 120) && (COLOR(2) >
0) && (COLOR(2) < 170) && (COLOR(3) > 0) && (COLOR(3) <
120)
    result = 2;
% BLUE 3
elseif (COLOR(1) > -50) && (COLOR(1) < 85) && (COLOR(2) > -
70) && (COLOR(2) < 210) && (COLOR(3) > 40) && (COLOR(3) <
285)
    result = 3;
% YELLOW 4
elseif (COLOR(1) > 220) && (COLOR(1) < 350) && (COLOR(2) >
180) && (COLOR(2) < 450) && (COLOR(3) > 100) && (COLOR(3) <
580)
    result = 4;
% NOT RECOGNIZED 5
else
    result = 5;
end
```

Read.m

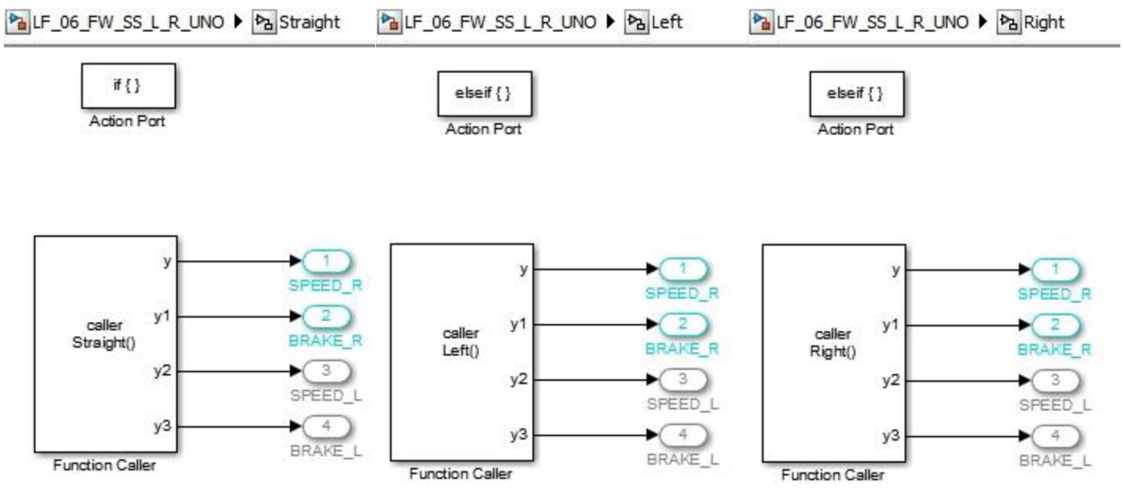
```
function avgR = Read(a)

 INPUT = 'A0';
 voltage = 0;
 value = 0;
 avgR = 0;
 koef = 51;

 for i = 1:1:10
     voltage = readVoltage(a, INPUT);
     % transform to 8-bit number 0-255
     value = koef * voltage;
     avgR = avgR + value;
     pause(0.1);
 end

 avgR = ceil (avgR / 10);
```

Příloha 2: Model pro úkol Sledování čáry

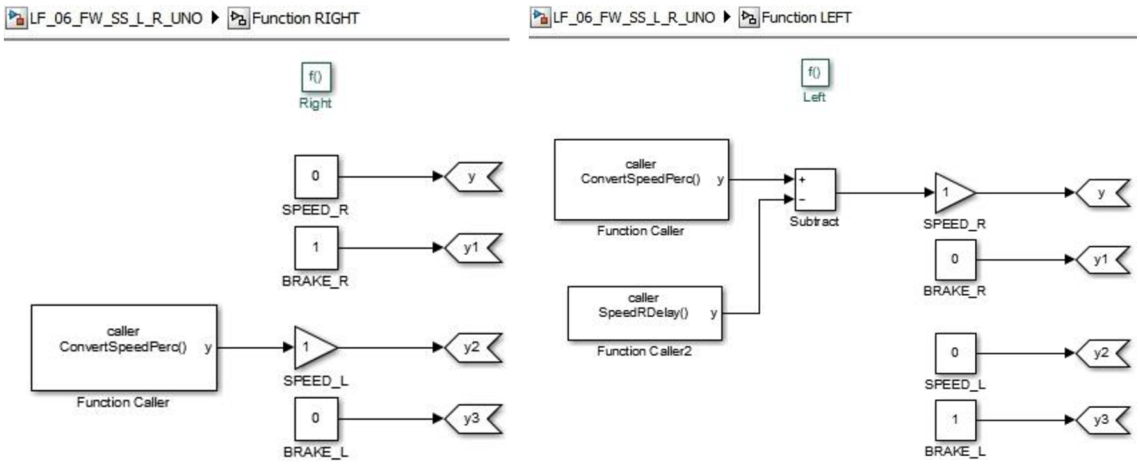
LineFollower_UNO.slx



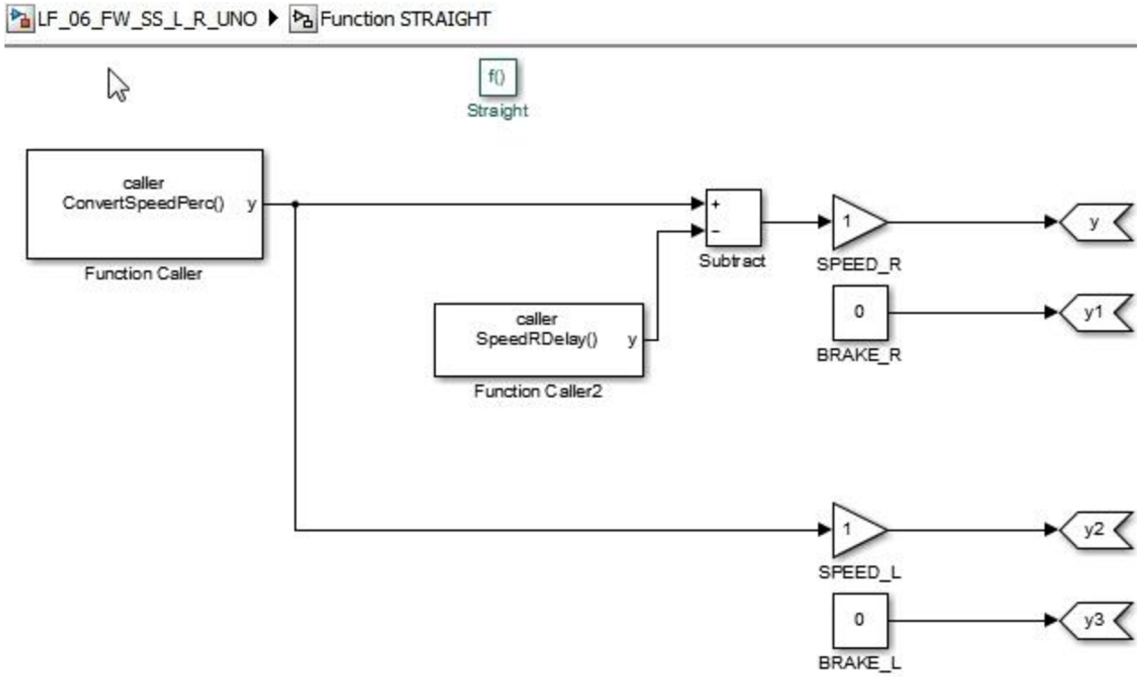
Bloky if{} **Straight** elseif{} **Left** elseif{} **Right**



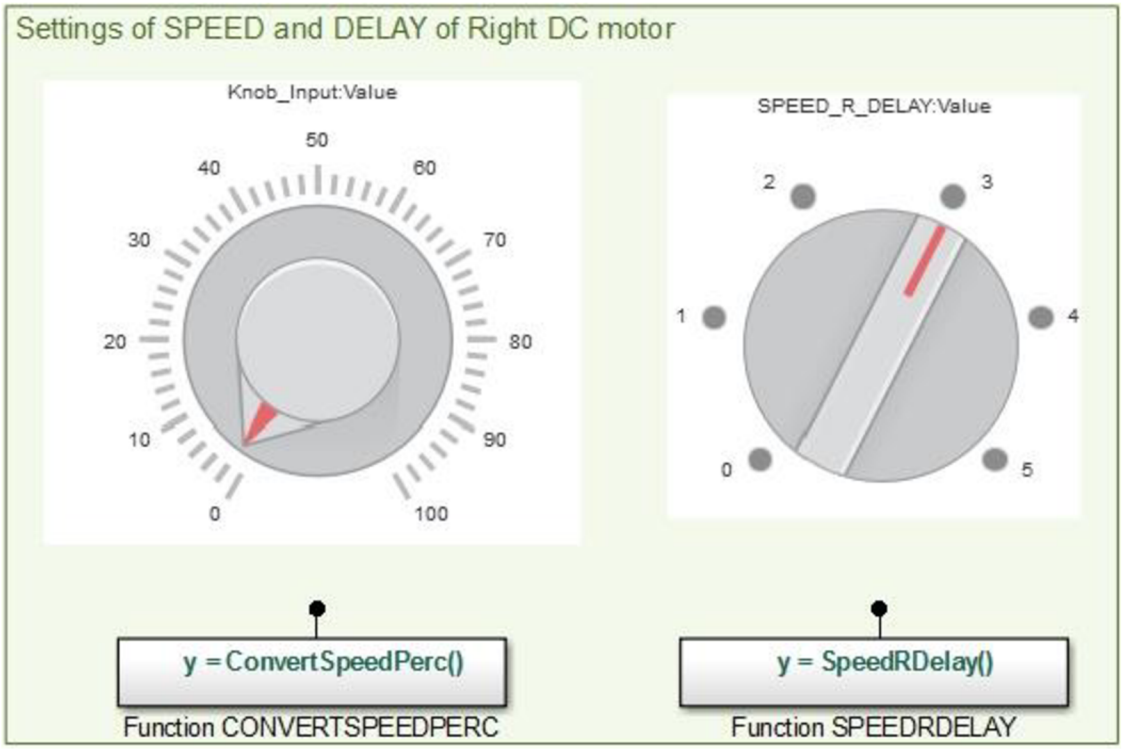
Funkce **Right** **Left**



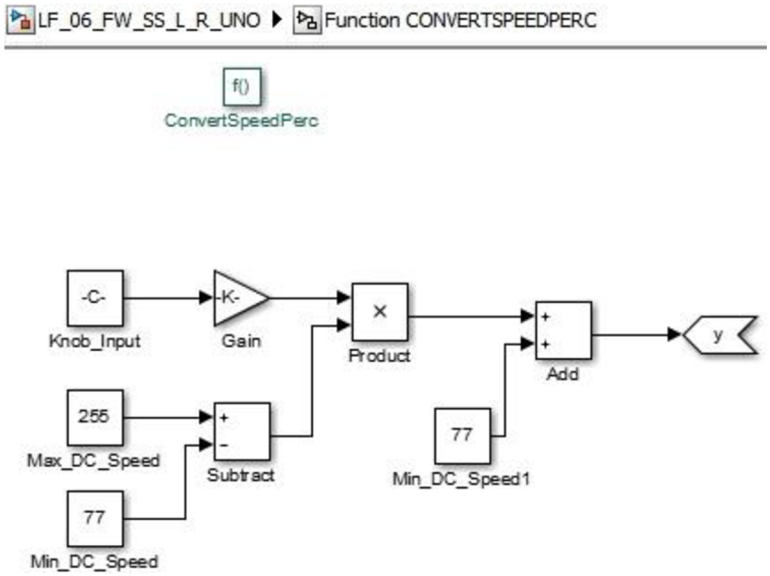
Funkce **Straight**



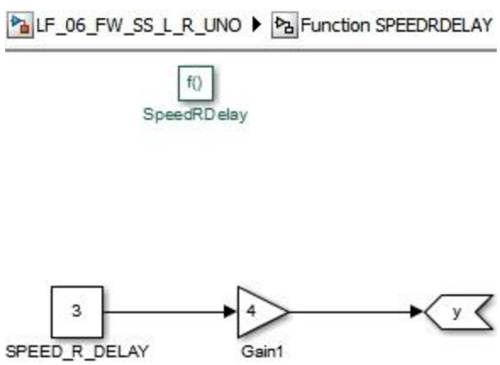
GUI



Funkce **ConvertSpeedPerc**



Funkce **SpeedRDelay**



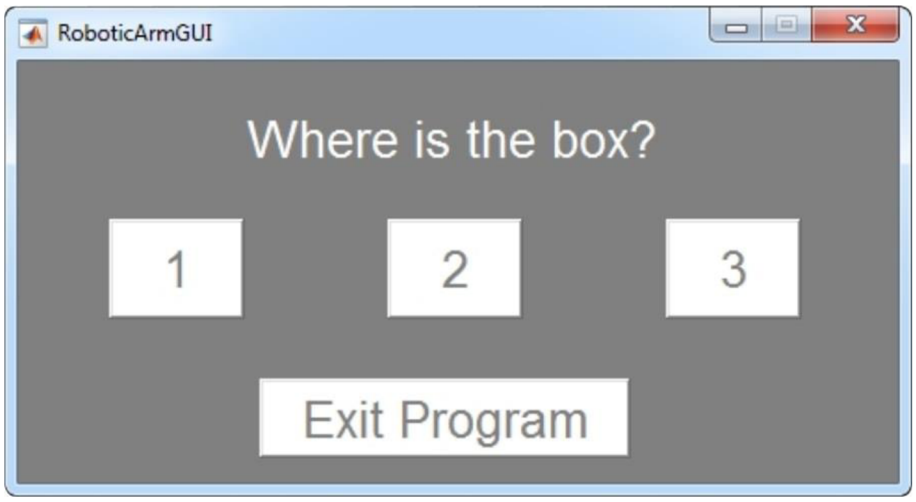
Příloha 3: Kód pro úkol Manipulátor

RoboticArm.m

```
clear all;clc;

%evaluate these later in loop
BoxPosition = 0;
Exit = 0;
Color = 0;

%servo positions
start = [0.50, 0.50, 0.50, 0.50];
ready = [0.50, 0.50, 0.50, 0.50];
position1 = [0.14, 0.14, 0.88, 0.40];
position2 = [0.30, 0.15, 0.94, 0.40];
position3 = [0.80, 0.12, 0.86, 0.40];
sensor = [0.50, 0.20, 0.88, 0.40];
white = [0.55, 0.88, 0.12, 0.50];
black = [0.28, 0.86, 0.12, 0.50];

%connect to arduino
a = arduino();

%assign digital pin as input from IR sensor
configurePin(a, 'D53', 'DigitalInput');

%assign digital PWM pins to servo motors
servo = [servo(a, 'D4'), servo(a, 'D5'), servo(a, 'D6'),
servo(a, 'D7')];

%READY position
writePosition(servo(4), ready(4)); pause(1);
writePosition(servo(3), ready(3)); pause(1);
writePosition(servo(2), ready(2)); pause(1);
writePosition(servo(1), ready(1)); pause(1);

%running program - ENDLESS LOOP (EXIT possible from GUI)
while ~Exit

    %show GUI
    RoboticArmGUI;
    uiwait(RoboticArmGUI);

    %EXIT test
    if Exit
        break;
    end

    %go to BOX POSITION and GRAB the BOX
    if BoxPosition == 1
        ToPosition(servo, position1);
```

```
    elseif BoxPosition == 2
        ToPosition(servo, position2);
    elseif BoxPosition == 3
        ToPosition(servo, position3);
    end %end if

    %move to SENSOR
    PositionToSensor(servo, sensor);

    %scan the color of the BOX
    pause(1.5);
    Color = readDigitalPin(a, 'D53');
    pause(0.5);

    %go to OUT POSITION and DROP the BOX
    if Color == 1 %WHITE
        ToPosition(servo, white);
    else    %BLACK
        ToPosition(servo, black);
    end %end if

    %back to READY position
    ToPosition(servo, ready);

    Color = 0;
    BoxPosition = 0;

end %end while

%END/START POSITION
writePosition(servo(4), start(4)); pause(1);
writePosition(servo(3), start(3)); pause(1);
writePosition(servo(2), start(2)); pause(1);
writePosition(servo(1), start(1)); pause(1);

clear all;clc;
```

RoboticArmGUI.fig



RoboticArmGUI.m

```
function varargout = RoboticArmGUI(varargin)
% Last Modified by GUIDE v2.5 02-May-2017 15:44:25
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn',
@RoboticArmGUI_OpeningFcn, ...
                   'gui_OutputFcn',
@RoboticArmGUI_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State,
varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before RoboticArmGUI is made visible.
function RoboticArmGUI_OpeningFcn(hObject, eventdata,
handles, varargin)
% Choose default command line output for RoboticArmGUI
handles.output = hObject;
% Update handles structure
guidata(hObject, handles);

% --- Outputs from this function are returned to the
command line.
function varargout = RoboticArmGUI_OutputFcn(hObject,
eventdata, handles
% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
assignin('base', 'BoxPosition', 1);
delete(handles.figureRoboticArmGUI);

% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
assignin('base', 'BoxPosition', 2);
delete(handles.figureRoboticArmGUI);
```

```
% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
assignin('base', 'BoxPosition', 3);
delete(handles.figureRoboticArmGUI);

% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles)
assignin('base', 'Exit', 1);
delete(handles.figureRoboticArmGUI);
```

ToPosition.m

```
function ToPosition(servo, position)

ServoPosition(servo(1), position(1));
ServoPosition(servo(3), position(3));
ServoPosition(servo(2), position(2));
ServoPosition(servo(4), position(4));
```

ToPosition.m

```
function PositionToSensor(servo, position)

ServoPosition(servo(3), position(3));
ServoPosition(servo(2), position(2));
ServoPosition(servo(1), position(1));
```

ServoPosition.m

```
function ServoPosition(servo, newPosition)

    position = readPosition(servo);

    if position < newPosition
        ServoPositionUp(servo, position, newPosition);
    elseif position > newPosition
        ServoPositionDown(servo, position, newPosition);
    end
```

ServoPositionUp.m

```
function ServoPositionUp(servo, oldPosition, newPosition)

for angle = oldPosition : 0.01 : newPosition
    writePosition(servo, angle);
    pause(0.01);
end
pause(1.5);
```

ServoPositionDown.m

```
function ServoPositionDown(servo, oldPosition, newPosition)

for angle = oldPosition : -0.01 : newPosition
    writePosition(servo, angle);
    pause(0.01);
end
pause(1.5);
```