

Mendelova univerzita v Brně  
Provozně ekonomická fakulta

---

# **Modernizace manipulačního robota**

**Bakalářská práce**

Vedoucí práce:  
Ing. Richard Klein

Tomáš Bajtek

Brno 2017

Na tomto místě bych rád poděkoval Ing. Richardu Kleinovi za odborné vedení práce a cenné rady. Dále mé poděkování patří také Tereze Vyorálkové, Petru Holubovi, rodině a všem, kteří mne podporovali.

## **Čestné prohlášení**

Prohlašuji, že jsem tuto práci: **Modernizace manipulačního robota** vypracoval samostatně a veškeré použité prameny a informace jsou uvedeny v seznamu použité literatury. Souhlasím, aby moje práce byla zveřejněna v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách ve znění pozdějších předpisů, a v souladu s platnou *Směrnici o zveřejňování vysokoškolských závěrečných prací*.

Jsem si vědom, že se na moji práci vztahuje zákon č. 121/2000 Sb., autorský zákon, a že Mendelova univerzita v Brně má právo na uzavření licenční smlouvy a užití této práce jako školního díla podle § 60 odst. 1 Autorského zákona.

Dále se zavazuji, že před sepsáním licenční smlouvy o využití díla jinou osobou (subjektem) si vyžádám písemné stanovisko univerzity o tom, že předmětná licenční smlouva není v rozporu s oprávněnými zájmy univerzity, a zavazuji se uhradit případný příspěvek na úhradu nákladů spojených se vznikem díla, a to až do jejich skutečné výše.

V Brně dne 18. května 2017

.....

## **Abstract**

BAJTEK, T. *Modernization of a handling robot.* Bachelor thesis. Brno, 2017.

This thesis deals with a complete modernization of a handling robot and contains the technical design, the analysis of a technology selection and describes the modernization process itself along with the robot function testing. For the purpose of the modernization, the modern and freely available technologies based on the Arduino platform are used. The realization was finished successfully, therefore the main benefit of the project is a functional handling robot, useful for a practical schooling. The financial matter of the modernization is summarized in a separate chapter.

**Keywords:** Arduino, handling robot, Java, modernization, stepper motor

## **Abstrakt**

BAJTEK, T. *Modernizace manipulačního robota.* Bakalářská práce. Brno, 2017.

Tato bakalářská práce řeší kompletní modernizaci manipulačního robota, zahrnující její návrh, výběr technologie a součástí, dále samotné provedení modernizace a následné ověření funkčnosti. Pro zpracování jsou využívány moderní a volně dostupné technologie založené na platformě Arduino. Realizace byla úspěšně provedena a dokončena, hlavním přínosem je tak funkční manipulační robot, kterého je tak možné využívat při praktické výuce. Finanční stránka celé provedené modernizace je shrnuta ve zvláštní kapitole.

**Klíčová slova:** Arduino, manipulační robot, Java, modernizace, krokový motor

## **Obsah**

<b>1</b>	<b>Úvod</b>	<b>7</b>
<b>2</b>	<b>Cíl práce</b>	<b>8</b>
<b>3</b>	<b>Analýza a literární rozbor</b>	<b>9</b>
3.1	Manipulační robot . . . . .	9
3.1.1	Současný stav . . . . .	9
3.1.2	Popis robotické ruky . . . . .	9
3.1.3	Akční členy . . . . .	10
3.1.4	Elektronika původního řízení . . . . .	11
3.1.5	Dosavadní řízení . . . . .	11
3.1.6	Senzory . . . . .	12
3.2	Ověření funkčnosti motorů . . . . .	12
3.3	Možnosti modernizace . . . . .	13
3.3.1	Raspberry Pi . . . . .	13
3.3.2	Arduino . . . . .	14
3.4	Návrh modernizace . . . . .	15
3.4.1	Řídicí jednotka . . . . .	15
3.4.2	Driver DRV8825 . . . . .	17
3.4.3	Ultrazvukový senzor . . . . .	18
3.4.4	Mikrospínačové senzory . . . . .	19
3.4.5	Reflexní opto senzory . . . . .	19
3.5	Komunikační rozhraní . . . . .	20
3.5.1	Komunikace po sériové lince . . . . .	20
3.5.2	Knihovna pro komunikaci po RS232 v jazyce JAVA . . . . .	20
<b>4</b>	<b>Metodika</b>	<b>22</b>
4.1	Software . . . . .	22
4.1.1	Popis prostředí Arduino . . . . .	22
4.1.2	Jazyk Java . . . . .	22
4.1.3	Programování v NetBeans . . . . .	23
4.2	Metodika vývoje SW . . . . .	23
4.2.1	Iterativní vývoj . . . . .	23
4.3	Návrhové vzory . . . . .	23
4.3.1	Singleton . . . . .	23
4.4	Zpracování signálů ze senzorů . . . . .	24
4.4.1	Mikrospínače . . . . .	24
4.4.2	Reflexní opto senzory . . . . .	24
4.5	Elektronické součásti . . . . .	24

<b>5 Praktická realizace</b>	<b>25</b>
5.1 Zapojení modulů . . . . .	25
5.1.1 Arduino Micro . . . . .	25
5.1.2 Drivery . . . . .	25
5.1.3 Senzory . . . . .	26
5.1.4 Schéma zapojení modulů . . . . .	26
5.2 Komunikační protokol . . . . .	26
5.2.1 Komunikace z počítače do mikroprocesoru . . . . .	26
5.2.2 Komunikace z mikroprocesoru do počítače . . . . .	28
5.3 Implementace procesorové části . . . . .	28
5.3.1 Struktura programu . . . . .	28
5.3.2 Ovládání motorů . . . . .	29
5.3.3 Senzory . . . . .	30
5.3.4 Komunikace . . . . .	31
5.4 Implementace knihovny . . . . .	32
5.4.1 Knihovna RXTX . . . . .	32
5.4.2 Kolekce dat . . . . .	33
5.4.3 Příprava dat pro odeslání . . . . .	34
5.4.4 Zpracování přijatých dat . . . . .	35
5.4.5 Rozhraní řídicí knihovny . . . . .	35
5.4.6 Grafické rozhraní . . . . .	38
<b>6 Zjednodušené ekonomické zhodnocení</b>	<b>39</b>
<b>7 Diskuse</b>	<b>41</b>
<b>8 Závěr</b>	<b>42</b>
<b>9 Literatura</b>	<b>44</b>
<b>Přílohy</b>	<b>48</b>
<b>A Schéma zapojení</b>	<b>49</b>
<b>B Modernizovaný manipulační robot</b>	<b>50</b>

## 1 Úvod

V současné době je vývoj a výroba technologických produktů na takové úrovni, že produkty velmi rychle zastarávají a jsou postupem času nahrazovány novějšími a modernějšími. Ruku v ruce jde s tímto trendem i postupné snižování výrobních cen, kdy může být cena za pořízení modernějšího zařízení nižší než investice do údržby starého zařízení.

U zařízení z oblasti elektroniky se toto stárnutí projevuje nejčastěji na úrovni zastarávání softwarové výbavy, kdy již není výrobcem dále podporováno, případně je postaveno na již nevyhovujících nebo neefektivních elektronických součástkách a prvcích. Převážně u zařízení, která v době vzniku této práce, dosahují stáří minimálně kolem patnácti let, bývá elektronika často založena na technologiích, které byly v době vývoje používané pouze jedním výrobcem, a jsou navzájem s jinými zařízeními nekompatibilní. V současné době je trend již opačný, a naopak se výrobcí snaží zařízení vytvářet na technologiích, které spolu dokáží komunikovat na základě jednotného protokolu, případně stačí implementovat jednoduchý softwarový převodník, který komunikaci přemostí a zařízení je tak stále použitelné.

Tato bakalářská práce se bude zabývat modernizací manipulačního robota, který byl v dřívější době pořízen Provozně ekonomickou fakultou, a nyní by po modernizaci mohl být opět využíván studenty při zpracování úloh v praktických cvičeních. Jedná se o typický případ, kdy již zastaralo elektronické vybavení, ale mechanické zůstalo bez problému nadále použitelné v rámci kategorie výukových prostředků.

Celá modernizace se bude týkat nahrazení původní řídicí jednotky za novou. Dalším krokem bude vytvoření komunikačního protokolu pro ovládání tohoto robota a jeho implementace do řídicí jednotky, která bude podle požadavků naprogramována. Následně práce bude vysvětlovat zpracování řídicí knihovny v programovacím jazyce Java, pro použití na libovolném počítači, který bude tuto ruku používat jako svoje výstupní zařízení. Závěrem bude celá modernizace zhodnocena a provedena také její ekonomická analýza.

## 2 Cíl práce

Cílem bakalářské práce je provést modernizaci manipulačního robota. Prvním krokem je seznámení s aktuálním stavem, jeho vybavením a funkčností, včetně otestování krokových motorů. Nutné je také provedení analýzy možností, které se k modernizaci nabízejí. Následně navrh a praktické provedení modernizace tohoto manipulačního robota. Modernizací bude nahrazeno původní řízení za nové, naprogramování řídicí jednotky, popsání komunikačního protokolu, kterým bude možno robota ovládat a vytvoření řídicí knihovny.

Další částí bakalářské práce je vytvoření řídicí knihovny v jazyce Java pro ovládání manipulačního robota z vyššího programovacího jazyka. Tato knihovna bude použita pro komunikaci pomocí sériového rozhraní, bude s ní možné kompletně ovládat celého robota a současně bude poskytovat zpětnou vazbu.

Teoretická část bude zaměřena na popis konstrukce a aktuálního stavu robota, včetně popisu možností, které řízení původně nabízelo. Následně bude provedeno srovnání existujících platform, kterými lze modernizaci provést, a na základě zvolené platformy bude proveden návrh modernizace. Tento návrh se bude soustředit na konkrétní zapojení motorů, jednotlivých modulů a senzorů, které budou využity. Dále bude navržen a popsán komunikační protokol, pomocí kterého bude komunikovat řídicí počítač s řídicí jednotkou umístěnou v manipulačním robotovi.

Praktická část se bude zabývat fyzickou realizací zapojení motorů, modulů a senzorů k řídicí jednotce. Dále se bude praktická část zaměřovat na popis chování řídicí jednotky. Tato jednotka se bude starat o řízení celého manipulačního robota, zpracování dat přijatých na vstupním rozhraní a vyhodnocení a odeslání dat ze senzorů zpět do řídicího počítače. Součástí praktické části bude také implementace a popis chování řídicí knihovny, díky které bude možné manipulačního robota ovládat z jakéhokoliv počítače pouze za použití zjednodušených příkazů a uživatel tak bude odstíněný od nepříliš přívětivého ovládání jednotlivých motorů zvláště. Tuto knihovnu bude mimo jiné možné využít jako předlohu pro vytvoření případné další knihovny v jiném programovacím jazyku. Nedílnou součástí, bude také příprava pro praktické využití ve výuce. Tato příprava bude zahrnovat testování a ověření funkčnosti realizované modernizace.

Posledním bodem, kterým je nutné se zabývat pro splnění všech podmínek práce, je vytvoření zjednodušené ekonomické kalkulace, ve které bude srovnána celková cena použitých součástí nutných k realizaci této modernizace s nákupem podobného manipulačního robota.

## 3 Analýza a literární rozbor

Před započetím praktické realizace je nutné provést teoretický rozbor úlohy, kterým se zabývá následující kapitola. Konkrétně bude provedeno zhodnocení stavu, ve kterém se manipulační robot aktuálně nachází, porovnání dostupných technologií a ujasnění požadované funkčnosti po modernizaci.

### 3.1 Manipulační robot

Manipulační roboti jsou stacionární zařízení umožňující manipulaci s předměty. Skládají se z podstavy a minimálně dvou ramen, kdy je každé z ramen ovládáno zvlášť. Na konci ramene, které je nejvzdáleněji od podstavy, je umístěno koncové zařízení plnící různé funkce, kterým můžou být čelisti, případně magnet, nebo úplně jiná zařízení.

#### 3.1.1 Současný stav

Robotická ruka, o jejíž modernizaci se v rámci této bakalářské práce jedná, byla pořízena již v dřívější době. Tovární označení je dle typového štítku NESA G60. Údaj o roku výroby na štítku chybí, tudíž není možné přesně určit stáří tohoto robota, ovšem dle elektronického vybavení, o kterém bude řeč později, lze stáří odhadnout minimálně na 10 let. Manuál k použití, případně obdobnou dokumentaci nebylo možné dohledat a tudíž bylo nutné vše zjistit vlastními pokusy a provést reverse engineering.

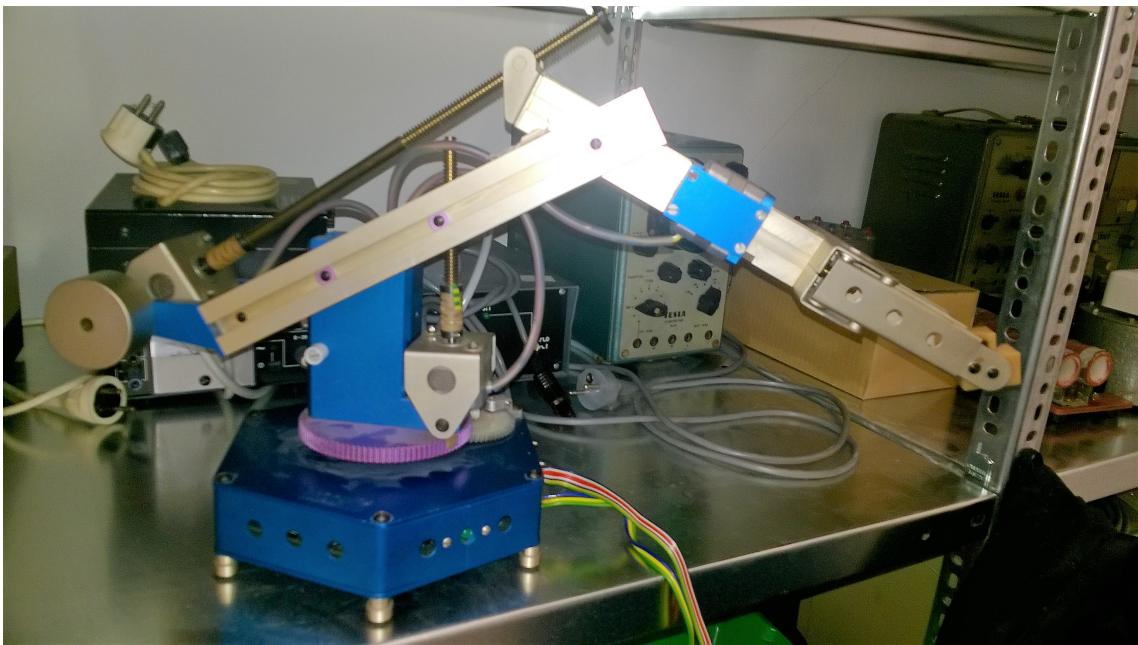
#### 3.1.2 Popis robotické ruky

Ruka se skládá ze čtyř pohybových částí. Šestiúhelníková základna obsahuje krokový motor, který ovládá rotaci přes dvě ozubená kola. Ruku není možné otáčet o celých  $360^\circ$ , ale pouze po části kružnicové výseče, která je na větším z ozubených kol omezena zarážkou, z důvodu zamezení přetáčení kabelů k dalším krokovým motorům. Dále základna obsahuje původní elektroniku, která se stará o řízení každého z krokových motorů. Tato elektronika zabírá většinu prostoru základny a je realizována z většiny integrovanými obvody. Dále se na jedné straně šestiúhelníkové základny nachází vstupní konektor pro připojení ovladače tohoto robota. Pro ilustraci je na Obrázku 1 fotografie manipulačního robota v původním stavu.

Na velkém ozubeném kole je připevněn podstavec pro celou konstrukci robotické ruky. Součástí podstavce je krokový motor, na který je připevněna krátká šroubovice. Pomocí této šroubovick je možné zvedat a sklápět první rameno.

První rameno je vyváženo pomocí závaží a krokového motoru, na který je opět napevno připevněna delší šroubovice. Tato šroubovicka zajišťuje pohyb druhého ramene a případné vyrovnávání polohy čelistí do požadované roviny.

Čelisti jsou umístěny na konci druhého ramene. Ovládány jsou opět pomocí krokového motoru a kratší šroubovicka. Samotné čelisti se skládají ze dvou kusů



Obrázek 1: Původní stav

a jsou upevněny asi v jedné třetině na ose. Obě části jsou spojeny ve spodní části pomocí osy na plastovém pohyblivém prvku, který se pohybuje po šroubovici.

Všechny šroubovice, ozubená kola a ostatních mechanické části manipulačního robota byly, podle fyzického vyzkoušení pohybu, bez problémů pohyblivé a nebylo tedy nutné provádět žádné další úpravy nebo opravy.

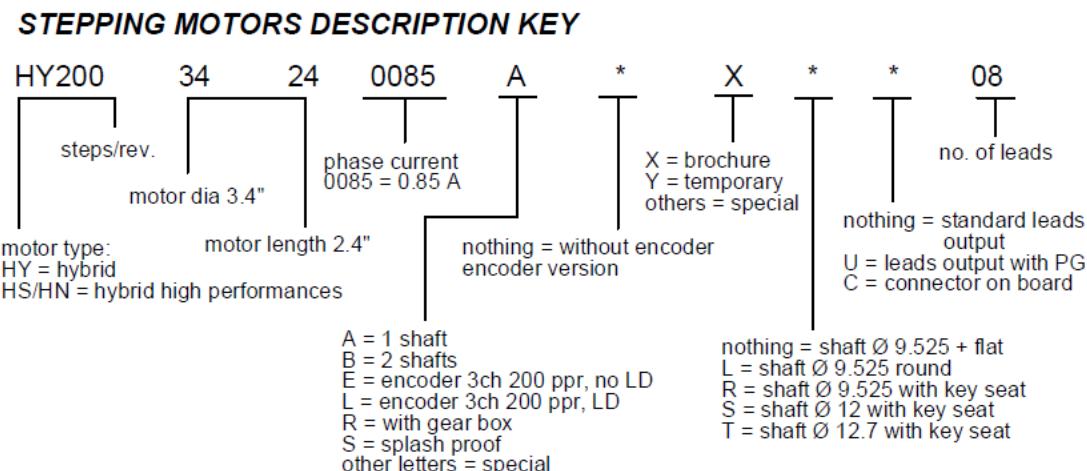
### 3.1.3 Akční členy

Manipulační robot má celkem 4 samostatně pohyblivé části, kdy je každá z těchto částí řízena krokovým motorem. Konkrétně se jedná o hybridní krokové motory s označením MAE HY-100 1713 020 A4.

O hybridních krokových motorech pojednává Novák, který uvádí, že hybridní krokový motor je představitel v současné době pravděpodobně nejužívanějšího typu. Stator je u zde popisovaného motoru tvořen osmi hlavními pólovými nástavci, každý je dále rozdělen na 5 zubů. Na každém hlavním pólovém nástavci je vinutí cívky. Rotor motoru je tvořen hřídelí z nemagnetické oceli, na které jsou nalisovány dva pólové nástavce složené z plechů. Dále popisuje, že mezi těmito pólovými nástavci je uložen permanentní magnet, takže rotorové pólové nástavce mají každý různou magnetickou polaritu. Pólové nástavce jsou rozděleny na 50 zubů o stejně šířce jako rotorové (Novák, 2005). Udané hodnoty platí pro krokové motory, které pro jednu otáčku potřebují 200 kroků, konkrétní použité motory umožňují pouze 100 kroků na otáčku.

Dokumentace k tomuto konkrétnímu typu motoru nebyla nalezena, ovšem byla nalezena dokumentace motorů ze stejné výrobní řady od stejného výrobce, která

obsahuje mimo jiné tabulkou, ze které je možné zjistit, dle označení, parametry konkrétního modelu. Tato převodní tabulka, znázorněna na Obrázku 2, platí i na model použitý v manipulačním robotu.



Obrázek 2: Označení motorů

Pro jednu otáčku těchto motorů je zapotřebí 100 kroků, v přepočtu má jeden krok úhel  $3,6^\circ$ . Motor zajišťující otáčení celé robotické ruky používá jako převod pro otáčení ozubené soukolí, kterým je pohybováno za pomocí ozubeného kola umístěného na motoru, který se nachází přímo v základně manipulačního robota. Ostatní motory jsou umístěny na samotné konstrukci ruky a pro plynulý pohyb jimi pohybované části používají šroubovice, které jsou pevně spojeny s rotory motorů.

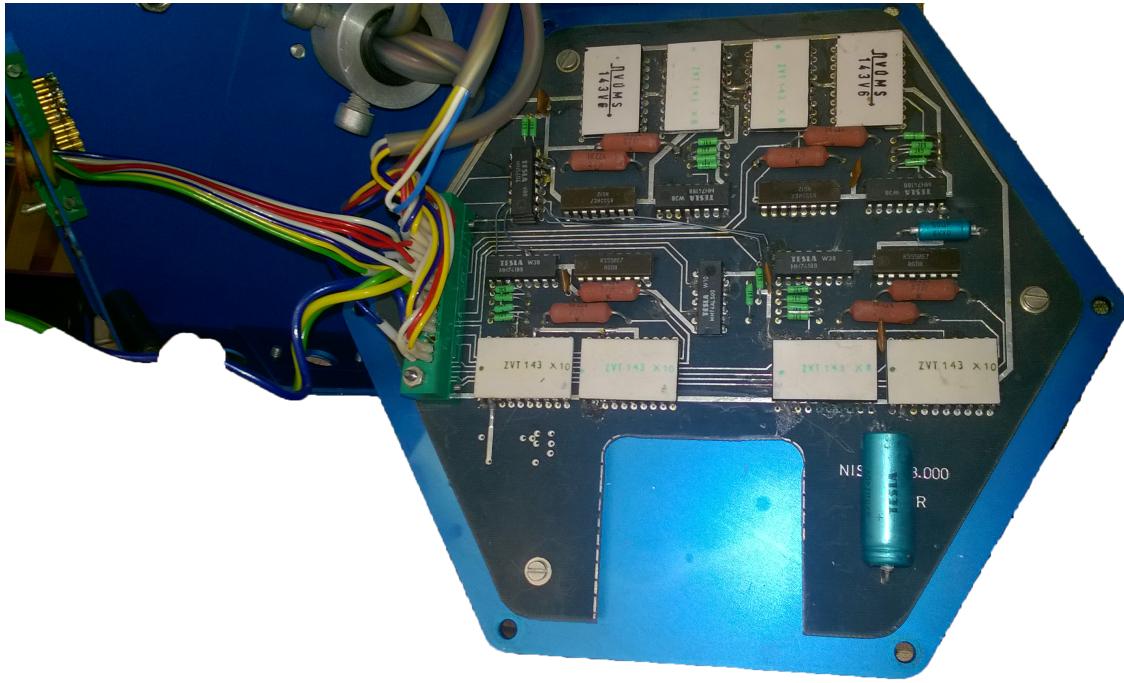
Použitím převodu z ozubených kol a šroubovic je zvýšena síla motorů a zároveň také odstraněn problém s nízkým počtem kroků pro jednu otáčku motoru.

### 3.1.4 Elektronika původního řízení

Dle analýzy použitých součástek na ovládací desce, která se nachází v základně, lze usoudit, že původní řízení, bylo řešeno pouze na principu přímého ovládání. Základová deska byla osazena integrovanými obvody Tesla MH74188, což jsou programovatelné 256 bitové paměti, dále integrovaným obvody K555NE7 a obvody ZVT 143 X10, o kterých již nebylo v současné době možné zjistit žádné bližší informace. Fotografie původní elektroniky se nachází na obrázku 3.

### 3.1.5 Dosavadní řízení

Původní ovladač se do současné doby nedchoval a kvůli absenci jakéhokoliv manuálu není ani možné přiblížit jaké funkce umožňoval. Lze tedy pouze usoudit, že ruka byla ovládána několika ovládacími prvky, kterými bylo možné napřímo ovládat každý motor zvlášť. Dále mohl ovladač umožňovat ukládání sekvencí, které



Obrázek 3: Původní elektronika

měly jednotlivé motory provést a tyto sekvence pak opakovat. Manipulační robot neobsahoval žádné senzory o aktuálním pohybu a ovladač tedy nemohl reagovat na případné neočekávané situace.

Později byl robot již jednou modernizován napojením převodníku z USB na RS232 na původní obvody. Nejspíše bylo možné použít pro ovládání ruky software, který pracoval přímo na počítači. Tento software mohl umožňovat pouze stejné ovládání jako fyzický ovladač. Ani k této úpravě nebyla zanechána žádná dokumentace, tudíž nebylo možné zjistit žádné další podrobnosti.

### 3.1.6 Senzory

Původní ani dříve modernizované řízení nebylo vázané na žádné senzory, veškerý pohyb ruky byl tedy závislý na rozhodnutí a úsudku uživatele manipulační ruky. Ruka neobsahovala žádný ze senzorových prvků, který by mohl její pohyb ovlivňovat a zamezovat tak krajinám situacím, ve kterých by mohlo docházet například k přetěžování krokových motorů. Lze usuzovat, že původní ovládání bylo pouze jednoduché, nezávislé řízení jednotlivých motorů.

## 3.2 Ověření funkčnosti motorů

Pro rozhodnutí, zda bude možné původní motory zachovat nebo budou muset být nahrazeny, muselo proběhnout jejich otestování. Toto testování muselo být realizo-

váno ještě před samotným návrhem řešení. Pro otestování byl vybrán driver, který bylo možné použít jak s původními krokovými motory, tak případně s motory, kterými by byly původní nahrazeny. Otestování muselo být provedeno pro každý motor zvlášť.

K driveru bylo připojeno napájení udávané datasheetem, samotný motor a vývojové Arduino Micro. Driver byl ovládán za pomocí tohoto naprogramovaného mikroprocesoru. Program spočíval v nekonečné smyčce, která obsahovala příkazy pro jednoduchý pohyb motoru v obou směrech.

Takto bylo ověřeno, že všechny čtyři motory jsou funkční a je možné je používat i nadále.

### 3.3 Možnosti modernizace

Před samotným návrhem a implementací modernizace je nutné provést srovnání aktuálně existujících platforem, které je možné pro tuto modernizaci využít.

Základním kritériem je nahrazení dosavadního řízení, řízením digitálním. Rozhodující je také dostupnost modulů, které budou při modernizaci použity, tyto moduly jsou nabízeny výhradně v digitálním provedení.

Každý z prvků, který bude zapojen v obvodu, je modul, který dokáže pracovat nezávisle na ostatních modulech a jeho funkcí řídí řídicí jednotka. Výhodou modulů je možnost sestavit je na základě konkrétních požadavků a případně také dle akčních členů.

Výhodou modulů, které jsou určeny pro řízení podobných koncových prvků, je také zpravidla podpora stejných komunikačních protokolů, díky kterým lze modul a koncový prvek fyzicky nahradit a řízení je nadále funkční i bez programového zásahu do řídicí jednotky. Programová obsluha modulů v řídicí jednotce je zpravidla řešena použitím knihovny, která je dodávána přímo výrobcem modulu, případně lidmi a společnostmi, které se zaměřují na prodej nebo výrobu dalších zařízení s těmito moduly. Knihovny jsou zpravidla dostupné pod volně šířitelnou licencí.

V důsledku použití digitálního řízení je možné využít řídicích jednotek, které nejsou jednoúčelové a je možné jejich chování upravovat přeprogramováním za pomocí jiného počítače. Je tak tedy zaručena možnost úpravy programu, v případě nalezení chyb nebo nekorektního chování. Dále je možné program upravit na základě požadavků na změnu chování nebo v případě přidání nového senzoru.

#### 3.3.1 Raspberry Pi

Aktuálně existuje mnoho platforem, na kterých lze řídicí jednotku pro manipulačního robota založit. Velmi oblíbeným a používaným minipočítačem je Raspberry Pi, které je nabízeno v několika různých výkonových variantách. Dle dokumentace tento minipočítač ve všech verzích pracuje s procesory ARM, RAM v řádech stovek MB, integruje v sobě grafické jádro a také vstupy a výstupy označované jako GPIO (Raspberry Pi, 2017). O GPIO se dokumentace zmiňuje jako o pinech, které

mohou být konfigurovány buď jako obecně použitelný vstup, výstup nebo jako jedno z šesti speciálních alternativních nastavení, jejichž funkce závisí na konkrétních pinech (Raspberry Pi, 2017). Tyto piny je možné obsluhovat i za pomocí vyšších programovacích jazyků jako je Java nebo Python.

Z dalších zařízení, která mají podobné rysy jako Raspberry Pi, lze jmenovat například Intel Edison nebo Banana Pi. Nevýhodou všech výše jmenovaných je především jejich cena, jedná se totiž o kompletní počítače, které jsou pro použití jako řídicí jednotky pro manipulačního robota velmi redundantní a obsahují některé komponenty, které by za celou dobu používání nebyly vůbec využity.

### 3.3.2 Arduino

Ve světě je asi nejrozšířenější platformou Arduino. To nabízí různé typy desek od méně výkonných a malých modelů po kompletní soustavy obsahující USB, HDMI, Ethernet, či audio porty (Voda, 2015).

Dále Voda upřesňuje, v dnešní době se prodalo již několik stotisíc desek Arduino. Důkazem, že tato platforma není mrtvá, může být i to, že nedávno byl ohlášen vývoj nové a výkonné desky Arduino Galileo, která vzniká ve spolupráci s Intelem. Za osm let vývoje již vzniklo spoustu různých typů Arduina. Jelikož se jedná o opensource projekt, vznikalo společně s hlavní linií projektu i spousta dalších, neoficiálních typů, takzvaných klonů (Voda, 2015).

Pro ovládání zařízení, která nevyžadují zpracování velkého množství dat, je možné využít mikroprocesory, do kterých je při naprogramování nahraný program, který je vždy po zapnutí napájení do mikroprocesoru spuštěn a nekonečně opakován. Tyto mikroprocesory se vyznačují nižší spotřebou, výkonem a především nižší cenou než u dříve zmínovaných minipočítačů. Nižší spotřeba je zajištěna menší výpočetní rychlostí, menší pamětí pro program, menší pamětí RAM i menší programovatelnou pamětí pro uložení dat i při vypnutém stavu. Tento nedostatek je ovšem kompenzován variabilitou jejich použití a velkým počtem digitálních a analogových vstupů a výstupů na jednotlivých pinech.

Samotná platforma Arduino umožňuje programovat i další mikroprocesory jiných plafotrem založených například na procesorech ESP8266 jako jsou konkrétně moduly WeMos D1 nebo NodeMCU. Tyto moduly běží na vyšších frekvencích a dosahují tak vyššího výpočetního výkonu, současně mají také větší operační paměť a flash paměť pro ukládání samotných programů (Suchánek, 2016).

Každý z těchto procesorů se liší svým výpočetním výkonem, velikostí pamětí, počtem vstupů a výstupů nebo integrovanými technologiemi. Různé procesory lze také datově spojit za pomocí komunikačních protokolů a mohou si tak mezi sebou vyměňovat data.

## 3.4 Návrh modernizace

Tato bakalářská práce se zabývá návrhem a praktickou realizací modernizace manipulačního robota. Práce vzniká pro potřeby Provozně ekonomické fakulty Mendelovy univerzity v Brně.

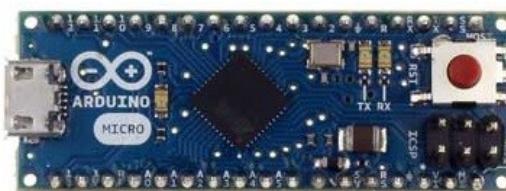
Pro řídicí jednotku byla zvolena platforma Arduino. Důvody pro výběr byly dostupnost a velmi levné pořizovacích náklady. Dále využívá logických úrovní 0V a 5V a tím umožňuje připojení přímo na moduly bez použití dalších úrovňových převodníků.

Z původního elektronického vybavení zařízení zůstaly zachovány pouze krokové motory, které v současné době nebylo nutné nahrazovat. Ostatní elektronika bude nahrazena moderními moduly, kterými bude možné ruku ovládat a současně zajistit částečně autonomní chování.

### 3.4.1 Řídicí jednotka

Arduino existuje v mnoha různých variantách, které je možné zvolit dle požadavků na počet vstupů a výstupů, rychlosť procesoru a podobně.

Na většině desek je mimo hlavního čipu ještě převodník, který umožňuje komunikaci mezi PC (USB) a čipem. Setkáme se však s typy, které převodník nemají. Může to být ze dvou důvodů. Prvním z nich je úspora místa a následná nutnost použití externího převodníku. Druhým typem jsou ty, jejichž čip má v sobě tento převodník zabudovaný (Voda, 2015).



Obrázek 4: Arduino Micro (Voda, 2015)

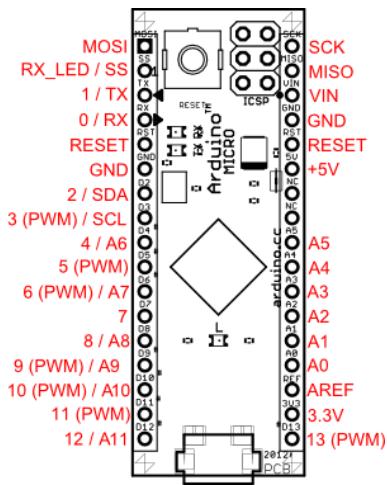
Zvolené Arduino pro praktickou realizaci, má označení Arduino Micro. Voda uvádí, srdcem každého Arduina je procesor od firmy Atmel, který je obklopen dalšími elektronickými komponenty (Voda, 2015). Konkrétně se jedná mikroprocesor Atmel ATmega32U4.

Následující tabulka popisuje základní parametry Arduino Micro.

Pro komunikaci s ostatními procesory nebo integrovanými obvody jsou na Arduinu vyhrazeny piny, které podporují různé komunikační protokoly. Těmito protokoly jsou nejčastěji komunikace po sériové lince, IIC nebo SPI.

Tabulka 1: Parametry Arduino Micro (Arduino - ArduinoBoardMicro, 2017)

Mikroprocesor	ATmega32U4
Pracovní napětí	5 V
Vstupní napětí (doporučené)	7-12 V
Vstupní napětí (limit)	6-20 V
Digitální I/O piny	20
PWM kanály	7
Analogové vstupní kanály	12
DC proud na I/O pin	20 mA
DC proud na 3.3V pin	50 mA
Flash paměť	32 kB (ATmega32U4) ze kterých jsou 4 kB využity bootloaderem
SRAM	2.5 kB (ATmega32U4)
EEPROM	1 kB (ATmega32U4)
Frekvence	16 MHz
LED_BUILTIN	13
Délka	48 mm
Šířka	18 mm
Hmotnost	13 g

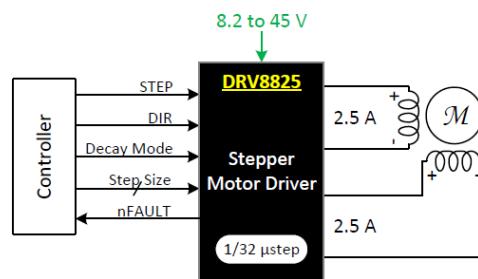


Obrázek 5: Schéma pinů Arduino Micro (Arduino - ArduinoBoardMicro, 2017)

Programovací jazyk Arduino podporuje, formou knihoven, také širokou škálu různých modulů, jako jsou například moduly měřící různé veličiny, komunikační rozhraní nebo ovládající koncová zařízení. Použitím knihovny je programátor odstíněn od řízení modulů na úrovni jednotlivých výstupů z procesoru. Je tak formou funkcí vytvářeno rozhraní, pro kompletní ovládání těchto zařízení.

### 3.4.2 Driver DRV8825

Driverů pro ovládání krokových motorů existuje mnoho druhů. Jednotlivé druhy se liší převážně maximálním napětím a proudem, který vyžadují řízené krokové motory. Pro modernizaci byl zvolen driver s integrovaným obvodem DRV8825. Parametry uvedeny v dokumentaci k tomuto integrovanému obvodu určují, že je obvod možné využít pro napětí v rozmezí 8,2V až 45V, dále může motoru dodávat proud až 2,5A na cívku při napětí 24V, rozhraní lze ovládat na logické úrovni 3,3V nebo 5V a dále podporuje tzv. mikrokroky o velikosti až 1/32, které ovšem v práci nebudou využívány (Texas Instruments, 2016).



Obrázek 6: Zjednodušené schéma driveru (Texas Instruments, 2016)

Ovládání těchto driverů ke krokovým motorům je v mnoha případech velmi podobné. Na výstup A1 a A2 je zapojena první z cívek motoru, na výstup B1 a B2 je zapojena druhá cívka motoru. Pro výkonovou část, na kterou je zapojen motor, je připojen vlastní napájecí zdroj na piny VMOT a GND. Mikroprocesor je nutné zapojit minimálně na vstupy STEP, DIR a spojit s GND.

Dále driver umožňuje využít různě velkých mikrokroků připojením různých kombinací logických jedniček na piny M0, M1 a M2. Další řídící piny jsou negovaný ENABLE, negovaný RESET a negovaný SLEEP. Pin ENABLE umožňuje přivedením logické jedničky vypnutí driveru, tzn., že driver v požadovanou chvíli nereaguje na vstupy STEP a DIR. Pin RESET slouží k resetování driveru, přivedením logické jedničky je resetování zrušeno. Pin SLEEP k zapnutí driveru, ve výchozím stavu je negovaný, a je tedy nutné pro zapnutí driveru přivést na tento pin opět logickou jedničku.

Posledním pinem tohoto driveru je negovaný pin označený jako FAULT, který při přepnutí do stavu logické nuly může procesoru signalizovat nefunkčnost konkrétního driveru, například z důvodu přehřátí, případně přetížení.

### 3.4.3 Ultrazvukový senzor

O měření vzdálenosti ultrazvukem pojednává také Novák, píše, že princip měření vzdálenosti k překážce je založen na principu měření doby mezi vysláním akustického signálu a přijetím odraženého akustického signálu – echa. Nejběžnější frekvence akustického signálu jsou hodnoty nad 40kHz. Dále popisuje konkrétní principu fungování, díky relativně nízké rychlosti zvuku (ve vzduchu) je doba mezi vysláním a příjemem signálu výrazně vyšší než u radarových, laserových nebo IR senzorů. Proto lze dosáhnout relativně vysoké přesnosti měření i bez extrémních nároků na vyhodnocovací obvody. Díky tomu je jejich cena poměrně nízká, ale perioda měření je vyšší (0,1 s). Nevýhodou je i vysoké tlumení, což omezuje praktický dosah na desítky metrů, běžně do cca 10 m (Novák, 2005).

Modul, který bude použit jako ultrazvukový senzor, je model s označením HC - SR04. Tento modul umožňuje měření od minimální vzdálenosti 2 cm až po 4 m s přesností na 3 mm (ElecFreaks Technology Ltd., 2016). Minimální vzdálenost i přesnost lze v tomto konkrétním použití, na čelistech manipulačního robota, považovat za dostačující.

Rozměry celého modulu jsou  $45 \times 20 \times 15$  mm, veškerá elektronika je umístěna na jednom plošném spoji. Na plošném spoji se nachází ultrazvukový vysílač, přijímač a řídicí jednotka. Připojení je realizováno přes čtyři piny označené jako VCC, Trig, Echo a GND. Pin Trig reaguje na vstupní impuls, pin Echo poskytuje výstupní impuls, pin VCC slouží k napájení celého modulu a GND uzemňuje celý modul. Řídicí jednotka na modulu pouze zajišťuje zpracování vyslaného a přijatého signálu, ostatní výpočty musí být prováděny na externím procesoru.

Tabulka 2: Parametry modulu HC - SR04 (ElecFreaks Technology Ltd., 2016)

Pracovní napětí	DC 5 V
Pracovní proud	15 mA
Pracovní frekvence	40 Hz
Maximální vzdálenost	4 m
Minimální vzdálenost	2 cm
Měřící úhel	15 °
Délka signálu na vstupu Trig	$10\mu\text{s}$ TTL puls
Výstupní signál	Vstup TTL signál a rozsah v proporce
Rozměry	$45 \times 20 \times 15$ mm

Jak uvádí datasheet, po přivedení logické jedničky po dobu  $10\mu\text{s}$  na pin Trig je vysláno osm ultrazvukových pulzů o frekvenci 40kHz. Poté modul čeká na odraz ultrazvukového signálu zpět. Ve chvíli, kdy přijímač zachytí vracející se ultrazvukový

signál, je výstup Echo přepnuto do logické jedničky, která trvá po celou dobu příjmu tohoto signálu. Dobu trvání logické jedničky na výstupu Echo je nutné v řídicí jednotce počítat, z této hodnoty je již možné provést přepočet na jednotky vzdálenosti (ElecFreaks Technology Ltd., 2016).

#### 3.4.4 Mikrospínačové senzory

Novák o těchto senzorech uvádí, že se jedná o nejjednodušší provedení senzoru. Fyzickým dotykem překážky je spínač aktivován a dojde k sepnutí nebo rozepnutí elektrického obvodu a ke změně logické úrovně, která je vyhodnocena. Pokud spínač není aktivní, je na příslušném výstupu, ke kterému je spínač připojený, nastavena logická úroveň jedna, při jeho aktivaci je na výstupu logická úroveň nula (Novák, 2005).

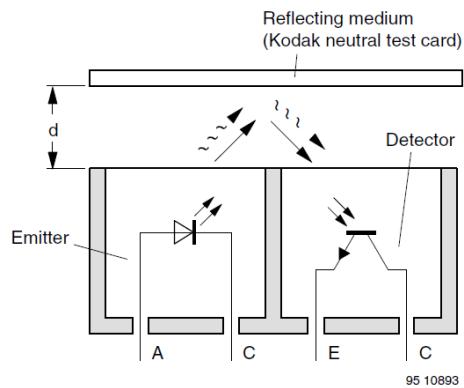
Každý mikrospínač bude zapojen zvlášť na jeden pin mikroprocesoru a jednotlivé piny tak budou v programu vyhodnocovány a zpracovávány dále.

Konkrétně budou použita malá dvou pinová tlačítka. Tato tlačítka budou nalepena přímo na ramena buď do míst, na kterých senzor zareaguje těsně před dotykem fyzické zarážky nebo do míst, ve kterých rameno při přibližování se k mezní pozici dokáže fyzicky senzor stlačit.

#### 3.4.5 Reflexní opto senzory

Reflexní opto senzory budou sloužit jako optické závory pro otáčení celé robotické ruky na základně.

Princip funkce reflexních opto senzorů vysvětluje Novák a spočívá v odrážení infračerveného světla od překážky. Světlo je emitováno infračervenou LED a jako detektor bývá použit fototranzistor citlivý v infračervené oblasti nebo infračervená fotocitlivá dioda. Tento senzor nabývá dvou stavů, kdy senzor buď detekuje, nebo nedetekuje odražený infračervený signál. Tyto senzory je možné využít pouze k detekování překážek v jejich blízkém okolí (Novák, 2005).



Obrázek 7: Princip fungování opto senzoru (CNY70, 2012)

Použité opto senzory mají označení CNY70. Datasheet zmiňuje, že tento senzor má vysílač i přijímač v jednom kompaktním pouzdře, jako přijímač je využívána IR LED a jako detektor fototranzistor, senzor pracuje na vlnové délce 950nm (CNY70, 2012).

## 3.5 Komunikační rozhraní

Jak bylo již dříve uvedeno, pro komunikaci s procesorem se využívá sériové rozhraní. Sériové rozhraní, RS232 je rozhraní pro přenos informací vytvořené původně pro komunikaci dvou zařízení do vzdálenosti 20 m. Přenos informací probíhá asynchronně, pomocí pevně nastavené přenosové rychlosti a synchronizace sestupnou hranou startovacího impulzu (Olmr, 2005).

### 3.5.1 Komunikace po sériové lince

Oficiální webové stránky projektu Arduino zmiňují, že v případě modulu Arduino Micro, sériová komunikace probíhá na pinech označených jako RX a TX, číselně 0 a 1 a využívá TTL logické úrovně 5V nebo 3,3V. Není tedy možné mikroprocesor připojit přímo na rozhraní RS232, toto rozhraní, dle specifikací, pracuje na napětí  $\pm 12$  V (Arduino Serial, 2017). Pro připojení procesoru přímo na port RS232 je nutné využít převodník logických úrovní typu MAX232.

Sériovou komunikaci, příjem a odesílání dat, zajišťuje knihovna Serial, která je součástí programovacího jazyka Arduino.

### 3.5.2 Knihovna pro komunikaci po RS232 v jazyce JAVA

Pro ovládání manipulačního robota je nutností naprogramování knihovny, která umožní komunikovat s tímto robotem prostřednictvím sériového rozhraní. Vhodným jazykem je pro naprogramování univerzální knihovny jazyk Java, jak zmiňuje

Herout Java umožňuje plnou přenositelnost na libovolnou platformu bez nutnosti překladu na této platformě, ke které existuje Java platforma (Herout, 2010).

Knihovna RXTX zajišťuje kompletní obsluhu sériového portu pomocí jazyka Java, tato knihovna existuje ve verzích pro většinu aktuálně používaných operačních systémů.

Hotovou knihovnu je potřeba zabalit do souboru JAR. Jak uvádí dokumentace Javy formát souborů JAR, umožňuje spojit více souborů do jednoho archivu, typicky obsahuje JAR soubor soubory tříd a externích zdrojů, které jsou potřeba pro funkčnost aplikace (Oracle Corporation, 2015). Tento soubor je následně možné importovat do požadované třídy, která bude obsluhovat chod manipulačního robota v návaznosti na vytvářené koncové aplikaci.

## 4 Metodika

Zadání bakalářské práce je nutné rozvrhnout do několika etap, které po jejich kompletním zpracování vytvoří funkční celek. Jelikož s jistotou neexistuje konkrétní řešení tohoto zadání, je nutné vyvinout řešení vlastní. Každý větší problém tvoří soustava problémů menších, pro které již řešení existovat může, případně je nutné v rozdělování problémů pokračovat tak dlouho, dokud pro tyto problémy již řešení existuje. Tato část rozdělení větších problémů na menší části se nazývá dekompozice. Realizace dekomponovaných problémů na sebe může a nemusí navazovat, proto jsou tyto problémy přiřazeny do jednotlivých časových etap a jsou v tomto pořadí také řešeny.

Pomůckou pro zpracování celé modernizace je také volba vhodné platformy, jelikož jednotlivé problémy již mohly být řešeny komunitou uživatelů, případně přímo v oficiální dokumentaci volně dostupné přes internet. Neexistuje-li konkrétní řešení pro konkrétní platformu, je třeba toto řešení převzít nebo vymyslet vlastní.

Vytváření návrhu je nutné podřídit požadavkům, které jsou na celý systém kladený a současně dalším omezením, které se musí při konkrétním řešení brát v úvahu. Dekompozicí tedy vzniká návrh řešení celého problému, pokud je tento návrh proveditelný je možné přejít k realizaci.

### 4.1 Software

Pro programování manipulačního robota bude použito Arduino IDE a NetBeans IDE.

#### 4.1.1 Popis prostředí Arduino

Arduino Integrated Development Environment nebo Arduino Software (IDE) – obsahuje textový editor pro psaní kódu, oblast zpráv, textovou konzoli, panel nástrojů s tlačítky pro běžné funkce a různá menu. Připojuje se k Arduino a Genuino hardware, nahrává programy a komunikuje s nimi (Arduino Software (IDE), 2015).

Přímo v prostředí Arduina je možné vybírat konkrétní mikroprocesor, pro který má být program zkompilován, dále prostředí umožňuje také správu instalovaných knihoven, které mohou být přidány do komplikovaných programů.

#### 4.1.2 Jazyk Java

Jak zmiňuje Herout Java je objektově orientovaný interpretovaný jazyk. Výhodou Javy je plná přenositelnost programů na libovolnou platformu, ke které existuje Java platforma, bez nutnosti jejich překladu na této platformě (Herout, 2010).

Hotovou knihovnu je potřeba zabalit do souboru JAR. Dokumentace Javy soubor JAR popisuje jako formát souboru, který umožňuje spojit více souborů do jednoho archivu, typicky JAR soubor obsahuje soubory tříd a externích zdrojů, které jsou potřeba pro funkčnost aplikace (Oracle Corporation, 2015).

### 4.1.3 Programování v NetBeans

NetBeans IDE umožňuje rychlý a snadný vývoj Java desktopových, mobilních a webových aplikací, stejně tak i HTML5 aplikace s HTML, JavaScriptem a CSS. IDE nabízí sadu nástrojů pro PHP a C/C++ vývojáře. Je zdarma a je open source a má velkou komunitu uživatelů a vývojářů (NetBeans, 2017).

Popis prostředí záleží na konkrétním nastavení zobrazení oken, ovšem obecně lze říci, že největší prostor zaplňuje okno pro zdrojový kód, při psaní kódu pomáhá také našeptáváč, který v NetBeans IDE pracuje na velmi vysoké úrovni. Další části okna jsou ovládací prvky určené pro kompliaci a spuštění vytvářeného programu, průzkumník souborů u aktuálně otevřeného projektu a konzole zobrazující výpis dat ze spuštěného programu.

## 4.2 Metodika vývoje SW

Dle rozsahu projektu a dalších požadavků je zvolena vhodná vývojová metoda software.

### 4.2.1 Iterativní vývoj

Vývoj probíhá v iteracích, kde každá iterace je vlastně malý vodopád složený z etap tzv. pracovních postupů (Rábová, 2008). Dále Rábová uvádí výhody použití této metody, je to včasná eliminace rizik, jednodušší řízení (a akceptovatelnost) změn, znovupoužití, školení týmu během celého projektu a lepší kvalita produktu (Rábová, 2008).

## 4.3 Návrhové vzory

Pecinovský o návrhových vzorech píše, návrhové vzory jsou doporučené postupy řešení často se vyskytujících úloh (Pecinovský, 2007).

### 4.3.1 Singleton

Jedináček specifikuje, jak vytvořit třídu, která bude mít nejvýše jednu instanci. Ta-to instance přitom nemusí být vlastní instancí dané třídy. Jako jedináčci jsou často implementovány různé fondy, zápisníkové paměti, dialogová okna, ovladače různých zařízení a řada dalších objektů. Implementace jedináčka mají jedno společné: definují konstruktor jako soukromý, čímž komukoliv cizímu zabrání ve vytvoření další instance. Požadovanou instanci pak vytvoří uvnitř třídy a odkaz na ni uloží do statického atributu. Jednotlivé varianty implementace se pak liší tím, kdy a jak objekt konstruuje a jak jej mohou ti ostatní získat (Pecinovský, 2007).

## 4.4 Zpracování signálů ze senzorů

Při zpracování signálů ze senzorů je nutné dbát na jejich správnost, počítat s jejich zkreslením, šumem a možnými nepřesnostmi.

### 4.4.1 Mikrospínače

Jedním z problémů u tlačítek, které zmiňuje dokumentace k Arduinu je časté generování falešných přechodů mezi stavy sepnutí/rozepnutí tlačítek, při stisknutí, kvůli mechanickým a fyzikálním problémům: tyto přechody mohou být čteny jako několik stisknutí ve velmi krátkém času a způsobit tak problémy (Arduino Debounce, 2015).

### 4.4.2 Reflexní opto senzory

Tento senzor poskytuje dvouhodnotový signál – detekuje odražený IR signál/nedetekuje odražený IR signál, respektive detekuje překážku/nedetekuje překážku. Nevýhodou IR senzorů pracujících na principu detekce odraženého IR světla je, že množství odraženého světla je závislé na barvě překážky a druhu povrchu (Novák, 2005).

## 4.5 Elektronické součásti

Jednotlivé elektronické moduly budou spojeny pomocí nerozebíratelných a rozebíratelných spojů. Nerozebíratelnými spoji budou pájeny piny na univerzální desky plošných spojů k propojení konkrétních modulů. Pro vytváření nerozebíratelných spojů bude využito pájení naměkkem (Bastian, 2004). U přichycování modulů k základně a konstrukci robota je nutné dbát na izolaci od vodivých částí a také eliminaci elektromagnetického rušení.

## 5 Praktická realizace

### 5.1 Zapojení modulů

Prvním krokem po seznámení s aktuálním stavem manipulačního robota bylo vytvoření seznamu modulů, které budou pro řízení potřebné. Moduly musely být zvoleny na základě specifikací použitych krokových motorů a kompatibility s platformou Arduino.

Před výběrem konkrétního typu Arduina následovalo nastudování minimálních zapojení těchto modulů, které zajistí požadovanou funkci modulu. Dále bylo vytvořeno blokové schéma kompletního zapojení, toto schéma znázorňuje i zjednodušení zapojení modulů k procesoru a je tak možné ušetřit vstupní a výstupní piny. Po provedení zjednodušení zapojení stačí počty požadovaných pinů pouze sečít.

#### 5.1.1 Arduino Micro

Arduino Micro disponuje 14 digitálními vstupy a výstupy a 6 analogovými vstupy. Tyto vstupy je ovšem možné použít také jako digitální vstupy nebo výstupy. Část z digitálních výstupů je navíc vybavena řízením PWM a umožňuje tak jejich využití jako analogových výstupů. Přesněji se jedná stále o digitální výstup, který pulzně šířkovou modulací vytváří různé úrovně napětí. Některé z pinů mají kombinovanou funkci a v případě využití jejich funkce je nutné je z dostupných vstupů či výstupů vyřadit.

Konkrétně jsou rezervovány piny D0 a D1 pro komunikační linku, přes kterou je provozována sériová komunikaci.

#### 5.1.2 Drivery

Zjednodušení lze demonstrovat na použitych čtyřech totožných driverech DRV8825. Každý z driverů vyžaduje pro minimální funkci zapojení pinů DIR a STEP a nastavení stálé logické jedničky na vstupech RESET a SLEEP. Pokud by byl každý z modulů zapojený na Arduino zvlášt, bylo by využito celkem osm výstupních pinů. Z důvodů mechanického zpoždění není nutné, a v případě použití knihovny ani možné, ovládat všechny drivery ve stejný čas, spočívá zjednodušení v paralelním zapojení pinů DIR a STEP všech driverů pouze na dva výstupní piny procesoru a použití pinu ENABLE pro aktivaci jednotlivých driverů. Tímto způsobem zapojení byly na řídicím procesoru ušetřeny dva piny, které je možné využít jinak.

Jak již bylo zmíněno řídicí jednotkou je procesor Atmel ATmega32U4 využívající platformu Arduino. K tomuto procesoru jsou paralelně připojeny čtyři drivery, vstup DIR na pin D8, vstup STEP na pin D9, každý z driverů má pro spuštění zapojený pin ENABLE a ten je zapojen na piny D4, D5, D6, D7. Pro sériovou linku jsou vyhrazeny piny D0 a D1, výstupy z těchto pinů lze zapojit přímo na převodník komunikující na TTL úrovních.

Drivery motorů používají pro napájení krokových motorů stejnosměrné napětí 12V, které je dodáváno z druhého externího zdroje, kladný pól napájení je přímo připojen na pin driveru VMOT a GND.

### 5.1.3 Senzory

Pro ochranu na dorazech šroubovice, u ramen zajišťující vertikální pohyb a následného případného přetěžování motorů jsou použity malé mikrospínáče. Tyto mikrospínáče jsou napájeny na malé desky plošného spoje, které jsou přilepené přímo na ramenech na místa, na kterých mohou být fyzicky stisknuty při přiblížení ramena do krajní pozice. Propojení s mikroprocesorem je realizováno jen pomocí kabelů.

Mikrospínáče spínají do pinů procesoru GND. Výhodou, pro toto konkrétní využití je, že mikrospínáče nemusí být zapojovány přes další rezistory, protože mikroprocesor umožňuje zapnout na vstupech již integrované rezistory, které eliminují parazitní jevy.

Reflexní opto senzory byly použity pro ochranu motoru zajišťujícího otáčení celého robota kolem své osy. Tyto senzory byly umístěny na základnu, do nejkrajnějších možných pozic, co nejblíže velkému ozubenému kolu. Na fyzickou zarázku, která je umístěna na velkém ozubeném kole, byla přilepena malá cuprexitová ploška opatřená povrchem, na který senzory dokáží reagovat a vyhodnotit tak koncovou mezní polohu. Pro předejití parazitních jevů, na které může mikroprocesor reagovat, je nutné pin procesoru uzemnit pomocí vhodného rezistoru a teprve k takto uzemněnému pinu připojit detekční fototranzistor, který je součástí jednotky CNY70.

### 5.1.4 Schéma zapojení modulů

Schéma zapojení se nachází v příloze.

## 5.2 Komunikační protokol

Komunikační protokol je předpis, který zajišťuje komunikaci mezi Arduinem a knihovnou napsanou ve vyšším programovacím jazyce, konkrétně v jazyce Java, který je jednoduše použitelný ve složitějších aplikacích.

Komunikační protokol je duplexní, nejdříve následuje na popis komunikace směrem z počítače do Arduina.

### 5.2.1 Komunikace z počítače do mikroprocesoru

Ovládání krokových motorů je řešeno zasláním datové zprávy. Datová zpráva se skládá z proměnného počtu segmentů, které jsou odděleny dvojtečkami. Přes proměnný počet segmentů si datové zprávy zachovávají stejnou strukturu, které se bude práce věnovat dále. Konec zprávy je značen znakem svislé čáry.

První segment je určen pro řídicí znak, na základě vyhodnocení tohoto segmentu se určuje způsob naložení se zbytkem zprávy. Předdefinované hodnoty včetně

jejich vysvětlení znázorňuje následující tabulka. Povinnost přidat hodnotu k řídicímu znaku je znázorněna v dalších sloupcích.

Tabulka 3: Příkazy pro komunikaci z počítače do mikroprocesoru

Znak	Číslo motoru	Počet kroků	Směr pohybu	Popis
C	Ano	Ano	Ne	Odeslání počtu kroků
S	Ano	Ne	Ne	Dotaz na zbývající počet kroků motoru

Následujícím segmentem po řídicím znaku je číslo motoru. Každý motor má přiděleno své číslo, kterým je jednoznačně identifikován, rozsah tedy vyplývá z celkového počtu motorů a to 0 až 3. V následující tabulce jsou popsány motory, které jsou k jednotlivým číslům přiřazeny. Toto přiřazení je nutné zachovat z důvodu kompatibility s knihovnou pro řízení robota.

Tabulka 4: Označení motorů a směr otáčení

Číslo motoru	Pohyb	Kladné kroky	Záporné kroky
0	Otáčení	Doleva	Doprava
1	Zvednutí ruky	Nahoru	Dolů
2	Přitažení druhého ramene ruky	Nahoru	Dolů
3	Čelisti	Otevřít	Zavřít

Třetí segment, je-li využit, v sobě nese hodnotu o počtu kroků, a to buď počet kroků, které mají být provedeny, nebo počet kroků, které zbývají pro dokončení pohybu. Kroky jsou vyjádřeny kladným, nebo záporným číslem, které současně určuje i směr otáčení daného motoru.

Čtvrtým segmentem je v případě chybové zprávy číslo senzoru, který způsobil přerušení pohybu motoru.

Datové zprávy o pohybu jednotlivých motorů lze zasílat v libovolném pořadí. V procesoru je vytvořena datová struktura, která slouží pro ukládání počtu kroků, které je nutné daným motorem provést. Počty kroků jsou sčítány a je tedy možné počet kroků motoru prodloužit ještě před dokončením aktuálního pohybu, případně naopak motor vrátit do původní pozice zasláním opačné hodnoty.

Motory jsou, i vzhledem ke způsobu jejich zapojení, ovládány střídavě tak, že v jednu chvíli je prováděn úkon pouze jedním z motorů. To je ovšem vzhledem k rychlosti procesoru a zpoždění, které vzniká mechanickým pohybem, naprostě zanedbatelné a pohyb se jeví jako naprostě nepřerušovaný, tudíž i pohyb všech motorů se jeví jako současný.

Na jiné zprávy procesor nereaguje a vrací chybovou zprávu, stejně tak je ošetřena možnost, kdy je zaslaný neplatný identifikátor motoru. Tyto chybové zprávy a veškerá zpětná vazba o pohybu je popsána v další kapitole.

### 5.2.2 Komunikace z mikroprocesoru do počítače

Zpětná vazba z procesoru je řízena stejnými datovými zprávami, o stejné struktuře, jako komunikace opačným směrem, pouze s tím s rozdílem, že ukončení zprávy je realizováno oddělovačem konce řádků \n.

V následující tabulce jsou opět sepsány řídicí znaky, které vysílá procesor na sériovou linku a které je nutné odchytávat a reagovat na ně v počítači.

Tabulka 5: Příkazy pro komunikaci z mikroprocesoru do počítače

Znak	Číslo motoru	Počet kroků	Směr pohybu	Popis
R	Ano	Ano	Ne	Potvrzení přijetí počtu kroků
F	Ano	Ne	Ne	Dokončení pohybu motoru
A	Ano	Ano	Ne	Odpověď na zbývající počet kroků motoru
E	Ano	Ano	Ano	Oznámení, případně chyba ze senzorů

## 5.3 Implementace procesorové části

Programování pro Arduino je možné ve stejnojmenném vývojovém prostředí. Toto prostředí je napsáno v jazyce Java a v novějších verzích umožňuje mnoho různých nastavení kompilátoru pro různé značky a modely mikroprocesorů. Výhodou používání na úkor uživatelského rozhraní, které je oproti konkurenčním IDE značně zjednodušené, je podpora kompilátoru a okamžitého nahrávání na podporované desky, včetně výpisu a možnosti odesílání dat do sériové konzole.

### 5.3.1 Struktura programu

Základní struktura programu pro Arduino je tvořena jedním textovým souborem, který se skládá z importu knihoven, deklarace konstant a proměnných, funkce `setup()` a funkce `loop()`. Funkce `setup()` je spuštěna za celou dobu běhu programu pouze jednou, a to při spuštění programu v procesoru. Příkazy ve funkci `loop()` jsou opakovány v nekonečné smyčce až do vypnutí procesoru, kdy je program ukončen.

Mimo tyto dvě funkce lze definovat funkce vlastní, které mohou být vykonávány na základě volání z funkcí `setup()` nebo `loop()`. Pro přehlednost projektu je možné jednotlivé funkce rozdělovat do souborů podle logických celků, které tvoří.

Další možnosti, která v jazyce Arduino existuje pro usnadnění programování je využívání existujících nebo vytvoření vlastních knihoven. Tyto knihovny jsou

zapisovány a mají stejné chování jako objekty v jazyce C++. Knihovny tedy obsahují konstruktor, metody a vlastnosti. Inicializace probíhá zavoláním konstruktoru, a případně předáním parametrů.

### 5.3.2 Ovládání motorů

Pro ovládání krokových motorů byla zvolena knihovna **StepperDriver**, která byla vytvořena přímo pro použití s krokovými motory typu DRV8825. Všechny 4 motory jsou, pro jednoduchý přístup k instancím, inicializovány na začátku programu do pole, kdy jednotlivé indexy odpovídají číselnému označení motorů. Zdrojový kód obsahuje inicializaci knihovny pro jednotlivé motory.

```
const DRV8825 motors[] = {
    DRV8825 (MOTOR_STEPS, DIR, STEP, 4),
    DRV8825 (MOTOR_STEPS, DIR, STEP, 5),
    DRV8825 (MOTOR_STEPS, DIR, STEP, 6),
    DRV8825 (MOTOR_STEPS, DIR, STEP, 7)
};
```

Jelikož jsou piny **STEP** a **DIR** driverů k sobě připojeny paralelně, je využívána další z funkce driverů, a to vypnutí reakce driveru pomocí pinu **ENABLE**. Podpora pro ovládání tohoto pinu je zahrnuta také v knihovně **StepperDriver**. V inicializační části programu jsou všechny motory nastaveny do stavu „vypnuto“ za pomocí volání funkce **disable()** na vytvořené instance jednotlivých motorů.

Dále je nastavena velikost mikrokroku, která je využívána v knihovně pro správné zasílání signálů do driverů.

Knihovna **StepperDriver** neumožňuje zastavení v průběhu pohybu motoru a bylo tedy nutné samotné řízení zabalit do funkce, která se navíc kromě samotného ovládání postará i o kontrolu a vyhodnocování stavu senzorů.

Informace o pohybu jednotlivých motorů se do řídicího procesoru robotické ruky mohou zasílat střídavě pro všechny motory bez ohledu na jejich aktuální pozici a pohyb. Byla vytvořena datová struktura, která bude udržovat aktuální pozici, s názvem **position**, a také druhá datová struktura, která bude udržovat počet kroků, který má daný motor provést, tato struktura je nazvána jako **offset**. Načítání do datové struktury **offset** zprostředkovává funkce **readData()**, která počet kroků pro daný motor přičte k aktuální hodnotě ve struktuře **offset**.

Samotný pohyb tedy zajišťuje funkce **rotateMotorMultiply()**, která je spouštěna při každém průchodu funkcí **loop()**. Zde je zkонтrolována hodnota ve struktuře **offset** pro jednotlivé motory a je-li rozdílná od 0 je tento počet kroků postupně po jednom kroku proveden.

### 5.3.3 Senzory

Pro zajištění ochrany motorů, zařízení manipulačního robota i okolí je manipulační robot osazen senzory. Výběr, použití a umístění jednotlivých senzorů je činnost, která závisí již na konkrétním využití manipulačního robota. V této práci byly zvoleny pouze základní senzory, zajišťující ochranu krajních stavů pohybu některých částí.

Stav každého ze senzorů je vyhodnocován vlastní funkcí, která pracuje se senzory přímo na nejnižší úrovni, jejíž návratovou hodnotou je logická hodnota. Tato funkce má tedy za úkol vyhodnotit, zda je konkrétní senzor aktivován nebo ne, případně zda byla překročena prahová hodnota, která by značila aktivaci senzoru, bez ohledu na stav dalších senzorů.

Volání funkcí, které kontrolují tyto senzory, probíhá ve funkci `checkSensors()` umístěné v odděleném souboru `sensors.ino`, jejímiž parametry jsou číslo aktivního motoru a směr pohybu. Právě jednotlivým motorům a směru pohybu jsou, v této funkci, přiřazeny požadované senzory, které mají v případě jejich aktivace ukončit pohyb daného motoru. Je tak možno k jednomu motoru a jednomu směru přiřadit i více senzorů, které mají být vyhodnocovány. Předávání informace o směru pohybu motoru je nutné z důvodu ošetření situace, kdy bude motor v jednom směru zastaven reakcí senzoru a následně by se měl začít pohybovat ve směru opačném, pokud by byl senzor přiřazen pouze na základě čísla motoru, nebyl by pohyb opačným směrem možný bez ukládání dodatečný informací přímo do paměti mikroprocesoru.

Zpracování dat z tlačítkových senzorů zajišťuje knihovna `Button`. Tato knihovna poskytuje informace o různých stavech mikrospínačů jako je stisknutí, uvolnění nebo změna stavu tlačítka. Integruje také tzv. debounce, tedy filtraci šumu, který je způsobován mechanickými a fyzikálními vlastnostmi mikrospínačů a je následně mikroprocesorem zachycen. Pro každý z mikrospínačových senzorů je vytvořena instance s nastavením čísla vstupního pinu, na který je tento mikrospínač připojen, dále konstruktor umožňuje také nastavení, zda bude vstupní pin generovat stisknutí po spojení pinu s logickou jedničkou nebo nulou. K tomuto nastavení se používají klíčová slova `PULLUP` nebo `PULLDOWN`. Samotné vyhodnocení je při každé iteraci kontrolováno funkcí `isPressed()` pro daný objekt. Je-li tlačítko stisknuto, nabývá návratová hodnota logické jedničky, není-li stisknuto, je tato hodnota nula.

Opto reflexní senzory jsou obsluhovány přečtením hodnoty pomocí funkce `analogRead()` z konkrétního pinu, ke kterému jsou připojeny. Jelikož se jedná o analogovou hodnotu ze senzoru, je tato hodnota nestálá a mění se mimo jiné na základě okolního prostředí. Převod na logickou hodnotu je tedy realizován podmínkou, kdy pokud je přečtená hodnota pod nastavenou mez, je vyhodnocena jako logická nula a naopak. Toto vyhodnocení dat z opto senzoru je znázorněno v následující ukázce.

```
boolean checkSensor6() {
    int v = analogRead(A4);
    return v > 700;
}
```

Dalším ze senzorů je ultrazvukový senzor, který slouží k měření vzdálenosti čelistí od objektu, ke kterému jsou čelisti přibližovány.

Pro vyhodnocení vzdálenosti objektu od senzoru je použita doba od vyslání zvuku po jeho návrat zpět. Rychlosť šíření zvuku se mění v závislosti na okolních podmínkách, v tomto konkrétním případě bude použita rychlosť pro teplotu okolí 20 °C. Fyzikální tabulky udávají pro teplotu 20 °C rychlosť šíření zvuku ve vzduchu 343m/s (J. Mikuláček, J. Charvát, M. Macháček, F. Zemánek, 2013). Pro výpočet konstanty je nutné tuto hodnotu převést na rychlosť v mm/ $\mu$ s, tedy 0,343mm/ $\mu$ s a následně vydělit dvěma, protože vyslaný signál urazí vzdálenost od vysílače k objektu a od objektu zpět do přijímače, v přepočtu tak bude použita hodnota 0,1715. Následující zdrojový kód provádí obsluhu ultrazvukového senzoru.

```
boolean checkSensor5() {
    digitalWrite(TRIGPIN, LOW);
    delayMicroseconds(2);
    digitalWrite(TRIGPIN, HIGH);
    delayMicroseconds(10);
    digitalWrite(TRIGPIN, LOW);

    float distance = pulseIn(ECHOPIN, HIGH, 30000);
    distance = distance*0.1715f;

    return distance < 100;
}
```

#### 5.3.4 Komunikace

Sériovou komunikaci s mikroprocesorem je možné realizovat zapojením pinů RX, TX a GND na sériový TTL převodník, který komunikuje na úrovních 0V a 5V. V této práci je využitý převodník, který je implementován přímo na desce Arduino Micro a umožňuje tak propojení přímo USB kabelem.

Nastavení mikroprocesoru pro naslouchání na sériové komunikaci je provedeno inicializací knihovny `Serial`, zvolená rychlosť pro komunikaci je 115200 baudů/s.

Zachycení komunikace skrz sériové rozhraní je zajištováno funkcí `readData()`, která je oddělena do vlastního souboru s názvem `serial.ino`. Tato funkce je volána při každém průchodu nekonečné smyčky `loop()`.

Nejdříve je zkонтrolováno, zda existuje komunikace na pinu RX, pokud ano, budou se do proměnné `readString` načítat znaky ze sériového rozhraní dokud nebude sekvence zakončena znakem `\`. Následně se provede parsování přijatých dat za pomocí funkcí `substring()`, `indexOf()` a `toInt()`. Maximální počet segmentů v datové zprávě, kterou procesor přijímá, je tři, zbytek zprávy je zahozen. První segment je uložen do znakové proměnné `command`, druhý segment do proměnné typu `integer servo` a třetí do proměnné stejného typu s názvem `value`. Následuje ukázka zpracování přijatých dat v řídící knihovně.

```
byte index1 = rs.indexOf(':');
byte index2 = rs.indexOf(':', rs.indexOf(':') + 1);

char command = rs.substring(0, index1)[0];
int servo = rs.substring(index1+1, index2).toInt();
int value = rs.substring(index2+1).toInt();
```

Proměnná **servo** symbolizuje číslo motoru a může nabývat hodnot 0 – 3. Držování těchto hodnot je ošetřeno podmínkou. Je-li zasláno neplatné číslo motoru, neprovádí se již žádná další činnost s konkrétní datovou zprávou a je vrácena chybová zpráva.

Následuje vyhodnocení proměnné **command**, která určuje jakým způsobem s proměnnými **servo** a **value** nakládat. Jsou-li všechny hodnoty zaslané v datové zprávě validní, jsou předány konkrétní funkci, která zařídí její zpracování.

První funkcí je funkce **setMotors()**, tato funkce přebírá dva parametry a to číslo motoru a hodnotu kroků. Hodnota kroků je přičtena do struktury **offset** a odeslána zpět do řídicího počítače potvrzovací zpráva.

Druhou funkcí je funkce **sendInfo()**, funkce vyžaduje jako jeden parametr číslo motoru. Na základě tohoto čísla je odeslána datová zpráva obsahující zbývající počet kroků v **offsetu**.

Funkce **sendFinish()** není volána přímo při vyhodnocení přijaté zprávy, ale ve chvíli dokončení pohybu motoru. Jediným parametrem je číslo motoru, u kterého byl pohyb dokončen. Pohyb motoru se považuje za ukončený ve chvíli, kdy je hodnota offsetu daného motoru rovna nule. Jelikož je nová hodnota offsetu přičítána k původní hodnotě, je nutné data odesílaná touto funkcí vyhodnocovat v řídicím počítači, aby bylo možné provést sekvenci pohybu a jednotlivé pohybové informace na sebe navazovaly.

## 5.4 Implementace knihovny

Pro ovládání robotické ruky je možné použít jakýkoliv software umožňující zasílání, přijímání a vyhodnocování datových zpráv na sériovou linku. Součástí bakalářské práce bylo vytvořit knihovnu, která bude umožňovat řízení a kterou bude možné implementovat v libovolném programu napsaném v jazyce Java.

### 5.4.1 Knihovna RXTX

Jak bylo zmíněno v teoretické části práce, je pro komunikaci po sériové lince využita knihovna RXTX. Tato knihovna, která byla dále vyvíjena firmou Mfizz Inc., je v poslední verzi 2.2 z roku 2008 a je dostupná pro platformy Windows a Linux. Tato knihovna je kompatibilní i s novými verzemi Javy.

Použití této knihovny by mělo zaručovat, že řídicí knihovna bude moci využívat konzistentní rozhraní pro přístup k sériovým portům a bude se chovat na všech počítačích s různými operačními systémy stejně.

RXTX je po přidání do projektu načtena importem balíku `gnu.io`. Balík obsahuje mimo jiné třídu `CommPortIdentifier`, která je použita pro vytvoření instance a připojení se k sériovému portu. Statickou metodou `getPortIdentifiers()` je vrácen výčet dostupných portů a jejich identifikátorů, tyto identifikátory jsou použity pro následné otevření komunikace se sériovým portem.

V konstruktoru třídy `NESADriver` je toto připojení vytvořeno na základě zvoleného portu a rychlosti komunikace. Sériový port může v jednu chvíli obsluhovat pouze jedna aplikace, metodou `isCurrentlyOwned()` tedy probíhá ověření, zda je port možné otevřít, nebo je již využíván. Následně je port otevřen za použití metody `open()` a instance tohoto portu je přiřazena do členské proměnné `serialPort`. Následuje nastavení parametrů komunikace za pomocí funkce `setSerialPortParams()`, touto funkcí je nastavena rychlosť komunikace, počet datových bitů, typ stop bitu a použití parity. Konkrétně je počet datových bitů stanoven na 8, jako stop bit se používá 1 bit a parita není použita. Po úspěšném vytvoření instance portu a jeho inicializaci jsou dostupné vstupní a výstupní streamy portu, kterými již probíhá samotné odeslání a přijetí dat za pomocí metod, které jsou k instancím streamu v Javě dostupné.

Pro nepřetržitou obsluhu vstupního streamu a možnost reagování na zpětnou vazbu, která jsou zaslána z mikroprocesoru do počítače, je tento stream předán třídě `SerialReader`, která je spuštěna ve vlastním vlákně. Toto vlákno tedy čeká v nekonečné smyčce na data přicházející do vstupního streamu. Data získávána ze vstupního streamu jsou ve formě posloupnosti bajtů převáděny na znaky a ukládány do proměnné typu `String`. Pokud jsou data ukončena znakem konec řádku je tento konec řádku z řetězce odstraněn a řetězec je předán funkci `parseData`, o této funkci a její činnosti se zmiňuje kapitola Zpracování přijatých dat.

Ukončení komunikace a uvolnění sériového portu probíhá na základě zavolání funkce `close()`.

Třída `NESADriver` dále zajišťuje ověření, zda je port otevřen a připraven pro komunikaci. Aby nevznikala situace, kdy je do jednoho motoru zasíláno současně několik informací a nepočká se na zpracování zprávy v procesoru, je v poli typu `boolean motorProgress` udržována informace, zda je možné aktuálně zaslat do jednotlivých motorů informace, nebo zda ještě nebyla data zpracována a je tedy nutné počkat. Ověření takového stavu motoru je možné zjistit metodou `inProgress()`, jejímž parametrem je právě číslo ověřovaného motoru.

#### 5.4.2 Kolekce dat

V datových strukturách knihovny jsou také ukládány počty kroků jednotlivých motorů v postupném pořadí. Tyto struktury jsou uloženy v instanci třídy `NESAData`, kde se o konkrétní implementaci stará struktura `ArrayList`. Tato struktura umožňuje zpracování různých směrů pohybu jednotlivých motorů, ovšem nezajíšťuje vyčkávání na vykonání činnosti, která není závislá na pohybu manipulačního robota, tzn. v případě, že je třeba rameno nastavit na určitou pozici a vyčkat na dokončení čin-

nosti jiného zařízení, je tuto skutečnost nutno řešit přímo v programu, ve kterém je knihovna implementována.

Třída `NESAData` je zpracována dle návrhového vzoru singleton, jelikož v jednu chvíli může pouze jedna instance a obecně pouze jeden program přistupovat k sériovému portu. V konstruktoru této třídy se inicializují datové struktury a vytváří nové vlákno, které tyto datové struktury následně zpracovává a předává instanci třídy `NESADriver`.

Po vytvoření instance je umožněno přidávat počty kroků pro jednotlivé motory za použití metody `addData(int i, int value)`, kde prvním parametrem je číslo motoru, pro které je určen počet kroků, který je reprezentován druhým parametrem. Alternativně lze využít metodu `addCoordinate(int x, int y, int z)`, která slouží pro zdávání souřadnic, na které se má rameno s čelistmi manipulačního robota přesunout. Tato metoda zajistí přepočet souřadnic na počet kroků, které mají být vykonány jednotlivými motory.

Další metodou je metoda `getMotorData(int i)`, která dle zadaného parametru vrátí počet kroků pro konkrétní motor, tento počet kroků je zaslán řídicímu mikroprocesoru, který jej následně zpracuje.

Poslední metodou této třídy je metoda `isNotEmptyQueue(int i)`, tato metoda vrací logickou jedničku, pokud ve struktuře náležící motoru, která je určena parametrem `i`, existuje nějaká hodnota, která může být zaslána řídicímu mikroprocesoru.

V návaznosti na třídu `NESAData` je ve vlastním vlákně vytvořena instance třídy `SendToDriver`. Tato třída musí implementovat metodu `run`, která je také jedinou metodou, která je ve třídě obsažena. V metodě běží nekonečná smyčka, která zajistuje okamžitě samotné předávání dat z datové struktury do instance třídy `NESADriver`. Pro zahájení předávání dat je nutné, aby byl korektně připojen řídicí mikroprocesor a aby odesal úvodní informaci o vyčkávání na příjem dat. Tuto skutečnost ověřuje metoda `isReady()`.

#### 5.4.3 Příprava dat pro odeslání

Odesílání dat do sériového portu probíhá formou zápisu pole bajtů do výstupního streamu portu. Metoda `sendData(int motor, int steps)` ve třídě `NESADriver` přebírá dva parametry, a to číslo motoru a počet kroků, který má daný motor provést. Tyto hodnoty jsou doplněny do datové zprávy odpovídající požadavkům procesoru. Této datové zprávě se věnovala již jedna z předchozích kapitol. Datová zpráva je za použití funkce `getBytes()` s parametrem `StandardCharsets.UTF_8` přetrasformována z běžného řetězce na pole datového typu `byte`. Následující ukázka kódu obsahuje přípravu a odeslání dat do výstupního streamu portu.

```
public boolean sendData(int motor, int steps) throws IOException {
    if(steps == 0) return false;
    if(!this.readyState) return false;
    this.motorProgress[motor] = true;
    this.os.write( ("C:" + motor + ":" + steps + "|")
```

```
    .getBytes(StandardCharsets.UTF_8) );
    return true;
}
```

Programovací jazyk Java pracuje s kódováním Unicode, naopak mikroprocesor ATmega32U4 podporuje komunikaci pouze na bázi jednobajtových hodnot a zpracování vícebajtových hodnot by bylo výpočetně zbytečné náročné. Při převádění řetězce na pole bajtů je použit parametr `StandardCharsets.UTF_8`. Tento parameter zajišťuje, že hodnoty, které je možné zapsat na jeden bajt, dle ASCII tabulky, jsou takto doopravdy převedeny a procesor je tedy dokáže zpracovat.

Následně je toto pole zapsáno do výstupního streamu za pomocí funkce `write()`, o samotné odeslání na sériový port se již postará zmínovaná knihovna RXTX.

#### 5.4.4 Zpracování přijatých dat

Zpracování dat, která byla přijata zpět z mikroprocesoru, probíhá ve funkci `parseData()`. Tato funkce je součástí vnořené třídy `SerialReader`, jejíž instance je spuštěna v samostatném vlákně a data tedy mohou být zpracována ihned po přijetí z mikroprocesoru.

Funkce `parseData(String data)` přebírá celý přijatý řetězec, který je ošetřen odebráním netisknutelných a řídicích znaků. Následně je tento řetězec rozdělen na jednotlivé segmenty, dle popisu komunikačního protokolu, který se nachází v kapitole Komunikace z Arduina do počítače. Přijatý řetězec se skládá z hodnot, které jsou odděleny oddělovačem `:`, lze tedy pro rozdělení na jednotlivé segmenty využít funkce `split()` právě se znakem dvojtečky jako parametrem. Rozdělením vznikne pole řetězcových hodnot, kde jsou jednotlivé prvky pole seřazeny dle pořadí v původním řetězci. Následně je první segment za pomocí funkce `equals()` porovnán s předem definovanými klíčovými znaky. Dle těchto řídicích znaků jsou, s dalšími prvky pole, provedeny konkrétní úkony. V případě nutnosti jsou řetězcové hodnoty převedeny na celočíselné za použití funkce `parseInt()`.

#### 5.4.5 Rozhraní řídicí knihovny

Rozhraní knihovny tvoří několik metod, které je možné volat z programu, který bude tuto knihovnu implementovat. Rozhraní obsahuje metody informační a výkonné.

Informační metody vrací různé stavové informace o úspěšném připojení a navázání spojení s řídicí jednotkou, o aktuální pozici motorů dle vnitřního stavu řídicí jednotky. Dále o tom, zda je určitý motor stále v pohybu, nebo již svoji činnost dokončil.

Výkonnými metodami jsou myšlené metody, které zasílají konkrétní pokyny do řídicího mikroprocesoru. Základní metodou je metoda `sendData()`, která zajišťuje odeslání příkazu obsahující identifikaci motoru a počet kroků, které má daný motor vykonat ve tvaru např. `C:2:100|`.

Dále knihovna umožňuje i zpracování dat ve formě souřadnic, které jsou pře-počítány na počty kroků, které má každý motor provést.

Osy pohybu jsou určeny tak, že souřadnice x a y obsluhují obě horní ramena a pro osu z je určen rotační pohyb celého robota. Souřadnice osy x a y jsou zadávány v milimetrech, souřadnice osy z ve stupních. Převodní koeficienty z délkových a úhlových jednotek na kroky jednotlivých motorů byly určeny na základě měření a výpočtů.

Počet kroků pro pohyb o  $1^\circ$  na ose z, byl určen změřením počtu kroků pro otočení o  $90^\circ$  a následným výpočtem z této naměřené hodnoty. Pohyb na osách x a y byl spočítán stejným způsobem. Pohyb zajišťují motory 1 a 2, tyto motory pouze ovlivňují úhly mezi základnou a prvním a druhým ramenem, je tedy nutné pro zadané souřadnice použít přepočet na velikost těchto dvou úhlů. Po výpočtu úhlů, o který mají motory ramena posunout, jsou tyto úhly pře-počítány na počty kroků.

Tabulka 6: Počty kroků pro posun ramen o  $1^\circ$

<b>Motor</b>	<b>Počet kroků</b>
0	11,223
1	42,02
2	52,356

Přepočet zajišťuje funkce `calcDegrees(Coordinates coordinate)`, následně jsou počty kroků matematicky zaokrouhleny na celá čísla.

Tento systém určování polohy souřadnic ovšem nelze považovat za naprostě ideální, v návrhu řídicí knihovnami se s možností úprav těchto výpočtů počítá a tudíž nebudou časově náročné.

Pro sevření nebo rozvření čelistí jsou připraveny metody `closeJaws()` a `openJaws()`, které mají nastavený statický počet kroků, které mají pro sevření vykonat. Jelikož robot není vybaven senzory v čelistech, musí být počet kroků určen při konkrétní aplikaci, nebo opakováním použitím těchto metod.

V následující tabulce jsou popsány jednotlivé metody, které je možné využít, včetně jejich činnosti, parametrů a návratových hodnot.

Tabulka 7: Příkazy pro komunikaci z mikroprocesoru do počítače

Návratová hodnota	Název metody	Parametry	Popis
void	NESADriver(String port, int rate)	String port – identifikátor portu int rate – rychlosť komunikace s řídicí jednotkou	Konstruktor třídy, který zajistí otevření připojení ke konkrétnímu portu.
void	close()		Metoda ukončí připojení do sériového portu a uvolní jej pro připojení jiné aplikaci.
boolean	inProgress(int motor)	int motor – určuje číslo motoru	Vrací logickou hodnotu vypovídající o tom, zda se motor, určený předaným číslem v parametru, pohybuje, nebo nikoliv. Logická hodnota true značí, že je motor aktuálně v pohybu.
boolean	isReady()		Pokud je počítače připojen k řídicí jednotce, je vrácena logická hodnota true.
Enumeration	listPorts()		Výčet dostupných portů v počítači.
void	sendData(int motor, int steps)	int motor – určuje číslo motoru int steps – počet kroků, která má motor provést	Připraví a odešle počet kroků do konkrétního motoru.
void	getActualState(int motor)	int motor – určuje číslo motoru	Dotáže se řídicí jednotky na konkrétní stav pohybu motoru.

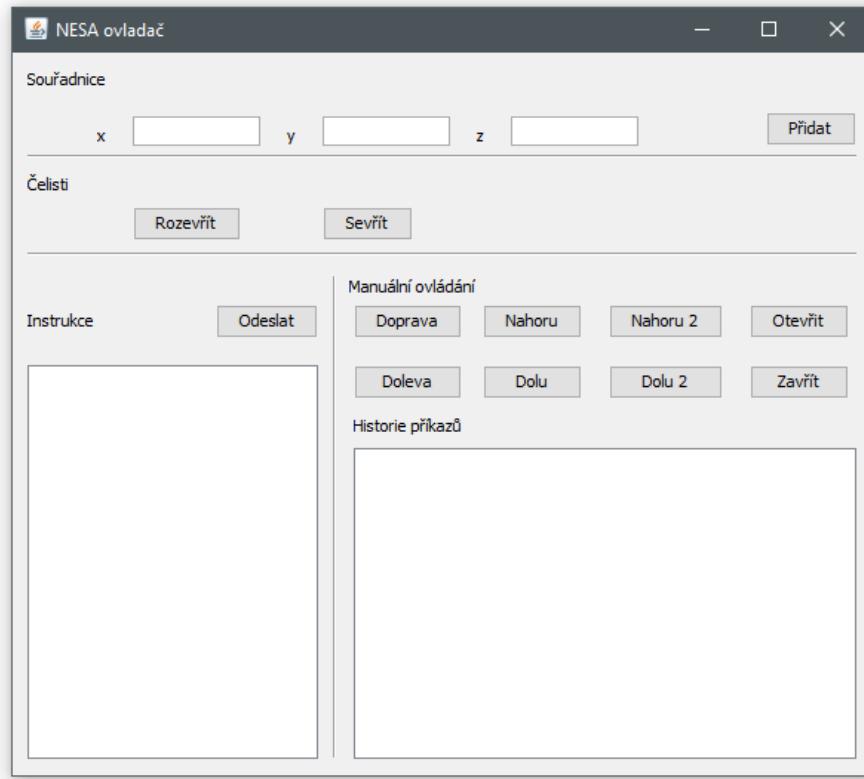
#### 5.4.6 Grafické rozhraní

Pro demonstraci a otestování řídicí knihovny bylo vytvořeno i grafické rozhraní, kterým je možné manipulačního robota ovládat. Toto grafické rozhraní využívá grafického enginu Swing, který je součástí jazyka Java.

Grafická reprezentace slouží pro základní demonstraci ovládání manipulačního robota, umožňuje přímé ovládání stiskem tlačítka i vytváření posloupností, které jsou zpracovávány postupně.

V horní části okna, je možné nastavit souřadnice, na které se mají přesunout čelisti robotické ruky a následně je přidat do seznamu. Dále je možné do seznamu přidat rozevření a sevření čelistí. Tato posloupnost se zobrazuje v okně označeném jako Instrukce, zda je možné jednotlivé instrukce mazat a následně kliknutím na tlačítko Odeslat odesílat posloupnost příkazů do manipulačního robota.

V dolní pravé části okna jsou tlačítka pro manuální, okamžité ovládání robotické ruky. Každá z provedených instrukcí je zaznamenána v okně Historie příkazů.



Obrázek 8: Grafické okno pro ovládání manipulačního robota

## 6 Zjednodušené ekonomické zhodnocení

Při každé modernizaci je nutno dbát na to, zda se modernizace vyplatí s ohledem na cenu pořizovaných součástí a údržbu v porovnání s cenou za nový produkt, u kterého je navíc samozřejmostí záruka a určitá podpora ze strany výrobce.

V této konkrétní modernizaci bylo rozhodnuto, že všechno elektronické vybavení mimo krokové motory bude nahrazeno za nové. Původní krokové motory zůstaly zachovány, jelikož byly stále funkční a jednalo by se o jednu z nejdražších položek při modernizaci. Celá finanční zátěž modernizace se tak znatelně snížila.

Pro tuto konkrétní modernizaci byla zvolena relativně levná technologie, která vyniká právě svou cenou za výkon. Dále modernizace spočívala ve využití univerzálních modulů a mikroprocesoru na platformě umožňující jednoduchý vývoj a také následné úpravy již existujícího programu.

Použitý materiál včetně cen, za které byl nakoupený, znázorňuje následující tabulka.

Tabulka 8: Ceny použitého materiálu

Název	Počet kusů	Celková cena
Arduino Micro	1	557,-Kč
Driver DRV8825	4	396,-Kč
Ultrazvukový senzor HC-SR04	1	74,-Kč
Mikrospínáče senzorů	4	23,-Kč
Reflexní optočlen CNY70	2	50,-Kč
<b>Celkem</b>		<b>1 100,-Kč</b>

Celková cena použitého materiálu činí 1 100,-Kč. Tato cena není konečná a je nutné ji rozšířit také o cenu práce, která byla věnována praktické realizaci této práce. Celková realizace, konkrétně, vytvoření návrhu, otestování původních motorů, nahrazení elektroniky, osazení senzory, vytvoření programu pro řídicí mikroprocesor a otestování bylo zapotřebí práce zhruba na 70 hodin. Budeme-li brát v úvahu pouze hrubou mzdu na pozici Programátor PLC dle serveru platy.cz v dubnu 2017 pobírá hodinovou sazbu 200,-Kč (Profesia CZ, s.r.o., 2017). Hrubá cena za práci by vycházela na 14 000,-Kč. Celková suma za modernizaci by se tedy pohybovala kolem 15 100,-Kč.

I přes toto navýšení ceny se modernizace vyplatila a přináší mnoho dalších výhod oproti zakoupení nového produktu, které nemusí být na první pohled zřejmé. Na manipulačního robota sice není poskytována žádná záruka, ale nyní k robotovi existuje dokumentace, která umožňuje provádění dílčích úprav a zároveň je robot postaven na dostupných a otevřených technologiích, které je možné upravit dle dalších požadavků.

Pro srovnání ceny za modernizaci s pořízením podobného nového manipulačního robota je možné jmenovat například robota, který je nabízen pouze osazený krokovými motory za cenu 4 695,-Kč od firmy ECLIPSERA s.r.o. (ECLIPSERA,

s.r.o., 2017). Do tohoto modelu je nutné opět vytvořit řídicí jednotku a cena se tak může zvýšit na částku srovnatelnou s částkou za provedenou modernizaci. Další kategorií jsou již profesionální roboti určení pro výukové účely, například od firmy FANUC, které ovšem svou cenou cenu za modernizaci znatelně převyšují.

## 7 Diskuse

Hlavním z cílů bakalářské práce bylo praktické provedení modernizace manipulačního robota.

Možností, kterými lze zadání této bakalářské práce řešit, jistě existuje velké množství. Ty se navzájem budou lišit jak ve volbě hardwaru, tak v softwarovém zpracování programu řídicího procesoru nebo softwaru řídicí knihovny.

Jedním z požadavků bylo osazení robota senzory. Osazení senzorů je velmi subjektivní záležitost a vždy záleží na konkrétním použití manipulačního robota. Aktuální počet použitých senzorů by bylo možné, dle konkrétních potřeb, rozšířit například o tlakové senzory v čelistech nebo o senzory detekující překážky při pohybu robotické ruky. Tím by byla mnohem více podpořena autonomita celého systému.

Při rozšiřování systému o další senzory, bude, z důvodu omezeného počtu vstupních a výstupních pinů na desce Arduino Micro, nutné využít obvod, sloužící jako multiplexor. Tento obvod zvýší počet vstupních a výstupních pinů, které bude možné skrze Arduino řídit.

Dalším rozšířením, které může být implementováno, je možnost ovládat manipulačního robota pomocí protokolu TCP/IP potažmo přímo protokolem HTTP, kdy se napojení do sítě provede ethernetovým rozhraním zapojeným přímo na mikroprocesor. Možností pro konkrétní realizaci existuje několik, řídicí jednotka může vystupovat jako server nebo se může periodicky dotazovat jiného serveru na aktuální příkazy, které mají být provedeny. Toto rozšíření by vyžadovalo pouze vytvoření funkce, která zajistí příjem dat, která mají být zpracována, a následně drobné úpravy již existujícího programu uloženého v řídicí jednotce.

Samotnou kapitolou, kterou lze dále upravovat a vylepšovat, je aplikace pro řízení manipulačního robota z počítače. Tato aplikace umožňuje v současné době pouze základní ovládání robota vytvořením posloupnosti pozic, mezi kterými má robot přecházet. Aplikaci by bylo možné rozšířit o vylepšené ovládání, propracovanější nastavení, lepší zacházení se souřadnicemi a například vizualizaci pohybu robota v prostoru.

Modernizovaného manipulačního robota bude moci využívat obor Automatizace řízení a informatika, při praktických cvičeních například v předmětech Měřicí systémy II nebo Technická kybernetika. Může tak být rozšířena výuka týkající se manipulační techniky.

## 8 Závěr

Cílem bakalářské práce bylo provést modernizaci manipulačního robota včetně její praktické realizace, vytvoření komunikačního protokolu a implementace tohoto protokolu do řídicí knihovny.

Před započetím činnosti na samotné modernizaci bylo provedeno seznámení s aktuálním stavem a původním využitím manipulačního robota. Dle použitých součástek se jedná o robota se stářím převyšujícím deset let a způsob využití je možné odhadnout jako učební pomůcku při praktických cvičeních. Pro rozhodnutí o zachování krokových motorů na robotovi bylo provedeno také testovací zapojení těchto motorů a úspěšné ověření jejich funkčnosti.

V bakalářské práci byl zpracován rozbor aktuálně dostupné literatury a zdrojů, vztahující se k tématu robotiky, elektrotechniky, programování mikroprocesorů a programování aplikací, díky kterému byl zpracován návrh řešení celého problému. Na základě tohoto rozboru mohly být pro samotnou modernizaci použity nejnovější technologie včetně využití zkušeností ostatních autorů, kteří se již, v různých oblastech, touto problematikou zabývali.

Návrh modernizace byl zpracován s důsledností na použití moderních a levných technologií, které byly a jsou v současné době kompatibilní s mnoha ostatními zařízeními tohoto druhu. Ovládání mohlo být řešeno od naprostého počátku a nemusel tedy být brán ohled na původní řízení, protože toto řízení bylo kompletně nahrazeno za řízení nové. Pro modernizaci byla zvolena platforma Arduino, kterou lze využít na mnoha různých mikroprocesorech, konkrétně verze Arduino Micro, které jako hlavní výpočetní jádro využívá mikroprocesoru Atmel ATmega32U4. Dále byly zvoleny a použity vhodné moduly pro ovládání samotných krokových motorů, komunikaci s počítačem a zvoleny vhodné senzory.

Pro komunikaci mezi řídicí jednotkou a počítačem byl navržen komunikační protokol, a to opět s důrazem na co nejvyšší jednoduchost a nízké zatížení přenosových kanálů. Současně má také možnost následného rozšiřování o další příkazy nebo návratové kódy.

Praktická část byla zaměřena již na samotnou realizaci této modernizace. Při realizaci bylo nutné provést zapojení řídicího mikroprocesoru, motorů a modulů. Tato část byla kompletně splněna. Senzory byly osazeny v minimálním možném počtu, alespoň pro zabezpečení pohybu ramen.

Naprogramování řídicího procesoru lze samozřejmě řešit mnoha různými způsoby. Zvolen byl opět co nejjednodušší přístup, aby jakékoli další úpravy tohoto programu mohl provádět kdokoliv po seznámení s programem. Navíc se tento přístup snaží také o co nejmenší výpočetní zátěž mikroprocesoru. Mikroprocesor se stará pouze o přijetí pokynů z komunikačního kanálu, jejich zpracování a předání do datových struktur. Zpracováním dochází již k samotnému odesílání pokynů do driverů motorů. Dále mikroprocesor zajišťuje také odeslání návratových hodnot, a to po dokončení pohybu motorů, anebo také v případě zastavení jejich pohybu reakcí

některého ze senzorů. Při programování a současném testování naprogramovaných bloků nedocházelo k žádným nekorektním situacím.

Nedílnou součástí bylo také praktické realizování celého návrhu modernizace a následné ověření správného fungování. Při praktické realizaci tak byly odstraněny i některé drobné chyby v návrhu, které nemusely být ihned zřejmé.

Pro usnadnění komunikace s manipulačním robotem z řídicího počítače byla vytvořena knihovna v jazyce Java. Tato knihovna usnadňuje připojení a zaslání instrukcí. Programátor, který by knihovnu implementoval do jakékoliv další aplikace, je tak odstíněn od komunikačního protokolu, který manipulační robot využívá. Dle této knihovny je také možné napsat knihovnu pro jakýkoliv jiný programovací jazyk.

Výhodou této modernizace je možnost jakýchkoli softwarových úprav v programu řídicí jednotky, které zajistí, že je možné v případě nevyhovující logiky tuto logiku libovolně upravit nebo změnit. Dále využití univerzálního komunikačního protokolu poskytuje možnost vytvořit knihovnu pro přístup k ovládání robotické ruky pro jakýkoliv jiný programovací jazyk, který umožňuje komunikaci skrze sériové rozhraní.

Závěrem lze tedy konstatovat, že cíl bakalářské práce byl splněn, realizovaná modernizace je funkční a splňuje všechny požadavky, které byly kladené v zadání této práce.

## 9 Literatura

ARDUINO *Arduino - ArduinoBoardMicro* [online]. 2017 [cit. 2017-03-25]. Dostupné z:  
<https://www.arduino.cc/en/Main/arduinoBoardMicro>.

ARDUINO *Arduino - Debounce* [online]. 2015 [cit. 2017-03-21]. Dostupné z:  
<https://www.arduino.cc/en/tutorial/debounce>.

ARDUINO *Arduino - Environment* [online]. 2015 [cit. 2017-04-01]. Dostupné z:  
<https://www.arduino.cc/en/Guide/Environment>.

ARDUINO *Arduino - Serial* [online]. 2017 [cit. 2017-03-25]. Dostupné z:  
<https://www.arduino.cc/en/reference/serial>.

BASTIAN, P. A KOL. *Praktická elektrotechnika* Praha: Europa-Sobotáles, 2004  
ISBN 80-86706-07-9.

ECLIPSERA, s.r.o. *Arduino robotická ruka* [online]. 2017 [cit. 2017-04-22]. Dostupné z:  
<http://arduino-shop.cz/arduino/978-arduino-roboticka-ruka-1424623023.html>.

ELECFAKES TECHNOLOGY LTD. *HC-SR04 User Guide* Shenzhen: ElecFreaks Technology Ltd., 2016.

HEROUT, P. *Učebnice jazyka Java* České Budějovice: Nakladatelství KOPP, 2010  
ISBN 978-80-7232-398-2.

MIKULÁČEK, J., CHARVÁT, J., MACHÁČEK, M., ZEMÁNEK, F. *Matematické, fyzikální a chemické tabulky a vzorce pro střední školy* Praha: Prometheus, spol. s. r. o., 2013  
ISBN 978-80-7196-264-9.

NOVÁK, P. *MOBILNÍ ROBOTY - pohony, senzory, řízení* Praha: Nakladatelství BEN, 2005  
ISBN 80-7300-141-1.

OLMR, V. *HW server představuje - Sériová linka RS-232 / Vývoj.HW.cz* [online]. 2005 [cit. 2017-04-28]. Dostupné z:  
<http://vyvoj.hw.cz/rozhrani/hw-server-predstavuje-seriova-linka-rs-232.html>.

ORACLE CORPORATION *Lesson: Packaging Programs in JAR Files* [online]. 2015 [cit. 2017-04-12]. Dostupné z:  
<https://docs.oracle.com/javase/tutorial/deployment/jar/>.

ORACLE CORPORATION *NetBeans IDE - Overview* [online]. 2017 [cit. 2017-04-01]. Dostupné z:  
<https://netbeans.org/features/index.html>.

PECINOVSKÝ, R. *Návrhové vzory* Brno: Computer Press, 2013  
ISBN 978-80-251-1582-4.

PROFESIA CZ, s.r.o. *Plat - Programátor PLC - Platy.cz* [online]. 2017 [cit. 2017-04-22]. Dostupné z:  
<http://www.platy.cz/platy/elektrotechnika-a-energetika/programator-plc>.

RASPBERRY PI FOUNDATION *GPIO - Raspberry Pi Documentation* [online]. 2017 [cit. 2017-04-13]. Dostupné z:  
<https://www.raspberrypi.org/documentation/hardware/raspberrypi/gpio/README.md>.

RASPBERRY PI FOUNDATION *Raspberry Pi - Raspberry Pi Hardware Guide requirements / Raspberry Pi Learning Resources* [online]. 2017 [cit. 2017-04-13]. Dostupné z:  
<https://www.raspberrypi.org/learning/hardware-guide/components/raspberry-pi/>.

RÁBOVÁ, I. *Podnikové informační systémy a technologie jejich vývoje* Brno: Tribun EU, 2008  
ISBN 978-80-7399-599-7.

SUCHÁNEK, M. *Arduino UNO WiFi vs. NodeMCU / Arduino.cz* [online]. 2016 [cit. 2017-04-15]. Dostupné z:  
<https://arduino.cz/arduino-uno-wifi-vs-nodemcu/>.

TEXAS INSTRUMENTS INCORPORATED *DRV8825 Stepper Motor Controller IC* Dallas: Texas Instruments Incorporated, 2016.

VISHAY SEMICONDUCTORS *CNY70* Selb: Vishay Semiconductors, 2012.

VODA, Z. *Průvodce světem Arduina* Bučovice: Martin Stříž, 2015  
ISBN 978-80-87106-90-7.

## Seznam obrázků

Obrázek 1: Původní stav	10
Obrázek 2: Označení motorů	11
Obrázek 3: Původní elektronika	12
Obrázek 4: Arduino Micro (Voda, 2015)	15
Obrázek 5: Schéma pinů Arduino Micro (Arduino - ArduinoBoard-Micro, 2017)	16
Obrázek 6: Zjednodušené schéma driveru (Texas Instruments, 2016)	17
Obrázek 7: Princip fungování opto senzoru (CNY70, 2012)	20
Obrázek 8: Grafické okno pro ovládání manipulačního robota	38
Obrázek 9: Schéma zapojení	49
Obrázek 10: Mikrospínačové senzory	50
Obrázek 11: Opto senzory	50
Obrázek 12: Mikrospínačové senzory	51
Obrázek 13: Elektronika umístěná v základně	51
Obrázek 14: Celkový stav manipulačního robota po modernizaci	52

## Seznam tabulek

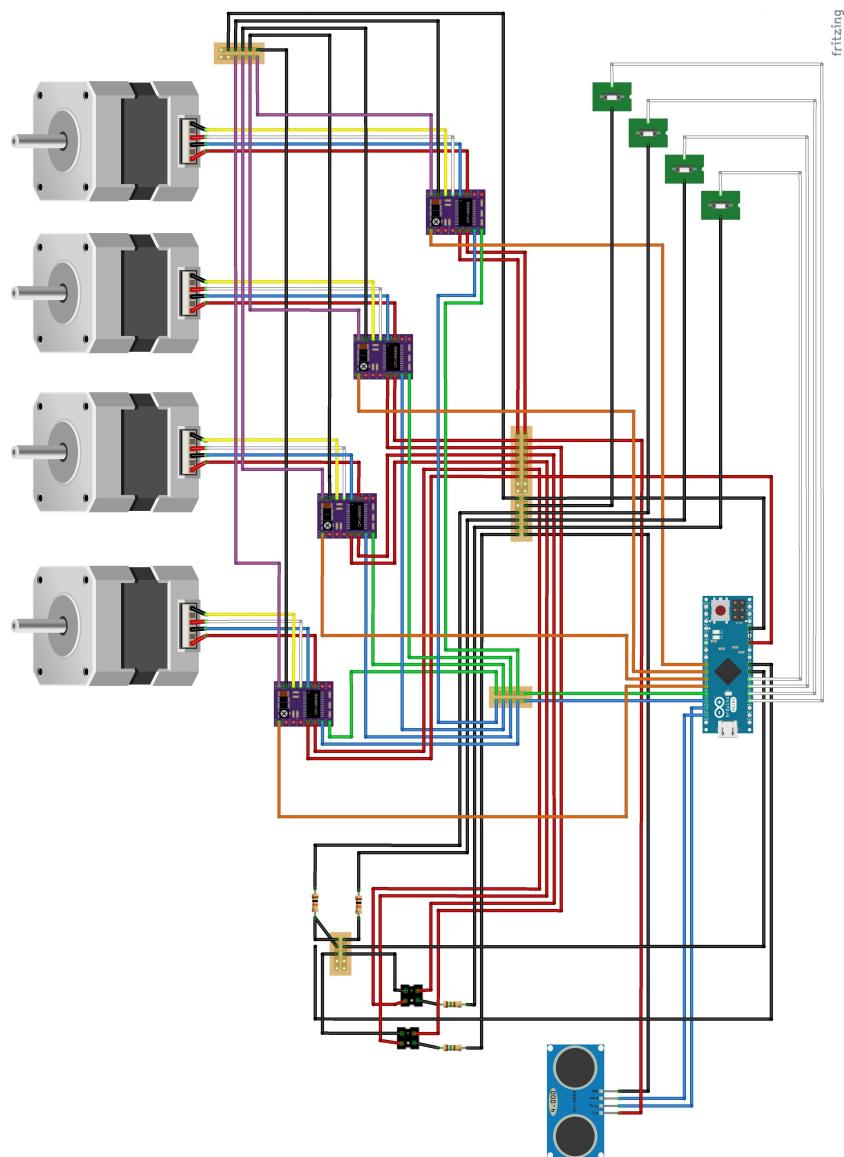
Tabulka 1: Parametry Arduino Micro (Arduino - ArduinoBoardMicro, 2017)	16
Tabulka 2: Parametry modulu HC - SR04 (ElecFreaks Technology Ltd., 2016)	18
Tabulka 3: Příkazy pro komunikaci z počítače do mikroprocesoru	27
Tabulka 4: Označení motorů a směr otáčení	27
Tabulka 5: Příkazy pro komunikaci z mikroprocesoru do počítače	28

---

<b>Tabulka 6: Počty kroků pro posun ramen o 1°</b>	<b>36</b>
<b>Tabulka 7: Příkazy pro komunikaci z mikroprocesoru do počítače</b>	<b>37</b>
<b>Tabulka 8: Ceny použitého materiálu</b>	<b>39</b>

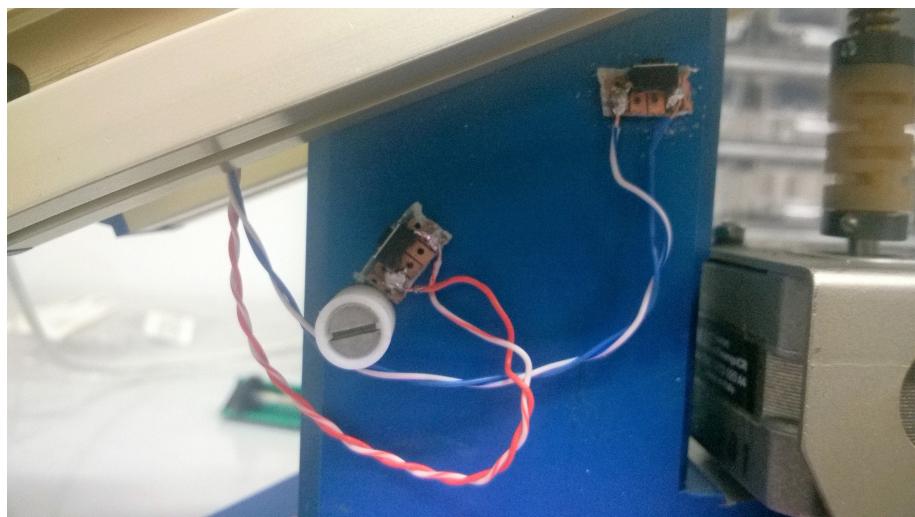
## **Přílohy**

## A Schéma zapojení

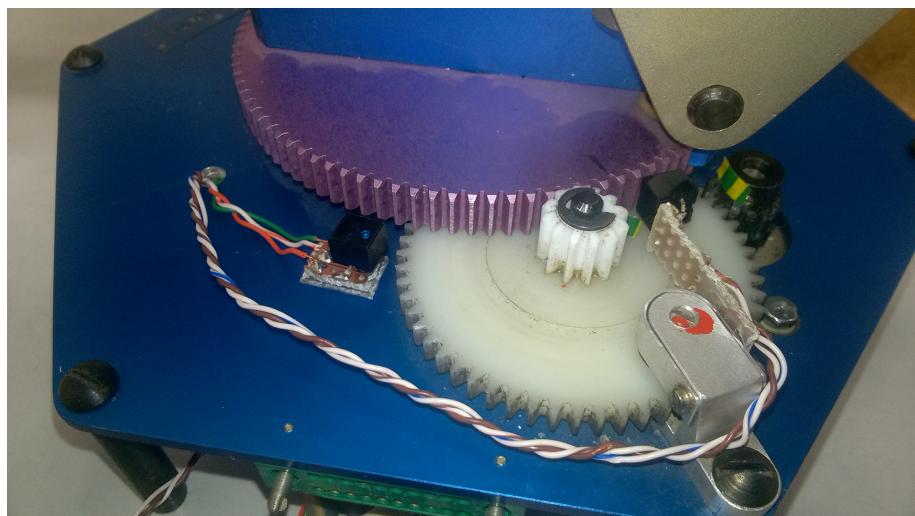


Obrázek 9: Schéma zapojení

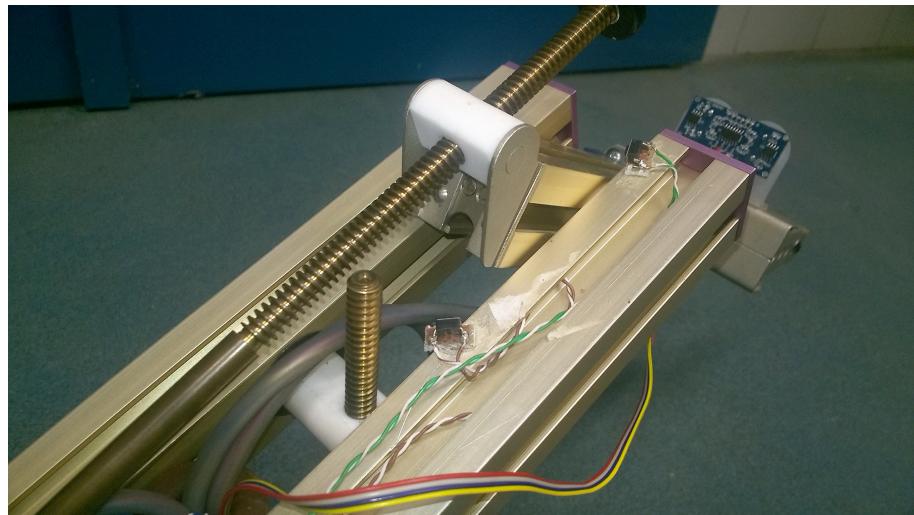
## B Modernizovaný manipulační robot



Obrázek 10: Mikrospínačové senzory



Obrázek 11: Opto senzory



Obrázek 12: Mikrospínačové senzory



Obrázek 13: Elektronika umístěná v základně



Obrázek 14: Celkový stav manipulačního robota po modernizaci