ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Systém pro správu řídících příkazů průmyslového robota
Student:	Jan Marvan
Vedoucí:	Dr. Ing. Sven Ubik
Studijní program:	Informatika
Studijní obor:	Softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2017/18

Pokyny pro vypracování

Cílem práce je analýza, návrh a implementace řídícího a ovládacího software pro malého průmyslového robota pro manipulaci s předměty (tzv. robotická ruka) typu Lynx Motion. Proveďte stručnou rešerši přístupů a implementací existujících softwarových řešení. Vytvořte softwarovou aplikaci pro ovládání robota umožňující nahrávání, editaci a přehrávání posloupnosti ovládacích příkazů. Vstupní zařízení bude typu gamepad. Uživatelské rozhraní bude webového typu. Ověřte činnost programu v praxi.

Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

prof. Ing. Pavel Tvrdík, CSc.
děkan

V Praze dne 4. ledna 2017

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra softwarového inženýrství

Bakalářská práce

Systém pro správu řídících příkazů průmyslového robota

Jan Marvan

Vedoucí práce: Dr. Ing. Sven Ubik

15. května 2017

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, avšak pouze k nevýdělečným účelům. Toto oprávnění je časově, teritoriálně i množstevně neomezené.

V Praze dne 15. května 2017 .....

Odkaz na tuto práci

Marvan, Jan. *Systém pro správu řídících příkazů průmyslového robota.* Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2017.

Abstrakt

Předmětem této bakalářské práce je vytvoření aplikace sloužící k ovládání malého průmyslového robota pro manipulaci s předměty. Aplikace je vyvinutá pro robota Lynxmotion AL5D, jehož typ lze označit jako robotická ruka. Ruka je ovládána připojeným gamepad a aplikace umožňuje i nahrávání sekvencí pohybu. Nahrané sekvence pak lze v aplikace editovat a následně přehrávat. Tímto postupem lze robota naprogramovat k provádění automatizované činnosti. Text práce se zaobírá stručným popisem daného robota a analýzou existujících, pro něho určených aplikací. V další části je pak popsán návrh mé aplikace a následně i způsob její implementace. Aplikace je napsána v programovací jazyku C++ a disponuje uživatelským rozhraním zobrazovaným ve webovém prohlížeči.

Klíčová slova robotická ruka, Lynxmotion, SSC-32U, ovládací aplikace, pohybové sekvence, gamepad, kartézský systém

Abstract

The subject of this bachelor thesis is creating an application for controling a small industrial robot used for object manipulation. The application is developed for the Lynxmotion AL5D robot, which can be referred to as, a robot hand. The robotic hand is controlled via a connected gamepad and the application also allows you to record motion sequences. These recorded sequences can then be edited and replayed in the application. By doing so, the robot can be programmed to perform an automated activity. The text of the thesis starts with a brief description of the robot and analysis of other existing applications. The next part describes the design process of my application and then highlights how my application is implemented. The application is written in C++ programming language and has an user interface displayed in a web browser.

Keywords robotic arm, Lynxmotion, SSC-32U, control application, motion sequences, gamepad, cartesian system

Obsah

Úvod **1**

1 Cíl práce **3**

2 Analýza **5**
2.1 Popis robota 5
2.2 Existující softwarové řešení 8

3 Návrh **13**
3.1 Požadavky na aplikaci 13
3.2 Použité technologie 15
3.3 Architektura 16
3.4 Ovládání ruky 17
3.5 Nahrávání a přehrávání sekvencí 20
3.6 Grafické uživatelské rozhraní 24

4 Implementace **27**
4.1 Čtení joysticků 28
4.2 Ovládání ruky 29
4.3 Nahraná sekvence 30
4.4 Grafické uživatelské rozhraní 31

5 Testování **33**
5.1 Testování funkčnosti pomocí případů užití 33
5.2 Testování přesnosti ovládání v osách kartézského systému . . . 35

Závěr **39**

A Obsah přiloženého CD **41**

B Instalační příručka **43**

C Uživatelská příručka **45**

Literatura **51**

Seznam obrázků

2.1 Lynxmotion AL5D ze SAGElab 6
2.2 Schéma pohyblivých částí ruky 6
2.3 Lynxmotion SSC-32U USB Servo Controller 7
2.4 FlowBotics Studio 9
2.5 Prostředí programu Visual Sequencer 9
2.6 SSC-32 Servo Sequencer Ultility 10
2.7 Program FlowArm PLTW 11

3.1 Jednoduchý doménový diagram 16
3.2 Diagram aktivit vlákna pro čtení joysticků 17
3.3 Diagram aktivit vlákna ovládajícího ruku 17
3.4 Schéma pohybu ruky v na sobě kolmých osách 19
3.5 Diagram aktivit pro proces nahrávání sekvencí 21
3.6 Diagram aktivit pro proces přehrání sekvence 22
3.7 Návrh grafického uživatelského rozhraní 24

4.1 Diagram aktivit při spuštění aplikace 27

5.1 Graf měření přesnosti ovládání v osách kartézského systému 36

C.1 Schéma ovládání gamepadem (v souřadnicovém systému) 47
C.2 Stránka uživatelského rozhraní 47
C.3 Úkázka dialogového okna v uživatelském rozhraní 48

Seznam kódů

3.1 Ukázka exportu nahrané sekvence do souboru 22
4.1 Struktura js_event 28
4.2 Ukázka kódu třídy Arm. Metoda changePos() 30
4.3 Struktura reprezentující nahraný event 30
4.4 Pseudokód metody generate ze třídy Record 31
4.5 Ukázka implementace GUI za použití widgetů 32

Úvod

S automatizovanými robotickými systémy se můžeme setkat především v průmyslové výrobě, kde s vysokou přesností manipulují či skládají komponenty v sériové výrobě. Na trhu lze ale také nalézt menší roboty, které se dají zakoupit ve skládacích soupravách. Příkladem takového robota může být robotická ruka AL5D od společnosti Lynxmotion. Po zakoupení a sestavení takového robota je dalším krokem najít vhodný software pro jeho ovládání.

V případě že bychom chtěli i praktické využití takového robota, tak se nespokojíme pouze s přímým ovládáním, ale budeme vyžadovat i možnost naprogramování robota k provádění samostatné činnosti. Jedním z jednodušších způsobu, jak toho dosáhnout, je nahrání pohybů ovládáním robota a následná kompozice pohybů do pracovní smyčky.

V mojí práci se zabývám vytvořením aplikace umožňující ovládání robota za použití ovládacího zařízení se dvěma analogovými joysticky, obecně známého jako gamepad. Aplikace také nabízí již zmíněné nahrávání a uspořádávání sekvencí za účelem vytvoření automatizované činnosti.

Robot kterého jsem si vybral, a pro kterého je určena moje aplikace, je již výše zmíněný Lynxmotion ALD5. V mojí práci se nevěnuji porovnávání s ostatními roboty, ani se příliš nezabývám hardwarovou stránkou robota samotného.

Práce je rozdělena do pěti kapitol. První kapitola Cíl práce vymezuje rozsah a směr mé práce. V druhé kapitole Analýza se věnuji výchozím podmínkám mojí práce, popisuji v ní tedy fungování robota samotného a také již existující softwarové řešení pro ovládání robota. Třetí kapitola je věnována návrhu mnou vyvíjené aplikace. Tato kapitola je nejobsáhlejší a řeší jaké jsou požadavky na aplikaci, dále pak architekturu aplikace a nakonec návrhy jednotlivých částí aplikace včetně návrhu grafického uživatelského rozhraní. Čtvrtá kapitola je Implementace. V této kapitole jsou popsány kroky které jsem prováděl při implementaci a obsahuje ukázky kódu který jsem v aplikaci použil. Pátá a poslední kapitola se zaobírá testováním aplikace, a to ve dvou krocích. V prvním kroku ověřuji že aplikace splňuje všechny dané požadavky a v druhém kroku

se věnuji přesnosti ovládání robota.

Součástí práce jsou také přílohy. K práci je přiloženo CD a popis jeho obsahu. Dalšími přílohami jsou pak instalační a uživatelské příručka.

Kapitola 1

Cíl práce

Cílem mé práce je vytvoření softwaru pro ovládání malého průmyslového robota Lynxmotion AL5D.

Součástí tohoto cíle je seznámení se s již existujícím softwarem pro ovládání zvoleného robota a jeho analýza. Vyzkoušení si různých přístupů k ovládání robota a následné využití těchto znalostí při návrhu vlastního softwaru.

Softwarová aplikace musí splňovat požadavky ze zadání práce. Hlavními požadavky jsou nahrávání, editace a přehrávání posloupnosti ovládacích příkazů. Dalšími požadavky jsou použití ovládacího zařízení typu gamepad a uživatelské rozhraní zobrazitelné ve webovém prohlížeči.

Aplikace by ve výsledku měla poskytovat možnost co nejintuitivnějšího ovládání ruky joystickami gamepadu. Přes toto ovládání by mělo být možné v aplikaci vytvářet sekvence pohybů. Tyto sekvence pak s přihlédnutím na hardwarovou přesnost robota mají být schopny vytvářet automatizovanou činnost robotické ruky.

Kapitola 2

Analýza

Kapitola je rozdělena na dvě části. V první části popisuji robota samotného a v druhé se věnuji popisu již existujících aplikací, které slouží k ovládání robota.

2.1 Popis robota

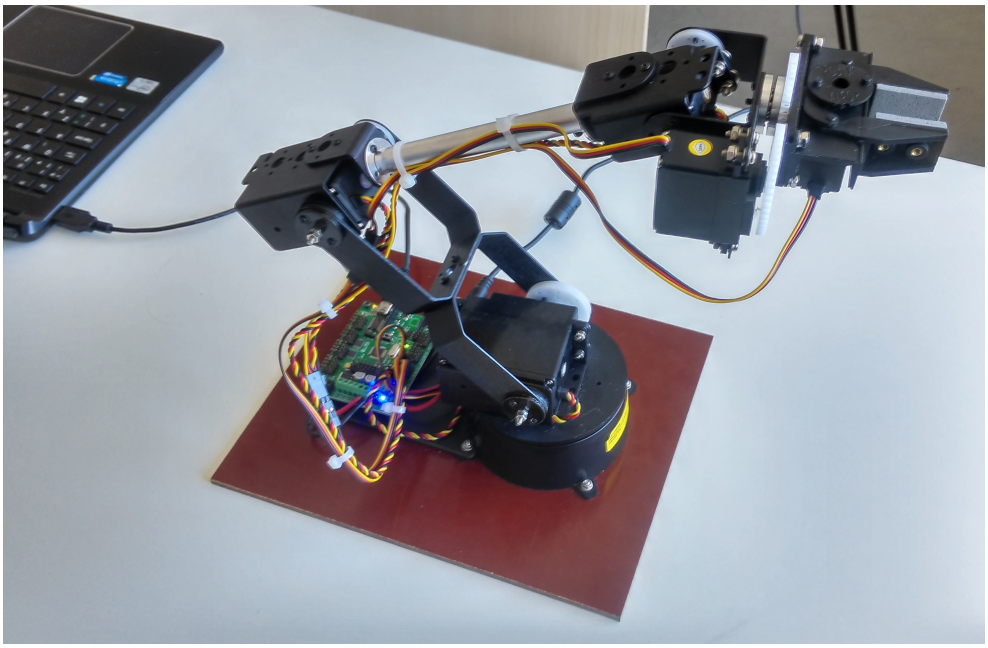
Pro moji práci jsem si vybral malého průmyslového robota od společnosti Lynxmotion, přesněji robotickou ruku AL5D (obr. 2.1). Tato robotická ruka je k dispozici v učebně SAGElab na Fakultě informačních technologií a zde jsem tedy na vývoji a testování aplikace pracoval.

Robotická ruka je složena ze šesti pohyblivých součástí (obr. 2.2). Všechny tyto součásti jsou jsou poháněny neprotáčivými servomotorky. To znamená že mají maximální krajní pozice do kterých se mohou dostat.

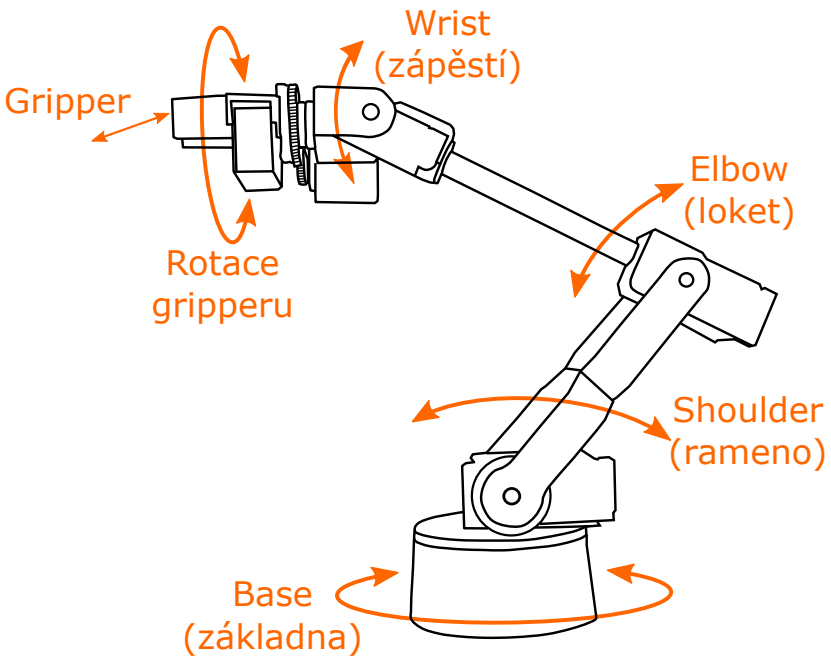
Výrobcem udávený maximální dosah ramene je až 26 cm,[1] při kterém by ruka měla udržet až 283 g.[2] Nosnost ruky se ale zvyšuje při zvedání v nižší vzdálenosti od základny.

2.1.1 SSC-32U

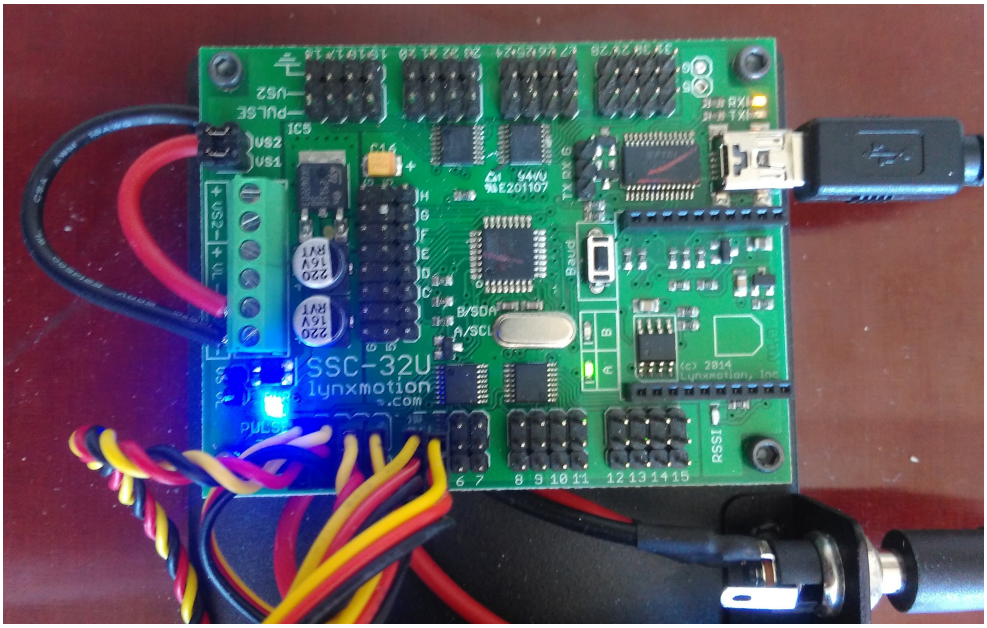
Ovládání robotické ruky je záležitost řízení jednotlivých kloubů ruky, což jsou elektronicky řízené malé servomotory. Pro řízení motorků na hardwarové úrovni se používá takzvaný kontroler, v mém případě jde o kontroler SSC-32U (obr. 2.3). Tento kontroler lze pak řídit přes sériové spojení z počítače, či jiného programovatelného zařízení schopného komunikovat na sériovém portu, například zařízení Arduino. Nejnovější verze kontroleru kterou mám k dispozici pak nabízí možnost připojení k počítači přes USB a vytvoření virtuálního sériového portu v počítači. V mé práci se pak již nevěnuji hardwarovým záležitostem, ale pouze řízení kontroleru připojeného přímo k počítači přes port USB.



Obrázek 2.1: Lynxmotion AL5D ze SAGElab



Obrázek 2.2: Schéma pohyblivých částí ruky



Obrázek 2.3: Lynxmotion SSC-32U USB Servo Controller

Kontroleru se posílají příkazy v následujícím formátu (kurzívou psané části jsou nepovinné):[3]

#<ch> P<pw> ***S<speed>*** ***T<time>*** **<cr>**

- <ch>: číslo kanálu/pinu na kterém je připojené servo (v případě této robotické ruky 0 až 5)
- <pw>: pozice do které chcete servo dostat (hodnota většinou mezi 500 až 2500)
- *<speed>*: rychlost posunu daná rozdílem hodnot pozic (vzdálenost) za sekundu
- *<time>*: čas v milisekundách za který se má posun realizovat
- <cr>: carriage return (ASCII 13)

Příklad 1: **#5P1500T100**

Tento příkaz pohne se servem připojeným na pinu #5 do pozice 1500 s dobou vykonání 100ms.

Existuje ale i druhý typ příkazu, který dokáže obsluhovat i víc serv najednou.

Příklad 2: **#2 P1500 #3 P1840 T1000**

Tento příkaz spustí pohyb obou serv #2 i #3 ve stejný čas. Se stejnou dobou pohybu 1 sekundy je pak dostane na požadované pozice, byť rozdílnou rychlostí.

2.2 Existující softwarové řešení

Sám výrobce nabízí několik různých programů pro ovládání robota. Pro pokročilé uživatele nabízí program FlowBotics Studio, který slouží k vytváření aplikací pro manipulaci s roboty. Dále pak nabízí tři základní programy, které disponují nahráváním a přehráváním sekvencí pohybů a jsou spíše určeny pro běžné uživatele. Všechny programy od výrobce jsou pouze pro operační systém Windows.

2.2.1 FlowBotics Studio

FlowBotics Studio je nejrozsáhlejší program který výrobce nabízí. Jde o vývojové studio aplikací zaměřených na robotiku. Ve studiu se používá Flowstone grafický programovací jazyk.[4] Aplikace se zde vytvářejí spojováním jednotlivých komponent v grafickém prostředí. Příkladem vstupní komponenty může být například výstup z ovladače gamepadu nebo třeba zpracovaný obraz z kamery umístěné na robotické ruce. Takovou komponentu je pak možno propojit s modulem, který jste si sami naprogramovali a který zařizuje logiku chování robota. Výstup tohoto modulu je pak například možné připojit na komponentu která již zařizuje pohyb robota.

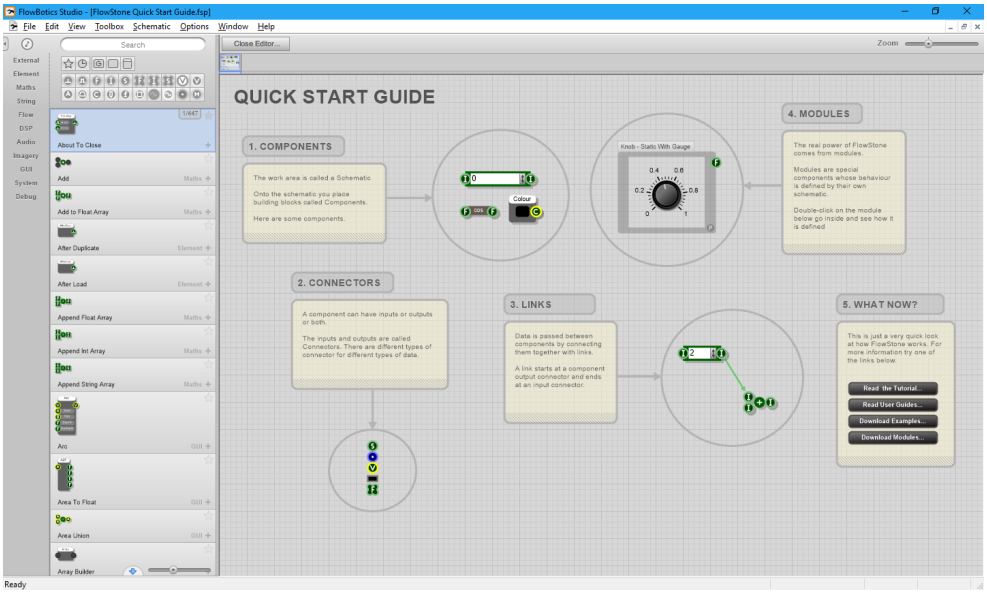
Ve FlowBotics Studiu je možné vytvářet i grafické aplikace s využitím komponent pro grafické prostředí. Tyto projekty pak lze exportovat jako samospustitelné aplikace. Příkladem aplikací dostupných ke stažení, které byly vytvořeny ve FlowBotics Studiu, jsou FlowArm PLTW a SSC-32 Servo Sequencer Ultility. Těmto aplikacím se samostatně věnuji níže.

Tento program podléhá licenčním poplatkům, cena uvedená na webu výrobce je $39.99.

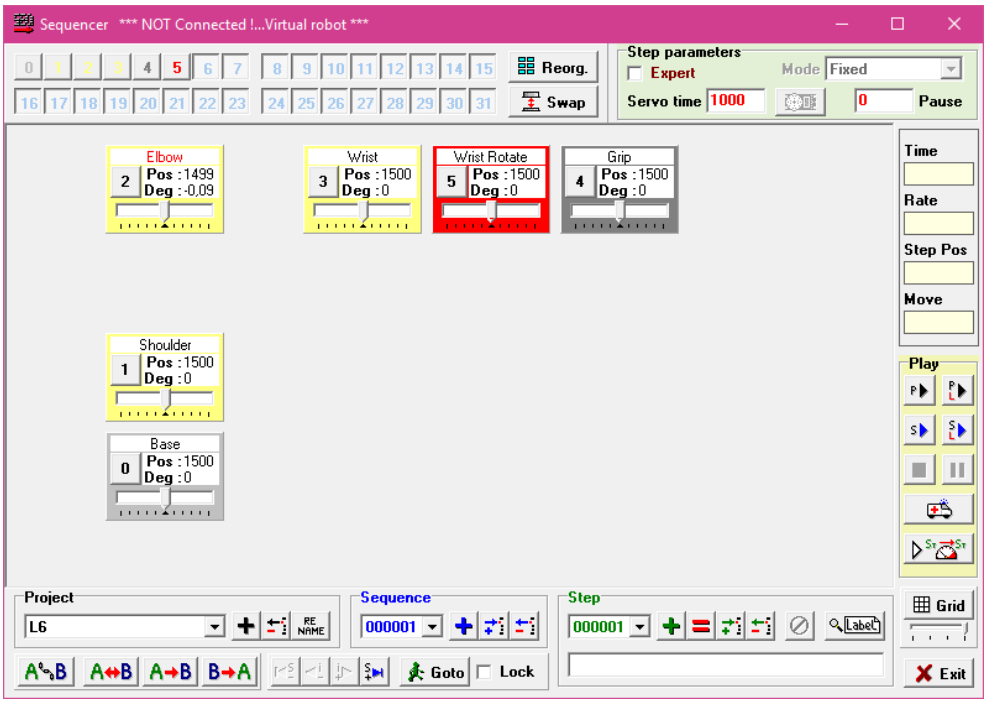
2.2.2 Ovládací aplikace

Lynxmotion Visual Sequencer

Tento program je výrobce označen jako legacy software a je volně dostupný na webu výrobce.[5] Tvoření sekvencí v programu probíhá po krocích, kde uživatel manuálně specifikuje hodnoty reprezentující pozice servomotorů a pak nastaví časový rozestup mezi jednotlivými kroky. Veškerá funkčnost tohoto programu je podle mého obsažena i v programu novějším a také volně dostupným SSC-32 Servo Sequencer Ultility.



Obrázek 2.4: FlowBotics Studio

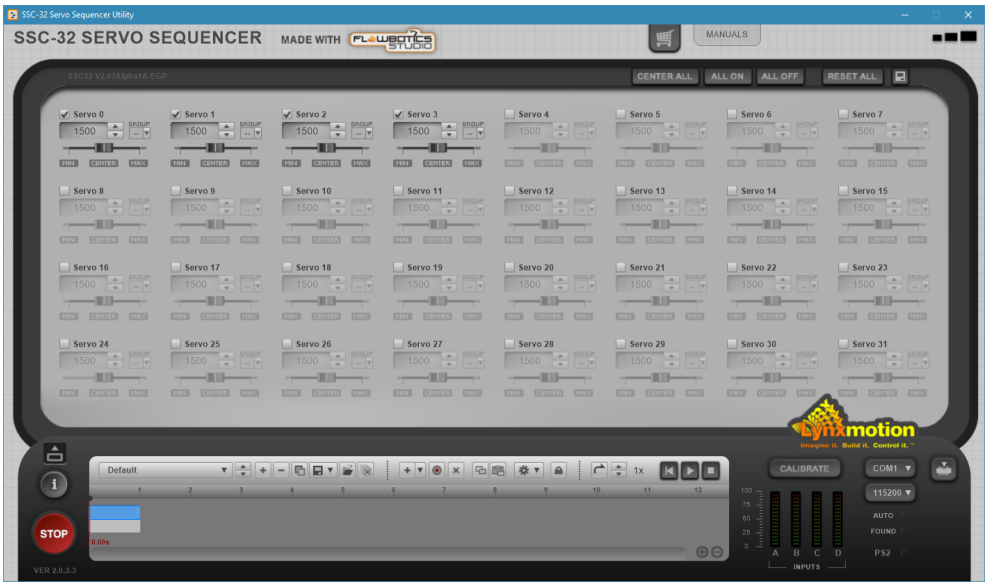


Obrázek 2.5: Prostředí programu Visual Sequencer

SSC-32 Servo Sequencer Ultility

Program SSC-32 Servo Sequencer Ultility lze také nalézt volně ke stažení na stránkách výrobce.[6] V programu je možné tvořit sekvence pohybu jednotlivých servomotorů a pak je přehrávat. Pro pohyb lze použít manuální upravování hodnot reprezentující pozice servomotorů, nebo lze využít připojení gamepadu.

Jde o nejzákladnější nástroj pomocí kterého jde ovládat jakékoliv zařízení s kontrolerem SSC-32U. Program byl vytvořen a je open source demo projekt v FlowBotics Studio. Dalšími funkcemi programu je kalibrace pozicí servomotorků a aktualizace firmware kontroleru.



Obrázek 2.6: SSC-32 Servo Sequencer Ultility

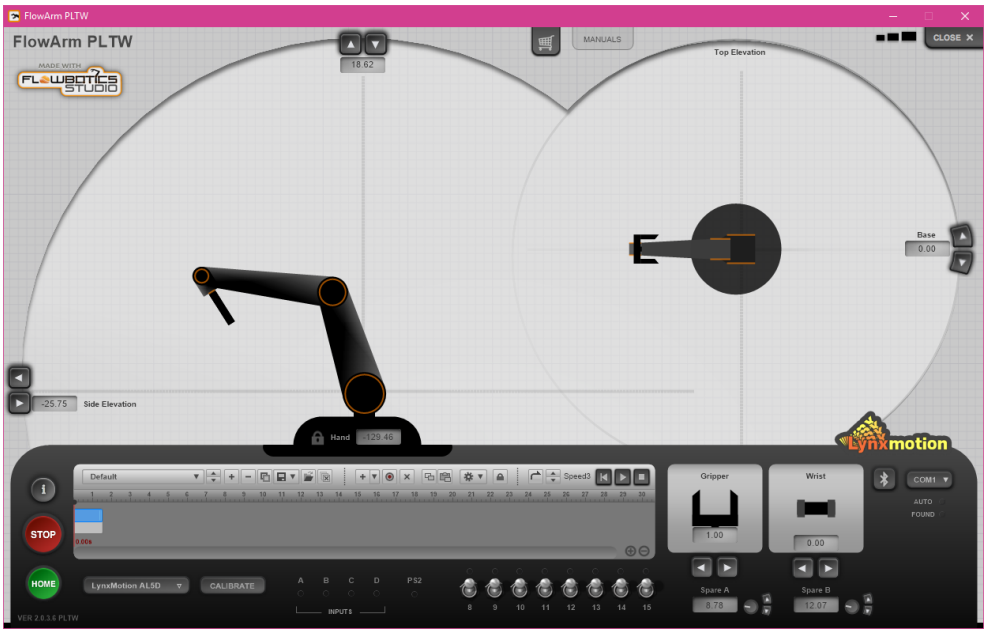
FlowArm PLTW

Poslední program je také projekt vytvořený v FlowBotics Studio.[7] Tento program je ale úzce specializovaný na robota AL5D, tedy na robotickou ruku. Pokud v programu provedete přesnou kalibraci vašeho robota, tedy jednotlivé klouby uvedete do pozic vodorovných či svírajících pravý úhel, podle toho jak je robot vyobrazen v grafickém prostředí, pak dostanete velmi užitečný nástroj. Kalibrovaného robota budete totiž schopni ovládat v osách kartézského souřadného systému. Této problematice se v mé práci také věnuji a to v kapitole návrhu v sekci 3.4.1.

Program disponuje stejným nahrávacím a přehrávacím systémem jako program SSC-32 Servo Sequencer Ultility, ale pro ovládání se využívá grafické prostředí ve kterém je vyobrazen model robotické ruky (obr. 2.7). Za pomoci

gamepadu nebo myši můžete ve virtuálním prostředí modelem pohybovat a při správné kalibraci by se pohyb měl přesně přenášet na robota samotného.

Tento program podléhá licenčním poplatkům, cena uvedená na webu výrobce je $9.99.



Obrázek 2.7: Program FlowArm PLTW

Kapitola 3

Návrh

Při návrhu aplikace jsem postupoval podle požadavků vycházejících ze zadání práce a také z požadavků které jsem si určil při analýze již existujících řešení. Aplikaci jsem se rozhodl vyvinout pro platformu Linux. Výrobce neposkytuje softwarové řešení pro operační systémy Linux a ani aplikaci z jiného zdroje jsem pro platformu Linux nenalezl.

Kapitolu návrhu jsem rozdělil na šest částí. V první části se věnuji požadavkům na aplikaci a mapováním případu užití na ně. V druhé části se pak zabývám použitými technologiemi. Třetí část je zaměřena na návrh architektury aplikace. Ve čtvrté části podobněji popisuji návrh samotného ovládání ruky a pátá část je věnována záznamu sekvencí. Poslední část se věnuje návrhu grafického rozhraní.

3.1 Požadavky na aplikaci

Požadavky na aplikaci jsou rozděleny do dvou kategorií. První kategorií jsou funkční požadavky, definující jaké funkce aplikace nabízí. Jde tedy o možnosti která aplikace musí obsahovat. Druhou kategorií jsou nefunkční požadavky, aplikace je těmito požadavky vázána, ale samy o sobě neposkytují funkčnost.

Zajištění všech funkčností aplikace zaručuje metoda mapování případu užití na požadavky. Nejdříve je nutno nadefinovat typické případy užití aplikace a to v dostatečném množství aby pokryly všechny funkční požadavky. Při testování se pak postupuje podle scénářů těchto případů užití a pokud všechny dopadnou podle očekávání tak je zaručeno že jsou všechny funkční požadavky splněny.

Seznam požadavků na aplikace plynoucí ze zadání práce:

Funkční požadavek FP1 Aplikace umožňuje ovládat robotickou ruku.

Funkční požadavek FP2 Aplikace umožňuje nahrávání posloupnosti ovládacích příkazů.

Funkční požadavek FP3 Aplikace umožňuje editaci posloupnosti ovládacích příkazů.

Funkční požadavek FP4 Aplikace umožňuje přehrávání posloupnosti ovládacích příkazů.

Nefunkční požadavek NP1 Aplikace využívá gamepad jako vsupní zařízení.

Nefunkční požadavek NP2 Aplikace nabízí uživatelské rozhraní zobrazitelné ve webovém prohlížeči.

Dále rozšířeno o požadavky vycházející z analýzy existujících řešení:

Nefunkční požadavek FP3 Aplikace umožňuje provádět pohyb vodorovný či kolmý k ploše základny.

Funkční požadavek FP5 Aplikace umožňuje export a import nahrané sekvence do/ze souboru.

Funkční požadavek FP6 Aplikace umožňuje měnit citlivost ovládání.

Funkční požadavek FP7 Aplikace umožňuje měnit rychlost přehrávání sekvencí.

Dále jsem si nadefinoval případy užití které pokrývají všechny funkční požadavky. Podrobné scénáře pro případy užití jsou popsány v kapitole testování (sekce 5.1).

- Uživatel ovládá pohyb ruky pohybem joysticků na gamepadu (UC1)
- Nahrání sekvence (UC2)
- Upravení části sekvence pohybů nahráním nového pohybu (UC3)
- Přehrání nahrané sekvence (UC4)
- Smazání nahrané sekvence (UC5)
- Přehrání více sekvencí ve zvoleném sledu (UC6)
- Export nahrané sekvence do souboru (UC7)
- Import nahrané sekvence ze souboru (UC8)
- Nahrání sekvence se zvýšenou citlivostí (UC9)
- Přehrání sekvence dvojnásobnou rychlostí (UC10)

Tabulka 3.1: Mapování případů užití na funkční požadavky

	UC1	UC2	UC3	UC4	UC5	UC6	UC7	UC8	UC9	UC10
FP1	x	x	x						x	
FP2		x	x						x	
FP3			x		x	x				
FP4				x		x				x
FP5							x	x		
FP6									x	
FP7										x

V tabulce můžeme vidět že všechny funkční požadavky (řádky) jsou pokryté alespoň jedním případem užití. Testování všech případů tedy zaručí kvalitu aplikace. Zároveň všechny případy užití (sloupce) pokrývají alespoň jeden funkční požadavek, tímto se tedy vyhneme testování které by nevykazovalo žádný námi tázaný výsledek.

3.2 Použité technologie

Práci jsem se rozhodl vyvíjet v jazyce C++, jelikož je mi tento jazyk nejbližší a mám s ním nejvíce zkušeností. V zadání jsem dále měl specifikované uživatelské rozhraní webového typu. Jelikož jsem chtěl aby aplikace byla více interaktivní a zároveň mi přišlo vhodné aby byla aplikace v jednom celku, zvolil jsem knihovnu Wt pro vytváření webových rozhraní pro aplikace psané v C++.

3.2.1 C++

Jazyk C++ je velmi populární rozšíření neméně populárního jazyka C. C++ je jazyk podporující OOP (objektově orientované programování) a jeho kód se kompiluje přímo do instrukcí cílového hardwaru. Programování v C++ nabízí programátorovi velkou kontrolu, ale zároveň se po něm očekává hlubší znalost jazyka. Díky jeho oblíbenosti je pro jazyk dostupné nepřeberné množství knihoven.[8]

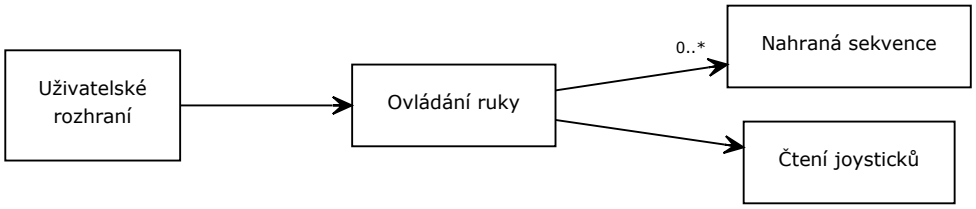
3.2.2 Wt: C++ Web Toolkit

Wt (vyslovuje se jako witty) je C++ knihovna pro vývoj webových aplikací. Knihovna používá systém widgetů, malých součástí GUI (graphical user interface), které se pak společně vykreslují ve webovém prohlížeči. Vývojáři nabízí možnost programovat web bez nutnosti znalosti spousty specifických detailů vývoje webu. Knihovna poskytuje zabudovaný HTTP server, který lze využít i jenom lokálně pro poskytnutí grafického webového rozhraní lokální aplikaci.[9]

Největší výhodou knihovny je možnost naprosté integrace do již existujícího C++ projektu. V takovém případě pak lze v jednotlivých widgetech volat přímo funkce uvnitř programu.

3.3 Architektura

Aplikaci je rozdělena do čtyř domén: řízení ruky, čtení joysticků, uživatelské rozhraní a záznam sekvence. Všechny součásti aplikace ale běží dohromady v jednom spustitelném programu. Architektura a jednotlivé domény vychází z funkčních a nefunkčních požadavků a také omezení daných způsobem ovládání ruky, které popisuji níže.

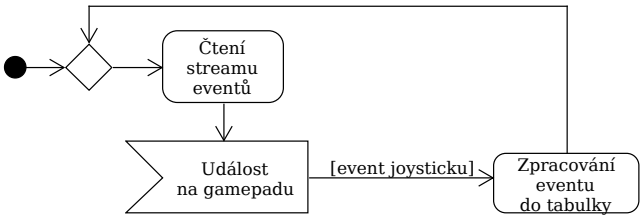


Obrázek 3.1: Jednoduchý doménový diagram

První doménou je ovládání ruky. Příkazy pro ruku jsou vždy posuny do absolutní pozice. V aplikaci bude potřeba třída, která bude řídit ruku a zároveň kontrolovat její aktuální pozici a držení se mezi krajními limity serv. Dále ovládání ruky neumožňuje přerušení již zaslané instrukce pohybu a je tedy zapotřebí, aby aplikace posílala ruce příkazy průběžně a pokud možno v co nejkratších časových intervalech. Pro tento účel bude tedy aplikace vícevláknová, kde samostatné vlákno bude zařizovat ovládání ruky. Návrhu procesu ovládání ruky se věnuji samostatně v této kapitole v sekci 3.4.

K ovládání ruky se používá gamepad. Počítá se s využitím takového, který má dva analogové joysticky pohyblivé ve dvou osách a jeden dvouosý digitální. Celkově se tedy počítá s pohybem v šesti osách. Ke čtení pohybu joysticků slouží linuxové joystick API (application program interface). Toto API funguje na principu čtení událostí, tedy zpráv které jsou vygenerovány při změně polohy joysticku. Vzhledem ke způsobu ovládání ruky se hodí spíše možnost čtení aktuální pozice jednotlivých joysticků. Za tímto účelem jsem navrhl třídu která bude obstarávat přepisování událostí z API do tabulky aktuálních pozic. Tento proces bude také probíhat v samostatném vlákně, jehož činnost je znázorněna v diagramu 3.2. V kapitole návrhu se již zvláště této doméně nevěnuji z důvodu, že jde o velmi jednoduchý proces. Implementace této třídy je ale popsána v příslušné kapitole (4.1).

Grafické uživatelské rozhraní, stejně jako část reprezentující nahranou sekvenci, jsou pak již poměrně přirozeně ve vlastních doménách. Návrhu těchto částí se v této kapitole věnuji níže.



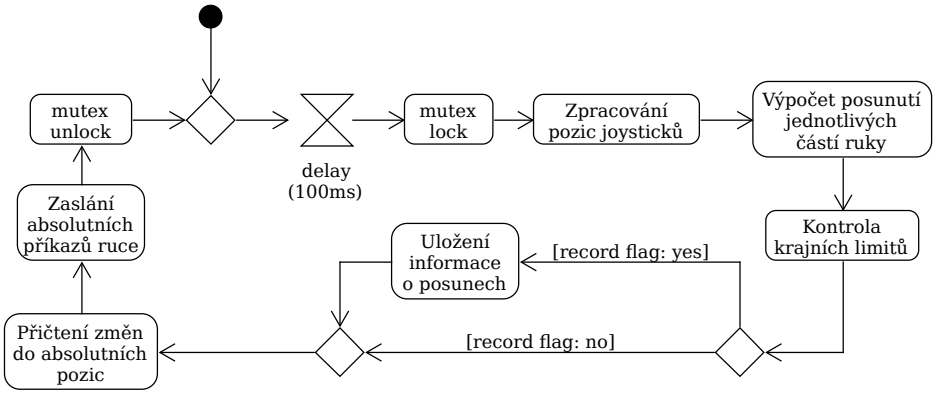
Obrázek 3.2: Diagram aktivit vlákna pro čtení joysticků

Běh aplikace je tedy navržen celkově ve třech vláknech, kde první vlákno procesu je použito k ovládání aplikace a vykresluje uživatelské rozhraní. Druhé vlákno přepisuje události z gamepadu do tabulky evidující polohy joysticků a třetí vlákno na základě této tabulky pohybuje s rukou.

3.4 Ovládání ruky

Ovládání ruky je nejdůležitější součást aplikace. V úvodu této sekce nejdřív popíšu návrh třídy která bude obstarávat zasílání instrukcí pro pohyb ruky. V podsekci níže se pak ještě věnuji návrhu přepočítávání pohybu ruky v osách kartézského systému.

Třída ovládání ruky podobně jako třída pro čtení pohybu joysticku obsahuje zacyklenou funkci pro spuštění v samostatném vlákně. Tato funkce pravidelně zasílá ruce pokyny pro přesun jednotlivých serv do nových pozic. Tento proces je znázorněn v diagramu 3.3 a popsán v odstavci níže.



Obrázek 3.3: Diagram aktivit vlákna ovládajícího ruku

Pokud by rozestup zasílání příkazů byl nastaven na 100 ms, funkce by

každých 100 ms zjistila v jakých pozicích se nacházejí joysticky, a podle toho vypočetla o kolik je potřeba posunout které servo. Posun je pak zároveň zkontrolován aby nedostal servo mimo jeho krajní limity a případně zmenšen do krajní pozice serva. Dále funkce rozpozná jestli je spuštěné nahrávání a pokud ano, tak relativní posun serva zašle k uložení třídě reprezentující nahrávku. Relativní posuny jsou pak přičteny k hodnotám aktuální pozice serv, které třída eviduje v tabulce. V posledním kroku je pak ruce zaslán příkaz, kde jsou vypsány všechny pozice serv podle nově aktualizované tabulky. V příkazu je pak uvedena doba přesunu, která se rovná rozestupu zasílání příkazů, tedy daných 100 ms.

V tomto režimu tedy servo kontroler dostane vždy dávku příkazů, jejichž doba plnění je rovna rozestupu zasílání těchto dávek. Ovládání ruky je ale také chráněno uzamykatelným mutexem, který se vždy v době čekání mezi zasíláním příkazů otevře a umožní jinému vláknu případně převzít kontrolu nad rukou. Tohoto je využito při přehrávání nahrávek, kdy se průběh této funkce na dobu přehrávání přeruší.

Třída pro ovládání ruky dále obsahuje uspořádaný seznam nahrávek a funkce pro přehrání celého seznamu či pouze jedné nahrávky ze seznamu. Tyto funkce společně s funkcí iniciující spuštění a ukončení nahrávání jsou prováděny z vlákna obsluhující ovládání aplikace a věnuji se jim v sekci 3.5.

3.4.1 Pohyb v osách kartézského systému

Možnost pohybovat s robotem rovnoběžně či kolmo s plochou základny je klíčová vlastnost potřebná k intuitivnímu ovládání robota. Myšlenka je taková, že pokud na gamepadu pohybujeme jedním z joysticků, tak pohyb tohoto joysticku v osách x a y se přenese na pohyb robota ve stejných osách, tak jak je znázorněno na obrázku 3.4.

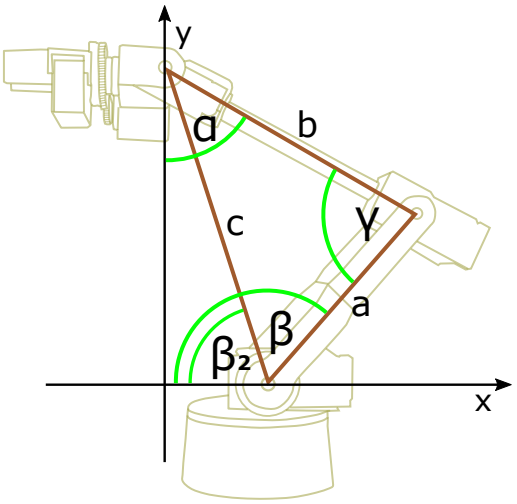
Pro takovouto funkčnost je nezbytné znát délku paže (vzdálenost mezi kloubem ramene a lokte), na obrázku označeno jako strana **a**. Druhou stranou kterou je nutné změřit je strana **b**, na robotovi sloužící jako předloktí (úsek mezi loktem a zápěstím). Strana **c** se mění v závislosti na aktuálních velikostech β a γ, tedy na aktuálních pozicích ramene a loktu.

Přesnou velikost úhlů β a γ získám pokud znám úhel v jedné z krajních pozic daného servomotorku a zároveň vím o kolik se změní hodnota vyjadřující pozici při posunu serva o jeden stupeň. Aktuální velikost úhlu β tedy například vypočítám následovně:

$$ShoulderMinAngle + \frac{(ShoulderPosition - ShoulderMinPosition)}{ShoulderPointsPerDegree}$$

$ShoulderMinAngle$ je změřený úhel který rameno svírá s podložkou v dolní krajní pozici.
$ShoulderPosition$ je aktuální hodnota pozice ramene.
$ShoulderMinPosition$ je hodnota dolní krajní pozice.

ShoulderPointsPerDegree je pak hodnota pozice o kterou se rameno změní při posunu serva o jeden stupeň.



Obrázek 3.4: Schéma pohybu ruky v na sobě kolmých osách

S těmito znalosti jsem již schopen vypočítat aktuální pozici zápěstí v osovém systému x, y. Prvně si vypočítám aktuální hodnoty x a y, ke kterým následně přičtu hodnoty získané z pozice joysticku. Poté z těchto nových hodnot x, y vypočítám zpět velikosti úhlů β a γ a změnu na těchto úhlech již můžu přepočítat do hodnot pro řízení posuvu ruky.

Pro výpočet strany **c** je využita Cosinová věta a následně Sinová věta[10] pro výpočet úhlu β_2.

$$c = \sqrt{a^2 + b^2 - 2 * a * b * \cos\gamma} \qquad \beta_2 = \beta - \arcsin\frac{b * \sin\gamma}{c}$$

Souřadnice jsou pak vypočteny za pomoci pravidel goniometrických funkcí v pravoúhlém trojúhelníku.[11]

$$x = c * \cos\beta_2 \qquad y = c * \sin\beta_2$$

Souřadnice jsou pak následně upraveny podle aktuální polohy joysticku a z nich zpět přepočítány úhly β a γ.

K výpočtu nové přepony **c** je využita Pythagorova věta, u β_2 je využito goniometrických funkcí, výsledný úhel loktu γ je vypočten přes Cosinovu větu a úhel ramene β je vypočítán za použití Sinové věty.

$$c = \sqrt{x^2 + y^2} \qquad \beta_2 = \arccos\frac{x}{c}$$

$$\gamma = \arccos \frac{a^2 + b^2 - c^2}{2 * a * b} \qquad \beta = \beta_2 + \arcsin \frac{b * \sin \gamma}{c}$$

Při změnách úhlů β a γ samozřejmě dochází i ke změně úhlu α, se kterým v těchto výpočtech nepracuji, ale tato změna má vliv na polohu zápěstí. Při přesunech ruky v osách x, y by bylo také vhodné, aby úhel který svírá gripper s osovým systémem byl zachován. Proto je změna úhlu α také přepočítána a servo zápěstí poté posunuto.

Úhel α prvně spočítám před přičtením změn a poté stejným způsobem spočítám nový úhel α po přičtení změn. Nejdříve je samozřejmě nutné spočítat novou přeponu **c** a nový úhel γ. Při výpočtu užívám Sinové věty a goniometrických funkcí.

$$\alpha = \arcsin \frac{a * \sin \gamma}{c} + \arccos \frac{y}{c}$$

Když již tedy znám všechny potřebné změny v úhlech při posunu v souřadnicích x a y, tak se znalostí změn hodnot při posunutí o jeden stupeň dokážu spočítat potřebné hodnoty pro pohyb ruky. Tyto hodnoty pak můžu již zpracuji standardním způsobem, případně je uložím při zapnutém nahrávání.

Přesnost tohoto ovládání není úplně stoprocentní, což je dáno i konstrukcí robota. Serva neleží přímo na ose dané konstrukcí, ale trochu mimo ní. Výsledkem toho je, že v různých polohách je vzdálenost mezi loktem a zápěstí rozdílná. Dalším faktorem bude i nepřesnost v měření rozdílů hodnot při posunu o jeden stupeň. Přesnosti tohoto ovládání se více věnuji v kapitole testování (sekce 5.2).

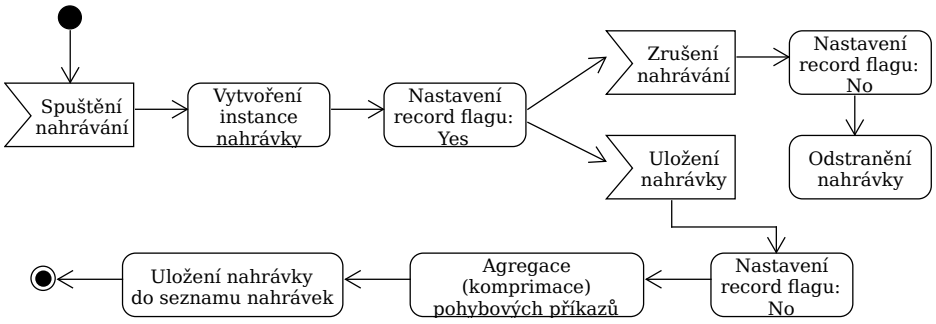
3.5 Nahrávání a přehrávání sekvencí

Způsob zaznamenání sekvencí vychází z ovládání ruky. Sekvence jsou ukládány přesně tak, jak byly generovány pohybem joysticků. Ukládány jsou jako relativní posuny, s tím že výchozí pozice ruky před startem nahrávání je také uložena. Prvně se v této sekci věnuji návrhu nahrávání sekvencí a v další podsekci pak přehrávání sekvencí. O podsekci níže je popsán formát uložení nahrávky v souboru. V poslední podsekci rozvádím jak je v navrhnuté aplikaci řešena editace posloupnosti ovládacích příkazů

3.5.1 Nahrávání sekvence

Nahrávání sekvence je inicializované z vlákna pro obsluhu aplikace. Činnosti popsané v diagramu 3.5 jsou tedy prováděny tímto vláknem. Vlákno které ovládá ruku pouze přeposílá veškeré provedené úkony instanci třídy reprezentující nahrávku.

Při spuštění procesu nahrávání je prvně vytvořena instance nahrávky. Nahrávka může být ve dvou stavech. První stav reprezentuje vznikající nahrávku



Obrázek 3.5: Diagram aktivit pro proces nahrávání sekvencí

a obsahuje uspořádaný seznam všech příkazů které ruka během nahrávání dostala. Druhý stav reprezentuje nahrávku kde jsou tyto příkazy komprimovány a která je již připravena k přehrání či exportu. Při vytvoření jsou nahrávce předány aktuální pozice servomotorků a taky informace o rozestupu, který je mezi dávkami příkazů ukládajících se do této instance třídy.

Po inicializaci začne vlákno ovládající pohyb ruky navíc také automaticky ukládat všechny příkazy zasílané ruce do této instance. Vlákno inicializující nahrávání pak čeká na uživatelův podnět k zastavení nahrávání a to buď zrušením či uložením nahrávky.

Při obdržení příkazu ukončení nahrávání pak vlákno přeruší nahrávání a pokud nahrávka není určena k zahození, tak spustí agregaci pohybových příkazů. Z proudu pravidelných příkazů ruky se vytvoří jednotlivé události, reprezentovány strukturou obsahující čas startu události, označení servomotorku, relativní hodnotu pohybu a dobu provedení tohoto pohybu. Agregovány jsou k sobě pouze pohyby s konstantní rychlostí. Pokud směr pohybu je stejný, ale rychlost se změní, pohyb se nesloučí do jedné události.

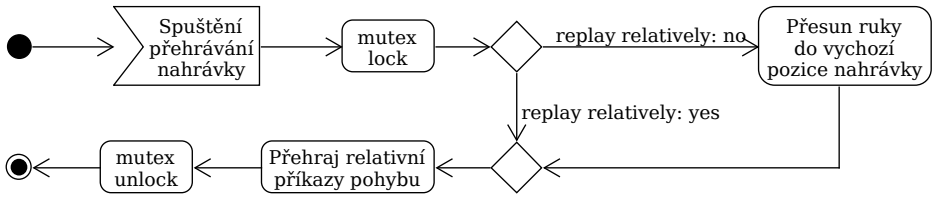
Po vygenerování sledu těchto událostí je nahrávka přesunuta do stavu reprezentující hotovou nahrávku a původní stream uložených příkazů je smazán. Instance nahrávky je taky přesunuta do seznamu nahraných sekvencí, který se nachází v instanci třídy ovládající pohyb ruky. Tento seznam je pak zobrazen v uživatelském rozhraní kde lze k jednotlivým nahrávkám přistupovat.

3.5.2 Přehrávání sekvence

Přehrávání sekvence je stejně jako nahrávání iniciováno z vlákna pro obsluhu aplikace, ale na rozdíl od nahrávání při tomto procesu vlákno obsluhy aplikace převezme plnou kontrolu nad rukou a zablokuje vlákno které ruku ovládá pomocí joysticků.

Při přehrávání je možné zvolit jestli se má sekvence přehrát z výchozí či aktuální pozice. Přehrávání z výchozí pozice pak ruce posílá příkazy výsled-

kem totožné s těmi které byly poslány při nahrávání sekvence. Přehrávání z aktuální pozice využívá způsobu uložení příkazů relativně, kdy příkazy jsou pak přehrávány tak jak byly joystickem nahrány (s přihlédnutím na krajní pozice servomotorků), ale s možností jiné počáteční pozice. Druhým parametrem přehrávání je pak rychlost přehrávání volitelná v procentech původní rychlosti nahrávky.



Obrázek 3.6: Diagram aktivit pro proces přehrání sekvence

Při přehrávání celého seznamu nahrávek je proces znázorněný v diagramu 3.6 a popsaný níže postupně opakován s každou další nahrávkou a jejími parametry (relativní přehrání, rychlost přehrání).

Po spuštění přehrávání si vlákno pro obsluhu aplikace převezme kontrolu nad robotickou rukou uzamknutím mutexu. Ten pak drží zamčený až do doby ukončení přehrávání a tím znemožňuje aby pohyb ruky byl ovlivněn pohyby na gamepadu. V dalším kroku pak v závislosti na parametru přehrávání přesune ruku do výchozí pozice ve které byla sekvence nahrána a nebo tento krok přeskočí. Poté začne s přehráváním samotných příkazů.

Příkazy jsou v nahrávce uloženy jako události na časové ose. Při přehrávání zvýšenou či sníženou rychlostí jsou časové hodnoty počátků událostí a také délky trvání událostí jednoduše vyděleny požadovanou rychlostí.

Na konci přehrávání vlákno odemkne mutex zamykající pohyb ruky a opět ho zpřístupní pro ovládání gamepadem.

3.5.3 Formát uložení

Pro uložení nahrávky do souboru je použit jednoduchý textový výpis jednotlivých událostí na časové ose. Na prvním řádku jsou vypsány výchozí poziční hodnoty jednotlivých serv oddělených mezerou. V dalších řádcích jsou pak jednotlivé příkazy pro ruku agregované do událostí.

Jedna událost je reprezentována čtyřmi hodnotami, které jsou oddělené mezerami. První údaj je časový kód v milisekundách, reprezentující počátek události. Druhým údajem je číslo reprezentující servo, kterému je příkaz určen. Třetí číslo reprezentuje hodnotu relativního posunu daného serva. Poslední, čtvrtý údaj, je doba provedení tohoto pohybu v milisekundách.

Kód 3.1: Ukázka exportu nahrané sekvence do souboru

```
1800 1229 2050 1480 550 1400
700 3 -34 400
1000 2 12 100
2300 3 -11 100
2300 1 -21 200
...
8600 -1 0 0
```

V první řádku jsou postupně vypsané pozice jednotlivých serv při spuštění nahrávání. Celkově je ruka poháněna šesti servomotorky a jsou číslované od nuly. V druhém řádku je již první uložená událost. Tato událost se při výchozí rychlosti spustí 700 ms po startu přehrávání a ruce se zašle pokyn aby přemístila serva číslo 3 o 34 bodů s délkou trvání přesunu 400ms.

Jak je v ukázce vidět, tak při zpracování událostí nevadí pokud jedno servo má začátek události dříve, než je konec události serva druhého. Také je možné aby dvě různé serva měli začátek události ve stejný moment. Na posledním řádku je uložena prázdná událost (reprezentována označením serva -1 a nulovou délkou trvání), která pouze reprezentuje časový údaj při kterém bylo nahrávání ukončeno. Pro správnou funkčnost přehrávání je třeba, aby byly události seřazeny dle jejich počátečního času.

Reprezentace nahrané sekvence v souboru je prakticky totožná s reprezentací nahrávky v programu, kde je každý řádek reprezentovaný jako jedna struktura události. Tyto struktury jsou pak uložené v seřazeném vektoru.

Při exportu do souboru se pouze neexportuje přednastavená rychlost přehrávání a volba pro relativní přehrávání. Tyto volby je při importu zpět do programu nutné u nahrávky nastavit znovu.

3.5.4 Editace posloupnosti ovládacích příkazů

Jednou z funkcí uvedených v zadání práce je editace nahrané posloupnosti ovládacích příkazů. Tato funkčnost je primárně řešena možností editace seznamu s jednotlivými nahrávkami. Pokud plánujeme nahrát nějaký komplexnější pohyb, který by vyžadoval připadnou editaci, tak je nejlepší si tuto sekvenci pohybů rozdělit a nahrát do více záznamů.

Příkladem může být sekvence složena ze dvou nahrávek. První nahrávka která by dostala ruku na určenou pozici, zatímco druhá nahrávka by provedla úkon v dané pozici. Pokud by jsme s pozicí ve které se má úkon provést nebyli spokojeni, je možné tuto společnou sekvenci editovat tím, že nahrajeme místo prvního záznamu novou sekvenci. Druhou sekvenci, která je pak například přehrávána relativně, je možno zachovat a není nutné přehrávat celou společnou sekvenci.

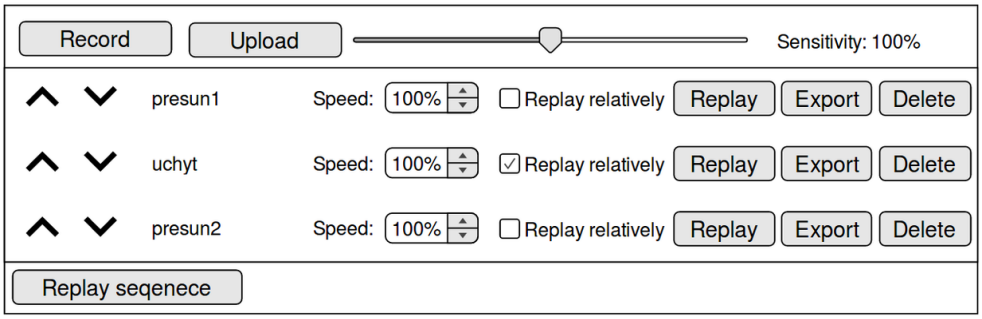
Editace samotné jedné nahrávky není v aplikaci možná, částečně kvůli neintuitivnosti, kdy měnění hodnot reprezentující úhly kloubů bez jakékoliv gra-

fického zobrazení ruky je poměrně nepoužitelné. Částečně také protože měnit hodnoty tímto způsobem lze provádět i v textovém editoru po vyexportování nahrávky do souboru.

Editace seznamu nahrávek nabízí přidávání, odebírání a měnění pořadí jednotlivých nahrávek. Dále lze změnit rychlosti přehrávání a také nastavit výchozí pozici ze které se nahrávka má přehrát. Způsob provádění těchto editací lze vidět na obrázku návrhu uživatelského rozhraní (3.7).

3.6 Grafické uživatelské rozhraní

Pro implementaci GUI jsem se rozhodl využít knihovnu Wt: C++ Web Toolkit. Knihovna nabízí vestavěný http server, který v mé aplikace bude spuštěn pouze lokálně. Při vstupu na port webserveru z prohlížeče se pak automaticky vygeneruje stránka poskytující ovládání aplikace. Grafické rozhraní jsem navrhl tak, že veškerá funkčnost je dostupná z jediné obrazovky (obr. 3.7).



Obrázek 3.7: Návrh grafického uživatelského rozhraní

Jednotlivé prvky mají pak následující funkce:

Tlačítko Record Po stisknutí tohot tlačítka aplikace začne nahrávat veškerý pohyb prováděný s rukou. V otevřeném dialogovém okně je pak potvrzovací tlačítko *Ok* pro uložení nahraného pohybu a tlačítko *Cancel* pro zahození nahrávaného pohybu. Při uložení pohybu se nová sekvence zobrazí na konci listu nahraných sekvencí.

Tlačítko Upload Otevře dialogové okno s tlačítkem *Browse*, které vyvolává systémového průzkumníka, ve kterém lze vybrat soubor k importu. Po potvrzení se sekvence z tohoto souboru přidá na konec listu.

Posuvník Sensitivity Slouží k nastavení citlivosti ovládání ruky.

Šipky pro posun sekvence Pomocí šipek lze změnit pořadí nahraných sekvencí.

Název sekvence Po kliknutí na název sekvence lze text editovat.

Pole Speed Slouží k nastavení rychlosti přehrání dané sekvence.

Zaškrtávací pole Replay relatively Při zaškrtnutí se sekvence přehraje z aktuální pozice ruky. V defautlním nastavení se ruka nejprve přesune do pozice ze které byla sekvence nahrána a pak se přehraje.

Tlačítko Replay Přehraje jednotlivou sekvenci pohybu s aktuálním nastavením rychlosti a startovací pozice sekvence.

Tlačítko Export Nabídne soubor s danou sekvencí k uložení.

Tlačítko Delete Odebere sekvenci ze seznamu.

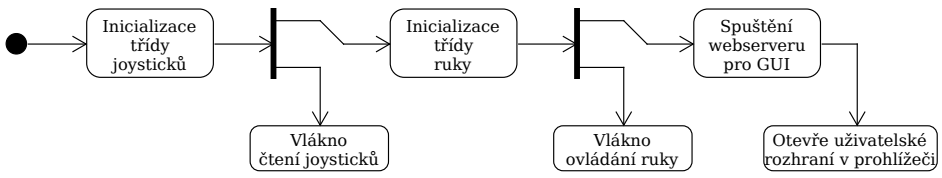
Tlačítko Replay sequence Otevře dialogové okno s možností zvolení počtu cyklů přehrání. Po potvrzení přehraje všechny sekvence v za sebou jdoucím pořadí, s jejich nastavením rychlosti a startovací pozice. Toto případně opakuje podle zvoleného počtu cyklů.

Kapitola 4

Implementace

V této kapitole se věnuji popisování implementačních detailů důležitých či zajímavých funkcí aplikace. Kapitola je rozdělena stejně jako aplikace samotná na čtyři části. První částí aplikace je třída čtení joysticku, druhou částí je třída obsluhující ovládání ruky, třetí část je třída reprezentující nahranou sekvenci a čtvrtá část je třída definující stránku GUI. Všechny tyto části jsou zvláště deklarovány ve svých hlavičkových souborech a dále pak definovány v souborech k nim příslušejícím. Části jsou tak kompilované zvlášť a následně jsou linkovány se spouštěcím programem.

Spouštěcí program aplikace nejprve vytvoří instanci třídy pro čtení joysticků a následně spustí nové vlákno ve kterém zavolá metodu obstarávající veškerou obsluhu čtení joysticku. V dalším kroku vytváří instanci třídy obsluhující ovládání ruky, které předá referenci na instanci třídy čtení joysticků. Program pak v novém vlákně spustí metodu této instance zařizující pohyb ruky na základě pozic joysticků. Jako poslední je spouštěn webserver knihovny Wt, který obstarává grafické prostředí. Posledním krokem při startu aplikace je pak zavolání systémové funkce, která otevře stránku pro ovládání ve webovém prohlížeči.



Obrázek 4.1: Diagram aktivit při spuštění aplikace

4.1 Čtení joysticků

Čtení joysticku zařizuje třída **Joystick** deklarovaná v souboru "joystick.h". Při implementaci této třídy jsem vycházel z dokumentace Joystick API[12].

Ovládací zařízení typu gamepad jsou v systému Linux dostupné jako datový stream obsahující jednotlivé události na ovladači. Tento datový stream lze pak v C++ programu otevřít jako jakýkoliv jiný stream či soubor. V adresářové struktuře ho lze nalézt v cestě */dev/input/jsX*, kde *X* je číslo přidělené systémem. Zařízení jsou číslovány od nuly vzestupně s dalšími přibývajícími zařízeními. Adresu na které je zařízení připojeno lze v aplikaci zvolit prostřednictvím parametru při spuštění.

Čtení událostí ze streamu lze provádět v blokujícím či neblokujícím režimu. V blokujícím režimu se při zavolání funkce *read()* zablokuje běh programu do doby, než se vygeneruje nová událost. V jednovláknové aplikaci by byl takový postup nežádoucí, ale v aplikaci kde čtení vstupů je zařízeno samostatným vláknem je toto naprosto vyhovující a proto jsem ho zde použil.

Při čtení se pak k uložení jednotlivých událostí z datového streamu používá předepsaná struktura, viz kód 4.1.

Kód 4.1: Struktura js_event

```
struct js_event e;
read (fd, &e, sizeof(e));

where js_event is defined as

struct js_event {
    __u32 time; /* event timestamp in milliseconds */
    __s16 value;/* value */
    __u8 type;  /* event type */
    __u8 number;/* axis/button number */
};
```

Takto uložená událost se dá již zpracovat. Prvně je třeba zjistit o jaký typ události jde. Gamepad generuje tři typy událostí: syntetický event, stisk/uvolnění tlačítka a pohyb joysticku. V mé aplikaci jsem zpracovával pouze události týkající se pohybu joysticků, jelikož ostatní funkčnost nebyla vyžadována.

Třída zařizující čtení gamepadu tedy disponuje funkcí, která v nekonečném cyklu čte a následně zpracovává události (viz diagram 3.2). Třída dále obsahuje pole jednotlivých os ve kterém jsou uchovávány hodnoty odpovídající aktuální pozici joysticků.

4.2 Ovládání ruky

Veškeré ovládání ruky je implementováno v třídě **Arm** deklarované v "arm.h". Jak jsem již popisoval v kapitole návrhu tato třída zastává vícero funkcí. V této sekci se budu věnovat popisu některých implementačních detailů těchto funkcí.

K třídě Arm spadá ještě jeden hlavičkový soubor a to "parts.h". V tomto souboru jsou definovány klíčové konstanty popisující jednotlivé části ruky. Jsou zde například uvedeny výchozí pozice serv při spuštění, minimální a maximální pozice do kterých můžou být jednotlivá serva posunuty a případně i úhly v těchto pozicích. Dále je zde zapsána vzdálenost loktu od ramene a zápěstí od loktu. Poslední zmíněné hodnoty se používají při výpočtu pohybu v kartézských souřadnicích.

Při vytváření instance třídy se spouští první důležitá funkce, a tou je inicializace sériového portu. Pro komunikaci na sériovém portu se v jazyce C++ na Linuxu používá knihovna termios, velmi dobře popsaná na webu Wikibooks[13]. Po úspěšném otevření sériového portu v aplikaci je portu je přidělen file descriptor a lze do něj psát obdobně jako do souboru či datového streamu. Před samotným používáním je třeba ještě nastavit parametry komunikace. Aplikace pak nastaví hodnoty serv do výchozích pozic a provede první odeslání příkazů pohybu ruce.

Jak již bylo popsáno v kapitole návrhu, aplikace pak v samostatném vlákně spustí funkci obstarávající pravidelné zasílání příkazu ruce v závislosti na pohybu joysticků. Tato funkce musí provádět kontrolu krajních limitů, ale vzhledem k tomu, že při nahrávání se ukládání relativní pohyb, je třeba provádět kontrolu ještě před přičtením k aktuální pozici. Pro tento účel třída využívá metodu kterou ukazuji v kódu níže (kód 4.2). Metoda bere jako první parametr relativní změnu přečtenou z pozice příslušné osy joysticku, či změnu vypočítanou přepočtem pohybu v souřadnicích a jako druhý parametr příslušné servo. Metoda provede změnu v tabulce aktuální pozice a zároveň vrací hodnotu relativního posunu, která je případně upravená pokud by se servo dostalo do krajní pozice. Tato metoda je postupně zavolána na každé servo a výstupy jsou v případě nahrávání zaslány instanci třídy Record.

K zasílání příkazů se pak použije dávkový příkaz obsahující požadované pozice všech serv. Tento příkaz je popsán v sekci popisující SSC-32U kontroler (2.1.1) u druhého příkladu příkazů.

Četnost provádění ovládací funkce, tedy čtení hodnot joysticků a převedení těchto hodnot do pohybu ruky je klíčová pro reakční čas při změně směru pohybu. Robot nemá problém s plněním rychlých příkazů přicházejícím ve velmi krátkých intervalech. Potíže s využitím krátkých intervalů nastávají až v případě nastavení vyšší senzitivity, kdy je po robotovi požadován pomalý pohyb. V takovém případě jsou odesílány příkazy v řádu malých jednotek bodů pohybu, což způsobuje trhavý pohyb robota. Časový rozestup mezi příkazy jsem tedy zvolil 50 ms, což je dostatečně krátký čas na to, aby uživatel nepoznal zpoždění v ovládání.

Kód 4.2: Ukázka kódu třídy Arm. Metoda changePos()

```
int changePos(int change, int part) {
int difference;
if (partsPos[part]+change > partsMaxPos[part]) {
   difference = partsMaxPos[part] - partsPos[part];
   partsPos[part] = partsMaxPos[part];
  return difference;
}
if (partsPos[part]+change < partsMinPos[part]) {
   difference = partsPos[part] - partsMinPos[part];
   partsPos[part] = partsMinPos[part];
   return difference;
}
partsPos[part] += change;
return change;
}
```

4.3 Nahraná sekvence

Nahraná sekvence je reprezentována instancí třídy **Record** deklarované v "record.h". Instance třídy může vzniknout dvěma způsoby.

Prvním případem je vytvoření prázdné nahrávky a následně její naplnění příkazy pro pohyb ruky. V tomto případě jsou při vytvoření uloženy pozice všech serv při startu nahrávání a taky údaj o aktuálním nastavení časového rozestupu mezi jednotlivými dávkami příkazů. Vlákno ovládající pohyb ruky rozpozná že je spuštěné nahrávání a začne do této instance posílat veškeré příkazy které zasílá i na ruku, ale v relativní podobě.

Po skončení nahrávání je při šesti různých servomotorcích a nastavení rozestupu příkazů 50ms ve třídě uloženo 120 integer hodnot za každou vteřinu nahrávání. Takovýto způsob uložení by nebyl praktický a proto se po ukončení nahrávání spustí metoda *generate()*, který převádí nahrávku do sledu událostí reprezentovaných strukturou RecEvent.

Kód 4.3: Struktura reprezentující nahraný event

```
struct RecEvent {
   int start; /*time of event starts*/
   int part; /*which servo to move*/
   int move; /*relative position change*/
   int len; /*length of the event*/
};
```

Metoda generate prochází postupně pole všech uložených příkazů a agreguje je do eventů. V pseudokódu popisuji zjednodušeně průchod tímto polem. V realitě program agreguje eventy pro všechny serva najednou.

Kód 4.4: Pseudokód metody generate ze třídy Record

```
for prikaz in polePrikazu
   if predchoziPrikaz != prikaz
      uloz event do vektoru
      if prikaz != 0
         vytvor novy event
   else if prikaz != 0
      pricti prikaz k eventu
   predchoziPrikaz = prikaz
uloz event do vekotru
```

Metoda v cyklu hledá totožné, po sobě jdoucí příkazy které mají nenulovou hodnotu. Sled stejných příkazů pak sčítá do doby než dojde ke změně hodnoty příkazu, pak event uloží do vektoru. Nulové příkazy přeskakuje. Na konci cyklu je pak nutné vektor seřadit podle startovacího časového údaje. Na rozdíl od ukázky v pseudokódu jsou totiž eventy pro jednotlivá serva smíchány a při přehrávání je nutné číst eventy seřazené. Po provedení agregace je původní pole příkazů smazáno a nahrávka je plně připravena k přehrávání či exportu.

Druhým způsobem vytvoření instance je že konstruktoru předám adresu k souboru ze kterého se má nahrávka importovat. Nahrávka uložená v souboru je již ve formátu eventů a tudíž ji není třeba nijak upravovat.

Metoda pro export nahrávky jako parametr bere místo jména souboru pouze stream do kterého má sekvenci pohybů zapsat. Tento způsob implementace je zvolen z důvodu že o vytváření dočasného souboru, který se nabídne uživateli k uložení, se stará webserver zabudovaný v knihovně poskytující GUI. Tato třída tedy pouze zapisuje do souboru již připraveného knihovnou Wt.

4.4 Grafické uživatelské rozhraní

Jak jsem již zmínil tak pro implementaci GUI využívám knihovnu Wt. V aplikaci zdaleka nevyužívám veškerou funkčnost této knihovny. Knihovna je navržena i pro vytváření rozsáhlých webových aplikací s aktivním generováním obsahu a celkově poskytuje široké možnosti využití.

V mojí aplikace knihovna generuje pouze jednu stránku na které se vykreslují ovládací prvky pro veškerou funkčnost. Tato stránka je navržena ve třídě **Interface** deklarované v "interface.h". Při otevření adresy a portu webserveru v prohlížeči dochází k zavolání konstruktoru této třídy a vytvoření jedné instance. O veškerou logiku komunikace webserveru a způsobu vykres-

lování stránky se stará knihovna Wt. Mojí úlohou bylo pouze implementování obsahu stránky vyplněním konstruktoru třídy a funkcí z něho volaných.

Způsob implementace podoby stránky probíhá skládáním takzvaných widgetů. Všechny na stránce viditelné prvky a některé další neviditelné prvky jsou widgety. Příkladem viditelného prvku může být jednoduché tlačítko či blok textu. Neviditelný widget je pak například container shlukující prvky. Velmi jednoduchou stránku obsahující dvě tlačítka pod sebou bychom vytvořili postupným vložením widgetu tlačítka, widgetu reprezentujícím konec řádku a znovu widgetu tlačítka do widgetu containeru. Widget containeru bychom pak vložili do hlavního containeru reprezentující danou stránku, v mém případě rovnou do widgetu *root*, který zajistí že veškerý obsah do něj přidaný se zobrazí při zobrazení úvodní stránky při připojení na webserver. Widgety containerů nemusí být pouze schraňovače prvků, ale mohou také poskytovat prvkům rozložení na stránce.

Většina prvků na stránce je pak schopna generovat signály. Signály jsou generovány při klikání na prvky, ale lze pracovat i se signály pouhého přejetí myší nad prvkem. Pro implementaci funkčnosti tlačítka pak stačí pouze napsat funkci obsluhující jeho stisknutí a pomocí funkce *connect()* ji připojit na vygenerovaný signál.

Kód 4.5: Ukázka implementace GUI za použití widgetů

```
WContainerWidget cont = new WContainerWidget();
WPushButton *recButton = new WPushButton("Record", cont);
recButton->clicked().connect(this, &Interface::recDialog);
cont->addWidget(new WBreak());
WPushButton *uplButton = new WPushButton("Upload", cont);
root()->addWidget(cont);
```

V krátkém kódu 4.5 je ukázán postup vytvoření tlačítka Record a navázání funkce *recDialog()* na toto tlačítko. Dále ukázka pokračuje instrukcemi pro vložení nového řádku, vložení druhého tlačítka a následné připojení na hlavní container stránky. Jak je v ukázce vidět vkládání do containerů lze provádět buď již při vytváření widgetu nebo i později dodatečně.

Obdobným způsobem je pak sestavena celá stránka uživatelského rozhraní. Rozhraní je naimplementováno tak jak bylo navrženo ve wireframu na obr. 3.7. Jeho finální podobu lze vidět na screenshotech v příloze Uživatelská příručka (obrázek C.2).

Při implementaci grafického rozhraní mi byla velmi nápomocná galerie widgetů[14] a pak dokumentace knihovny, kde jsou všechny widgety podrobně popsány[15].

Kapitola 5

Testování

Testování aplikace v mém případě znamenalo praktické ověření ovládání robota a funkcí aplikace. Veškerou funkčnost aplikace jsem otestoval podle scénářů případů užití.

V druhé sekci pak popisuji měření přesnosti ovládání, které jsem provedl v osách kartézského systému.

5.1 Testování funkčnosti pomocí případů užití

V této sekci rozepisuji podrobné scénáře jednotlivých případů užití. Tyto případy užití jsem si nadefinoval v kapitole návrhu a slouží k otestování kvality aplikace, tedy ke kontrole že aplikace splňuje všechny funkční požadavky.

- **Uživatel ovládá pohyb ruky pohybem joysticků na gamepadu (UC1)**
  1. Použij levý joystick ve všech směrech.
  2. Při pohledu na robota ze strany, kdy jeho základna je na pravé straně a gripper na straně levé, pohyb robota by měl odpovídat pohybu joysticku.
  3. Použij pravý joystick v horizontálním směru.
  4. Robot se otáčí kolem své základny.
  5. Použij pravý joystick ve vertikálním směru.
  6. Sklon zápěstí odpovídá pohybu joysticku.
  7. Použij digitální joystick v horizontálním směru.
  8. Zařízení pro úchyt předmětů (gripper) se přibližuje a oddaluje.
  9. Použij digitální joystick ve vertikálním směru.
  10. Zápěstí se otáčí kolem své osy.

- **Nahrání sekvence (UC2)**
  1. Stiskni tlačítko Record.
  2. Spustí se nahrávání. Zároveň se objeví dialogové okno s možností ukončení nahrávání s uložením či bez uložení.
  3. Potvrď uložení nahrávání.
  4. Nahraná sekvence se zobrazí v seznamu.
- **Upravení části sekvence pohybů nahráním nového pohybu (UC3)**
  1. Měj nahranou posloupnost sekvencí kterou chceš upravit.
  2. Nahraj novou sekvenci.
  3. Za pomocí šipek na levé straně přesuň novou sekvenci na určenou pozici.
- **Přehrání nahrané sekvence (UC4)**
  1. U zvolené sekvence stiskni tlačítko Replay.
  2. Robot provede posloupnost příkazů z nahrané sekvence.
- **Smazání nahrané sekvence (UC5)**
  1. U zvolené sekvence stiskni tlačítko Delete.
  2. Zvolená sekvence zmizí ze seznamu.
- **Přehrání více sekvencí ve zvoleném sledu (UC6)**
  1. Měj nahranou posloupnost více sekvencí.
  2. Uprav sled sekvencí za pomoci šipek na levé straně.
  3. Stiskni tlačítko Replay sequence.
  4. Potvrď počet opakování přehrání.
  5. Robot provede posloupnosti příkazů ze všech sekvencí ze seznamu.
- **Export nahrané sekvence do souboru (UC7)**
  1. U zvolené sekvence stiskni tlačítko Export.
  2. Aplikace nabídne soubor k uložení.
- **Import nahrané sekvence ze souboru (UC8)**
  1. Stiskni tlačítko Upload.
  2. Stiskni tlačítko Procházet a vyber soubor se záznamem.
  3. Záznam se automaticky nahraje a přidá do seznamu.

- **Nahrání sekvence se zvýšenou citlivostí (UC9)**

  1. Za pomoci posuvníku v horní části obrazovky změň citlivost ovládání.
  2. Nahraj novou sekvenci.
  3. Sekvence se objeví v seznamu.

- **Přehrání sekvence dvojnásobnou rychlostí (UC10)**

  1. U zvolené sekvence zapiš do pole Speed číslo 200.
  2. Spusť přehrávání zvolené sekvence.
  3. Robot přehraje posloupnost příkazů v polovičním čase.

Kontrola provedení a výsledků jednotlivých scénářů je triviální. Výsledný efekt scénáře je kontrolován na robotovi samotném či v GUI aplikace. Všechny testy na aplikaci proběhly korektně a aplikace tedy splňuje všechny dané funkční požadavky.

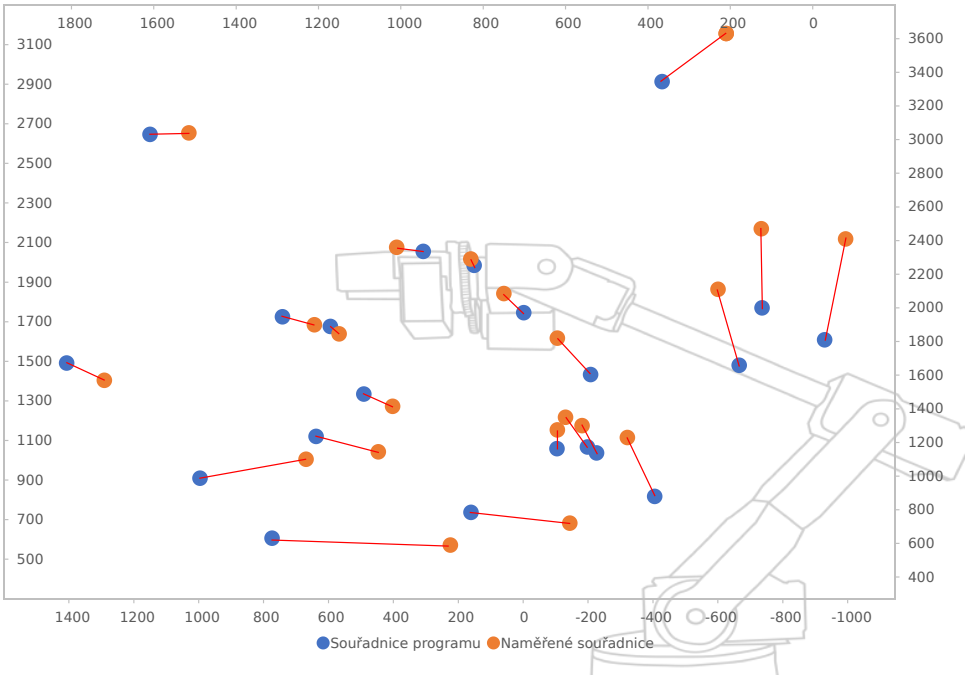
5.2 Testování přesnosti ovládání v osách kartézského systému

V závěru práce jsem na aplikaci provedl měření přesnosti ovládání v osách kartézského systému. Jak jsem již uvedl v kapitole návrhu tak souřadnicový systém uvnitř aplikace se od reálného vždy bude trochu lišit. Pro měření jsem náhodně vybral několik pozic ruky. V každí poloze jsem zapsal souřadnice spočítané programem a pak jsem změřil souřadnice reálné.

Souřadnice v programu jsou počítány jako vzdálenost kloubu zápěstí od kloubu ramene na osách x a y. Průnik těchto os a tedy jejich nulový bod je umístěn v kloubu ramene. Osa x je rovnoběžná s podložkou ruky a její směr je dán směrem natočení celé ruky. Osa y je pak kolmá k ose x. Pro výpočet souřadnic v programu je použit postup popsaný v kapitole návrhu v sekci 3.4.1. Jednotlivé změřené vzdálenosti a úhly ruky pro program jsou uloženy v definičním souboru "parts.h".

Měření reálných hodnot v ose y jsem prováděl od podložky robota, nikoli od výšky kloubu ramene. Ve výsledku jsou ale tyto dva souřadnicové systémy od sebe hodnotami poměrně vzdálené a pro zobrazení v grafu jsem tedy byl nucen osy těchto dvou měření upravit (na sebe posunout). Rozdíl ve způsobu měření tedy není úplně rozhodující, měření spíše slouží pro demonstraci velikosti rozdílů v závislosti na poloze robota.

Hodnoty souřadnic v programu na grafu reprezentuje dolní stupnice pro hodnoty x a stupnice na levé straně pro hodnoty y. Naměřené hodnoty x a y jsou pak na stupnici v horní, respektive pravé částí grafu. Hodnoty na



Obrázek 5.1: Graf měření přesnosti ovládání v osách kartézského systému

stupnicích (a v tabulce níže) jsou v desetinách milimetru. Graf si také lze představit jako prostor ve kterém je schopna se ruka pohybovat.

Jednotlivé osy byly v grafu upraveny tak aby hodnoty ve středu grafu, tedy v místě nejvzdálenějším od krajních pozice ruky, byly sobě co nejbližší. Z grafu lze pak pozorovat v kterých místech se hodnoty liší nejvíce.

Například v dolní části grafu lze vidět že čím níže se ruka pohybuje, tím více je třeba ji nutit do posunu dále od základny. V této pozice je tedy třeba neúměrně zvyšovat vzdálenost v programu vzhledem ke vzdálenosti kterou ruku reálně urazí.

Další viditelné rozdíly jsou znát v oblasti nad a za základnou, kde je umístěno rameno. V těchto místech je ruka vždy reálně ve vyšší pozici než program předpokládá.

Na druhou stranu poměrně přesné ovládání můžeme pozorovat kromě středové oblasti i v oblastech vzdálenějších při plném natažení ruky.

Při odečítání z grafu je nutno počítat s vyšší odchylkou měření. Na měření reálných vzdáleností jsem použil jednoduchý vysouvací metr a pravoúhlé pravítka. Již při měření bylo poznat že hodnoty nejsou zdaleka odpovídající těm vypočteným v programu, ale při udání jisté výchozí pozice (tedy posunutí os na sebe) lze z měření vyčíst určité trendy.

Tabulka 5.1: Měření přesnosti ovládání v osách kartézského systému

X v programu	Y v programu	X naměřené	Y naměřené
1407	1491	1720	1570
1150	2647	1515	3040
996	910	1230	1100
774	606	880	590
742	1725	1210	1900
638	1121	1055	1145
594	1676	1150	1845
491	1335	1020	1415
308	2055	1010	2360
161	737	590	720
151	1985	830	2290
-2	1746	750	2085
-104	1059	620	1275
-198	1067	600	1350
-208	1433	620	1820
-226	1037	560	1300
-405	818	450	1230
-428	2913	210	3630
-666	1480	230	2110
-736	1770	125	2470
-929	1609	-80	2410

Závěr

Cílem práce bylo vytvoření aplikace pro ovládání robotické ruky Lynxmotion. Funkce této aplikace měli uživateli poskytnout prostředek kterým lze naprogramovat robota k automatizované činnosti.

Z mého pohledu byly cíle práce splněny. Ovládání robota je na velmi slušné úrovni. Aplikace také umí nahrávat, svým způsobem editovat a nakonec přehrávat ovládací příkazy.

Co se týče automatizované činnosti, tak v tomto směru jsou zde jisté nedostatky. Pro takovouto funkci je bezprostředně důležitá přesnost. Vytvoření sekvencí pohybu za pomoci ovládacího zařízení na tom bude s přesností vždy hůře než například ovládání za využití kamery a algoritmů obrazového rozpoznávání. I v případě nahrání dokonalých sekvencí by bylo pro využití aplikace v praxi třeba, aby robot i výchozí pozice objektů byly vždy na naprosto stejném místě, což by vyžadovalo namontování robota na větší nepohyblivý objekt. Vyzkoušení aplikace s takovouto instalací robota jsem ale nezrealizoval.

Omezení v přesnosti tedy není způsobené aplikací samotnou, ale spíše hardwarovými nedostatky.

Jako největší plus aplikace vidím možnost ovládání robota přepočítáváním souřadnic v kartézském systému. Intuitivnost a tedy přesnost ovládání se díky této vlastnosti několikanásobně zvedla.

Aplikace je dle mého názoru využitelná ve stejné míře, jako tento robotický systém sám o sobě. Je třeba přihlédnout k tomu, že mezi robotem tohoto typu a roboty využívanými v průmyslu a při sériové výrobě je stále veliká propast, to jak v jejich schopnostech, tak cenově. Možnost pracovat s takovýmto kusem hardwaru je nicméně velmi zajímavá a v mnohých ohledech i poučná.

V případě budoucího rozšíření aplikace by bylo užitečné přidat možnosti upravení ovládání. V současné chvíli je ovládání natvrdo mapováno na jednotlivé osy joysticků a nelze jej v bežící aplikaci měnit. Další možnou úpravou by pak byla možnost nahrávání a přehrávání příkazů přesunu po souřadnicovém systému, místo nahrávání příkazů ovládající přímo serva. V takovém případě by ale bylo třeba více odladit pohyb v souřadnicovém systému.

Příloha **A**

Obsah přiloženého CD

Obsahem přiloženého CD je složka BP s pdf souborem bakalářské práce a také zdrojovými soubory LaTeX pro jeho vygenerování. Ve druhé složce RoboticArm je pak umístěna samotná aplikace.

- /
  - `BP` ..... adresář s bakalářskou prací
    - `source` .. zdrojové soubory latex pro vygenerování bakalářské práce
      - `graphics` ..... složka s obrázky a diagramy bakalářské práce
      - `BP_Marvan_Jan_2017.tex` ..... Zdrojový soubor LaTeX pro bakalářskou práci
      - `deskyBP_Marvan_Jan_2017.tex` .... Zdrojový soubor LaTeX pro desky bakalářské práce
      - `zadani_prace.pdf` ..... Naskenované zadání práce
    - `BP_Marvan_Jan_2017.pdf` ..... Bakalářská prace ve formátu PDF
  - `RoboticArm` ..... adresář s aplikací
    - `resources` ..... adresář s grafickými prvky pro GUI
    - `source` ..... adresář se zdrojovými soubory aplikace
    - `makefile` ..... soubor aplikace make pro instalaci aplikace
    - `wt_config.xml` konfigurační soubor pro webserver poskytující GUI
  - `install.txt` ..... instalační manuál aplikace

Příloha **B**

Instalační příručka

Pro instalaci a spuštění aplikace je zapotřebí pouze C++ kompilátor podporující C++11 standard a stažení knihovny Wt. Pro kompilování bych doporučil kompilátor gcc verze 4.8.4 nebo vyšší.

Nejjednodušším způsobem instalace knihovny Wt je použití nástroje apt-get. Stažení a instalaci provedeme zadáním následujícího příkazu do terminálu:

```
$ sudo apt-get install witty witty-dev witty-doc witty-dbg
```

Pokud by z nějakého důvodu instalace knihovny za použití apt-get nebyla vhodná, tak na oficiálním webu knihovny jsou popsány další způsoby stažení a instalace: https://www.webtoolkit.eu/wt/download.

Samotnou aplikaci nainstalujeme ze zdrojových kódů na přiloženém médiu následujícím způsobem:

1. Překopíruj adresář „RoboticArm“ do libovolného adresáře na lokálním disku.
2. V terminálu přejdi do výsledného adresáře „RoboticArm“.
3. Za pomoci příkazu „make all“ zkompiluj všechny zdrojové kódy aplikace.
4. Pro spuštění hlavní aplikace lze pak použít příkaz „make run“. Pokročilejší možnosti spuštění jsou popsány v příloze Uživatelská příručka.

Pro správnou funkčnost bude ještě pravděpodobně zapotřebí přidat aktuálního uživatele do skupiny dialout. Ve většině linuxových distribucí nemá uživatel defaultně přístup k sériovým portům počítače a aplikace pak nedokáže komunikaci na sériovém portu otevřít. Přidání uživatele do skupiny provedeme následujícím příkazem:

```
$ sudo adduser $USER dialout
```

Po přidání uživatele do skupiny je ještě třeba provést restartování systému, či se pouze přihlásit a odhlásit, za účelem nabráni efektu nových systémových práv.

Příloha C

Uživatelská příručka

Spuštění aplikace

Po nainstalování lze aplikaci spustit příkazem „make run“, který obsahuje všechny potřebné parametry pro spuštění aplikace s GUI.

Příkaz „make run“ spouští program následujícím způsobem:

```
./gui --docroot . --http-address 0.0.0.0 \
      --http-port 9090 --approot .
```

Pro správnou funkčnost aplikace je třeba vždy program spouštět s argumentem „–docroot .“ a „–approot .“, aby došlo k načtení zdrojových souborů pro zobrazení webového GUI a správného konfiguračního souboru. Dále grafické rozhraní vyžaduje zadání parametru „–http-address“ a „–http-port“ poskytující adresu a port na kterém se má lokální webserver spustit. Tyto parametry jsou povinné a je vždy třeba s nimi aplikaci spouštět. Všechny ostatní parametry pro webserver lze zjistit při spuštění aplikace s parametrem „-h“ či „–help“, ale jejich využití pro tuto aplikaci není podstatné.

Je důležité aby argumenty pro webserver byly napsány vždy před argumenty ovlivňující ovládání ruky či gamepadu. Funkce knihovny spouštějící webserver požaduje aby výše zmíněné parametry byly předávány přes příkazovou řádku při spuštění a pokud by mezi argumenty pro webserver byl neznámý argument nedošlo by ke spuštění GUI.

Argumenty pro nastavení ruky a gamepadu jsou nepovinné a jsou následující:

- **-classic**
  Při použití tohoto argumentu bude aplikace používat přímé ovládání ruky a nebude přepočítávat pohyb v osách kartézského souřadnicového systému

- **-switchx**
  Prohodí směr osy x při používání ovládání v kartézském systému

- **-joystick path**
  Nastavení cesty k připojenému gamepadu, defaultně: "/dev/input/js0"

- **-armport path**
  Nastavení cesty k sériovému portu kde je robot připojen, defaultně "/dev/ttyUSB0"

V případě neúspěšného připojení gamepadu či sériového portu se při spuštění aplikace vypíše v terminálu chybová hláška. Aplikace přesto ale pokračuje ve spouštění GUI. Pokud se tak stane a ruka nereaguje na pohyb joysticků, aplikaci ukončete (v terminálu za pomocí ctrl+c) a zkontrolujte na jakém portu je gamepad připojený. V případě problému s připojením sériového portu zkontrolujte zda je uživatel ve skupině dialout, nebo zkuste spustit aplikaci v superuživatelském režimu. Pokud problém stále přetrvává, zkontrolujte Baud rate nastavení SSC-32U kontroleru, viz literatura [3], strana 34. Aplikace vyžaduje Baud rate v hodnotě 115200.

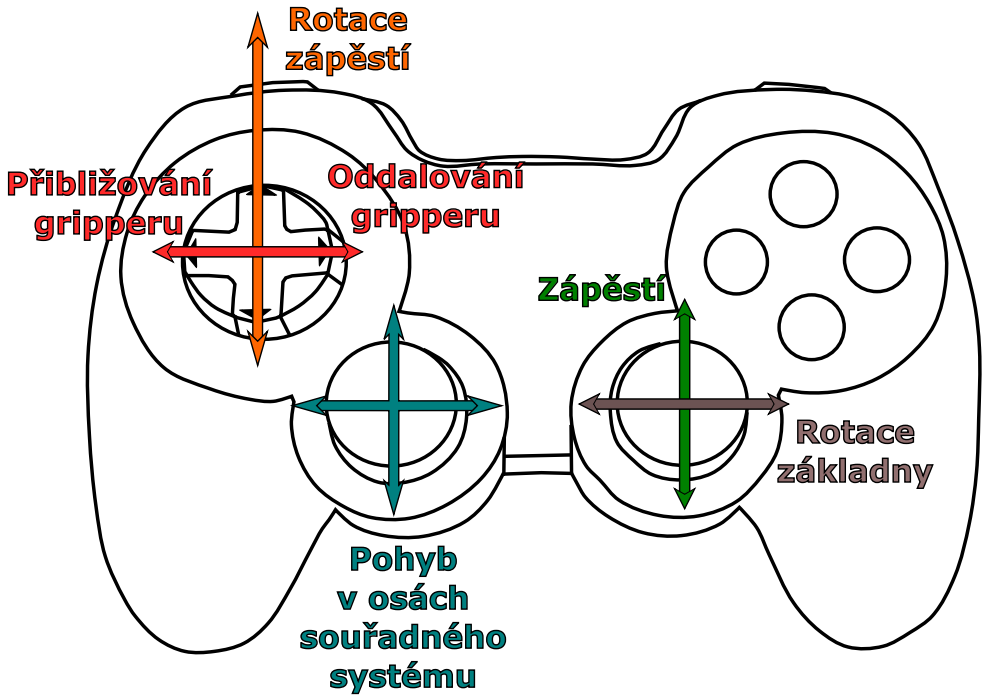
Při spouštění aplikace nejdříve dojde k inicializaci všech tříd. Při inicializaci třídy ovládající ruku dojde k rychlému přesunutí ruky do výchozí pozice. Vzhledem k tomu, že není technicky možné znát polohu serv při spuštění, tak zaslání prvního polohovacího příkazu pohne rukou bez možnosti kontroly rychlosti pohybu.

Po inicializaci všech tříd aplikace zavolá systémový příkaz, který spustí webový prohlížeč. Ve webovém prohlížeči se automaticky otevře stránka uživatelského rozhraní.

Ovládání robotické ruky

Ovládání ruky je popsané na obrázku C.1. Schéma popisuje ovládání v případě používání pohybu v souřadnicovém systému. V případě používání přímého ovládání serv (parametr -classic) je nastavení následující:

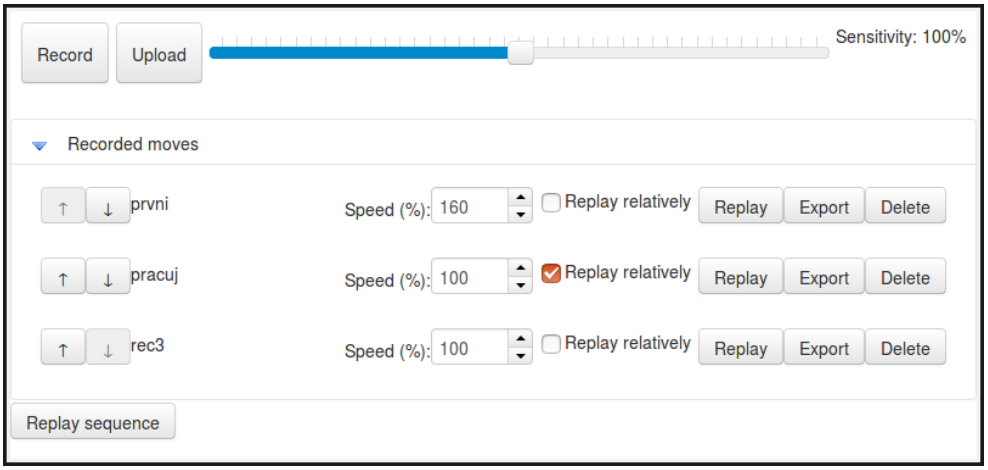
- **Levý joystick, osa X:** rotace základny
- **Levý joystick, osa Y:** rameno
- **Pravý joystick, osa X:** loket
- **Pravý joystick, osa Y:** zápěstí
- **Digitální joystick, osa X:** rotace zápěstí
- **Digitální joystick, osa Y:** gripper



Obrázek C.1: Schéma ovládání gamepadem (v souřadnicovém systému)

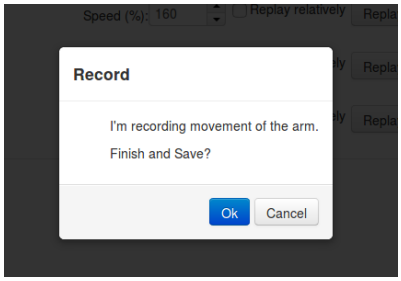
Ovládání aplikace

Ovládání aplikace probíhá v jediném okně, které je vidět na obrázku C.2. Vzhled aplikace i funkce tlačítek ve výsledné aplikaci odpovídá návrhu uživatelského rozhraní.



Obrázek C.2: Stránka uživatelského rozhraní

Pro nahrání nové sekvence pohybu slouží tlačítko Record. Po stisknutí tohoto tlačítka se automaticky spustí nahrávání a objeví dialogové okno (obr. C.3) umožňující ukončení nahrávání a to buď zahozením nahrávky či jejím uložení.



Obrázek C.3: Úkázka dialogového okna v uživatelském rozhraní

Při větším počtu nahrávek se s použitím šipek dají jednotlivé sekvence řadit. Tato vlastnost je využita při použití tlačítka Replay sequence, které přehraje všechny sekvence v seznamu a to i s patřičným nastavením. U jednotlivých sekvencí se dá nastavit rychlost přehrávání a defaultní pozice přehrávání zaškrtávacím boxem Replay relatively. Pokud je box zaškrtnutý tak se sekvence vždy přehraje z aktuální pozice ruky, v opačném případě se ruka před provedením sekvence vždy přesune do pozice ve které byla sekvence nahrána.

Tlačítko Upload slouží k importování záznamů uložených v souboru. Pro uložení nahrávky do souboru slouží tlačítko Export. Posuvník v horní části obrazovky upravuje senzitivitu ovládání ruky.

Použití aplikace bez GUI

Po zadání příkazu „make all“ se vygenerují i dva další spustitelné soubory. Tyto soubory slouží k používání klíčových funkcí aplikace bez spouštění GUI.

Soubor **rec** (spustíte „./rec“) slouží k nahrávání pohybů. Záznamy jsou pak automaticky uloženy do souborů v aktuálním adresáři. Parametry spuštění tohoto programu jsou:

- **-classic** použití přímého ovládání ruky (bez přepočtu souřadnic)
- **-switchx** prohodí směr osy x při používání ovládání v kartézském systému
- **-joystick path** nastavení cesty k připojenému gamepadu, defaultně: "/dev/input/js0"
- **-armport path** nastavení cesty k sériovému portu kde je robot připojen, defaultně "/dev/ttyUSB0"

- **-s arg** nastavení senzitivity ovládání v procentech

Soubor **replay** (spustíte „./replay“) slouží k přehrávání nahrané sekvence. Tento program umí přehrát vždy pouze jednu nahrávku a to pouze z pozice ze které byla nahrána. Parametry spuštění tohoto programu jsou:

- **-f path** cesta k souboru nahrávky která se má přehrát
- **-armport path** nastavení cesty k sériovému portu kde je robot připojen, defaultně "/dev/ttyUSB0"
- **-s arg** rychlost přehrávání v procentech

Literatura

[1] RobotShop: Lynxmotion AL5D 4 Degrees of Freedom Robotic Arm [online]. *robotshop.com*, ©2016, [cit. 2017-04-27]. Dostupné z: http://www.robotshop.com/en/lynxmotion-al5d-robot-arm-hardware-4dof.html

[2] Lynxmotion: Lynxmotion AL5D [online]. *lynxmotion.com*, ©2017, [cit. 2017-04-27]. Dostupné z: http://www.lynxmotion.com/c-130-al5d.aspx

[3] Lynxmotion: *SSC-32U User Guide [online]*. srpen 2015, [cit. 2017-04-14]. Dostupné z: http://www.lynxmotion.com/images/data/lynxmotion_ssc-32u_usb_user_guide.pdf

[4] Lynxmotion: FlowBotics Studio V2 [online]. *lynxmotion.com*, ©2017, [cit. 2017-04-18]. Dostupné z: http://www.lynxmotion.com/p-883-flowbotics-studio-v2-cd.aspx

[5] Lynxmotion: Lynxmotion Visual Sequencer [online]. *lynxmotion.com*, ©2017, [cit. 2017-04-18]. Dostupné z: http://www.lynxmotion.com/p-443-lynxmotion-visual-sequencer-seq-01.aspx

[6] Lynxmotion: SSC-32 Servo Sequencer Utility [online]. *lynxmotion.com*, ©2017, [cit. 2017-04-18]. Dostupné z: http://www.lynxmotion.com/p-895-free-download-ssc-32-servo-sequencer-utility-created-using-flowbotics-studio.aspx

[7] Lynxmotion: FlowArm PLTW [online]. *lynxmotion.com*, ©2017, [cit. 2017-04-18]. Dostupné z: http://www.lynxmotion.com/p-1018-flowarm-pltw-download.aspx

[8] Albatross: The Features of C++ as a Language [online]. *cplusplus.com*, ©2000-2017, [cit. 2017-04-23]. Dostupné z: http://www.cplusplus.com/info/description/

[9] EMWEB: Wt: an introduction [online]. *webtoolkit.eu*, ©2017, [cit. 2017-04-23]. Dostupné z: `https://www.webtoolkit.eu/wt/`

[10] Vydavatelství Nová média: Sinová a cosinová věta [online]. *Matematika.cz*, ©2006—2014, [cit. 2017-05-08]. Dostupné z: `http://www.matematika.cz/sinova-cosinova`

[11] Vydavatelství Nová média: Pravoúhlý trojúhelník [online]. *Matematika.cz*, ©2006—2014, [cit. 2017-05-08]. Dostupné z: `http://www.matematika.cz/pravouhly-trojuhelnik`

[12] Espinosa, R. H.: *Joystick API Documentation [online]*. srpen 1998, [cit. 2017-04-20]. Dostupné z: `https://www.kernel.org/doc/Documentation/input/joystick-api.txt`

[13] Wikibooks.org: Serial Programming/termios [online]. *Wikibooks.org*, duben 2016, [cit. 2017-05-08]. Dostupné z: `https://en.wikibooks.org/wiki/Serial_Programming/termios`

[14] EMWEB: Wt Widget Gallery [online]. *webtoolkit.eu*, ©2017, [cit. 2017-05-10]. Dostupné z: `https://www.webtoolkit.eu/widgets`

[15] EMWEB: Wt: Wt documentation [online]. *webtoolkit.eu*, březen 2017, [cit. 2017-05-10]. Dostupné z: `https://www.webtoolkit.eu/wt/doc/reference/html/index.html`