VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky

Bakalářská práce

2019/2020 **Daniel Seidel**

VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra kybernetiky a biomedicínského inženýrství

Řízení testovacího robotického pracoviště
Control of a laboratory robotic stand

2019/2020 **Daniel Seidel**

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra kybernetiky a biomedicínského inženýrství

Zadání bakalářské práce

Student: **Daniel Seidel**

Studijní program: B0714A150001 Řídicí a informační systémy

Téma: Řízení testovacího robotického pracoviště
Control of a Laboratory Robotic Stand

Jazyk vypracování: čeština

Zásady pro vypracování:

Bakalářská práce se zabývá návrhem řídicí robotické aplikace. Šestiosý robot od firmy Mitsubishi Electric je umístěn v laboratorní instalaci. Umožňuje manipulaci s díly, jednoduchou montáž a testování.
Cílem práce bude vytvořit aplikaci zajišťující specifikovaný úkol.

1. Seznámení se s konstrukcí laboratorního pracoviště s šestiosým robotem.
2. Analýza funkcí laboratorního pracoviště.
3. Návrh a realizace řídicí aplikace robota.
4. Testování funkčnosti robotického pracoviště.
5. Zhodnocení výsledků.

Seznam doporučené odborné literatury:

[1] ČSN EN ISO 10218. *Roboty a robotická zařízení - Požadavky na bezpečnost průmyslových robotů.* Praha: Úřad pro technickou normalizaci, metrologii a státní zkušebnictví, 2012. Třídicí znak 186502.
[2] Technická literatura firmy Mitsubishi Electric.
[3] BOUCHARD, Samuel. *Lean Robotics: A Guide to Making Robots Work in Your Factory.* [s.l.]:Samuel Bouchard, 2017. ISBN 1775082903.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **doc. Ing. Jiří Koziorek, Ph.D.**

Datum zadání: 01.09.2019
Datum odevzdání: 30.04.2020

doc. Ing. Jiří Koziorek, Ph.D.
vedoucí katedry

prof. Ing. Pavel Brandštetter, CSc.
děkan fakulty

Prohlášení studenta

Prohlašuji, že jsem tuto bakalářskou/diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě dne: *15. května 2020*

podpis studenta

Poděkování

Rád bych poděkoval doc. Ing. Jiří Koziorek, Ph.D. za odbornou pomoc a konzultaci při vytváření této bakalářské/diplomové práce.

Abstrakt

Práce je o seznámení se s daným robotickým pracovištěm a technickým zvládnutím jeho ovládání a manipulaci, při průběhu práce by měl být robot schopen složit a rozložit jednodušší stavbu z lego kostek a umisťovat ji mezi zásobníkem, odkládacím panelem a karuselem, dále by měl mít předpřipraveny další programy pro možnost výuky či předvedení provozuschopnosti pro další práci na tomto stroji. Mezi dodatek patří pokus o předávání materiálu skrze dvířka pro následující stroj od Mitsubishi tak aby s ním bylo dále možné pracovat.

Klíčová slova

Robotické rameno; lego kostka; zásobník; odkládací panel; karusel

Abstract

The work is about getting acquainted with the robotic workplace and the technical mastery of its control and manipulation, during the work, the robot should be able to assemble and unfold a simpler structure from lego cubes and place it between the storage tank, storage panel and karusel. Furthermore, he should have prepared more programs for the possibility of teaching or demonstrating operability for further work on this machine. The addition includes an attempt to pass the material through the door for the next machine from Mitsubishi so that it can be further used.

Key words

Robotic arm; lego cube; karusel; storage tank; storage panel

Obsah

Seznam obrázků

Úvod

Cílem této práce je vytvořit řídicí aplikace pro robotické rameno Mitsubishi a možnost spolupráce s vedlejším robotickým pracovištěm. Jedná se o robotické rameno, které slouží převážně k manipulaci s lehkými předměty. Jeho úkolem je uchopení předmětu a jeho následné přesunutí z místa na místo. K uchopení předmětu slouží kleště nebo jiné námi zvolené zařízení. Robotické rameno obsahuje několik pomocných prvků, které slouží k jeho ochraně. Ochrana předdefinovanými prvky však není dostatečná, a proto využíváme ochrany konstrukce, ve které je robotické rameno umístěno.

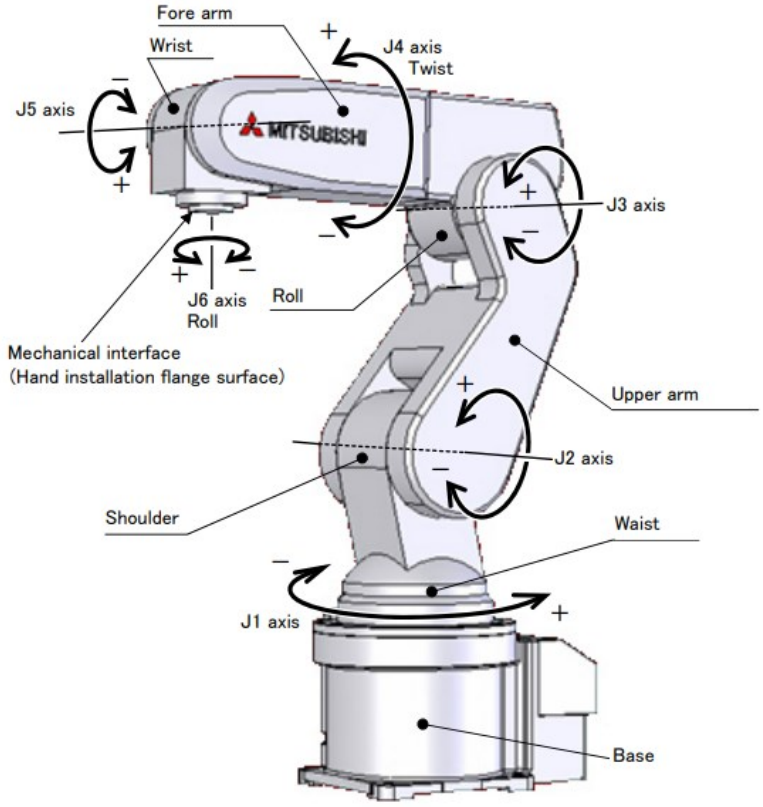
Návrh konstrukce byl vytvořen na základě požadavků, pro vytvoření budoucích aplikací. Konstrukce obsahuje zásobníky kostek, překladiště kostek a karusel. Zásobníky kostek slouží ke vkládání či odebírání kostek uživatelem. Překladiště umístěná na bočních stěnách, slouží k uchopení kostek pod správným úhlem. Karusel bude využíván k vytvoření stavby, neboť se jedná o desku Lego. Veškeré komponenty byly testovány v době návrhu konstrukce, a za pomoci získaných výsledků byly některé parametry změněny pro lepší dostupnost robotického ramene.

1 Seznámení se s konstrukcí laboratorního pracoviště s šestiosým robotem

Laboratorní pracoviště obsahuje robota s šestiosým ramenem, pracovní název je RV-2F-Q a skládá ze dvou částí, robotického ramene a kontroléru.

1.1 Robotické rameno, kabely

Jedná se o šestiosý typ RV-2F od firmy Mitsubishi z řady MELFA,RV-F je zároveň nejnovější generace této řady, vybaven vyspělou technologií a snadnými ovládacími prvky.



Obr.1.1: Robotické rameno RV – 2F – Q popis pohybových os

Mezi základní doručené prvky patří šestiosé rameno a kabely pro C750 a CR751 kontrolery, zákazník dále může požádat o nadstandardní vybavení:

- set elektromagnetických ventilů (1E-VD01/1E-VD01E, 1E-VD01/1E-VD01E),
- ručním vstupním panelem (1S-HC30C-11),
- ručním výstupním kabelem (1E-GR35S),
- setem ručních prstencovitých kabelů (1E-ST0402C, 1E-ST0404C)
- uzávěr na omezení provozní vzdálenosti (J1 osa: 1S-DH-11J1, J2 osa: 1S-DH-11J2, J3 osa: 1S-DH-11J3)

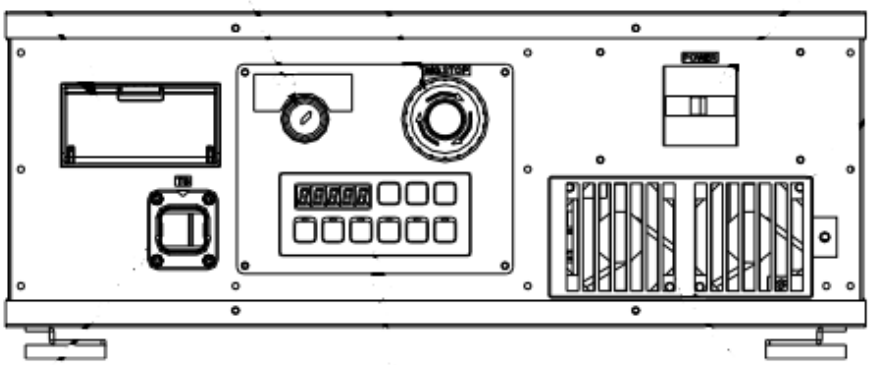
1.2 Kontroler

Kontrolér obsahuje několik zařízení které k němu mohou být připojeny,mezi základní dodané zařízení patří:

- pohonná jednotka (Skládající se z CR750-02VQ-1 a CR751-02VQ),
- robotické CPU(Q172DRCPU), baterie(Q170DBATC)
- kabely na propojení robotického CPU(TU, DISP, EMI, SSCNET III)

Mezi další možnosti patří:

- Výukový ovladač(Simple T/B, Highly efficient T/B)
- Instrukční manuál a CD s RT ToolBox2(mini).



Obr.1.2: Kontroler

Propojení s počítačem je možné několika způsoby:

- USB - přímé spojení počítače a robotického ramene.
- Ethernet – Připojení za pomoci sítě (pro nás za pomoci ethernet kabelu či Wi-Fi sítě)
- RS-422 – Sběrnice dat

My využíváme připojení za pomoci sítě ethernet(Wi-Fi) a tudíž je možné se k robotickému ramenu připojit odkudkoliv z učebny, v níž je umístěn, a také jej ovládat pokud daný počítač má nainstalovanou sadu RT Toolbox3 a vytvořeno základní prostředí pro manipulaci s robotem.

1.3 Výukový ovladač

Výukový ovladač je využíván jako manuální, snadno použitelný ovládací přístup pro robota. Může sloužit k vytváření jednoduchých programů, k seznámení se s možnostmi pohybu robota a dalším možným ukázkovým akcím.

Obr.1.3: Highly efficient T/B(výukový ovladač)

1.4 Základní informace

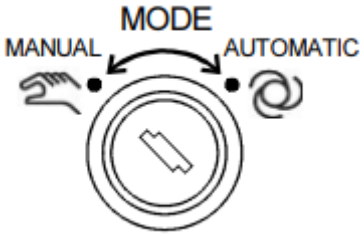
Mezi základní prvky, se kterými jsem se již seznámil, patří nově využívaný RT ToolBox3, 6osé rameno, Skříň s magnetickými zámky, operační panel, dotykový panel, karusel a výukový ovladač. Celý robot může být řízen za pomoci výukového ovladače kdy operační panel je přepnut do stavu MANUAL nebo za pomoci programu RT ToolBox3 při přepnutí do stavu AUTOMATIC. RT ToolBox3 Bude nejběžnější pracovní prostředí se kterým se při této bakalářské práci setkám, samostatně má 3 módy a to jsou Offline,Simulator a mód Online při kterém si jde zobrazit i virtuální panel na ovládání jednotlivých motorů, přepínání typu pohybu a vkládání programu.
Samotný šestiosý robot je složen z 6 motorů a je zde možnost výměny úchopového zařízení. Rameno jsou přiděleny 3 typy pohybů, které mohou mít samostatné využití. Mezi ně patří pohyb PTP (Point to point), LIN (Lineární) a CIRC (Pohyb po kružnici). Každý z těchto pohybů má nějaké výhody a zároveň se každý zapisuje jiným kódem do programu, nejrychlejší pohyb je vykonán při PTP kdy se rameno nepohybuje po přímce ale po obloucích, nejkratší trasu zajišťuje pohyb LIN kdy se pohyby všech motorů kombinují a vytvářejí přímkový pohyb a poslední pohyb je CIRC kdy se v programu zadávají 3 body kterými bude následně při rotačním pohybu procházet a to počáteční bod, bod průchodu a koncový bod, tímto způsobem se můžeme vyvarovat nepříjemnostem které nám může způsobit pohyb PTP u kterého si nejsme jisti jestli si při zvolení nejkratší dráhy bude rameno rotovat po či proti směru hodinových ručiček.

1.5 Základní ovládání

Na ovládání robotického ramene se může využít výše uvedeného výukového ovladače, který se dá efektivně využívat pro ukázkové či výukové pohybové modely, nebo využít program vyvinutý firmou Mitsubishi s vlastním jazykem pro jednoduché programování s obsaženou možností pohybů samostatných kloubů robotického ramene.

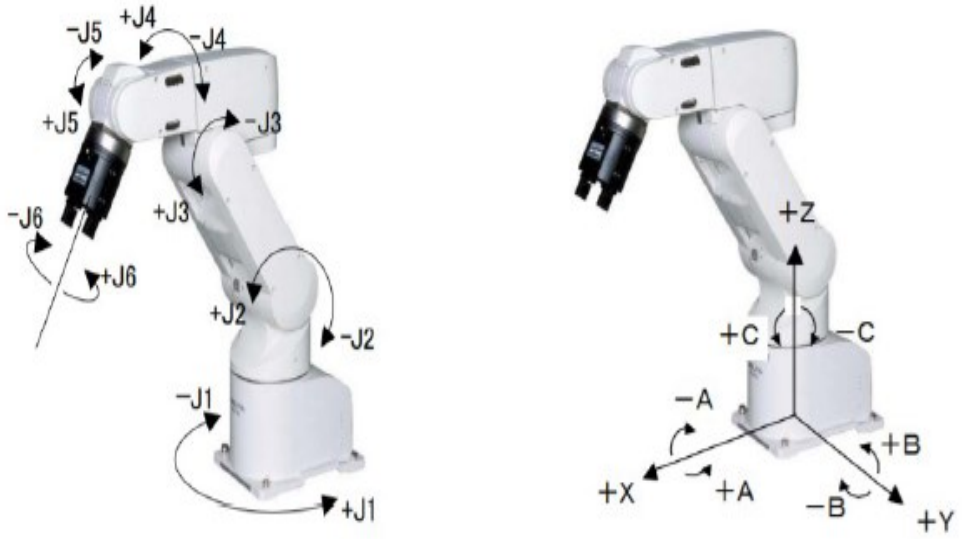
1.5.1 Ovládání za pomoci výukového ovladače

Při ovládání přes výukový ovladač je nutné mít přeplý operační panel do stavu MANUAL.

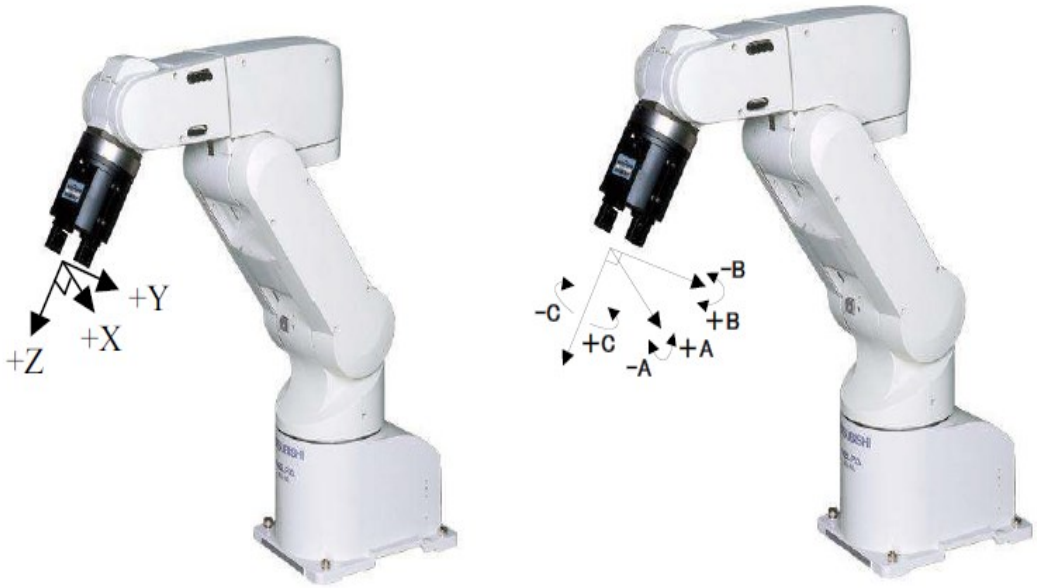


Obr.1.5.1: Přepnutí módu MANUAL/AUTOMATIC

Zobrazí se vám výpis chyb které nyní robot má, pokud nemá žádnou stačí zmáčknout reset a zobrazí se vám panel na ovládání. Je důležité, pokud chcete s motorem hýbat v RT držet spodní tlačítko které slouží ke kontrole že jej ovládá někdo kdo má na přístroj dohled, jinak se nepodaří spustit servo motory.
Na výukovém ovladači jde hýbat s jednotlivými servo motory zvlášť,měnit typ pohybu mezi JOINT, XYZ a Tool coordinates, rychlost pohybu motorů či dokonce psát kód pro automatické ovládání (nedoporučuje se,pomalé z hlediska zápisu) či zobrazení knihovny předchozích programů a jejich spuštění.



Obr. 1.5.1: World Vlevo: pohyb JOINT Vpravo: pohyb XYZ



Obr. 1.5.1: Tool Vlevo: pohyb XYZ Vpravo: pohyb ABC

1.5.2 **Ovládání za pomoci RT Toolbox 3**

Při ovládání přes RT Toolbox 3 je nutné mít přeplý operační panel do stavu AUTOMATIC. Tento mód dovolí vzdálené ovládání. Při přepnutí RT Toolboxu do online módu se vám zobrazí aktuální pozice chapadla a přesná pozice těla dle souřadnicového systému. Lze si zde zobrazit i přesný ovládací panel jako je uveden na výukovém ovladači se stejnými funkcemi, psaní kódu je však díky klávesnici snadnější.

Díky ochranně se vás program pokaždé, když změníte rychlost pohybu, zeptá jestli tuto změnu potvrzujete. Při změně rychlosti totiž hrozí možné poškození prostředí popřípadě robotického ramene, tomu lze předejít za pomoci nastavení ochrany User-defined area a Free plane limit které nám při možnosti srážky či nemožného pohybu zastaví celý program a upozorní nás na chybu v programu jez je potřeba odstranit či upravit pohybovou dráhu abychom této situaci nadále předešli.

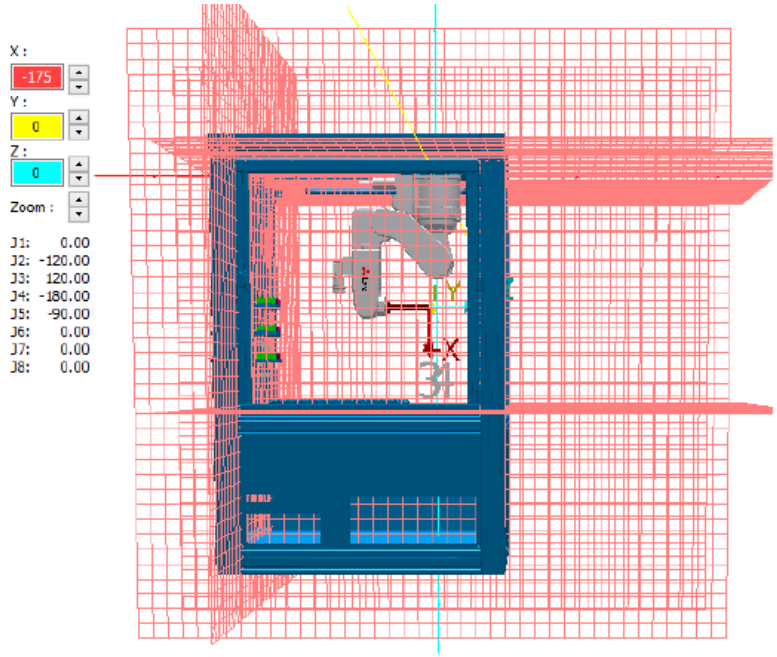
1.6 **Ochrana**

Mezi funkce patří i bezpečností ochranné prvky, ty mohou být buďto sogtwarové nebo hardwarové. Hlavními softwarovými prvky je takzavý Free plane limit a User-defined area které zabraňují kolizi mezi chapadlem a předem určeným ohraničením ve kterém se mohou nacházet různé objekty, stěny či jiná zařízení a dle nastavení se může rameno zastavit a vykázat hlášení o chybě nebo rovnou zastavit všechny servo motory. Nevztahuje se to bohužel na tělo robota kdy chapadlo se do vybrané zóny nedostane ale díky přetáčení tělo ano a může způsobit kolizi s předměty při snaze dostat se tam, kam bylo programem určeno, dokud nedojde k přetížení

servo motorů. Hlavním hardwerovým prvkem je jednoduché STOP Button, které má typickou červenou barvu s žlutým podkladem. Slouží k úplnému zastavení robota a nachází se na přední straně kde je snadno přístupné

1.6.1 **Free plane limit**

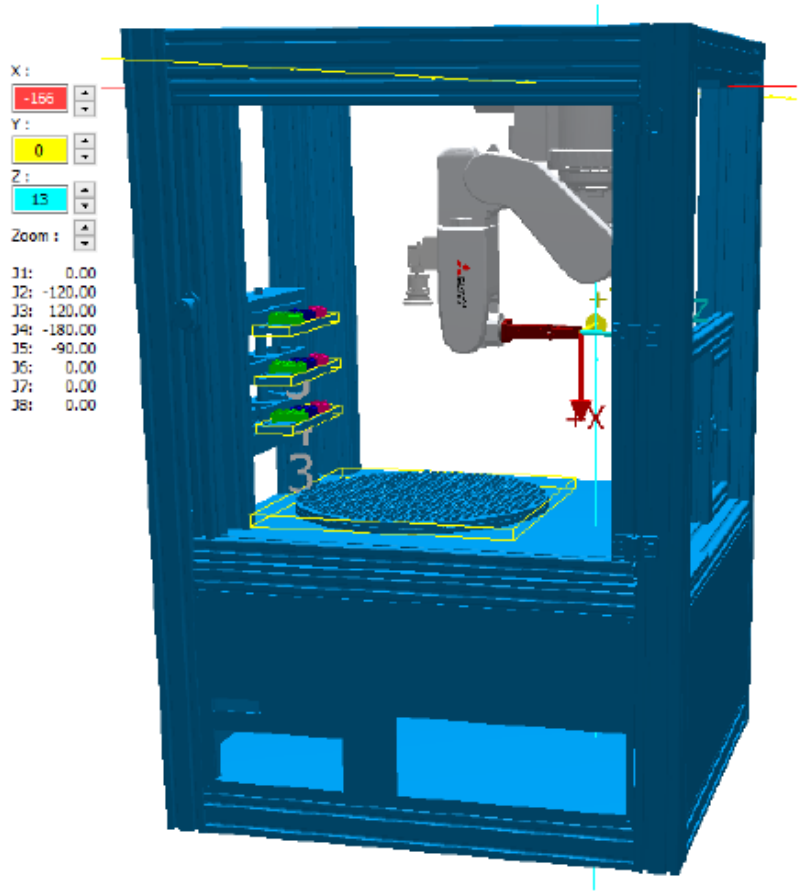
Vytváříme zde zóny které brání kolizím chapadla a prostředí které má free plane limit chránit. Tato plocha je určena třemi body v prostoru a díky nim se vytvoří stěnová mřížka. Tímto si robot zapíše souřadnice a je schopen upozornit na fakt že se chapadlo dostalo do těchto souřadnic zvukovým varováním či úplným zastavením servo motorů (záleží na původním nastavení).



Obr. 1.6.1: Free plane limit

1.6.2 **User-defined area**

Zóny mimo pracovní prostor vyobrazeny jako kvádry. Jsou zadávány 2 body v prostoru které následně tento kvádr vytvoří. Je zde typ informační a zakázaný. Informační způsobí že pokud chapadlo najede do této zóny, spustí se akustická signalizace. V případě zakázané se okamžitě vypnou servo motory a vyvolání chybové situace která zastaví jakýkoliv další pohyb, v této situaci se může stát, že robot nebude možné dostat z této chyby ovládáním a tak jej musíte mechanicky lehce posunout z této oblasti aby bylo možné rameno vyresetovat a dále s ním pohybovat.



Obr. 1.6.2: User-defined area

1.6.3 **Hardwarové prvky**

Stop button se nachází na přední straně konstrukce robota, aby bylo snadno přístupné a obsluha mohla reagovat rychle při chybovém stavu. U robotů si umístění STOP tlačítek definuje výrobce na rozdíl od tlačítek, které se nacházejí v halách, u těch je nutné dodržovat norrmy které jsou stanovené. Příklady uvedení pár norem:

- ISO 12100:Bezpečnost strojních zařízení - základní pojmy a všeobecné zásady pro konstrukci.
- IEC 51508:Funkční bezpečnost elektrických, elektronických a programovateln ých elektrických systémů souvisejících s bezpečností. Základní norma pro bezpečnostní funkce.

2 Analýza funkcí laboratorního pracoviště

Využití tohoto pracoviště je možné díky jeho funkcím a možným dodatkům vidět hlavně v průmyslu ve kterém může nahradit snadnou manuální práci jako je například předávání výrobku z jednoho pásu na druhý, třídění podle velikosti či barvy nebo následné ukládání do zásobníků. Výše uvedené funkce může zajistit kamera s rozpoznáváním objektů či laserový měřič vzdálenosti. Díky těmto funkcím a dodatkům je tedy možné aby tento robot pracoval s velmi velkou přesností ale pouze s omeznou váhovou hodnotou.

2.1 Způsoby prgramování

Robotické programování se od kdysi původních metod programování, jako je vytlačování děr do pásku z papíru, posunulo kupředu a stalo se mnohem intuitivnějším. To bylo způsobeno touhou udělat programování rychlejší a jednodušší pro operátora. Používají se tři populární metody, které určitě nepotřebují děrovaný papír.

Jak tedy zvolit správný způsob? Nejlepší metoda záleží na úkolu, jaký je zadaný robotu a na požadavcích. Je potřeba využít poznatků uvedených níže ve výhodách a nevýhodách a na základě těchto prvků se rozhodnout kterou metodu použít. Pokud je potřeba snížit prostoj, pak tedy bude ideální použít simulaci, pokud máte úkol vyžadující nepravidelné tvary, pak je ideální učení demonstrací.

2.1.1 Výukový ovladač

Je to nejpopulárnější metoda programování. Podle britské Automatizace a Robotická asociace, přes 90% robotů je programováno právě touto metodou. Novodobé výukové ovladače jsou převážně s dotykovou obrazovkou. Při programování za pomoci výukového ovladače operátor přemisťuje rameno z pozice na pozici a ukládá je stroji do paměti, jakmile je celý tento proces hotov, robot si jej může zpětně přehrát v plné rychlosti.

Výhody:

- Většina tradičních robotů je dodána s výukovými ovladači, ty přijdou povědomé technikům
- Dovolují dosáhnout velmi přesných pozice díky tomu, že robot může být naprogramován pomocí číslicových souřadnic.
- Jsou velmi dobré pro jednoduchý pohyb, jako je kresba po rovné přímce či velké ploše

Nevýhody:

- Robot musí být nastaven do „učícího módu“ kdy je nemožné jej jakkoliv jinak využít, to znamená zastavení všech procesů.
- Je zapotřebí trénink abychom byli schopni efektivně učit a programovat.
- Může být náročné pro řemeslníky, kteří nejsou seznámeni s programováním.

2.1.2 **Simulace/offline mód**

Nejčastěji využíváno v robotickém výzkumu, kdy je zapotřebí se ujistit, že pokročilé řídící algoritmy jsou prováděny správně, než se převedou na reálného robota. Také se ale využívají v průmyslu na snížení prostojů a zvýšení efektivnosti. V offline módu se programuje za pomoci nasimulovaného robota a úkolu. To může být rychlá cesta, jak otestovat nápad před tím, než jej přeneseme do robota

Výhody:

- Sníží prostoj, který je potřebný pro programování. Program je psán offline a robota je potřebné zastavit až tehdy, kdy se program nahrává´a je testován.
- Může být intuitivní, hlavně pokud máme 3D model a prostředí ve kterém jsou možné dané úkony.
- Jednoduché na testování několika různých přístupů ke stejnému problému, který by v případě online programování byl neefektivní.

Nevýhody:

- Virtuální modely nebudou nikdy schopny reprezentovat reálný svět, a proto musí být programy upraveny po tom, co jsou do robota nahrány.
- Celkový čas strávený nad problémem může být delší z důvodu vývoje prostředí a stejně jako testování na daném robotu.
- Někdy se může stát, že strávíte více času nad řešením simulačního procesu než nad řešením výrobního problému.

2.1.3 **Učení demonstrací**

Tento způsob přidává intuici ke klasickému výukovému ovladači. Tato metoda obsahuje pohyb robota prostředím za pomoci joysticku či silového senzoru. Mnohé robotické společnosti zavedli tento způsob programování do svých robotů z důvodu jednoduchého ovládání pro operátora a začít jej okamžitě využívat.

Výhody:

- Rychlejší než výukové ovladače, odstraňuje problém s nutností zmáčknout několik tlačítek najednou. Místo toho operátor jednoduše hýbe robotem do potřebné pozice.
- Mnohem intuitivnější než oba předchozí způsoby. Tento způsob nevyžaduje žádnou znalost programování nebo 3D prostředím
- Velmi dobré pro datailní úkoly, kde by bylo potřeba velké množství řádků kódu jako například kresba složitých tvarů.

Nevýhody:

- Je zapotřebí reálné robotické pracoviště, to způsobí, že není možné snížit prostoj robota tak jako v offline módu.
- Pohyb robota do přesně daných souřadnic není tak přímočarý tak jako u zbylých metod. Toto je hlavně pravda při využívání systému se základním využitím joysticku, kde není možné vložit číselnou hodnotu.
- Není vhodné pro úkoly, které jsou algoritmické od původu. Na ukázku, pokud by měl robot nakreslit plochu pohybem horizontálním přes povrch, následně se posunout o centimetr dolů a pohybovat se zase horizontálně v opačném směru a tak dále. Bylo by velmi náročné toto učit robotu rukou a také neefektivní.

2.2 Základní omezení pro využití

Pro základní projekt, který by měl obsahovat robotické rameno, je první omezení pořizovací cena celého projektu. První, co je zapotřebí je dané robotické rameno a jeho součásti, cena tohoto prvku se může pohybovat v rozmezí jednoho až dvou miliónů korun. Dále systémové prvky, další materiály a klientská technika vyjdou na další 3-4 miliony. To dává celkovou cenu v rozmezí 5-7 miliónů za celkový projekt. V takovém případě nemusí být vratná doba dostatečně rychlá na to, aby se vyplatila daná pozice automatizovat.

Jsou zde 4 důvody proč je tak náročné rozhodnout, zda je vhodné využít robota na danou pozici.

2.2.1 Vysoká množství, malá variabilita změn

Původně byli roboti sestrojeni k obsluze a výrobě automobilových karoserií. Karoserie jsou produkovány v obrovských množstvích a téměř se neliší, proto jeden program může sloužit řadu let beze změny či jen s malou úpravou. To znamená, že výrobci karoserií mohou amortizovat cenu jejich systémových prvků nad počtem vyrobených kusů.

2.2.2 Chybí normy, podle kterých by se řídilo

Jako výsledek zde máme, že každý výrobce robotů používá svůj vlastní ovládací a operační systém, a každý z nich podporuje jiný komunikační protokol, který ve většině případů si musíte zaplatit jejich používání.

Proto při práci s různými typy robotů od různých firem je tak náročné vytvořit jednotnou komunikaci pro všechny, tím že každý výrobce má vlastní komunikační protokol, který není kompatibilní s protokoly jiných výrobců se vytváří obrovská překážka, která musí být vyřešena ještě před tím, než se začne vyvíjet aplikace.

2.2.3 Fyzický svět

Oproti počítačům se roboti setkávají i s fyzickým světem. IT svět je přehledný a čistý (skládá se z 1 a 0), kdežto fyzický svět ohrožuje prach, nepřehlednost, možnost výskytu nepředvídatelných akcí a podobné.

Aktuální roboti jsou dobře přizpůsobeni ke zvládání opakovaných a logických úkolů. Celkově ale selhávají v neuspořádaném prostředí, aplikování nových a improvizovaných metod.

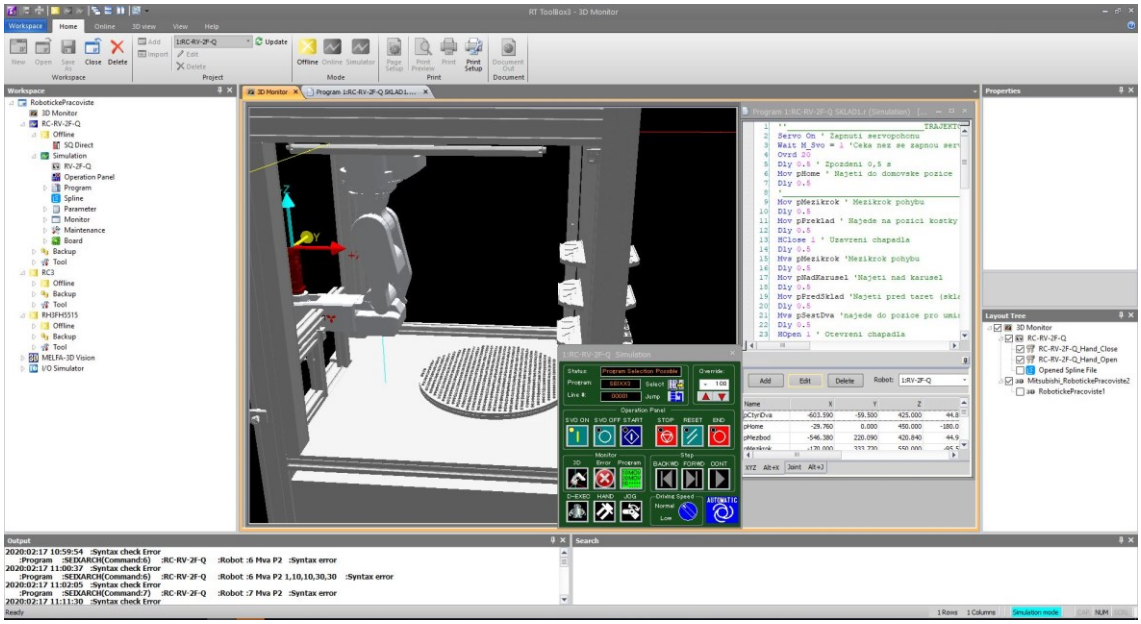
2.2.4 Nedostatek zaměstnanců vzdělaných v robotice

Toto je jeden ze základních problémů, nedostatek zaměstnanců, kteří jsou schopni roboty ovládat a pracovat s nimi. Ve většině malých a středních firem se nenachází ani jeden expert na toto téma. To způsobuje, že firma je limitována nejen tak, že nedosáhne na nějaký velký projekt, ale často je problém získat už jen jediného robota.

2.3 RT Toolbox3

Toto je programovací prostředí, které používají všichni roboti od firmy Mitsubishi s využitím programovacího jazyku Melfa IV, V a VI (využití typu záleží na modelu robota). RT ToolBox3 podporuje všechny procesy od spuštění systému do ladění a obsluhy, včetně programování a úpravy, ověření rozsahu operací před zavedením robota, odhad doby taktování, ladění robota před spuštěním a monitorování podmínek a poruch robotů během operací.

Pomocí programu lze připravit program jak v offline režimu, tak v simulaci reálného robota s jeho grafickým 3D modelem či následné spuštění pro online nahrání programu nebo jednoduché ovládání jako ukázkový model.



Obr.2.3: Úvodní panel pro RT Toolbox3

2.4 MELFA-BASIC

MELFA-BASIC je programovací jazyk, který dále rozšiřuje a vyvíjí příkazy potřebné pro řízení robota.

V MELFA-BASIC může rozšíření příkazu, stejně jako paralelní zpracování nebo strukturování, které bylo obtížně realizovatelné v jazyce BASIC, umožnit snadné ovládání MELFA.

<Example of a Pick & Place program>

```
Mov Psafe          'Move the evasion point
Mov Pget,-50       'Move the workpiece
                    extraction position up
Mvs Pget           'Move the workpiece
                    extraction position
Dly 0.2            'Wait 0.2-sec. on standby
Hclose 1           'Close the hand
Dly 0.2            'Wait 0.2-sec. on standby
Mvs Pget,-50       'Move the workpiece
                    extraction position up
Wait M_In(12)=1    'Wait for a signal
Mov Pput,-80       'Move the workpiece
                    position up
Mvs Pput           'Move the workpiece
                    position
Dly 0.2            'Wait 0.2-sec. on standby
Hopen 1            'Close the hand
.....
```

Classification	Main functions
Operation-related	Joint, linear, and circular interpolation, optimal acceleration/ deceleration control, compliance control, collision detection, and singular point passage
Input/output	Bit/byte/word signals, interrupt control
Numerical operations	Numerical operations, pose (position), character strings, logic operations
Additional functions	Multi-tasking, tracking, and vision sensor functions

Obr. 2.4:Ukázka základních příkazů [3]

2.5 Způsob programování chodu robota

Programování je až druhý bod, na který by se programátor měl zaměřit, prvním prvkem je zjištění požadavků klienta co by měl být program a robot schopný udělat. U našeho projektu bude zapotřebí vytáhnout lego kostky ze zásobníku, uchopení a její vyzvednutí. Další krok je následné přenesení kostky na odkládací okraj popřípadě rovnou na karusel pokud víme, že se nám tato kostka okamžitě hodí a není potřeba si ji odkládat na později. V případě že se nám kostka nehodí a víme, že bude použita později, umístíme ji na odkládací pozici. Celý proces se bude opakovat, dokud nebude postavena daná stavba, v našem případě ale předpokládám, že kostky budeme přemisťovat okamžitě na karusel, pokud nenastanou žádné potíže.

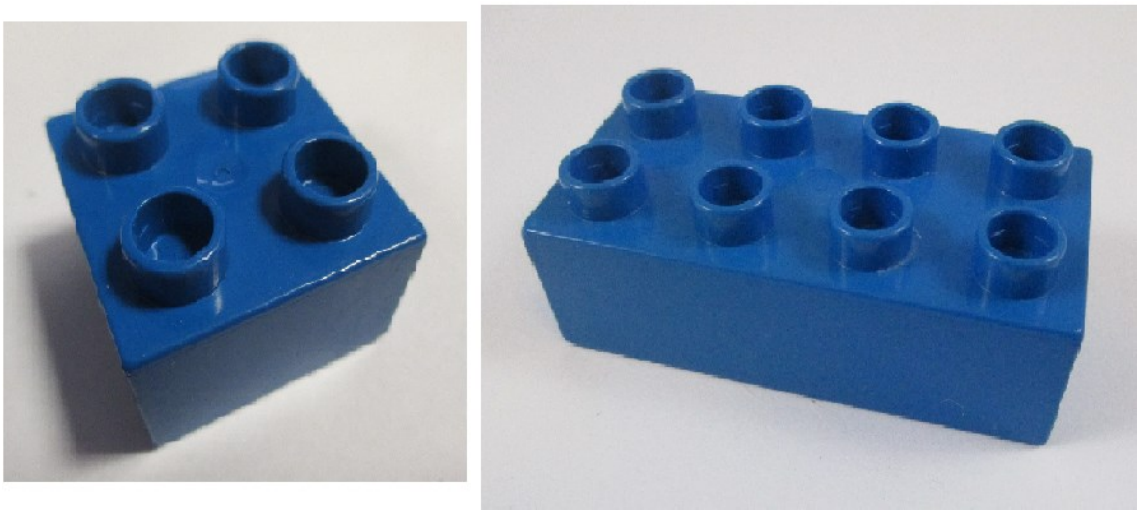
Celý tento proces nám umožňuje snadné programování pomocí RT Toolbox3 a jeho programovacího jazyku MELFA-BASIC. Díky příkazům na uchování pozice můžeme předem mechanicky nastavit základní pozice, které následně využijeme při programování a odkazování se na pohyb na tyto pozice. Následný pohyb nemusí být přímo na tuto pozici, lehkým přidáním závorek a do nich daných hodnot můžeme tuto pozici posunout pouze pro tento jeden pohyb, do závorek se uvede o kolik se má změnit hodnoty X, Y a Z.

3 Návrh a realizace řídicí aplikace robota.

Tato kapitola se zabývá návrhem a realizací řídící aplikace robota. Způsob ovládání, který bude použit a očekávaný výsledek dané aplikace.

3.1 Dopravení kostek ze zásobníku

Jako materiál použitý na stavbu tvarů byly zvoleny Lego kostky s třemi různými rozměry. Tyto kostky mají rozměry 2·2 a 2·4 výběžky.



Obr.3.1:Příklady použitých kostek

Úvodním úkolem bude zapotřebí přenést požadované kostky na dané pozice karuselu a podkladu, pokud bude možné je nabrat v zásobníku bez přetěžování a přílišného přetáčení ramene, pak je můžeme rovnou dostat na potřebnou pozici. Pokud ovšem nebude možné tyto požadavky splnit pak tedy využijeme odkládací plochy připevněné po bocích ochranné konstrukce pro robotické rameno.

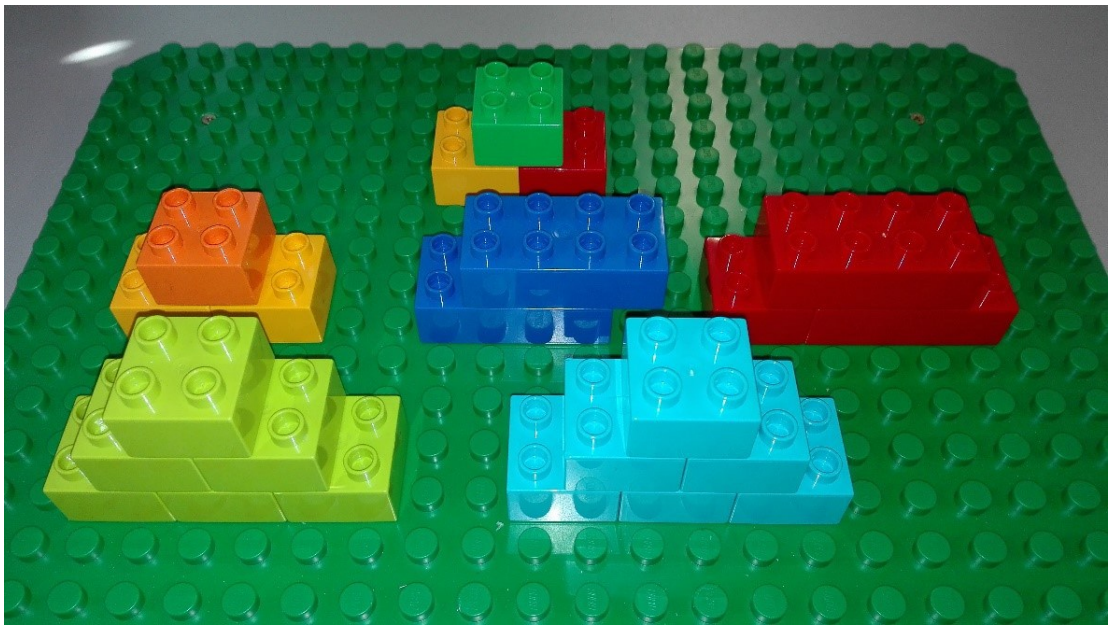
Budu využívat způsob programování v simulaci/offline módu i když tento úkol není tlačen časově. Při programování přesunu kostek na podklad je zapotřebí znát jak chceme, aby výsledná stavba vypadala, v úvodních řádcích každého programu pro různé stavby budou vždy ty samé příkazy:

- Spuštění servo motorů
- Čekání na spuštěný servo motor
- Nastavení počáteční rychlosti ramene
- Vyhlazení dráhy robota
- Najetí do výchozí pozice

Další příkazy budou už dále obvykle jiné a budou sloužit k vyzvednutí potřebných kostek ze zásobníku a využití při stavbě daného tvaru, který bude žádán (Vždy v názvu programu).

3.2 Pozice při stavbě

Při stavbě je potřeba kontroly aby na sebe kostky doléhaly a pokud použijeme kostky 2·4, je zapotřebí, aby dané kostky nebyly skládány přímo na sebe, musí být skládány aspoň s částečným posunem, pokud chceme předejít k možné kolizi a nestabilitě stavby, vhodné sestavení je znázorněno v obrázku 3.2



Obr.3.2: Spojení lego stavby

Pro správné dosednutí je vždy potřeba dané síly, která nám kostky spojí dostatečnou silou. Toho bychom mohli dosáhnout, pokud ukládací pozici nastavíme níže než je aktuální reálná pozice, díky poznatkům z předchozí práce s tímto robotem vím, že úchopovému rameni tyto kostky lehce kloužou, to zamezí zadření a bude dále možné rameno posouvat i když bude kostka již dolehlá.

Díky možnostem, které RT Toolbox nabízí, by bylo vhodné zkusit vytvořit mnoho základních programů, které by se následně šli využít k sestavení složitějších programů pouze za pomoci těch základních. V praxi by měl kdokoliv po úvodním seznámení mít možnost vytvořit si program dle vlastních potřeb, nebo daného zadání.

Typy navržených programů:

- Přenos kostky z daného patra a dané pozice zásobníku na překladiště
- Převzetí kostky z překladiště a přenesení nad karusel
- Převzetí kostky z karuselu a následné uložení na překladiště
- Převzetí kostky z překladiště a uložení do zásobníku

Během těchto programů nepředpokládáme, že by v boxu bylo celkově více než 5 kusů od každé velikosti jednotlivých kostek pokud nebude zprovozněn otočný zásobník.

3.3 Sestavení programu

Při sestavování programu bude potřeba napočítat, kolik a jaký typ kostek bude zapotřebí použít, například pro danou krychli, která by byla s výplní, by v nejlehčím případě bylo zapotřebí 64 kostek s 2·4 výstupy, pokud nebude požadována výplň, pak nám bude stačit pouze 48 kostek s 2·4 výstupy.

Program bude sestavován pro přesně dané parametry objektů, Jeho obsah se bude sestávat z PTP a lineárního pohybu.

Příklad programu složených z těchto dvou typů pohybů:

1. HOpen 1 Otevření chapadel
2. Mov pHome PTP na home pozici
3. Mov pHornisklad
4. Mvs pHornisklad1 LIN na Horní sklad kostky 1
5. Mvs pHornisklad1, -15 LIN na Horní sklad kostky 1, Y = -15 mm
6. HClose 1
7. Mvs pHornisklad1
8. Mvs pHornisklad
9. Mov pKostka1
10. Mvs pKostka1, -20
11. HOpen 1

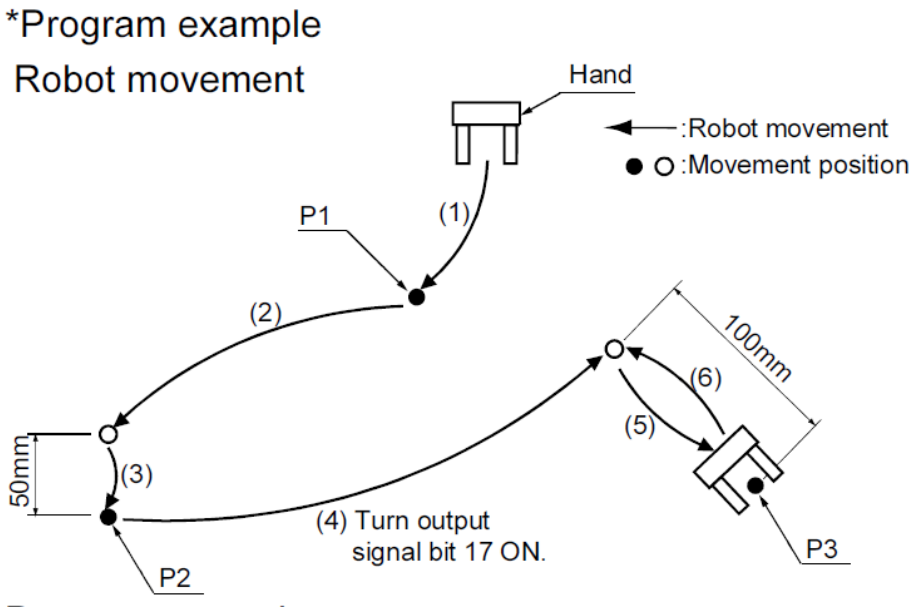
3.3.1 PTP

Tento pohyb bude sloužit k delším převozům z místa na místo jako je například pohyb z home pozice na pozici před překladištěm či skladištěm, budeme u něj měnit hodnotu rychlosti, která se bude pohybovat v rozmezí 50-100 %. Tento pohyb se v programování zaznačuje soupisem: Mov.

Příklad programu:

1. Mov P1 Pohyb na pozici P1
2. Mov P2, -50 Pohyb na pozici P2 s umístěním -50 mm v ose Z pro Tool
3. Mov P2 Pohyb na pozici P2
4. Mov P3, -100 Pohyb na pozici P3 s umístěním -100 mm v ose Z pro Tool
5. Mov P3 Pohyb na pozici P3
6. Mov P3, -100 Pohyb na pozici P3 s umístěním -100 mm v ose Z pro Tool

7. End Konec programu



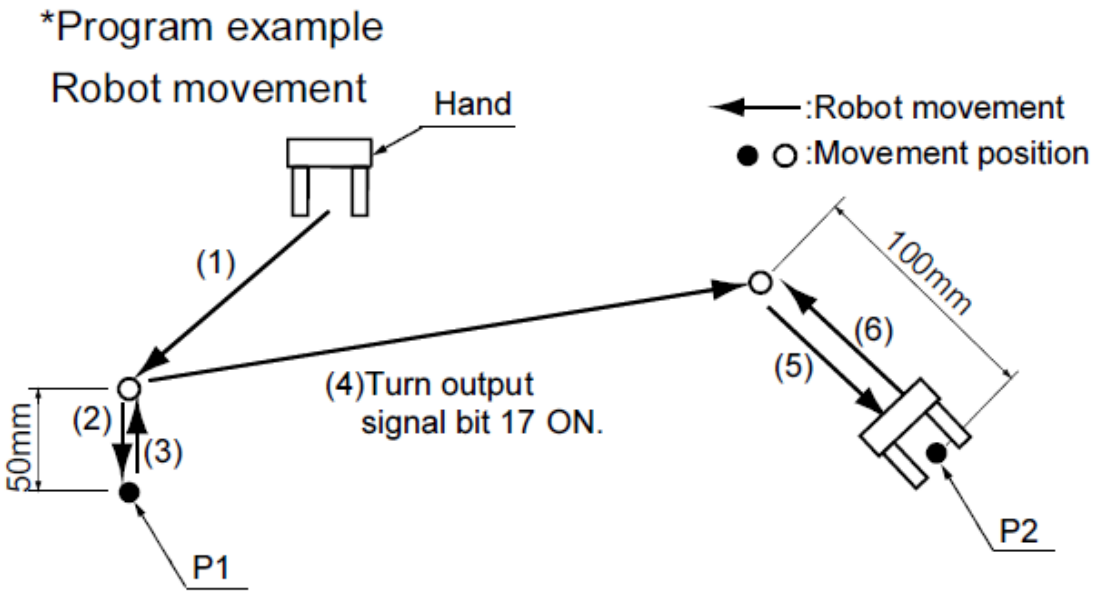
Obr.3.3.1: PTP pohyb

3.3.2 **Lineární**

Tento druh pohybu bude sloužit k převozu kostek na krátkou vzdálenost, kdy bude potřeba dosáhnout lineárního pohybu. Příkladem je vyzvednutí kostek ze skladiště a přepravu kostky do pozice před skladiště kde se pohyb změní zpět na PTP. Rychlost tohoto pohybu bude dle potřeby mezi 10-50 %. Tento pohyb se v programování zaznačuje soupisem: Mvs.

Příklad programu:

1. Mvs P1, -50 Lineární pohyb na pozici P1 s umístěním -50 v ose Z pro Tool
2. Mvs P1 Lineární pohyb na pozici P1
3. Mvs ,-50 Lineární pohyb na pozici -50 v ose Z pro Tool od aktuální pozice
4. Mvs P2, -100 Lineární pohyb na pozici P2 -100 v ose Z pro Tool
5. Mvs P2 Lineární pohyb na pozici P2
6. Mvs , -100 Lineární pohyb na pozici -100 v ose Z pro Tool od aktuální pozice
7. End



Obr.3.3.2: Lineární pohyb

3.3.3 **Cylindrický**

Tento druh pohybu není moc využívaný, je pro něj zapotřebí více místa díky poloměru kružnice, který tento pohyb opisuje. Při jeho využití se s množstvím potřebného prostoru počítá a využívá se při pohybu jen jednoho jointového kloubu. Při aplikaci tohoto typu pohybu je zapotřebí 3 bodů v prostoru, které se následně za command napíší. V Melfa V existují hned 4 typy tohoto pohybu a projeví se v nich jakým způsobem se přes tyto tři body přejede.

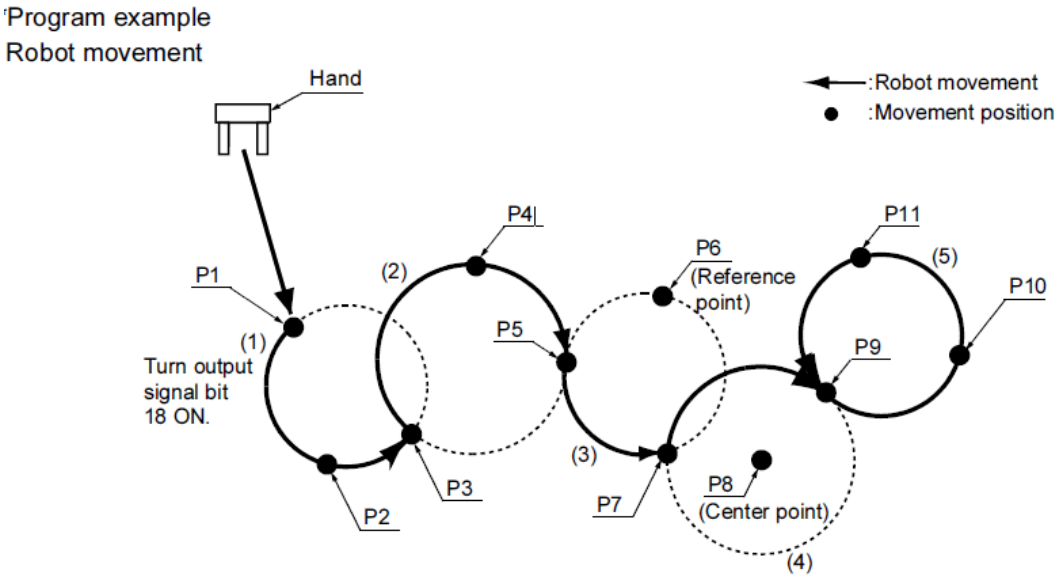
Varianty příkazu:

- Mvr – Určujeme počáteční bod, bod průjezdu a koncový bod. Robotické rameno projede přes všechny tyto body
- Mvr2 – Určujeme zde bod počáteční referenční a koncový. Robotické rameno projede pouze přes počáteční a koncový bod přičemž neprojíždí přes referenční bod.
- Mvr3 – Určujeme bod počáteční, koncový a následně středový který slouží jako střed kružnice po které se bude pohybovat
- Mvc – Určujeme počáteční bod (koncový), bod průjezdu 1 a bod průjezdu 2. Rameno najede na počáteční bod, následně na bod průjezdu 1, bod průjezdu 2 a dokončí pohyb na bod koncový.

Příklad programu:

1. Mvr P1, P2, P3 Wth M_Out(18) = 1 Cylindrický pohyb 1 mezi P1 a P3
2. Mvr P3, P4, P5 Cylindrický pohyb 1 mezi P3 a P5
3. Mvr2 P5, P7, P6 Cylindrický pohyb 2 mezi P5 a P7
4. Mvr3 P7, P9, P8 Cylindrický pohyb 3 mezi P7 a P9

5. Mvc P9, P10, P11 Cylindrický pohyb 4 mezi P9 a P9
6. End



Obr.3.3.3: Cylindrický pohyb

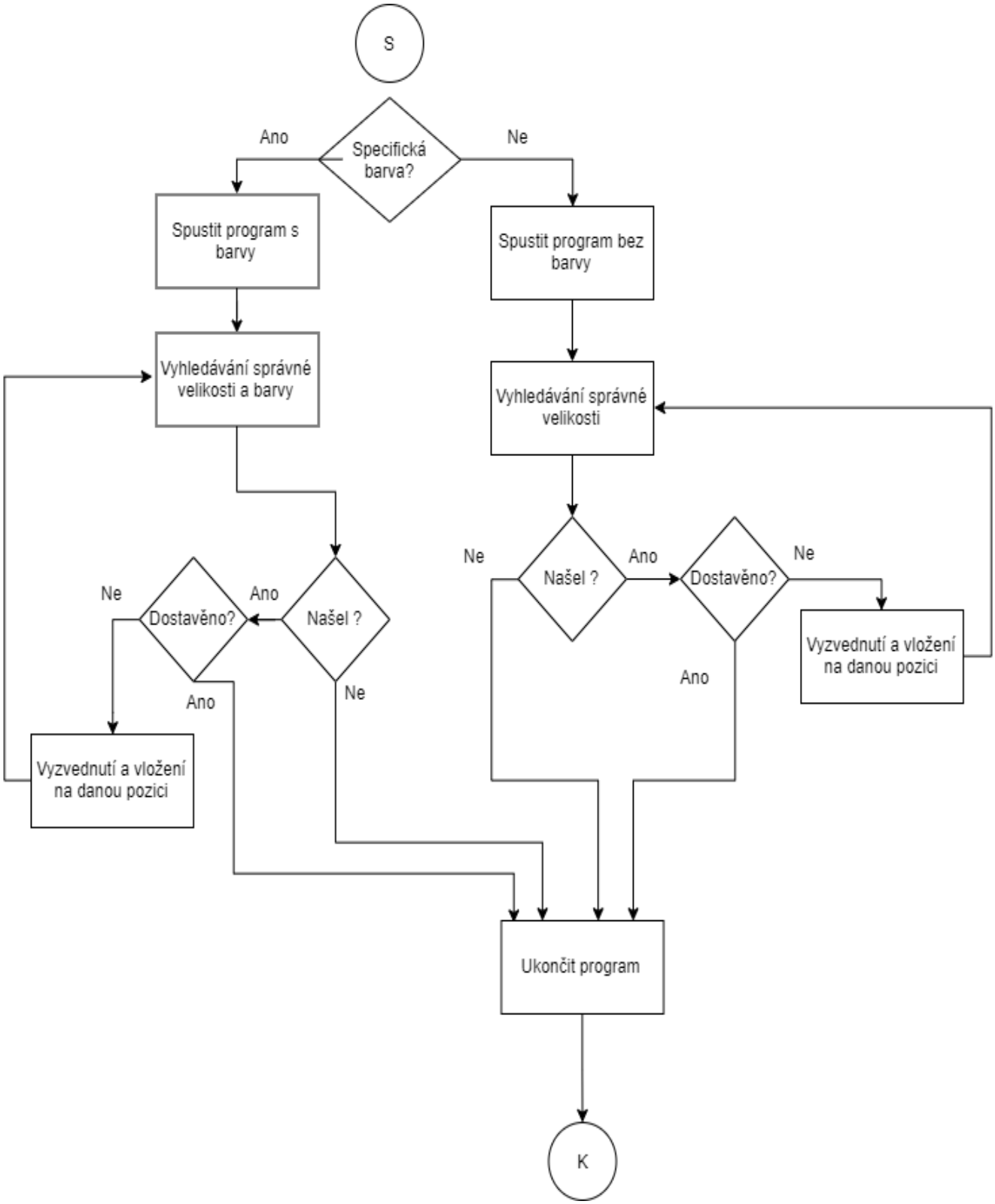
3.4 Využití programů

Využití předpřipravených programů by bylo možné jako:

- Výše uvedené ukázkové prostředí.
- Po dalších domluvách by bylo možné některé úlohy přepracovat k využití jako učební prostředky k výuce robotiky.

Tyto programy jsou snadno přístupné v programovacím prostředí RT Toolboxu 3 nebo výukovém ovladači kde se zobrazují aktuální nahrané programy v robotu a které je zde možné spustit. Jejich pozměnění by bylo snadné aspoň se základními programovacími schopnostmi a základnímu porozumění prostředí jako je kde dané programy najít, jak je uložit nebo nahrát do robota.

Za pomoci kamery a zpracování jejího obrazu bychom z programů, které mají zadané přesné souřadnice kostek potřebných na stavbu, mohli vytvořit programy, které by prohledávaly tyto souřadnice, ale nevěděli by co na daných pozicích je, při zpracování obrazu by zjišťoval požadavky vyplývající ze zadání jako je barva kostky a její velikost.

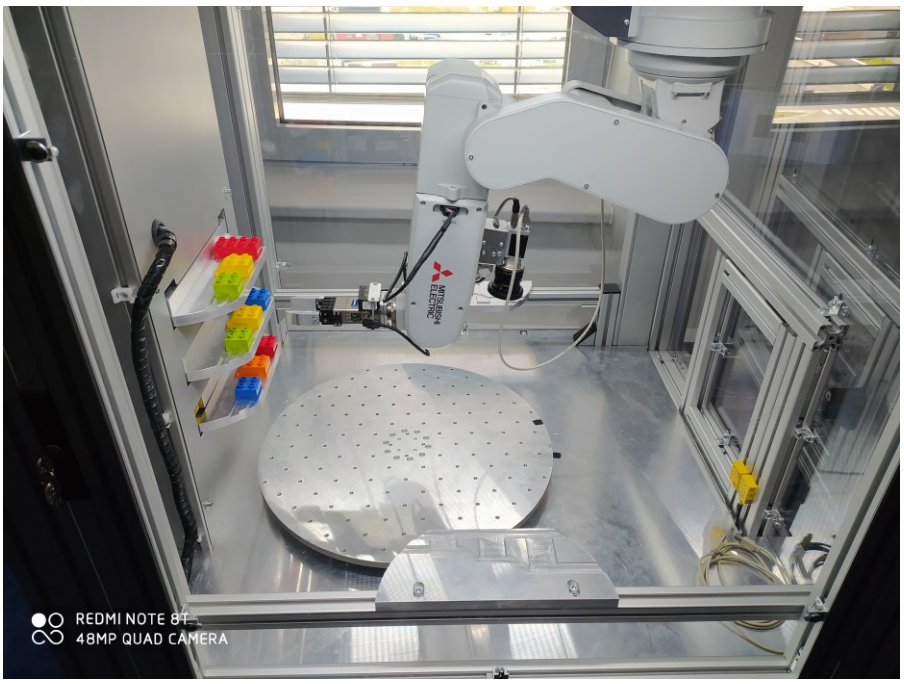


Obr.3.4:Blokové schéma funkce s kamerou

4 Testování funkčnosti robotického pracoviště.

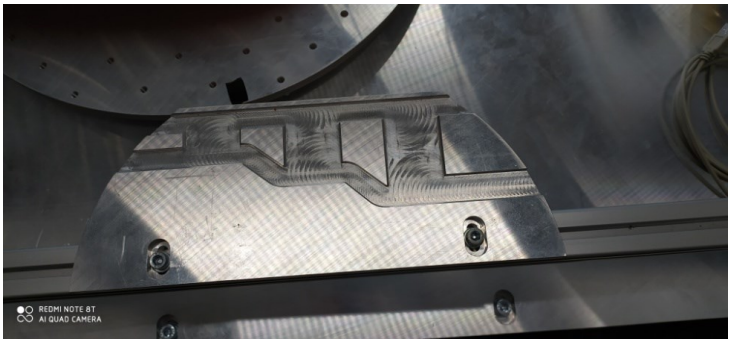
4.1 Dosažitelnost reálného prostředí

Při testování za pomoci výukového ovladače a virtuálního prostředí jsem se pokoušel prozkoumat prostory reálného prostředí našeho boxu ve kterém je robotické rameno umístěno.



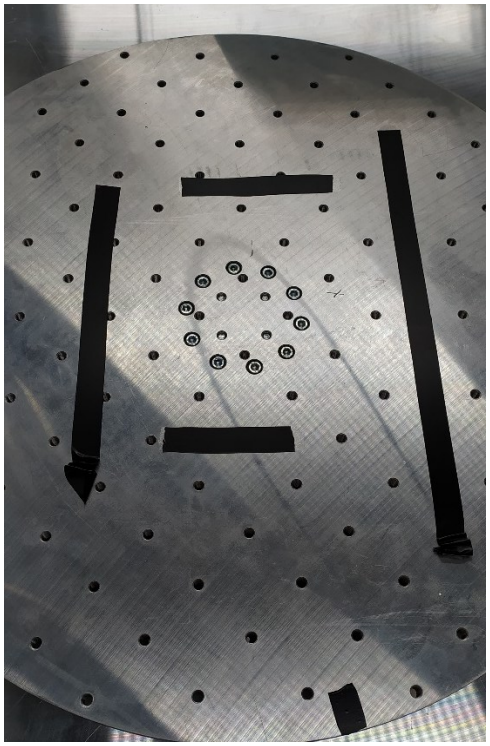
Obr.4.1: Box, karusel, překladiště, zásobník a robot

V tomto prostředí se nachází několik zásobníků a 2 odkládací panely, každý na jedné straně boxu, u těchto odkládacích ploch jsem se snažil dosáhnout pozic před odkládací plochou, nad ní a těsně nad ní z důvodu vyzvednutí kostky. První dvě pozice jsou dosažitelné a po chvilce manévrování se dají přesně zjistit souřadnice a otestovat jejich dostupnost z námi zvolených bodů. Problém je ovšem se třetí pozicí, do které se nedá dostat za pomoci lineárního krátkého pohyb, ten je ideální a žádoucí pro rychlý průběh programu. Snahou bude, aby se robotické rameno do potřebných pozic dostalo, může se však stát, že do této pozice se půjde dostat pouze úplným přetočením různých kloubů robota, a to by nebyl efektivní proces pro odkládání a překládání.



Obr.4.1: Překladiště

Dále jsme zjistili že na překladišti je velmi omezený prostor kde se dá s robotem dostat až na těsný dotek s karuselem, to nám omezuje možnosti velikosti a tvaru stavby. Ten jsme postupným posunem robotického ramene zjistili a označili černou páskou abychom si zjednodušili orientaci na karuselu pro následné programování pozic.

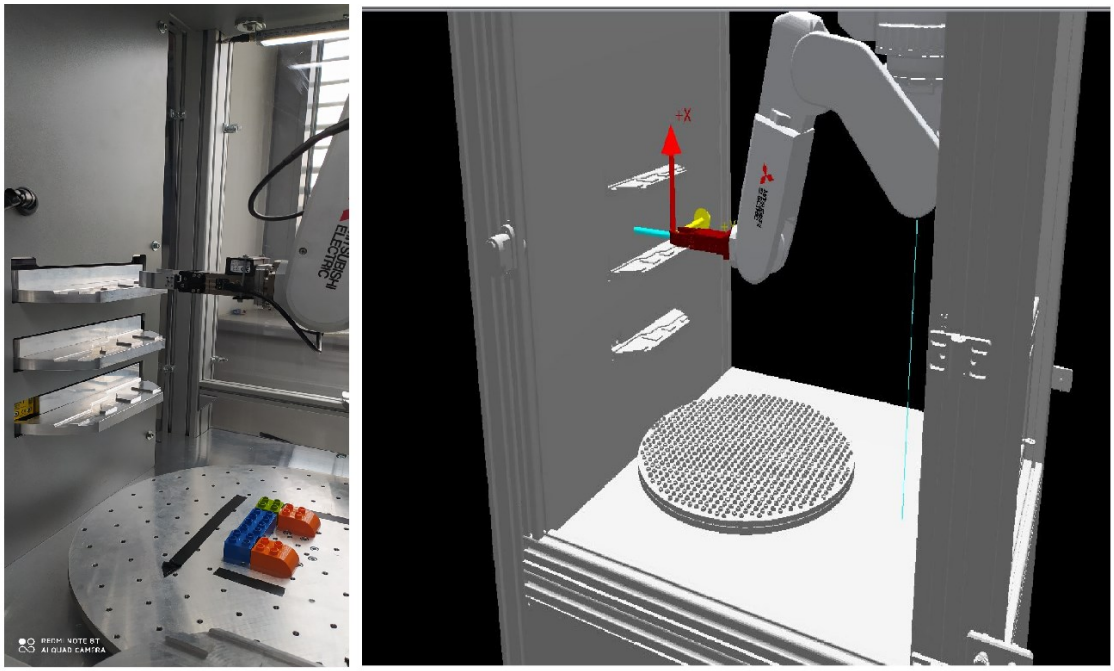


Obr.4.1: Prostor karuselu

4.2 Programování v online módu

Důvodem, který mně vedl k programování v online módu je, že vytvořený 3D model prostředí robotického ramene a boxu, v němž je robot umístěn, neodpovídá velikostí, umístěním zásobníků a vzdálenostmi skutečnému prostředí viz Obr.4.2 ve kterém se odkazuji na rozdíl fyzického a simulačního prostředí. Mezi základní rozdíly patří

odlišné pozice pro 2 a 3 patro zásobníku, dále také neodpovídají pozice jednotlivých velikostí kostek, které se zde nacházejí.

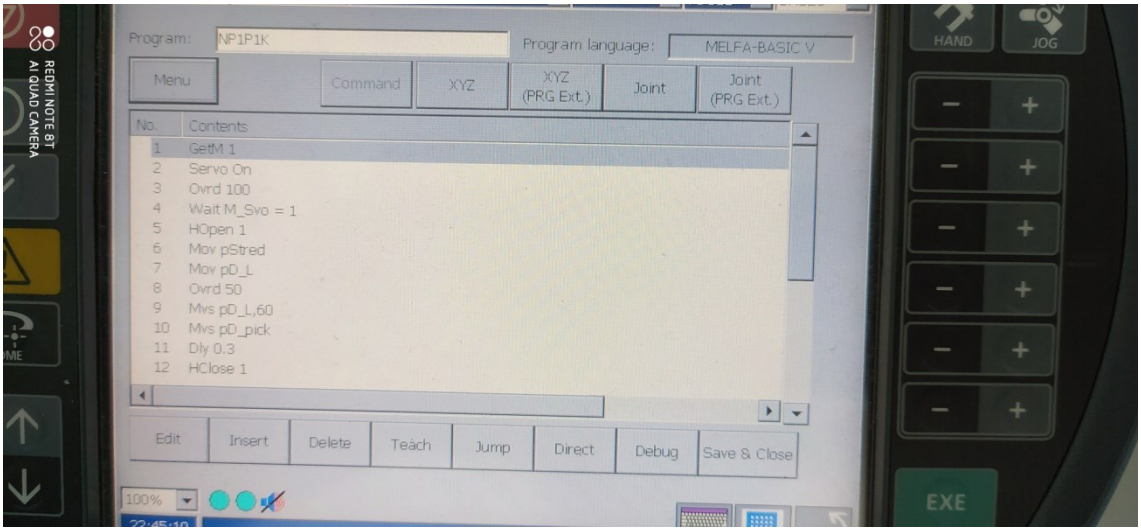


Obr.4.2: Zobrazení reálného prostředí 3 patra v simulaci

Nutností je tedy nyní hlídat každou změnu parametrů pohybu, aby nenastala kolize, popřípadě mít připravenou reakci na zmáčknutí kontrolního stop tlačítka. Zároveň po každém samostatném programu, který je sepsán je následně okamžitě odzkoušen a může být při nastalých potížích upraven a jsou řešeny chyby. I samostatný robot se chrání proti kolizím za pomoci force sensorů a je okamžitě zastaven, pokud překoná hodnotu zatížení, na kterou je dimenzován. V našem případě je robot dimenzován na tři kilogramy.

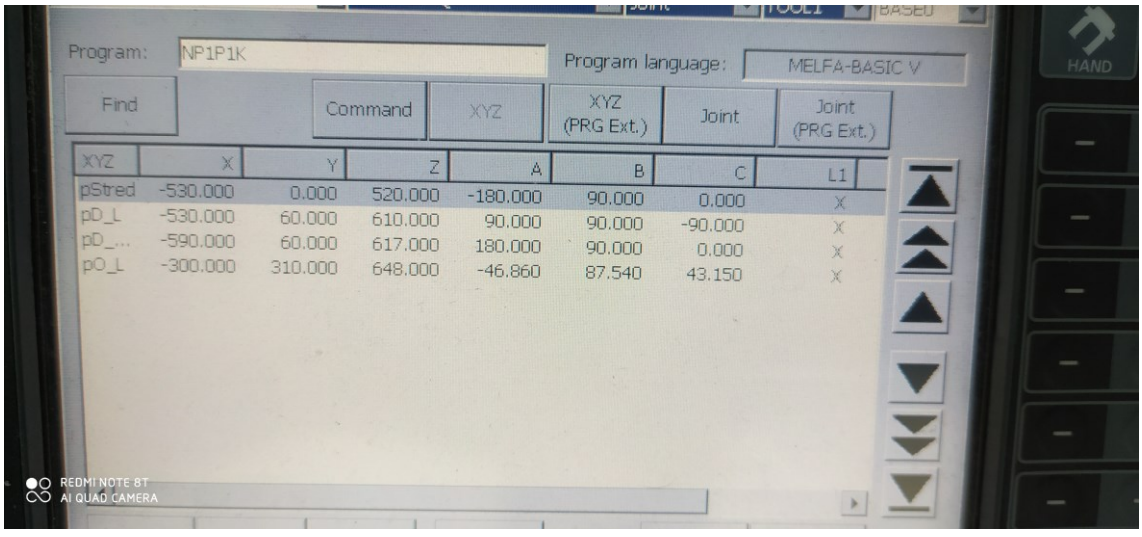
4.3 Obsah samostatného programu

Program nám obsahuje odblokování servo pohonů příkazem „*Servo On*“, dále se zde může nacházet hodnota na změnu rychlosti pohybu robota „*Ovrd*“, ta je určena v procentech a je možné ji v programu použít kolikrát je potřeba. Pokud není použita tak se bere její základní hodnota, která je 100. Dále se v programu mohou nacházet zpoždění, označovány jsou „*Dly*“, hodnota je zadávána v sekundách, hodnota jako je půl sekundy se zapisuje s desetinou tečkou. Program je vždy zakončen příkazem „*End*“, který nám ukončí celý program a vrátí se na jeho začátek, aby jej začal provádět znovu.



Obr.4.3: Program zobrazený ve výukovém ovladači

Vlastní programy mám tvořeny z jednoduchých pozic, zároveň jsem se snažil o redukci co nejvíce adres pozic za možnosti úprav adres v kódu. Jednou z těchto úprav je zadání hodnoty po zapsání jména adresy, tato hodnota je následný posun koncového efektoru v Z-ose Tool soustavě. Rozdíl je oproti world soustavě v tom že není neměnná. Soustava se vztahuje pouze na nastavení úhlu Tool, díky tomuto může osa Z u Tool působit i ve směru X v soustavě world. Tento rozdíl si musíme uvědomit při používání této featury v programu.



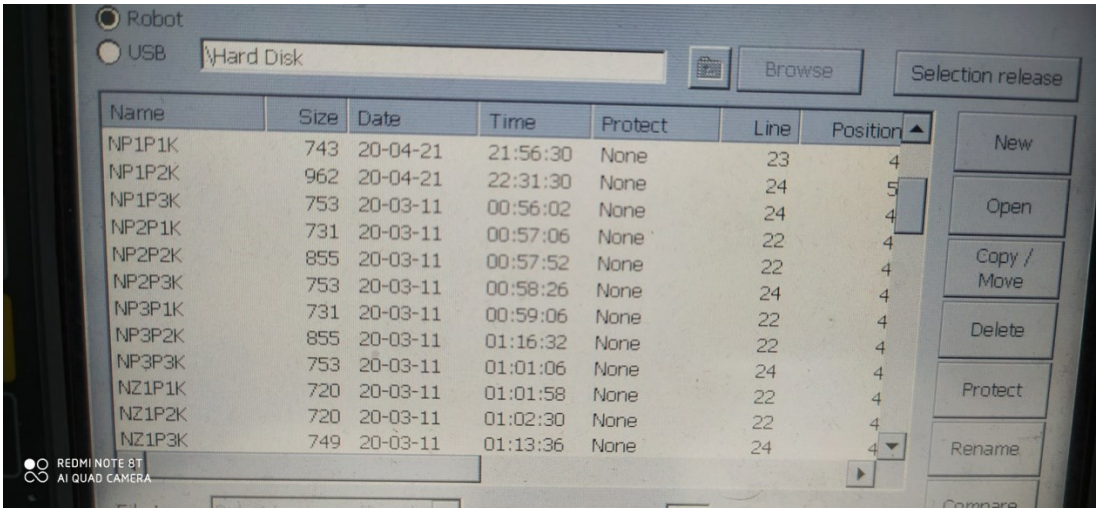
Obr.4.3: Adresy programu NP1P1K

4.4 Pojmenovávání a využití programů

Programy jsou pojmenovány podle funkce, adresy kostky v prostoru a jejím typu. Například NZ3P3K: NZ - Na zásobník, 3K – kostka číslo tři (číslovány zleva do prava), 3P – třetí patro (patra jsou číslována od spodního k hornímu)

Na zásobník	Na překladiště
NZ1P1K	NP1P1K
NZ1P2K	NP1P2K
NZ1P3K	NP1P3K
NZ2P1K	NP2P1K
NZ2P2K	NP2P2K
NZ2P3K	NP2P3K
NZ3P1K	NP3P1K
NZ3P2K	NP3P2K
NZ3P3K	NP3P3K

Tab.4.4: Tabulka programů pro zásobník a překladiště



Obr.4.4: BoxSeznam programů ve výukovém ovladači

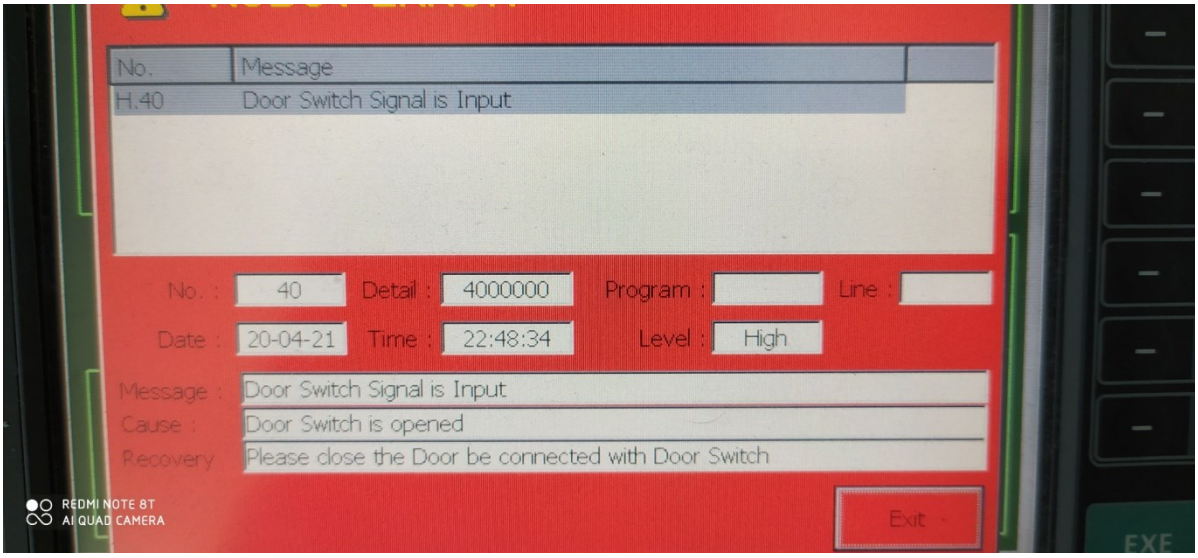
Programy označené jako NZ berou kostky z odkladače podle toho, jaký typ kostky to je, pokud je to tedy například program NZ3K2P, vezme nám kostku 3 která je největší. Překládací plocha se totiž skládá jen z tří pozic, na které je možno kostku odložit. Stejný princip platí pro programy, které je pokládají na překladiště. Zde se však specifikuje odkud ze zásobníku se má kostka odebrat a vkládaná pozice je nastavena dle velikosti kostky. Aby se nestalo, že budeme na 1 pozici odkládat více kostek, tak si v každém programu pošleme bitovou informaci na output dle našeho výběru. V každém programu, který odkládá na určitou pozici, zároveň budeme kontrolovat, zda již na této pozici nějaká kostka odložená není. Tudíž na začátku programu dáme podmínku, že pokud se bitu, který odpovídá naší pozici, nachází 0, pak může program pokračovat. Pokud se na pozičním bitu nachází 1, program se zastaví a nastane chyba či varování.

4.5 Testování chybových stavů

Sám robot se brání stavům, kdy nastávají chyby nebo situace, které mohou způsobit zranění obsluhy, požkození robota nebo odkázání na adresu, již není robot schopný dosáhnout.

Ochrana obsluhy

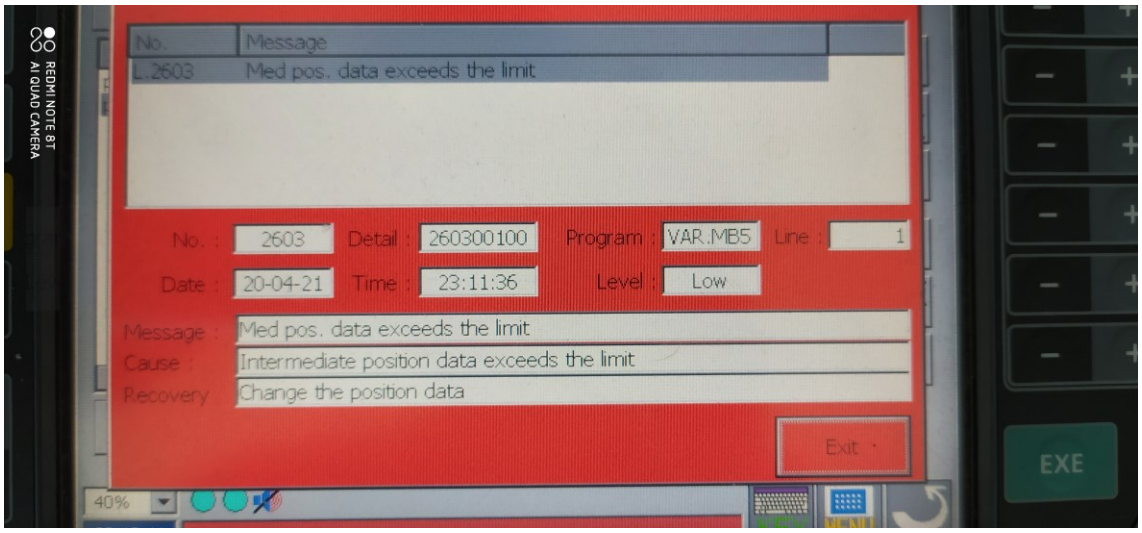
Situace s ochranou obsluhy a vyskočeným alarmem jsme testovali jednoduchým otevřením dveří s magnetickým zámkem během provozu jednoho z programů. Nastal nám chybový režim, který nás odkázal do errorového stavu. Robot se okamžitě zastavil, přerušil se program a vypla se všechna serva. Po zavření a zacvaknutí dveří jsme mohli všechny error stavy potvrdit a resetovat. Po dokončení resetu jsme program mohli dále zpustit a pokračovat od poslední uložené pozice dále v programu.



Obr.4.5:Chybová hláška otevřených dveří

Nedosažitelnost adresy

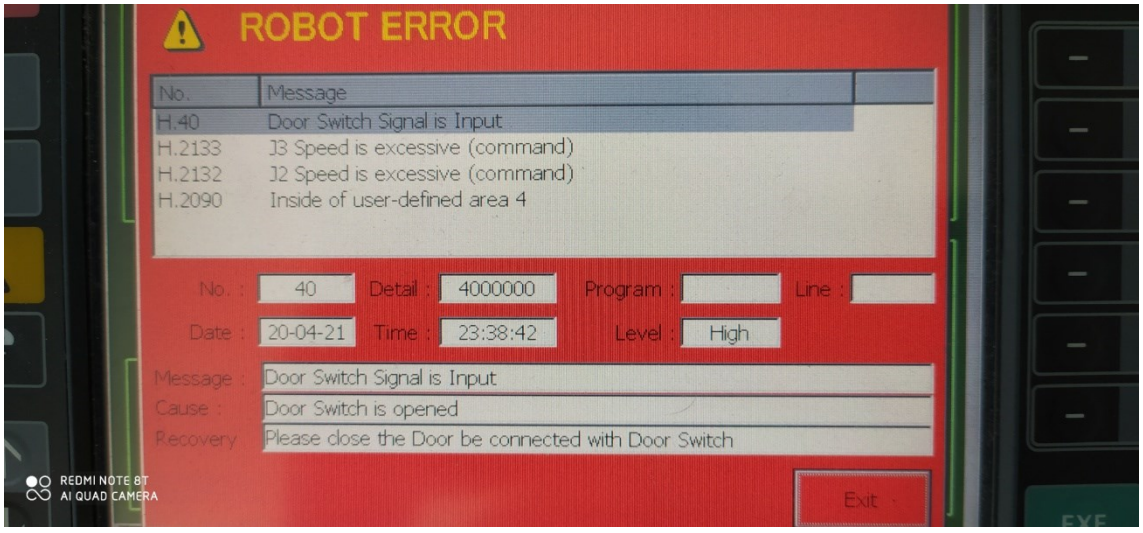
V tomto stavu je operátorem zadána adresa, které není možné současným polohováním kloubů dosáhnout. Robot se zastaví na nejkrajnější hranici svého dosahu pomocí kloubů a odkáže nás na chybovou hlášku s oznámení, že daná adresa je mimo dosažitelnost kloubového otočení. V tomto případě je postup podobný ochranně obsluhy, stačí nám zkontrolovat otevřené error okno a resetovat jej, pokud nám nějaké errory zůstaly, pak je nutné je vyřešit, pokud jsme po resetování bez jakékoliv error hlášky, můžeme následně pokračovat v pohybech s robotickým ramenem.



Obr.4.5: Chyba adresování mimo dosah

Přetížení force senzoru

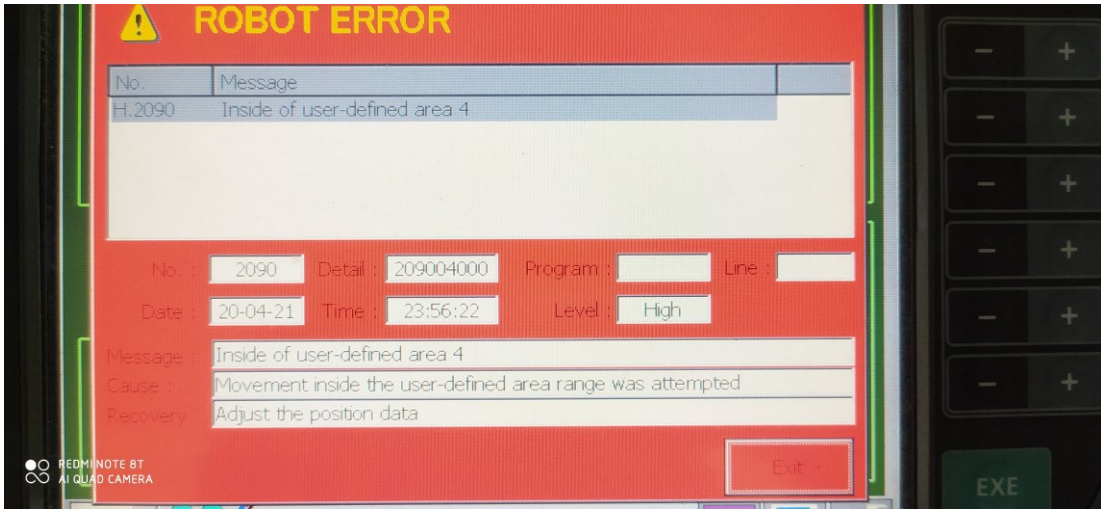
Náš robot má nosnost do 3 kg, pokud tedy na něj během pohybu začne působit síla větší než únosná, vypíše nám chybovou hlášku o přetížení servo motoru. Tento stav se nám podařilo simulovat díky programu, ve kterém nebyla ošetřena odstupná výška při vyzvedávání kostky z překladiště, kostka začala působit na lůžko s výstupkem na překladišti, při tomto manévru se vzpříčila kostka a robotické rameno se stále snažilo dostat na další adresu v programu a přetížilo se. Díky tomuto chybovému hlášení jsme mohli implementovat do programů odstupnou adresu díky které se vyhneme dalšímu kontaktu s překladištěm. Po odstranění problému z programu jsme vyresetovali robota a při dalším spuštění nám již tento problém nenastal.



Obr.4.5: Chybová hláška přetížení force sensoru

Pozice v User-defined area

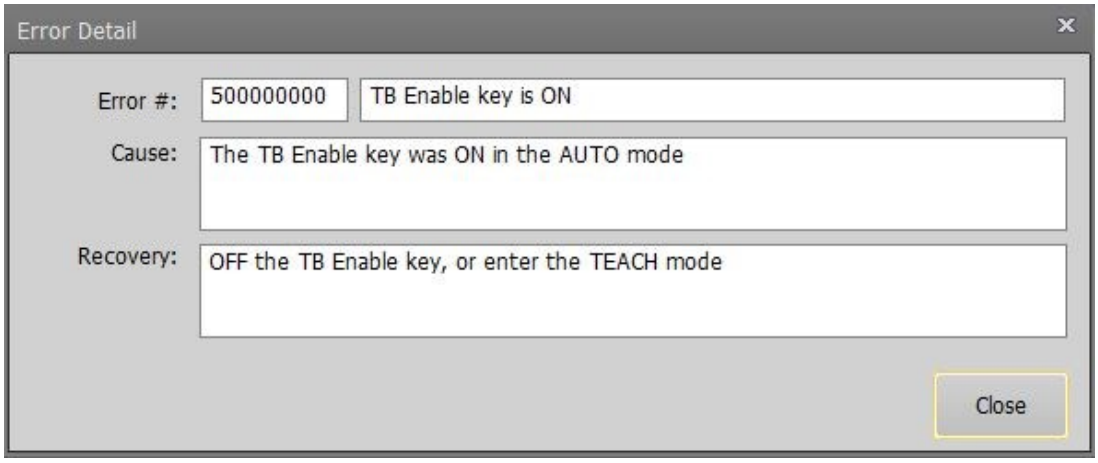
Tato chybová hláška nastává, pokud se robotické rameno dostane do oblasti, která je definována jako User-defined area, tyto oblasti jsou vytvořeny jako ochrana prvků, které se zde mohou nacházet, nebo na definování oblasti překážky. Pokud by byla cesta nastavena před a za touto oblastí a robot by při výpočtu dráhy narazil na pozici která se zde nachází tak nám proces přepočítá, aby se této oblasti vyhnul.



Obr.4.5: Chybová hláška pozice robota v User-defined area

Programování v automatic se zapnutým Teach tlačítkem

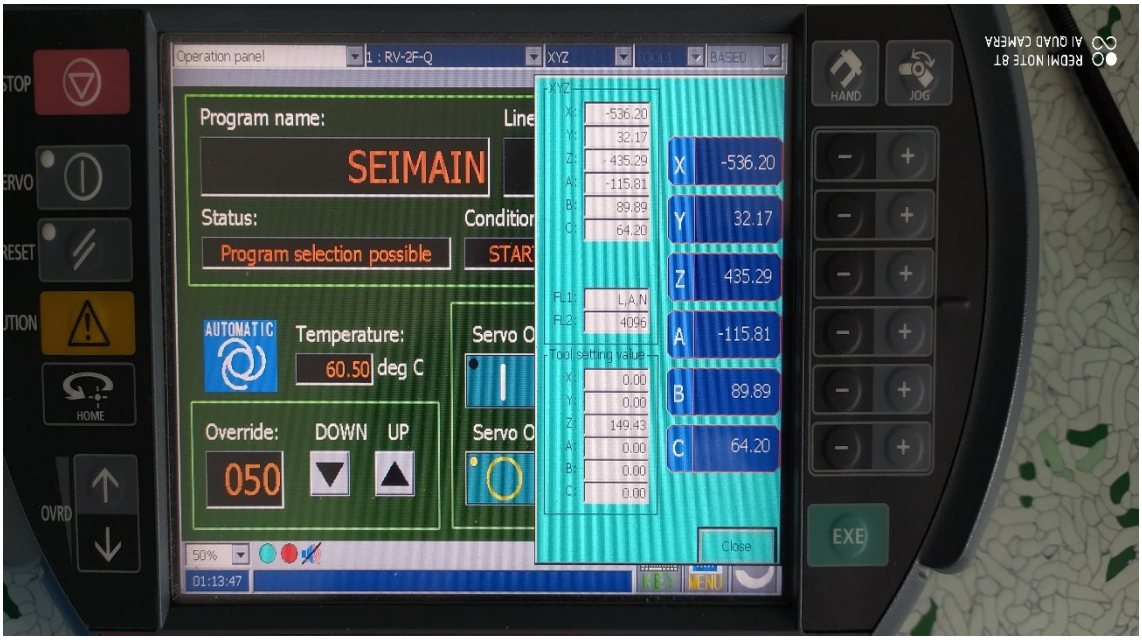
Tato chyba je způsobena hlavně nepozorností při přepínání ovládání mezi manual a automatic, není nijak těžké ji překonat a odstranit.



Obr.4.5: Chybová hláška zapnutého Teach tlačítka

4.6 Práce s výukovým ovladačem

Během práce, kdy jsme sestavovali programy, bylo vhodné využít výukový ovladač na pohybové operace a nastavování přibližných adres pozic. Průběh byl takový, že při sestavování programu jsme zapli v RT Toolbox vytvoření nového programu, ten jsme pojmenovali a následně jsme přepli robota do manuálního módu abychom jej mohli ovládat přes výukový ovladač. Ten jsme přepli na operační panel a zapli servopohony společně s držením deadman tlačítka. Následně jsme mohli zapnout Jog ovládání a přepli jsme na adresování za pomocí souřadnic XYZ, díky tomuto se robot pohyboval lineárně, mohli jsme najet do požadované pozice s robotickým ramenem, hodnotu v softwaru RT Toolbox zapsat do programu a následně pojmenovat podle potřeby. Teach pendantu jsme hlavně využívali z důvodu pomalejší odezvy robota na změnu dat z operačního panelu robota v programu, také se stávalo že tento operační panel „spadnul“ během změny adresy v programu a díky tomuto zjištění se také začalo ovladače využívat více. Přepnutí mezi manual a automatic je přece jen méně náročné a efektivnější než po každém „spadnutí“ operačního panelu v programu jej znova otevírat.



Obr.4.6: Operační panel výukového ovladače

5 Testování funkčnosti hlavního programu

Během testování funkčnosti hlavního programu bylo zapotřebí ověřit i funkčnosti předpřipravených programů, aby nebylo zapotřebí větší délky kódu a všechny předpřipravené programy mohli být znovu použity pro jiný typ úlohy nebo jako materiál k výuce s tímto robotickým pracovištěm. Hlavní program se skládá ze 2 hlavních typů podprogramů kdy každý je dále dělen mezi přenesení kostky na dané stanoviště, anebo přepravu kostky z tohoto stanoviště zpět na původní pozici. Mezi takto připravené programy patří 2 které zajišťují přepravu kostek mezi zásobníkem a překladištěm a následně další 2 které slouží na přepravu kostek mezi překladištěm a pozicí nad karuselem.

5.1 Mezi zásobníkem a překladištěm

Přeprava mezi zásobníkem a překladištěm je na zásobení překladiště kostkami a možnosti jejich přechycení, jelikož při vyzvedávání kostek je úchopová hlavice ve vodorovné poloze s osou X, v takovéto pozici není vhodné ukládat kostku na karusel, jelikož by se velmi snadno mohlo stát, že zde hlavice narazí do další, již položené, kostky. Proto využíváme překladiště, abychom mohli kostku uchytit pro nás výhodnějšího úhlu jenž podpoří snadné umístění kostky na karusel bez většího omezení. Programy jsou sestaveny tak aby každý jednotlivě odpovídal dané pozici na zásobníku, tudíž je každý z programů označen zkratkami, kam je kostka převážena, z jaké/na jakou pozici a typ kostky. Tudíž máme dva typy pohybu, NP (Na překladiště) a NZ (Na zásobník). Funkčnost těchto typů programů byla odzkoušena samostatně a následně i v hlavním programu pro zjištění návaznosti na další programy.

– NP: Při tomto typu programu se kostka přenáší ze zásobníku na překladiště, kdy na překladišti jsou definovány tři pozice pro jednotlivý tvar přenášených kostek. V úvodu programu je tedy napsáno, z jakého patra je kostka odebrána a jaký typ kostky to je. NP1P1K odpovídá programu jenž přenese kostku z prvního patra na první pozici a přenese ji na pozici v překladišti jenž obsahuje kostky této velikosti.

```
GetM 1
Servo On
Ovrd 100          'Rychlost pohybu nastavena na 100%'
Wait M_Svo = 1
HOpen 1           'Otevření úchopné hlavice'
Mov pStred        'Pohyb na základní pozici'
Mov pD_L
Ovrd 50
Mvs pD_L,60
Mvs pD_pick       'Pozice odebrání kostky ze zásobníku'
Dly 0.3
HClose 1          'Uzavření úchopné hlavice'
Dly 0.3
Mvs pD_L,60
Ovrd M_NOvrd      'Zadání rychlosti pohybu na normálové které je ze základu 100%'
Mvs pD_L
Mvs pStred
Mov pO_nL
Mvs pO_L          'Pozice odkladu kostky'
Dly 0.3
HOpen 1           'Otevření úchopné hlavice'
Dly 0.3
Mvs pO_nL
Mov pStred        'Pohyb na základní pozici'
End
```

Obr.5.1: Program NP1P1K

- NZ: Zde se program provádí obráceně oproti předchozímu typu, je zde zadáno, kam má být kostka převezena a podle koncové adresy se určuje odkud z překladiště se má kostka odebrat.
  NZ1P1K odpovídá programu který odebere kostku z překladiště jenž odpovídá velikosti pozice 1P1K a přenese ji na zásobník do zadané pozice.

```
1  GetM 1
2  Servo On
3  Ovrd 100
4  HOpen 1
5  Dly 0.3
6  Mov pStred          'Pohyb na základní pozici'
7  Mov pO_L
8  Ovrd 50
9  Mvs pO_L_Pick       'Pozice odebrání kostky z překladiště'
10 Dly 0.3
11 HClose 1            'Uzavrení úchopné hlavice'
12 Dly 0.3
13 Mvs pO_L
14 Ovrd 100
15 Mov pStred
16 Mvs pD_L
17 Mvs pD_L,60         'Pozice odkladu kostky na zásobníku'
18 Dly 0.3
19 HOpen 1             'Otevrení úchopné hlavice'
20 Dly 0.3
21 Mvs pD_L
22 Mvs pStred          'Pohyb na základní pozici'
23 End
```

Obr.5.1: Program NP1P1K

5.2 Mezi překladištěm a karuselem

Během této přepravy je nutné nastavit změnu strany pro odebrání kostky kvůli odkládání na karusel kdy rameno nastavíme do pozice vodorovné s osou Z nad kostkou, u tohoto pohybu se nám projevila chyba při maximální rychlosti, která nám kvůli přetížení zastavila program, proto jsme následně tento pohyb zpomalili na čtvrtinovou rychlost. Následně se nám vyzvedne kostka a přenese se na definovanou pozici nad karuselem kterou používáme i jako počáteční pozici v hlavním programu, tudíž si při sestavě jiného programu stačí tuto pozici zadat jako počáteční díky tomu že již obsahuje uchopenou kostku dle našeho výběru. Na této pozici je kostka stabilizována a v dostatečné výšce, aby neohrozila případnou vyšší stavbu která se zde nachází nebo má být teprve sestavena.

```
GetM 1
Servo On
Wait M_Svo = 1
Ovrd 100
Dly 0.3
HOpen 1          'Otevření úchopné hlavice'
Dly 0.3
Mov pStred
Mov pO_L
Ovrd 25
Mvs pO_nL        'Přetočení do vodorovnosti s osou Z'
Ovrd 30
Mvs pO_nL,30
Dly 0.2
HClose 1         'Sevření úchopné hlavice'
Dly 0.2
Mvs pO_nL
Ovrd 50
Mov pK           'Koncová pozice'
End
```

Obr.5.1: Program NP1P1K

5.3 Hlavní program

Hlavní program je sestaven z předem připravených programů, které nám požadované velikosti kostek přenesou až na souřadnici nacházející se nad karuselem, odsud již musíme sestavit program sami. To znamenalo zjistit nejlepší mechanické umístění kostek pro náš tvar stavby a následné adresování, při tomto jsme hlavně využívali výukového ovladače, následně jsme díky nastavených pozic kostky postupně odkládali na přiřazené pozice, které byly podle typu a umístění pojmenovány. Příklad adresy je pMP kdy toto označení odpovídá malé kostce umístěné vpravo.

```
GetM 1
Servo On
Wait M_Svo = 1
'Preprava kostek na prekladište '
CallP "NP1P1K"
CallP "NP1P2K"
CallP "NP1P3K"
'Uložení velké kostky'
CallP "NK3K"
Mov pKP
Mov pV
Ovrd 20
Mvs pV,35
Dly 0.2
HOpen 1
Dly 0.2
Mvs pV
Ovrd 100
Mov pKP
'Ukládání kostky malá levá'
CallP "NK1K"
Mov pKL
Mov pML
Ovrd 20
Mvs pML,35
Dly 0.2
HOpen 1
```

Obr.5.3: Program SEIMAIN

6 Závěr

Funkce robotického ramene odpovídá předpokládané funkčnosti, která byla ze začátku zamýšlena. Při programování jsme se setkali s chybami, které bylo potřeba uvést a nadále se přizpůsobit jejích vlastnostem a vyhýbat se opětovnému opakování těchto chyb. Zároveň bylo zjištěno že Ovrd který slouží k určování rychlosti pohybu v programu je navázán na Ovrd nacházející se ve výukovém ovladači, tím pádem vzniká rozdíl rychlosti programu pokud bude na výukovém ovladači nastavena rychlost 10% nebo 100%. Ideální zvolená rychlost byla 50% na výukovém ovladači po potřebném testování kdy 100% výkon lehce pohyboval s celou sestavou.

Sestavená aplikace slouží jako ukázka využití předpřipravených programů a jejich zápis, tudíž jakýkoliv následující program bude moct být sepsán s využitím těchto předpřipravených programů s ohleduplností na čas který by byl potřebný k znovuzapisování potřebných pozic. Zároveň může sloužit jako základ pro učební materiály v budoucnosti s obsahem a popisem základních úkonů.

Propojení a ovládání pracoviště je možné pomocí RT Toolboxu a připojením na stejnou síť, nastal problém, kdy nebylo možné připojit se na Online mód, a to bylo způsobeno špatnou IPV 4 adresou na které měl počítač se stanovištěm komunikovat, následně byl problém odstraněn a komunikace byla opět možná.

Použitá literatura

[1] Technické informace o RV-2F-Q. *Mitsubishi: Mitsubishi industrial robot* [online]. [cit. 2020-05-03]. Dostupné z: http://www.int76.ru/upload/iblock/645/645fed8bc1d630a32fce5c803d662ecd.pdf

[2] RT ToolBox3: Uživatelský manuál. *Mitsubishi electric: Mitsubishi electric industrial robots* [online]. [cit. 2020-05-03]. Dostupné z: https://lbsbr2.gerionline.at/images/pdf_dateien/RT3Manual.pdf

[3] RT ToolBox3: Melfa Basic. *Mitsubishi electric* [online]. [cit. 2020-05-03]. Dostupné z: https://www.mitsubishielectric.com/fa/products/rbt/robot/smerit/rt2/function.html#pageUnit02

[4] RT ToolBox3: Instrukční manuál. *Mitsubishi electric: Mitsubishi Industrial Robot* [online]. [cit. 2020-05-03]. Dostupné z: https://www.allied-automation.com/wp-content/uploads/2015/02/MITSUBISHI_CR750CR751-Controller-Instruction-Manual-Detailed-Explanations-of-Functions-and-Operations.pdf

[5] BOUCHARD, Samuel. *Learn robotics: A Guide to Making Robots Work in Your Factory* [online]. [cit. 2020-05-10]. Dostupné z: https://iptech1.com/pdf/Leanroboticsbook-download.pdf

Seznam příloh

Video funkce robota