TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Editor testovacích tras pro systém Syslogeum 3000

Diplomová práce

Studijní program: N2612 – Elektrotechnika a informatika
Studijní obor: 3906T001 – Mechatronika

Autor práce: **Bc. Jiří Kulich**
Vedoucí práce: Ing. Leoš Beran Ph.D

Liberec 2020

TECHNICAL UNIVERSITY OF LIBEREC
Faculty of Mechatronics, Informatics
and Interdisciplinary Studies

SysLogeum 3000 Test Route Editor

Master thesis

Study programme: N2612 – Electrotechnology and informatics
Study branch: 3906T001 – Mechatronics

Author: **Bc. Jiří Kulich**
Supervisor: Ing. Leoš Beran Ph.D

Liberec 2020

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Zadání diplomové práce

Editor testovacích tras pro systém SysLogeum 3000

Jméno a příjmení: **Bc. Jiří Kulich**
Osobní číslo: M18000165
Studijní program: N2612 Elektrotechnika a informatika
Studijní obor: Mechatronika
Zadávající katedra: Ústav mechatroniky a technické informatiky
Akademický rok: **2019/2020**

Zásady pro vypracování:

1. Seznamte se s řešením skladového systému SysLogeum 3000 a řídicím firmwarem robotu.
2. Navrhněte vizuální podobu editoru testovacích tras.
3. Navrhněte jednoduchý jazyk pro textové příkazy.
4. Realizujte program, který bude interpretovat textové příkazy, které budou zadávány z navrženého editoru.
5. Napište návod na ovládání editoru pomocí vytvořené vizualizace.

Rozsah grafických prací: dle potřeby dokumentace
Rozsah pracovní zprávy: 40–50 stran
Forma zpracování práce: tištěná/elektronická
Jazyk práce: Čeština

Seznam odborné literatury:

[1] ŠMEJKAL, Ladislav a Marie MARTINÁSKOVÁ. PLC a automatizace. 1. díl. Praha: BEN, 1999. ISBN 80-8605- 58-9.
[2] JOHN, Kharl-Heinz; TIEGELKAMP, Michael. IEC 61131-3 Programming Industrial Automation Systems : Concepts and Programming Languages, Requirements for Programming Systems, Decision – Making Aids. 2nd.
[3] Download SinuTrain for SINUMERIK Operate V4.8 – Basic. Siemens [online]. [cit. 2018-10-09]. Dostupné z: https://www.industry.siemens.com/topics/global/en/cnc4you/cnc_downloads/sinutrain_downloads/Pages/download-sinutrain-for-sinumerik-operate-v48-bas
[4] REISING, D. V. (2013). Effective console operator HMI design: ASM consortium guidelines. Houston, TX, Abnormal Situation Management (ASM).

Vedoucí práce: Ing. Leoš Beran, Ph.D.
Ústav mechatroniky a technické informatiky

Datum zadání práce: 10. října 2019
Předpokládaný termín odevzdání: 18. května 2020

L.S.

prof. Ing. Zdeněk Plíva, Ph.D.
děkan

doc. Ing. Milan Kolář, CSc.
vedoucí ústavu

V Liberci dne 10. října 2019

Prohlášení

Byl jsem seznámen s tím, že na mou diplomovou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé diplomové práce pro vnitřní potřebu TUL.

Užiji-li diplomovou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Diplomovou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím mé diplomové práce a konzultantem.

Současně čestně prohlašuji, že tištěná verze práce se shoduje s elektronickou verzí, vloženou do IS STAG.

Datum:

Podpis:

Abstrakt

Tato práce je součástí vývoje robotického skladu Syslogeum 3000 společnosti Systematic a.s. Cílem práce je vytvořit software v řídícím PLC, který umožní snadné testování skladovacího robota. Za normálních okolností je robot řízen z nadřazeného systému. V určitých chvílích je však třeba robota ovládat nezávisle na tomto systému. Záměrem práce je umožnit servisnímu technikovi naprogramovat libovolnou trajektorii robota a tím testovat jeho funkčnost. Pro obsluhu softwaru byla vytvořena vizualizace, která obsahuje grafický a textový editor. Součástí práce je vytvoření jednoduchého programovacího jazyka, kterým jsou popisovány jednotlivé pohyby robota. Tento jazyk umožňuje cyklické vykonávání zvolených částí trajektorie a pokrývá veškerou funkcionalitu robota. Vytvořené pohybové sekvence je možné ukládat a načítat z vnitřní paměti PLC.

Klíčová slova:

Automatizovaný sklad, Systematic, Syslogeum 3000, robot, B&R, MappView, Vizualizace, programování, PLC

Abstract

This work is part of the development of the robotic warehouse Syslogeum 3000 of Systematic a.s. The aim of the work is to create software in the control PLC, which will allow easy testing of the storage robot. Under normal circumstances, the robot is controlled from a superior system. At some point, however, the robot must be controlled independently of this system. The aim of the work is to allow the service technician to program any trajectory of the robot and thus test its functionality. To control the developed software was created visualization, which contains a graphic and text editor. Part of the work is the creation of a simple programming language, which describes the individual movements of the robot. This language allows cyclic execution of selected parts of the trajectory and covers all the functionality of the robot. The created motion sequences can be saved and read from the internal memory of the PLC.

Key words:

Automated warehouse, Systematic, Syslogeum 3000, robot, B&R, MappView, Vizualization, programming, PLC

Poděkování

Chtěl bych v prvé řadě poděkovat svému vedoucímu, panu Ing. Leošovi Beranovi Ph.D, za jeho ochotu a velmi vstřícný přístup. Ačkoli byla situace pandemií koronaviru SARS-CoV-2 značně komplikovaná, právě díky jemu jsem mohl práci v řádném termínu dokončit. Ať již po telefonu či přes email, vždy byl k dispozici a připraven pomoci. Vděčný jsem mu také za téma této diplomové práce.

Dále bych chtěl na tomto místě poděkovat kolegovi Tomáši Kubíčkovi, který mi byl nejednou nápomocen. Děkuji mu především za pomoc s Gitem, se kterým jsem se v průběhu práce zaučoval. Téměř spásou pro mě byla jeho rada při pátrání po chybě ve vizualizaci. Nebýt jí, asi bych postrádal svou flintu hluboko v žitě.

Obsah

Seznam obrázků

Seznam zkratek

AS	Automation Studio
BMS	battery management system
EAN	European Article Number
FB	Funkční blok
FTP	File Transfer Protocol
FM	Fakulta mechatroniky, informatiky a mezioborových studií Technické univerzity v Liberci
HDD	Hard Disk Drive
HMI	Human Machine Interface
OPC UA	Open Platform Communications – Unified Architecture
PLC	Programmable Logic Controller
SOC	State of charge
TUL	Technická univerzita v Liberci

1 Úvod

Od roku 2018 je na Fakultě mechatroniky vyvíjen systém automatického skladu SysLogeum 3000. Na tento projekt je vázáno více studentských závěrečných prací. Tématem této práce je vytvořit software a k němu příslušící uživatelské prostředí (vizualizaci), které umožní řídit skladovacího robota pomocí definovaných testovacích tras. Účelem je obejít existující automatizovaný systém pro ovládání robota, tzv. plánovač. Ten řídí pohyb robota tak, aby splnil příkaz uživatele skladu. Jedná se tedy o vychystání požadované krabice nebo uskladnění nové. Cílem této práce je umožnit servisnímu technikovi ovládat robota tak, jak by jej ovládal plánovač.

Aby však bylo vůbec možné robota ovládat, je třeba se nejprve seznámit s řídícím firmwarem robota. Od základu je skladovací systém postaven na PLC od společnosti B&R. S touto rakouskou společností fakulta spolupracuje již delší dobu, ale do studijních osnov se výuka s těmito PLC začala objevovat teprve nedávno. Z důvodu absence zkušeností s těmito stroji je žádoucí před samotnou prací účast na školících kurzech, pořádaných českou pobočkou společnosti B&R v Brně a Praze.

Pro jednodušší práci s vytvořenými pohybovými sekvencemi (tj. jejich ukládání, otvírání a editace) je nutné, aby vzniklý editor tras mohl zpracovávat program i ve formě textu. To vyžaduje vytvoření speciálního programovacího jazyka, kterým bude pohybová sekvence popisována. Součástí tedy musí být i kompilátor. Proto je nutné vytvořit uživatelsky přívětivé rozhrání, ve kterém bude možné vytvořit a editovat program pohybu robota a řídit průběh jeho vykonávání. Nedílnou součástí práce je tedy vizualizace a s ní i návod k její obsluze.

2 Skladová logistika

Každá firma, která ke své činnosti vyžaduje pohyb materiálu, dílů či zboží, musí nutně řešit logistický problém, a tím je skladování. Manipulace se skladovaným materiálem probíhá povětšinou ve čtyřech základních krocích – přejímka, uskladnění, vychystání a expedice [1]. Každý jednotlivý krok s sebou neodmyslitelně nese určité provozní náklady. Přirozená snaha je náklady na skladování minimalizovat.

V minulosti byl pojem sklad téměř vždy spojen s pozicí skladníka. To však nebyla jediná pozice nutná k chodu skladu. Výše zmíněné kroky při manipulaci se skladovaným materiálem ke svému chodu vyžadují jedno celé firemní oddělení – oddělení logistiky. Personální náklady jsou tedy významnou položkou finanční zátěže při provozu skladu.

Další náklady jsou spjaty se samotným skladovacím prostorem. Čím více materiálu je třeba uskladnit, tím větší prostor je zákonitě nutné vyčlenit. A zde nejde pouze o místnosti. Není neobvyklé, když skladovacím prostorem je extra stan, hala nebo celý komplex budov. Efektivita zaskladnění prostoru je tedy další věcí při snižování celkových nákladů. To však nelze dělat bez jistých vedlejších efektů.

V logistice je velmi důležitým faktorem čas. Pokud by výroba musela stát jen kvůli tomu, že potřebné díly jsou zarovnány v tom nejodlehlejším rohu skladu, kde se sotva protáhne skladník, natož pak s vysokozdvižným vozíkem, finanční ztráty s tím spjaté by rychle vedly k otázce, zdali by se s tím skladem nemělo něco dělat. Čas při uskladňování a vychystání materiálu nelze tedy opomíjet. Je zde ale ona nepřímá úměra s efektivitou zaskladnění prostoru. Čím lépe je prostor využit, tím déle zřejmě potrvá manipulace s uskladněným materiálem.

Jsou zde i další, již ne tak patrné, zdroje nákladů. Je-li sklad obsluhován vysokozdvižným vozíkem, jsou s ním spjaty provozní náklady, servis, školení atd. Je-li třeba skladový prostor chladit, odvětrávat či udržovat v čistotě, jsou to další náklady. Takto by se dalo pokračovat, ale stěžejními faktory při snižování nákladů na skladování jsou: personál, kapacita, rychlost. Všechny tyto položky spolu více či méně souvisejí.

2.1 Automatizace skladu

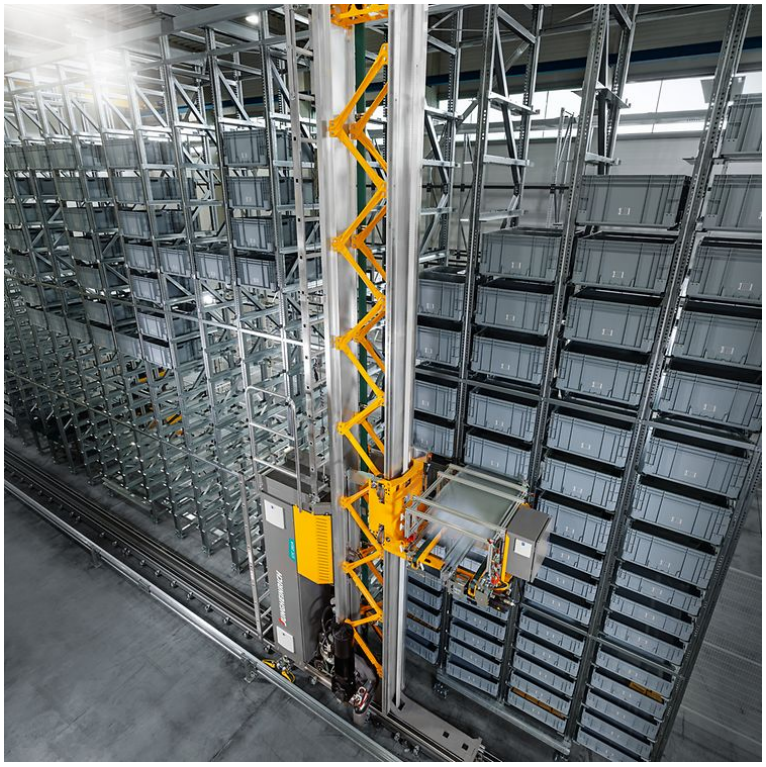
Když se začne mluvit o snižování nákladů a škrtech, zaměstnanci oprávněně zbystří. Snižování personálních nákladů vede logicky k propouštění. Lidská práce je v dnešním liberálním kapitalismu velmi drahá a je-li někde možné jí nahradit, nahradí se. Ostatně, o tom je celá industrializace, dnes již v nastupující verzi 4.0. Ani skladům se tento trend nevyhnul. Pokud je tendence snížit vliv lidského faktoru v procesu skladování, je tím nutně vyvíjen tlak na automatizaci. Automatizovaných skladů je dnes celá řada různých druhů, přičemž každý se hodí na jiný způsob použití a sortiment skladovaného materiálu.

Ať už je však skladováno cokoliv, vždy je kladen důraz na efektivitu využití prostoru. Každý chce, aby sklad pojmul co nejvíce věcí, ale zabral nejmenší prostor. Tedy mít co možná nejvyšší hustotu zaskladnění. Ovšem, jak již bylo zmíněno, je nutné brát zřetel na onu nepříjemnou úměru s časem.

Ve výrobním průmyslu velmi často využívaným druhem automatizovaného skladu je systém tzv. karuselu. Nejedná se zde o žádný soustruh, ale o sklad na rotačním principu (z franc. *caroussel*, kolotoč). V základě se jedná o to, že pokud je třeba dostat určitou skladovanou položku k výdejnímu místu, kde bude posléze vyzvednuta obsluhou, mechanismus orotuje celý skladový prostor, dokud se požadovaná položka nebo segment skladu nedostane k výdejnímu místu. Může se jednat o horizontální i vertikální uzpůsobení.

Obrázek 2.1: Horizontální karuselový sklad Kardex Remstar. [2]

Jako příklad výrobce karuselových skladů lze jmenovat např. spol. Kardex Remstar. Ta vyrábí jak sklady horizontálního, tak i vertikálního typu. S vertikálním typem jsem se již v jedné firmě osobně setkal. Prostorově je to velmi úsporné řešení, jelikož většina zabraného prostoru, jak již název napovídá, je ve výšce. Podmínkou je tedy vysoký strop, což ovšem výrobní haly mívají. Jeden sklad může mít i více výdejních míst (např. v přízemí a patře). Jedna police se může zatížit i více jak 500 kg (dle typu). Nutné ovšem je brát zřetel na rotační vyvážení skladované zátěže [2]. Kapacitně však jde spíše o menší sklady. Obsluha je jednoduchá a vychystání velmi rychlé. Bez pochyby se jedná o velice funkční řešení, tedy až do chvíle, kdy fungovat přestane. Byl jsem svědky toho, kdy práce celého týmu lidí stála pouze na tom, že se z automatizovaného skladu ozývaly znepokojivě skřípavé zvuky, načež ovládací terminál skladu překrylo chybové hlášení. Ano, spolehlivost je dalším důležitým faktorem, který přichází s automatizací skladů.

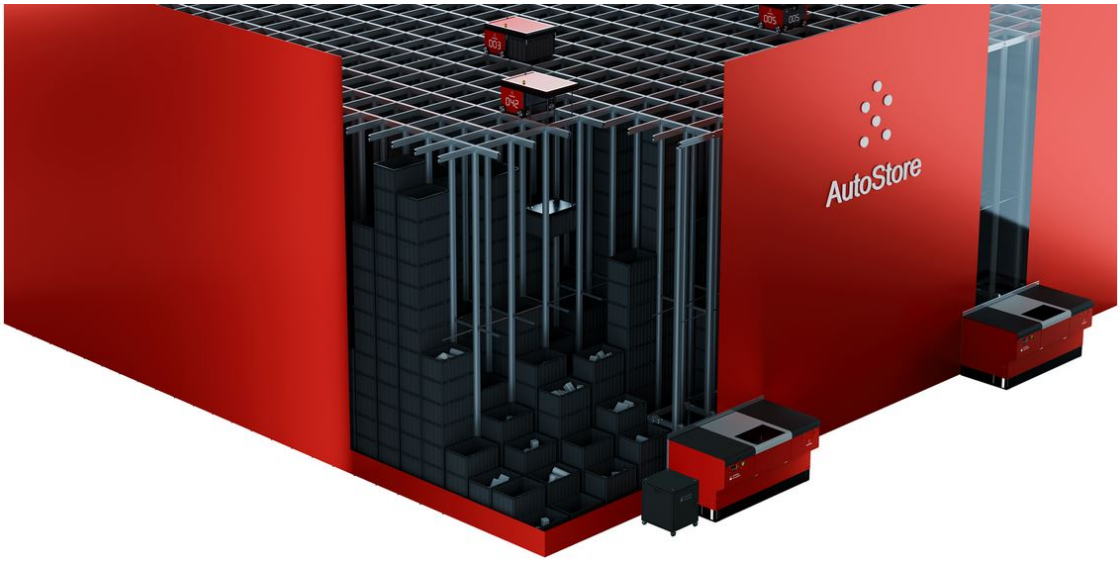


Obrázek 2.2: Regálový sklad spol. Jungheinrych. [3]

Další, hojně využívaný typ je regálový sklad. Jedná se o dobře známý systém palet, ovšem v automatizované verzi odpadá práce řidiče vysokozdvižného vozíku. Ta je nahrazena buď sloupovým zakladačem, který je v každé uličce mezi regály, nebo robotickým vysokozdvižným vozíkem. Tímto se zabývá např. spol. Jungheinrich, která na zakázku staví celé automatizované skladovací linky [3]. Tento typ skladu je vhodný pro velké úložné kapacity. Hustota zaskladnění je dobrá, musí být ovšem přítomna mezera mezi regály pro přístup zakladače. Čas pro uskladnění a vychystání je velice závislý na velikosti skladu. Jednoznačně však za karuselovými

sklady pokulhává.

Na trhu se v nedávné době objevil další typ skladu s názvem AutoStore. AutoStore má modulární mřížkovou konstrukci, což umožňuje téměř absolutně využít dostupný prostor. V každém sloupci mřížky jsou stohovány krabice, jedna vedle druhé. Manipulaci s krabicemi obsluhují pojízdní roboti, jezdící na úplném vrcholu skladu. Roboti mají funkci jakéhosi jeřábu, přičemž jezdí po kolejnicích, umístěných na vrcholu skladu. Pokud je třeba vychystat krabici, která je vespod, roboti musí horní krabice přeskládat. To je samozřejmě velmi časově náročné, a proto jsou tyto sklady vhodné spíše pro méně obrátkové použití. Jsou-li některé položky skladu přeci jen častěji používané, systém je dokáže ve volných chvílích automaticky přerovnat tak, aby byly ihned dostupné. To dokáže velmi významně zkrátit doby vychystání. Navíc je možné, aby na jednom skladu operovalo více robotických jeřábů. Mohou tedy navzájem spolupracovat. [4][5]



Obrázek 2.3: Sklad typu Autostore. [4]

3 Systém SysLogeum 3000

Projekt Archeion, společnosti Systematic a.s., je koncept moderního skladování, postaveném na plně automatizovaném skladovacím systému SysLogeum 3000. Základní princip tohoto systému je stohování plastových euroboxů, stejně jako u AutoStore. Tyto boxy jsou zasazovány do hliníkové konstrukce skladu tak, aby byl co možná nejvíce využit dostupný prostor. Horní část konstrukce je opatřena kolejnicemi, po kterých se pohybují obslužní roboti.

Součástí projektu Archeion je také softwarové aplikační řešení „E-Archiv“. Jedná se o elektronický evidenční systém, který dává uživateli přehled o evidovaných dokumentech.

Tento skladovací systém je, vzhledem k omezené velikosti boxů a časové náročnosti potřebné při vychystání, vhodný především pro archivaci písemných dokumentů, HDD atp. [6]

Obrázek 3.1: Model skladu projektu Archeion. [6]

3.1 Konstrukce

Konstrukte celého skladu je tvořena hlavním rámem, fixujícím modulární síť hliníkových L profilů, které slouží jako opora pro přesné umístění boxů na sebe. Tím je zabezpečeno, že jednotlivé boxy do sebe přesně zapadnou a robotický jeřáb je posléze může bezpečně uchopit. Modularita celého skladu umožňuje maximálně využít

dostupný prostor. Nemusí se tedy jednat jen o obdélníkový půdorys. Na horní rámové konstrukci jsou v osách X a Y kolejnice, po kterých se pohybují obslužní roboti. Boxy jsou dostupné pouze z vrchu a robot s nimi manipuluje pomocí zvedací plošiny.

3.2 Robot

Roboti systému SysLogeum 3000 se pohybují po kolejnicích v osách X a Y. Přepínání mezi osami je řešeno pomocí dvou podvozků, z nichž jeden je výsuvný. Pevný podvozek slouží k pohybu v ose X. Při pohybu v ose Y výsuvný podvozek zasune svá kola do kolejnice a zvedne celého robota spolu s pevným podvozkem.

Obrázek 3.2: Skladový robot systému SysLogeum 3000.

Pro přesné polohování je robot opatřen senzorem, který signalizuje hranu kolejnice. Přesnost polohy je velice důležitá pro spolehlivý chod skladu. Pokud by robot nenajel přesně na buňku gridu, při změně podvozku nebo manipulaci s boxem by mohlo dojít k jeho vykolejení.

Každý robot je opatřen výsuvnou uchopovací plošinou. Ta je zavěšena na ocelových svinovacích pásících. V principu robot funguje jako pojízdný jeřáb. Při zvedání boxu ze stohu robot sjede s plošinou na úroveň horní hrany krabice a poté vysune zámky, které přesně zapadnou do uchopovacích otvorů boxu. Následně robot box zvedne nad úroveň rámu, čímž je možné s ním dále manipulovat.

Obslužní roboti jsou napájeny z vnitřních akumulátorů a dle potřeby se mohou sami dobíjet. K tomu slouží vyhrazená místa na okraji konstrukce s nabíjecím konektorem.

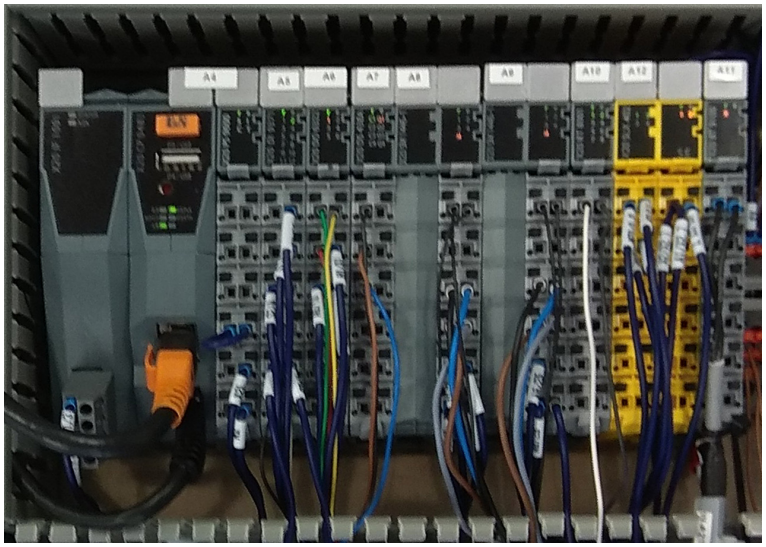
Dobíjecí stanice jsou roboty obsluhovány automaticky. Příkazy od řídícího systému (tzv. plánovače) jsou robotům zasílány bezdrátovou sítí.

3.2.1 Řídící systém

Hardware řídícího systému robota je postaven na produktech společnosti B&R. Je použito PLC typu X20CP0483 rozšířené o periferní moduly.

Software robota se skládá z více řídících procesů, obsluhující dílčí funkce. Jsou zde procesy obstarávající BMS, safety funkce a procesy řídící motory a polohy jednotlivých os, nad kterými je proces *AvatarCOM*. Tento proces je stěžejní pro řízení robota. Obstarává příjem příkazů a jejich vykonávání. Právě s tímto procesem komunikuje externí nadřazený řídící systém skladu, tzv. plánovač, který zabezpečuje naskladnění, vychystání a reorganizaci skladovaných boxů. Robot je tedy řízen z nezávislého vnějšího systému, se kterým komunikuje bezdrátově pomocí serveru OPC UA.

Řídící PLC robota obsahuje též vizualizační komponentu mapp View, která obstarává HMI systému robota.



Obrázek 3.3: Hardware řídícího systému robota.

3.3 Box

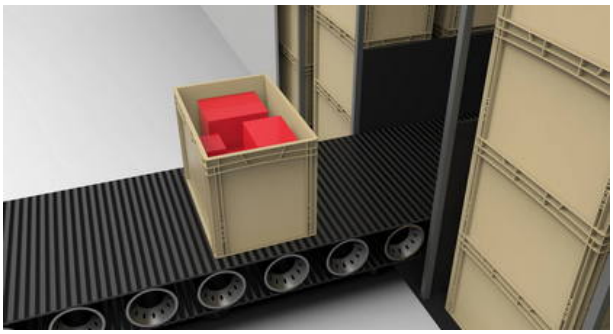
Skladovací systém využívá standardizované plastové boxy s vysokou statickou zatížitelností, nutnou pro jejich následné stohování. Systém je nastaven na plastové boxy o rozměrech 600 x 400 x 420 mm, které pojmou až 1 m^3. Každý box je opatřen kódem EAN, který je uložen v databázi skladu.

Obrázek 3.4: Normalizovaný plastový box. [6]

3.4 Předávací místo

V každém skladu jsou integrována předávací místa, kam roboti doručují požadované boxy. Předávací místo „IN/OUT“ může sloužit současně pro vyskladnění a naskladnění boxů. Předávací místo typu „Table“ je otevřený stůl, který slouží pro přístup k boxu bez opuštění skladovacího prostoru.

Při naskladnění nového boxu je pásovým dopravníkem dopraven do oblasti předání, kde je box zvážen a přečtením čárového kódu uložen do databáze. Robot jej následně zvedne a dopraví na volnou pozici skladu.



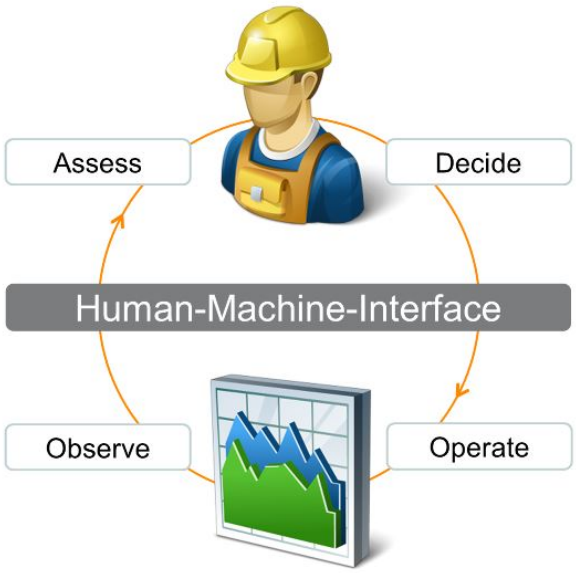
Obrázek 3.5: Předávací místo. [6]

4 Vizualizace

Aby bylo možné systém editoru tras robota obsluhovat, je třeba, aby obsahoval uživatelské rozhrání, tzv. vizualizaci. Pro snadné pochopení a rychlou orientaci v prezentovaných informacích hraje vizualizace významnou roly. Je to právě zrak, ze kterého lidský mozek čerpá nejvíce informací o svém okolí. Tato skutečnost je nutně respektována i v průmyslových procesech.

Vizualizace je přesná transformace dat (skutečnosti) do viditelného obrazu pro jejich snadnější získání, poznání a pochopení. Při její tvorbě platí zásady názornosti.

Probíhá-li interakce mezi obsluhou a strojem (procesem), jedná se o tzv. human-machine interface, zkráceně HMI. Jedná se o komunikační most mezi rozdíly lidského a strojového jazyka. HMI řízeného procesu musí splňovat různorodé očekávání a požadavky uživatelů, a tím poskytnout nejvyšší možný provozní komfort. [8]



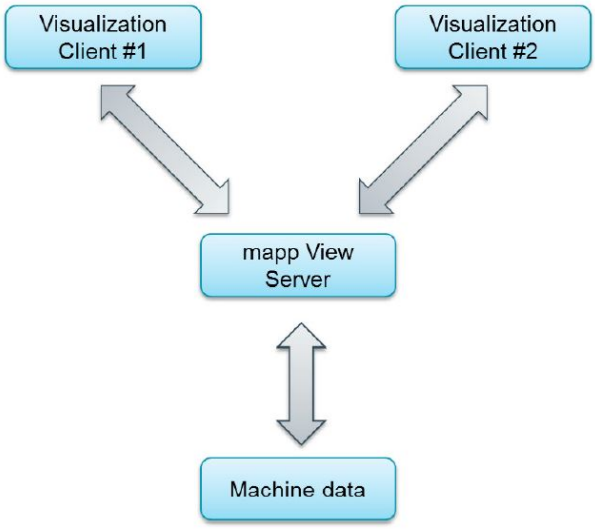
Obrázek 4.1: HMI - rozhraní mezi člověkem a strojem. [8]

4.1 Mapp View

Společnost B&R vytvořila pro své vývojové prostředí Automation Studio technologii „mapp“, která významně zjednodušuje tvorbu systémů pro průmyslové stroje a zařízení. Jedná se o předpřipravené modulární programové struktury, které se nakonfigurují a implementují jako funkční celek. Technologických balíků mapp je k dispozici mnoho, např. pro řízení pohonů je určen mapp Motion, pro bezpečnostní prvky mapp Safety a pro tvorbu grafického rozhraní mapp View. Všechny mapp moduly jsou spojeny skrze mapp Services.

Utilita mapp View je určena pro vývoj HMI systémů v B&R automatizačních aplikací. Mapp View je plně integrováno ve vývojovém prostředí Automation Studio společnosti B&R, ale podléhá licenci. V něm vytvořená vizualizace je založena na webovém prostředí. Tím je dosažena maximální kompatibilita se zobrazovacími zařízeními různých typů a značek. Není však nutné znát webové programovací jazyky – mapp View zdrojový kód generuje sama.

Procesy zabezpečující vizualizaci jsou integrovány ve firmwaru PLC (tzv. Runtime). Běží tedy nezávisle na uživatelských procesech, a to v době jejich nečinnosti (tzv. Idle time). K samotnému webovému rozhraní vizualizace se připojuje přes IP adresu PLC, kde je standardně vyhrazen port 81. Vizualizace podporuje připojení více zařízení (multi-client) a uživatelů (multi-user). V základní licenci je možné v jednu chvíli připojení pouze jednoho klienta. Přístup dalších je podmíněn na zakoupení licence. [9]



Obrázek 4.2: Architektura mapp View. [9]

4.2 Základy tvorby vizualizace

Grafická stránka vizualizace se v AS skládá z tzv. Layout – definovaného prostoru (rozlišení), ve kterém je vizualizace zobrazována, Areas – oblastí v Layout, kde jsou umístěny jednotlivé definice obsahů stran tzv. Contents. Do Content se umísťují tzv. Widgets, tedy elementární grafické položky vizualizace jako jsou texty, číselné hodnoty, obrázky atp. Content je následně zobrazen v HMI na tzv. Page, tedy stránce výsledné vizualizace.

Vizualizace může mít přístup k proměnným v PLC přes server OPC-UA. Veškeré vazby widgetů na proměnné jsou definovány v souboru Binding každého contentu. Ten je psán v XML kódu a příklad definice vypadá následovně:

```
<Binding mode="twoWay">
   <Source xsi:type="opcUa" refId="::AvatarCOM:executeStandBy"attribute="value" />
      <Target xsi:type="brease" contentRefId="SequencerContent"
        widgetRefId="StandByButton" attribute="value" />
</Binding>
```

Tímto je definována vazba mezi přepínacím tlačítkem „StandByButton“ a proměnnou „executeStandBy“ v procesu „AvatarCOM“. Atribut bindingu „mode“ určuje, zdali je přístup read/write (twoWay) nebo pouze read (oneWay).

Některé prvky vizualizace mohou iniciovat události, tzv. events. Ty jsou definovány v souboru Eventbinding pro příslušný content. Možné události se liší prvek od prvku. Nejčastěji využívanou událostí je „Click“, která je vyvolána kliknutím na daný prvek. Taková událost je definována následovně:

```
EventBinding id="SequencerContent.Button2.Click">
<Source contentRefId="SequencerContent" widgetRefId="Button2"
   xsi:type="widgets.brease.Button.Event" event="Click" />
      <EventHandler>
         <Action>
            <Target xsi:type="opcUa.NodeAction" refId="::Sequencer:Plus\_btn" >
               <Method xsi:type="opcUa.NodeAction.SetValueBool" value="true" />
            </Target>
         </Action>
      </EventHandler>
</EventBinding>
```

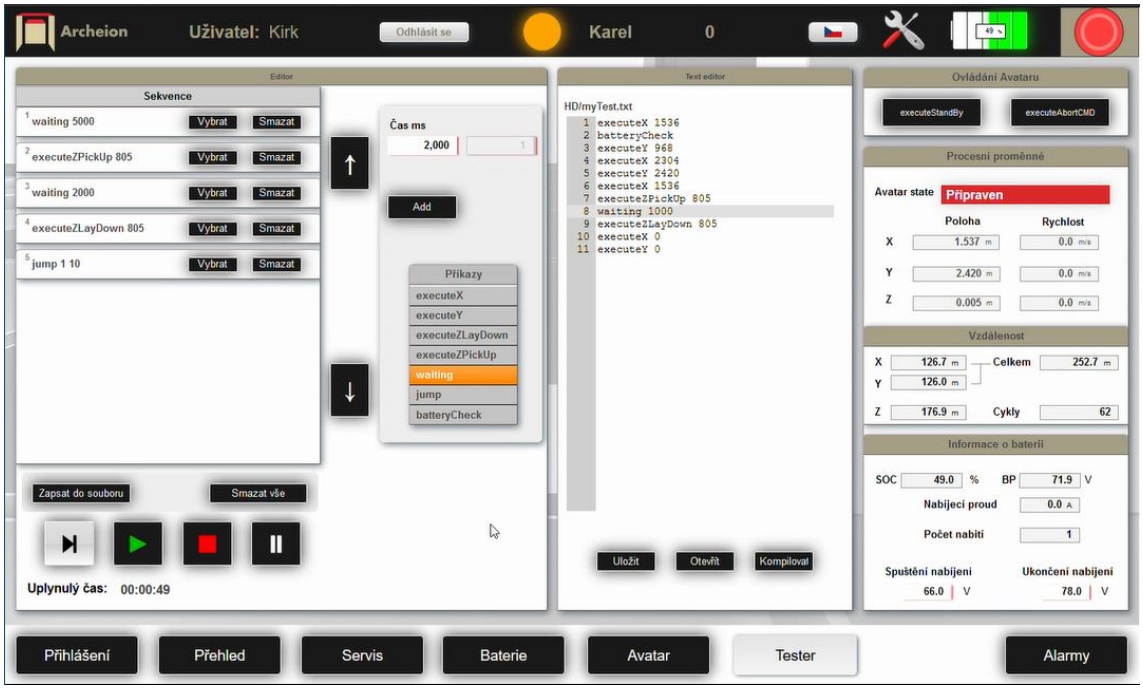
Po kliknutí na tlačítko „Button2“ je vyvolána akce, která nastaví na OPC UA serveru proměnnou „Plus_btn“ v procesu „Sequencer“ na hodnotu „true“.

Každý soubor s bindingem a eventbindingem je nutné uvést v souboru „Visualizat.vis“, definující použité soubory vizualizace.

5 Rozhraní Servisního testeru

Rozhrání Servisního testeru se skládá z jedné strany vizualizace PLC robota. Ta byla zhotovena v české i anglické variantě. Pro její snadnou obsluhu byl vytvořen uživatelský manuál, který je součástí přílohy této práce. Vizualizaci lze rozdělit do několika základních funkčních celků. Jde o následující funkcionality:

- **Grafický editor**
- **Textový editor**
- **Ovládání Avataru**
- **Procesní proměnné**.



Obrázek 5.1: Rozhraní Servisního testeru.

Následující kapitoly se budou jednotlivým prvkům vizualizace věnovat. Nejprve je však nutné definovat, s čím vlastně Servisní tester pracuje – tedy příkazy.

5.1 Seznam příkazů

Hlavním úkolem Servisního testeru je umožnit servisnímu technikovi zadat požadovanou sekvenci příkazů a uvést tím následně robota do pohybu. Aby bylo možné s příkazy pracovat, musí být jednoznačně definovány. Každému příkazu je tedy přiřazen jeho název, vycházející z jeho pojmenování v obslužném procesu *AvatarCOM*. Seznam možných pohybů a jejich příkazů je následující:

- Pohyb v ose X - **executeX**
- Pohyb v ose Y - **executeY**
- Zvednutí krabice - **executeZPickUp**
- Položení boxu - **executeZLayDown**
- Připojení k nabíječce - **batteryCheck**
- Odpojení od nabíječky – **batteryCheck**.

Příkaz pro připojení a odpojení od nabíječky je jeden a ten samý. Je to z toho důvodu, že tento příkaz, jak už název napovídá, kontroluje stav baterie a podle toho vykoná potřebný pohyb.

Většina příkazů, krom připojení a odpojení nabíječky, vyžadují stanovení cílové hodnoty. U příkazů pro pohyb v osách X a Y se jedná o cílovou polohu – jde tedy o absolutní polohování od stanovené nulové pozice. Údaj je zadáván v mm. Hodnota pozice jednotlivých stohů krabic je dána velikostí skladového gridu. V řídícím procesu *AvatarCOM* je tento argument ukládán do proměnných *xRequired* a *yRequired* a následně zpracováván v procesech řízení os.

Příkazy *executeZPickUp* a *executeZLayDown* mají argument hloubky vrchní hrany krabice v mm, měřené od polohy plně zasunuté uchopovací plošiny. Tato hodnota je závislá na konstrukci skladu a může se tedy lišit.

Aby bylo možné řídit chod programu, byly vytvořeny další dva příkazy. Jedná se o příkazy:

- Cyklický skok – **jump**
- Pozastavení běhu programu – **waiting**.

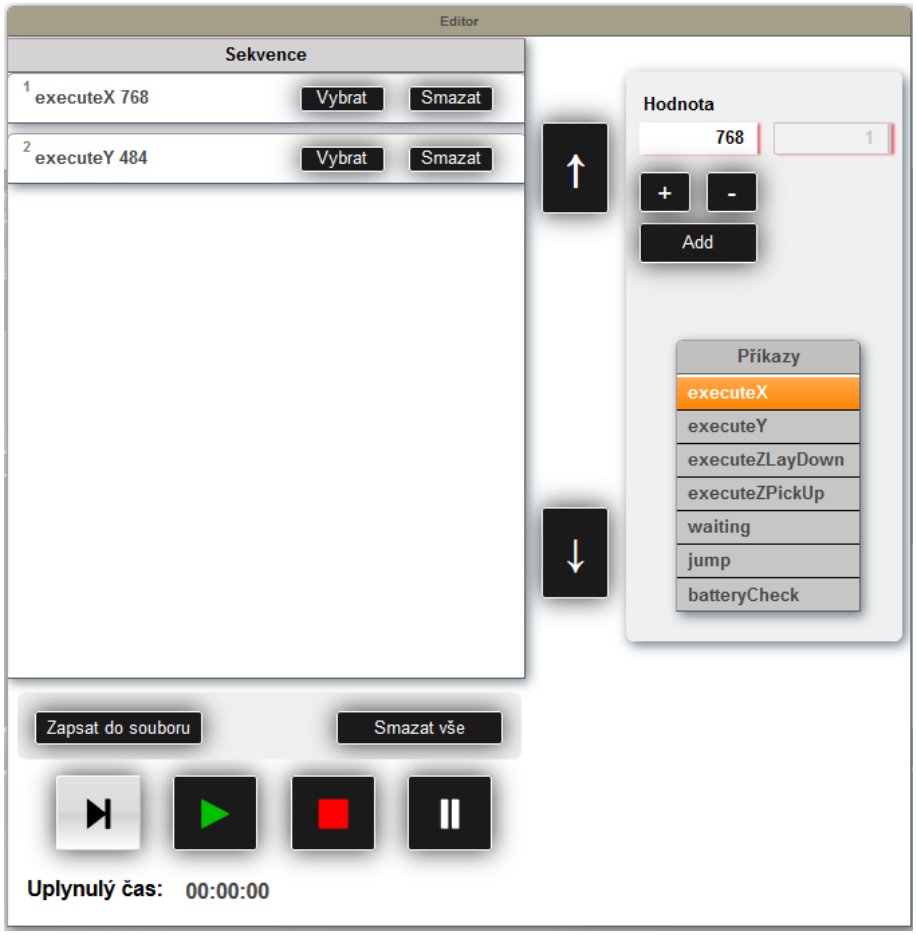
Pro opakované testování pohybu robota po určité trajektorii je nezbytný příkaz pro zacyklení programu. Příkaz *jump* má dva argumenty – číslo příkazu, na který se má skočit (Cíl/Target) a počet opakování cyklu (Opakování/Repetitions). Program Servisního testeru umožňuje libovolný počet vnořených cyklů, omezený pouze

maximálním počtem příkazů v sekvenci. Příkaz *waiting* pozastaví vykonávání programu na stanovený čas v ms.

Toto je tedy kompletní sada příkazů programu Servisního testeru, ze které je možné naprogramovat pohybovou sekvenci. Maximální počet příkazů v sekvenci je omezen na 50. Z empirického hlediska je tento počet plně dostačující. Pohybová sekvence se může vytvářet dvěma možnými způsoby – z grafického nebo textového editoru.

5.2 Grafický editor

Základním způsobem obsluhy Servisního testeru je grafický editor. V něm je možné „naklikat“ program sekvence a následně jej jednoduše editovat. Součástí grafického editoru je i spouštění programu a řízení jeho vykonávání.



Obrázek 5.2: Rozhraní grafického editoru.

Grafický editor se skládá z jednotlivých funkčních částí. Na následujících stranách jsou představeny a popsány.

5.2.1 Sekvence

Při tvorbě pohybové sekvence se příkazy postupně přidávají do boxu s názvem Sekvence. Najednou lze v boxu zobrazit maximálně 10 příkazů, jeli příkazů více, je možné seznam posouvat šipkami umístěnými na pravé straně boxu.

V boxu je zobrazen aktuální sled příkazů, kde každý příkaz má své pořadové číslo, název, hodnoty argumentů a vlastní položku s ovládacími prvky.

Každý příkaz má v levém horním rohu zobrazeno své pořadové číslo v programu. To je důležitý údaj pro snadnou orientaci v celém programu a adresování skoků v příkazu *jump*.

Obrázek 5.3: Položka příkazu

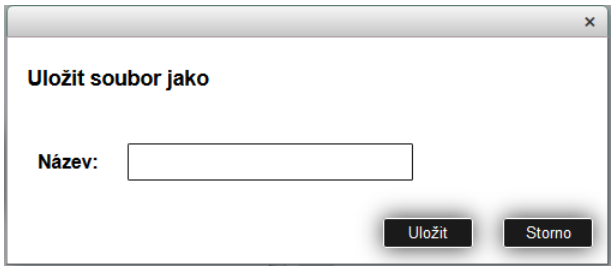
Informace o příkazu jsou zobrazeny v tomto pořadí: název příkazu – 1. argument – 2. argument. Počet argumentů se liší dle typu příkazu.

Tlačítko „Vybrat“ slouží pro dodatečnou editaci daného příkazu. Takto vybraný příkaz se zvýrazní žlutě. Před vybraným příkazem lze také přidat další. To umožňuje dodatečně upravovat sekvenci v libovolném místě. Tlačítko „Smazat“ vymaže příkaz ze seznamu.

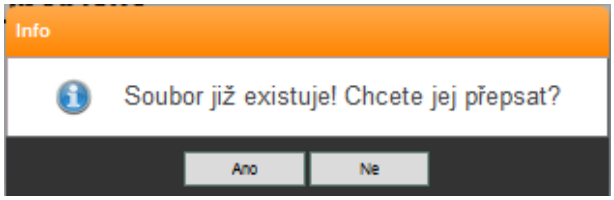
Obrázek 5.4: Vybraný příkaz zvýrazněn žlutě.

Na spodní straně boxu Sekvence je možné tlačítkem „Zapsat do souboru“ aktuální program uložit do vnitřní paměti PLC ve formě textu. Po jeho stisku se zobrazí dialog pro zadání názvu souboru. Název je zadáván bez přípony souboru. Přípona typu souboru je přidána automaticky na „.txt“.

Soubory jsou ukládány do vnitřního úložiště PLC v adresáři „/Sequencer“. Pokud je zadán název souboru, který již v adresáři existuje, je vyvolán dialog, ve kterém se uživatel musí rozhodnout, jestli soubor přepsat či nikoli.



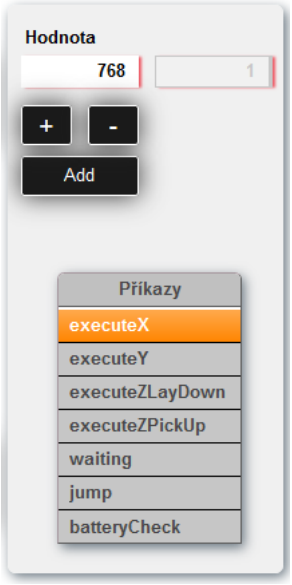
Obrázek 5.5: Ukládací dialog.



Obrázek 5.6: Dialog pro přepsání existujícího souboru.

Po úspěšném dokončení ukládání souboru je zobrazena potvrzující zpráva.

5.2.2 Nástrojový panel

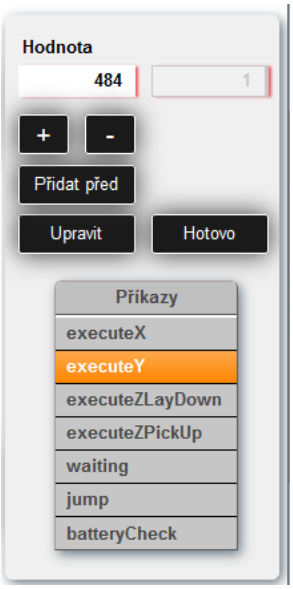


Obrázek 5.7: Nástrojový panel.

Uživatel z nástrojového panelu přidává příkaz či edituje již existující. Na jeho spodní straně je seznam dostupných příkazů. Po kliknutí na požadovaný příkaz jsou na horní straně nástrojového panelu nastavovány hodnoty parametrů. Příkazy *executeX* a *executeY* mají navíc zobrazeny tlačítka pro přičtení/odečtení inkrementu

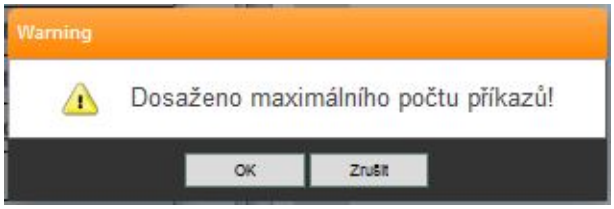
skladového rastru. Jedná se o konstantu vzdálenosti skladových buněk a tedy i kolejnic. Tato vzdálenost je v ose X 768 mm a v ose Y 484 mm.

Stiskem tlačítka „Add“ je příkaz přidán na konec aktuální sekvence. Je-li u některého příkazu v listu stisknuto tlačítko „Vybrat“, nástrojový panel se upraví do módu editace. Automaticky se zvolí typ vybraného příkazu a vyplní nastavené hodnoty (viz obr. 5.8/str. 30). V tomto momentě je možné měnit hodnoty příkazu nebo změnit samotný příkaz. Úpravy se potvrdí stiskem tlačítka „Upravit“. Pro ukončení editace bez úprav slouží tlačítko „Hotovo“. Tlačítko „Přidat před“ umístí nastavený příkaz před příkaz vybraný v seznamu.



Obrázek 5.8: Nástrojový panel při vybrání příkazu.

Maximální počet příkazů je omezen na 50. Pokud se uživatel pokusí přidat více příkazů, zobrazí se upozorňující dialog.

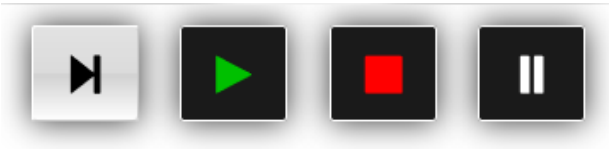


Obrázek 5.9: Dialog při dosažení maximálního počtu příkazů.

5.2.3 Ovládání běhu programu

Spouštění a následné ovládání běhu programu je umístěno na spodní straně grafického editoru. Jedná se o čtyři tlačítka v následujícím pořadí: *Step, Play, Stop*

a *Pause*. Jsou popsány pouze grafickým vyobrazením se symbolikou jejich funkce. Jedná se o dobře známé ovládání například kazetových či CD přehrávačů.



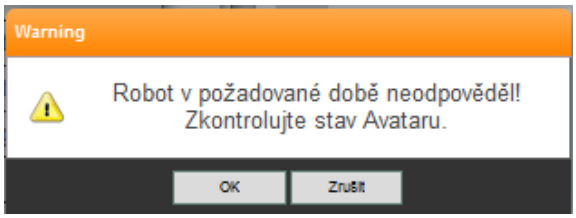
Obrázek 5.10: Ovládání běhu programu.

Kliknutím na tlačítko *Play* se spustí první příkaz sekvence a obnoví se a aktivují stopky umístěné na spodní straně Grafického editoru. Hodnota uplynulého času je zastavena po ukončení testu a zůstává zobrazena. Při vykonávání programu je zobrazeno tlačítko s grafickou značkou „«“ (viz obr. 5.11/str. 31). Toto tlačítko je defaultně aktivováno. Je umístěno mezi rolovacími šipkami.

Obrázek 5.11: Tlačítko Track pro sledování vykonávaného příkazu.

Pokud je aktivní, list sekvence je automaticky rolován podle toho, kde se nachází právě vykonávaný příkaz. Je-li tedy seznam delší, než je možné najednou zobrazit v seznamu (více jak 10 příkazů), je i přes to zabezpečena orientace uživatele v právě vykonávané části programu.

Během vykonávání programu je kontrolována odezva robota. Jedná se o jakýsi watchdog timer. Pokud robot do 5 vteřin nezareaguje na požadovaný příkaz, běh programu se zastaví a uživatel je informován skrze dialog.



Obrázek 5.12: Chybové hlášení při nereagování robota.

Příkazy jsou zvýrazňovány barvami podle toho, co se s daným příkazem děje. Tím je velice názorně uživatel informován o stavu běhu programu. Za normálních okolností se právě vykonávaný příkaz zabarví zeleně. Zůstává zabarven po celou

dobu vykonávání. Po jeho úspěšném vykonání je barva vrácena na bílou a zvýrazní se následující příkaz.

Obrázek 5.13: Vykonávaný příkaz zvýrazněn zeleně.

V případě výskytu chyby se příkaz, u kterého k ní došlo, zvýrazní červeně. To se může stát, pokud obslužný proces *AvatarCOM* odmítne příkaz (např. z důvodu požadavku pohybu na již dosaženou polohu), nebo pokud spadne do erroru (např. chybou pohonů či neurčité polohy robota). V této situaci je vykonávání sekvence příkazů zastaveno.

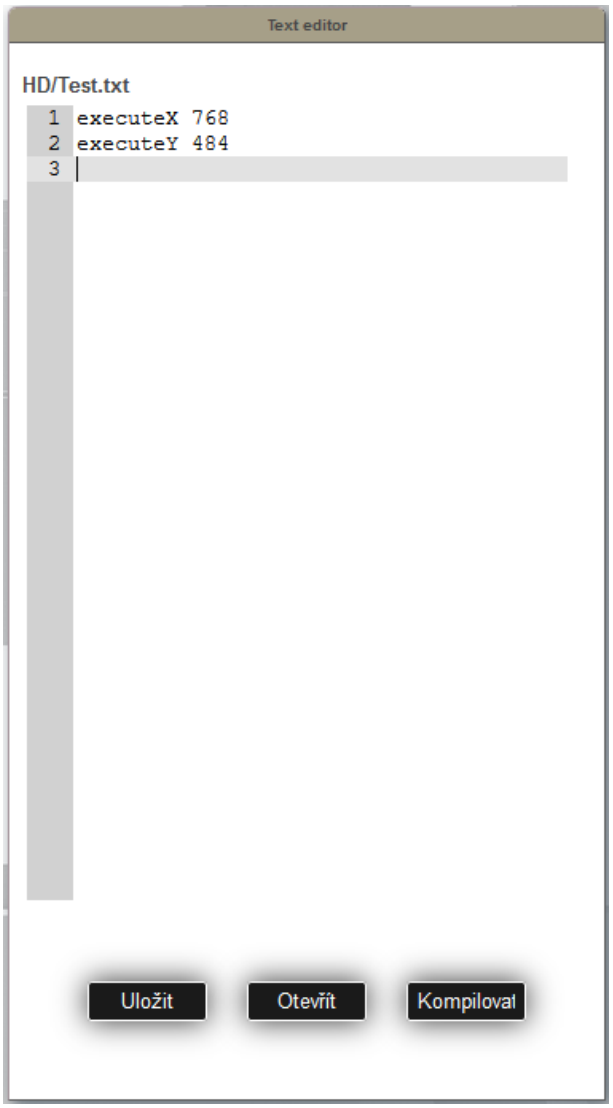
Obrázek 5.14: Příkaz zvýrazněn při chybě červeně.

Je-li vykonávání stiskem tlačítka *Pause* pozastaveno, dokončí se právě vykonávaný příkaz a následující příkaz se zvýrazní oranžově a již se nevykonává. Při pozastavení programu je zaktivováno tlačítko *Step*. V tuto chvíli má uživatel dvě možnosti, jak pokračovat v programu. Při stisku tlačítka *Play* se chod programu opět spustí a bude pokračovat rychlostí vykonávání příkazů robotem. Stiskem tlačítka *Step* se vykoná pouze pozastavený příkaz a následující se znovu pozastaví. Tímto má uživatel vykonávání programu plně pod kontrolou pomocí jeho krokování.

Obrázek 5.15: Příkaz zvýrazněn při pozastavení oranžově.

5.3 Textový editor

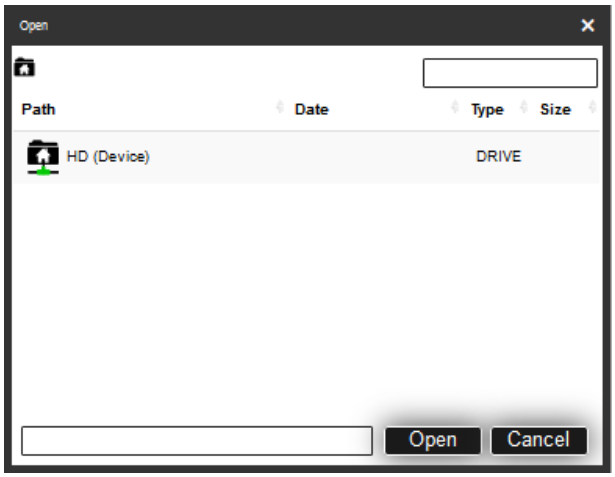
Pro ruční psaní sekvence příkazů slouží textový editor. Jeho rozhraní je poměrně jednoduché. Jeho základem je vizualizační prvek *TextPad*. Ten slouží pro práci s textovými soubory. Jeho implementace do vizualizace vyžaduje konfigurační soubor *mpfilemanager* (viz obr. 6.2/str. 41), umožňující přístup widgety do definovaného adresáře v paměti PLC.



Obrázek 5.16: Rozhraní textového editoru.

Grafický prvek *TextPad* může pracovat s pamětí PLC pomocí integrovaného průzkumníku. Mohou se otvírat existující soubory a ukládat nové. Má-li tedy uživatel uložené některé testovací trajektorie v paměti PLC, může si je tímto způsobem znovu otevřít a spustit.

K otevírání slouží tlačítko „Otevřít“, umístěné na spodní straně textového editoru. Po jeho stisku se zobrazí souborový průzkumník, ve kterém si uživatel vybere v adresáři „HD (device)“ patřičný soubor a tlačítkem „Open“ jej TextPad načte a zobrazí. Tlačítkem „Uložit“ může uživatel aktuální program uložit do paměti PLC. *TextPad* může zobrazovat čísla řádků, což je velmi žádoucí při psaní programu. Pokud uživatel ctí štábní kulturu a umísťuje každý příkaz na samostatný řádek, může snadno adresovat cíl cyklu příkazu *jump*. Také lze lehce nalézt chybu při kompilaci, kdy je uživateli sděleno číslo chybného příkazu.



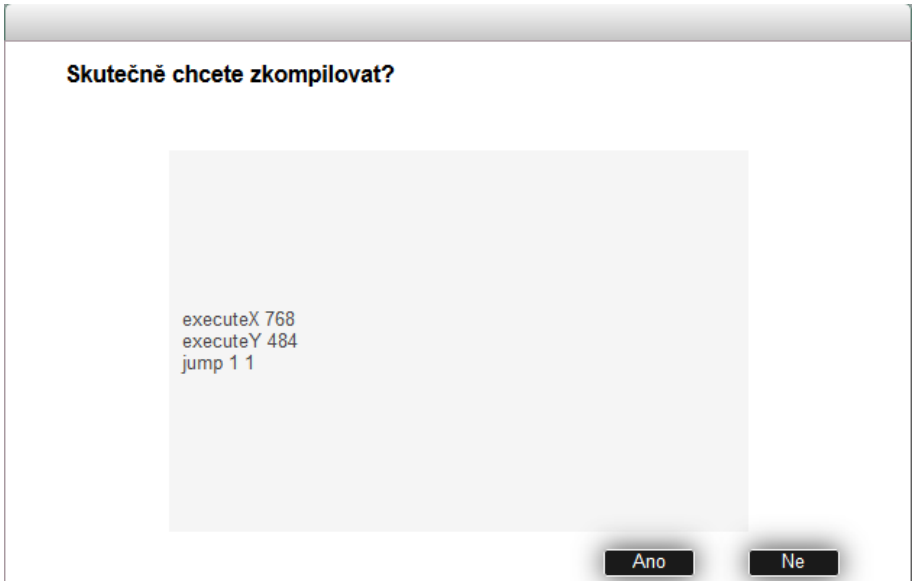
Obrázek 5.17: Souborový průzkumník.

Naposledy otevřený/uložený soubor je zobrazen na horní straně textového editoru. Tato informace je podstatná při editaci již existujícího souboru programu, aby mohl uživatel při ukládání vyhledat původní soubor a případně jej přepsat.

5.3.1 Kompilace

Chce-li uživatel program napsaný v textové formě spustit, musí jej nejprve zkompilovat. Kompilovat jde z důvodu vnitřní funkce TextPadu pouze text ze souboru. Je tedy nutné napsaný program nejprve uložit. Není-li program uložen, nelze kompilaci spustit. Kompilace se vyvolá stisknutím tlačítka „Kompilovat“.

Po kompilaci je zobrazen dialog, informující o jejím výsledku. Pokud je kompilace bez chyb, je zobrazen seznam příkazů tak, jak byl přeložen při kompilaci. Uživatel má v této chvíli možnost zkontrolovat výsledek kompilace sám. Může totiž dojít např. k tomu, že místo číselné hodnoty dá jako argument hodnotu nečíselnou. Pak je hodnota brána jako 0. Uživatel se může rozhodnout, zdali s kompilací souhlasí či nikoli. Pokud ano, sekvence příkazů se převede to grafického editoru, kde se s ní může dále pracovat.



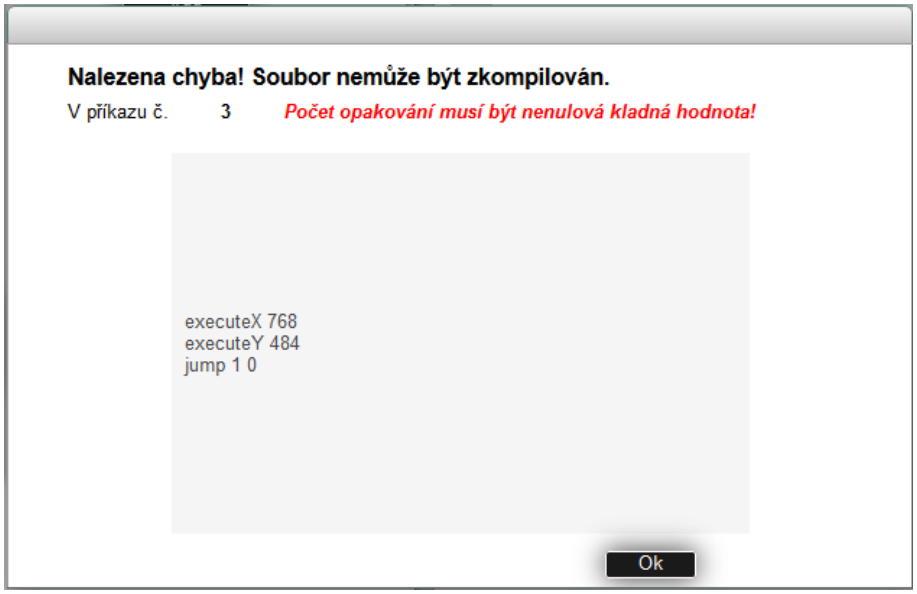
Obrázek 5.18: Dialog po bezchybné kompilaci.

Pokud nastane při kompilaci chyba, je o ní uživatel v dialogu informován. Je zobrazeno číslo pořadí, ve kterém byla chyba identifikována, a základní popis chyby. Kompilátor identifikuje několik možných chyb. Jde o chyby s následujícími popisy:

- Chybný cíl skoku!
- Dosaženo maximálního počtu příkazů!
- Hodnota nesmí být záporná!
- Neznámý příkaz!
- Počet opakování musí být nenulová kladná hodnota!

První uvedená chyba se týká příkazu *jump*. Při kompilaci je kontrolováno, zdali cíl skoku odkazuje na existující příkaz. Skok může odkazovat i na příkaz, který je až za příkazem *jump*. Pokud by však cíl odkazoval na příkaz, který neexistuje, mohlo by dojít k chybnému zásahu do paměti a tím k pádu celého softwaru PLC. Tato kontrola je tedy nezbytná.

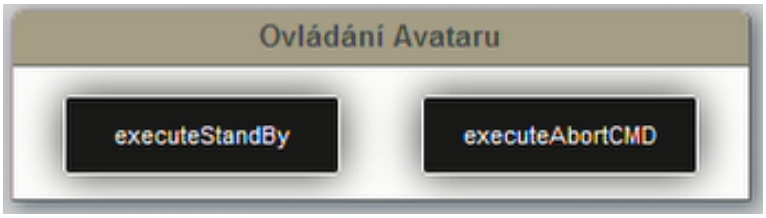
Všechny příkazy mohou mít za argument pouze kladné hodnoty. V grafickém editoru je toto ošetřeno limitovaným intervalem zadávané hodnoty, ovšem v textovém editoru je toto možné kontrolovat až při kompilaci.



Obrázek 5.19: Dialog po chybné kompilaci.

5.4 Ovládání Avataru

Na stránce servisního testeru je možné zadávat základní příkazy řídícímu systému robota, tzv. *Avataru*. Box s příkazy se nalézá v pravém horním rohu.



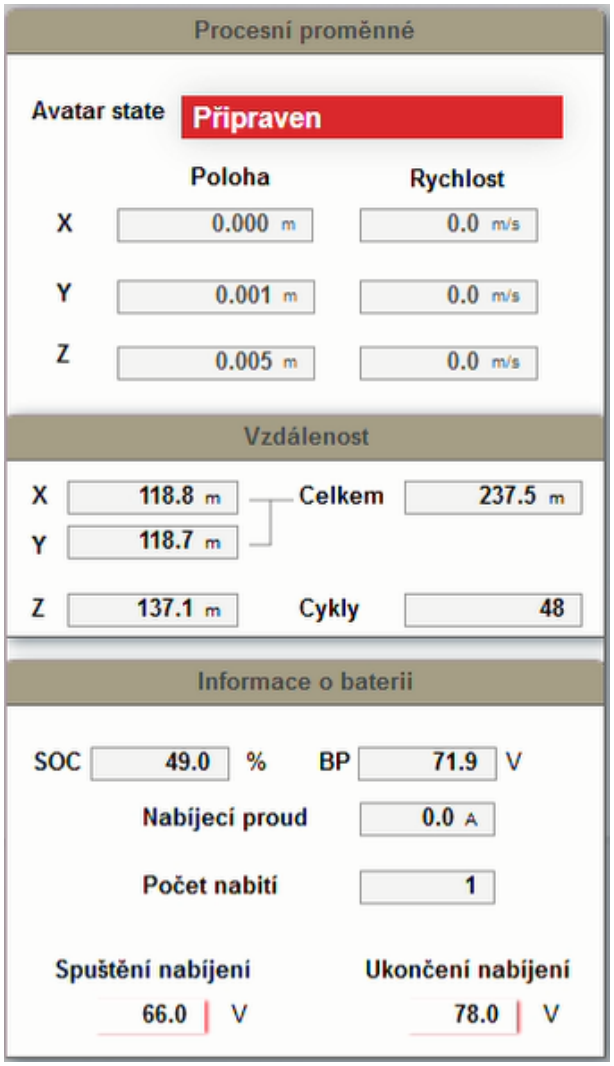
Obrázek 5.20: Ovládání Avataru.

Tlačítko „executeStandBy“ uvede robota do režimu „Stand by“. Pokud se v tomto režimu již nachází, lze ho stiskem tlačítka uvést do režimu „Připraven“.
Tlačítko „executeAbortCMD“ slouží k uvedení robota do stavu „Připraven“ při chybovém stavu „Příkaz zamítnut“.

Toto jsou pouze základní příkazy, které jsou potřeba při práci v Servisním testeru. Kompletní sada příkazů je na jiné straně vizualizace s názvem Avatar. Ta ovšem není součástí této práce.

5.5 Procesní proměnné

Aby měl uživatel kompletní informace o stavu robota, jsou na pravé straně vizualizace Servisního testeru umístěny informační boxy.



Obrázek 5.21: Procesní proměnné.

V horní části je zobrazen aktuální stav Avataru a poloha všech os robota, spolu s jejich rychlostmi. Poloha je uvedena v metrech a rychlost v metrech za sekundu.

Pod touto informací je box s názvem „Vzdálenost“. Zde je uveden celkový počet najetých metrů jednotlivých os. Osy X a Y mají zobrazen součet vzdáleností v údaji „Celkem“. Osa Z má navíc počítadlo uskutečněných cyklů – jeden cyklus je zvednutí a položení krabice.

Ve spodní části jsou informace o baterii. Je uvedena hodnota napětí akumulátoru

v % (hodnota SOC) a ve voltech (hodnota BP). Procentuální hodnota stavu baterie je počítána z maximální (79 V) a minimální (65 V) hodnoty napětí. Dále je zobrazen nabíjecí proud v ampérech a počet najetí na nabíječku.

Poslední hodnoty „Spuštění nabíjení“ a „Ukončení nabíjení“ může uživatel nastavovat. Jedná se o hodnoty napětí, při kterých příkaz *batteryCheck* spustí a ukončí nabíjení.

6 Program Servisního testeru

Kód Servisního testeru se skládá z jednoho procesu, napsaného v jazyce ST (Structured text). Tento proces je pojmenován „Sequencer“ a zařazen do 20ms cyklické třídy 2#. Z větší části se jedná o stavový automat o celkovém počtu 30 stavů. Pro vykonávání každé funkcionality je definována posloupnost příslušných stavů.

Strukturu programu lze rozdělit na několik dílčích celků, které obstarávají určité funkce. Rozdělení je následující:

- Inicializace programu
- Mimostavová část
- Práce se soubory
- Práce s příkazy
- Vykonávání sekvence
- Ostatní stavy

6.1 Inicializace programu

Každý program v PLC má svoji inicializační část. V Servisním testeru jsou v této části programu inicializovány některé proměnné a nastaveny časovače.

6.2 Mimostavová část

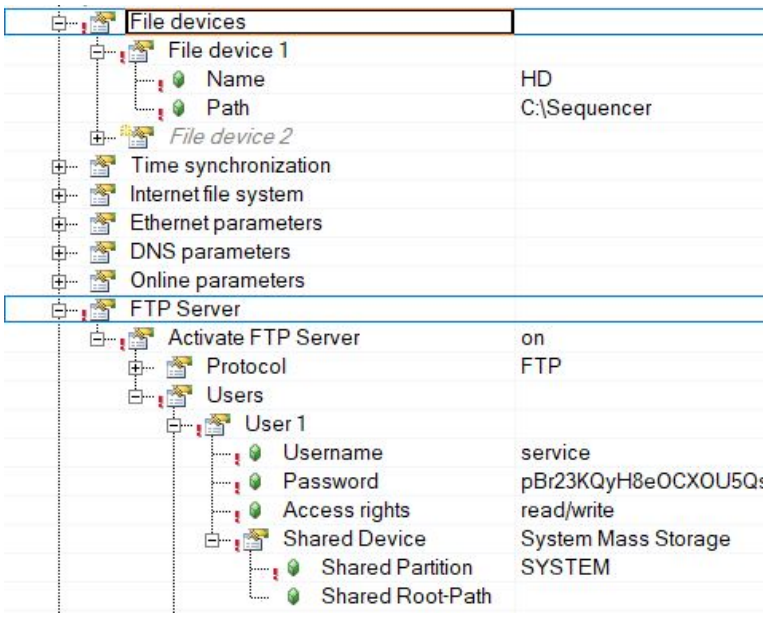
Tato část programu je stále aktivní bez ohledu na stav. Je zde přepis globální struktury, ovládající Avatar, do lokální a naopak. To je pozůstatek po řešení chyby, která zapříčiňovala nepředvídatelné měnění stavů procesu *AvatarCOM*. Chyba byla ovšem způsobena špatným zápisem v binding grafického prvku, zobrazující stav Avataru, která umožňovala zpětně přepisovat tento stav. Mód bindingu byl nastaven na ”twoWay”, namísto ”oneWay” (viz kapitola 4.2/str. 24).

Je zde nastavována viditelnost některých prvků vizualizace v závislosti na vybraném příkazu (pomocí akce *PropertySet*). Týká se to grafických prvků NumericInput pro hodnoty příkazů, jejich popisky a tlačítka inkrementu gridu. Dále je zde tento inkrement vyhodnocován. Je tady také spouštěn FB časovače pro hlídání odpovědi robota a příkazu *waiting*.

6.3 Práce se soubory

Proces Servisního testeru umožňuje uživateli (servisnímu technikovi) ukládat vytvořené pohybové sekvence pro jejich pozdější opakované použití. Sekvence je v textové podobě ukládána do souboru ve vnitřní paměti PLC. Použité PLC disponuje 1GB integrované flash paměti. V hlavním adresáři této systémové paměti je pro potřeby archivace souborů vytvořena složka s názvem „Sekvencer“, která je v programu viditelná pod pojmenováním „HD“.

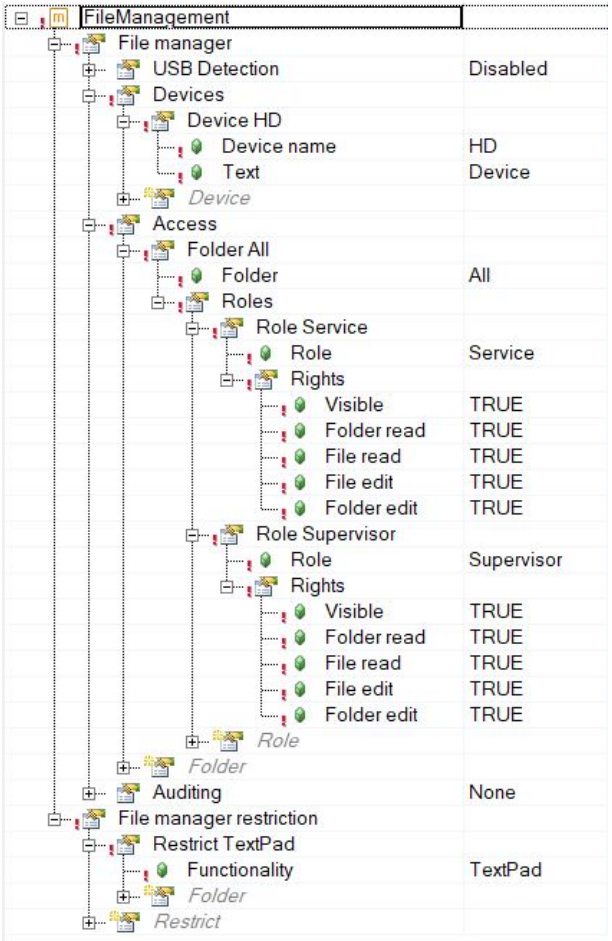
Systémový adresář je přístupný přes FTP server a lze tedy pracovat se soubory nezávisle na vizualizaci Servisního testeru. FTP server je přístupný přes IP adresu PLC pod přístupovým jménem ”service” a heslem stejným, jako má uživatel s rolí supervisora.



Obrázek 6.1: Konfigurace souborového úložiště a FTP serveru.

Program pracuje se soubory dvěma rozdílnými přístupy, jelikož prvek *TextPad* v textovém editoru může pracovat pouze s textovým řetězcem ve formě souboru. Jeden přístup je tedy pomocí implementované funkce grafického prvku *TextPad*.

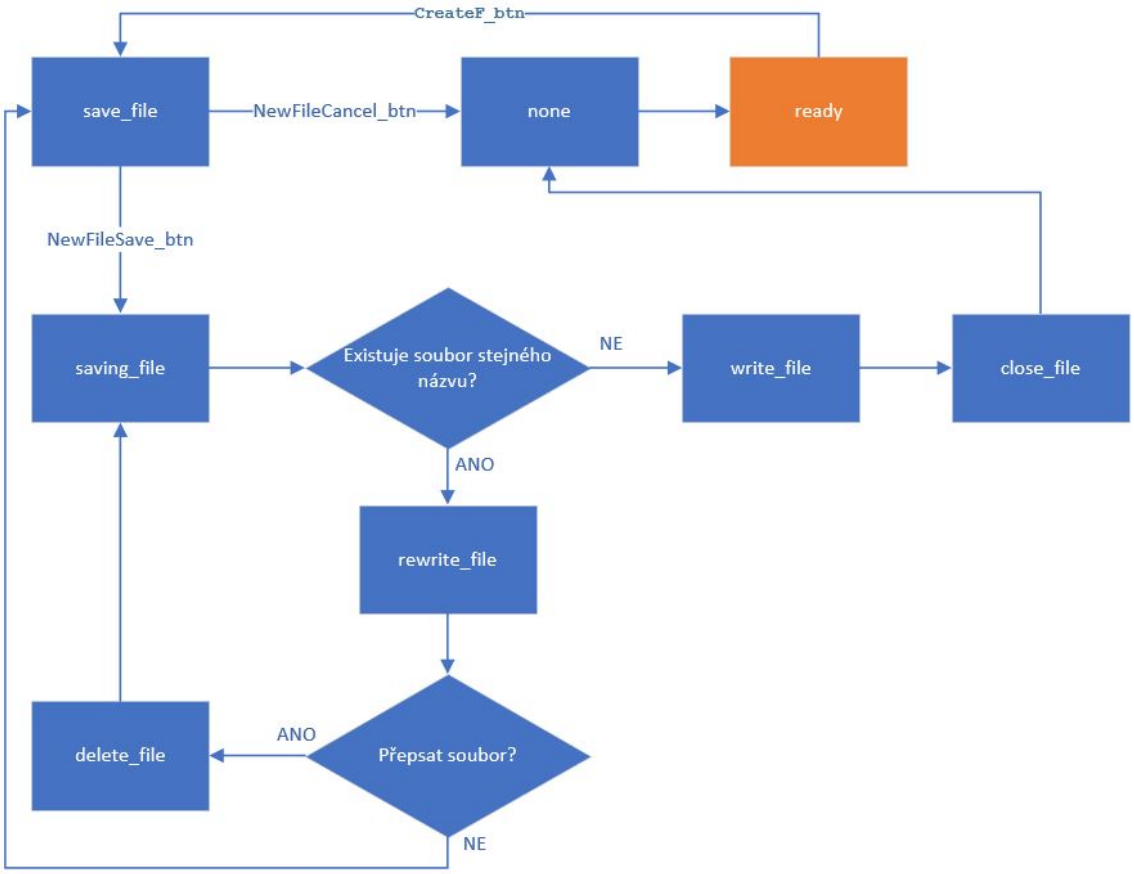
Přístup tohoto prvku k souborům je nastavován konfiguračním souborem *mpfilemanager* utility mappServices. Tím se zajistí, že souborový průzkumník prvku *TextPad* bude mít přístup k nastavenému adresáři, a bude v něm viditelný. To se ovšem i přes opakované přenastavování nestalo. Ve vývojovém prostředí integrovaném simulátoru adresář přístupný byl, ale v reálném PLC nikoli. Řešením bylo až přehrání paměti PLC prázdným projektem a opětovným nahráním původního. Problém zřejmě tkvěl v tom, že před nahráním projektu do PLC se nejprve porovnává rozdíl nahrávaného s již nahraným softwarem a přehraje se pouze jejich rozdíl. To velice urychlí proces nahrávání, ovšem v tomto případě to nejspíš vedlo k nepříjemné chybě.



Obrázek 6.2: Konfigurace v souboru mpfilemanager.

Druhý přístup k souborům je z grafického editoru, který je obsluhován stavovým automatem procesu Servisního testeru. Je využito funkčních bloků *FileOpen*, *FileRead*, *FileWrite*, *FileCreate*, *FileClose* a *FileDelete*. Pro ukládání souboru slouží sekvence stavů, popsaná v následujícím zjednodušeném stavovém diagramu.

6.3.1 Ukládání souboru



Obrázek 6.3: Stavový diagram pro ukládání souborů z grafického editoru.

Výchozím stavem, ze kterého se volají ostatní, je stav *ready*. V něm je stavový automat většinu svého času. Ke každému tlačítku vizualizace je přiřazena proměnná typu *BOOL*, která slouží jako příznak. Všechny příznaky tlačítek jsou vyhodnocovány právě ve stavu *ready*.

Jeli stisknuto tlačítko „Zapsat do souboru“, je aktivován příznak *CreateF_btn*. Po vyhodnocení se přejde do stavu *save_file*. V něm je zobrazen dialog (viz obr. 5.3/str. 28), kde uživatel může zadat název souboru. Může stisknout tlačítko „Uložit“ (příznak *NewFileSave_btn*) nebo „Storno“ (příznak *NewFileCancel_btn*).

Při stisku „Storno“ se přejde do stavu *none*. Tento stav zakončuje každou stavovou sekvenci. V něm jsou všechny příznaky stisku tlačítek, všechny FB (vyjma časovačů) a příznaky otevření dialogů deaktivovány. Ze stavu *none* se vždy přejde do stavu *ready*.

Pokud je stisknuto tlačítko „Uložit“, přejde se do stavu *saving_file*, kde je soubor vytvořen. Pokud již soubor existuje, přejde se do stavu *rewrite_file*, který vyvolá

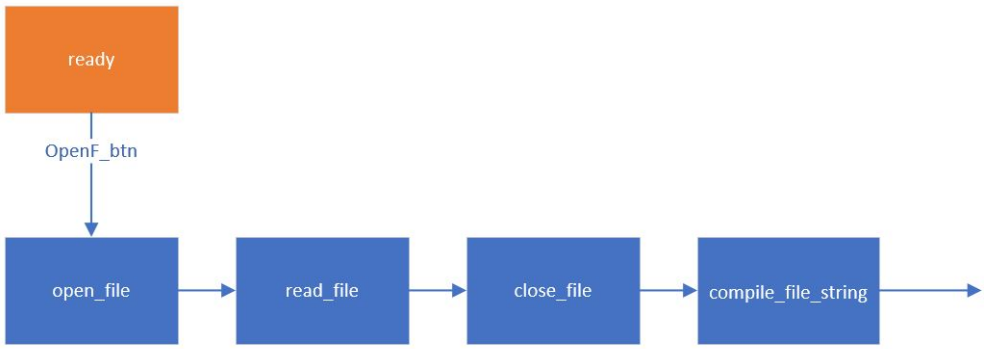
dialog s možností nahrazení souboru (viz obr. 5.6/str. 29). Souhlasí-li uživatel s přepsáním, existující soubor je ve stavu *delete_file* smazán a následně znovu vytvořen ve stavu *saving_file*. Nesouhlasí-li uživatel s nahrazením souboru, stavový automat se vrátí do stavu *save_file*, kde je opět vyvolán dialog vyžadující název souboru.

Neexistuje-li soubor se stejným názvem, přejde se ze stavu *saving_file* do stavu *write_file*, kde se do vytvořeného souboru zapíše textový řetězec příkazů pohybové sekvence robota. Soubor je následně ve stavu *close_file* uzavřen a přechází se do stavu *none*.

Tento stavový automat je výrazně zjednodušen. Ve skutečnosti se v každém stavu, ve kterém se využívají FB pro práci se soubory, vyhodnocuje jejich výstupní hodnota. Pokud nastane chyba, ukládací sekvence stavů je ukončena a chyba zapsána do proměnné.

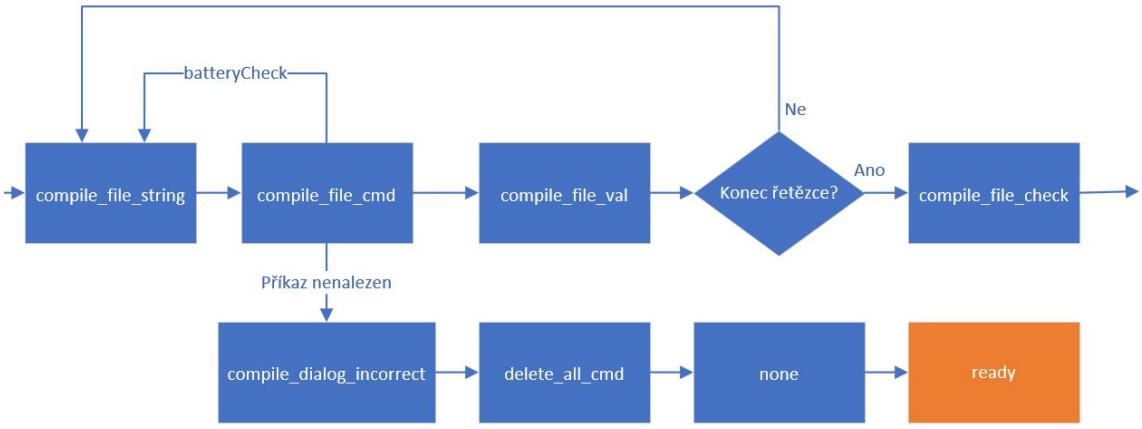
6.3.2 Kompilace souboru

Kompilace je vyvolána stiskem tlačítka „Kompilovat“ v boxu textového editoru. Tím se převedou příkazy v podobě řetězce do struktury v grafickém editoru.



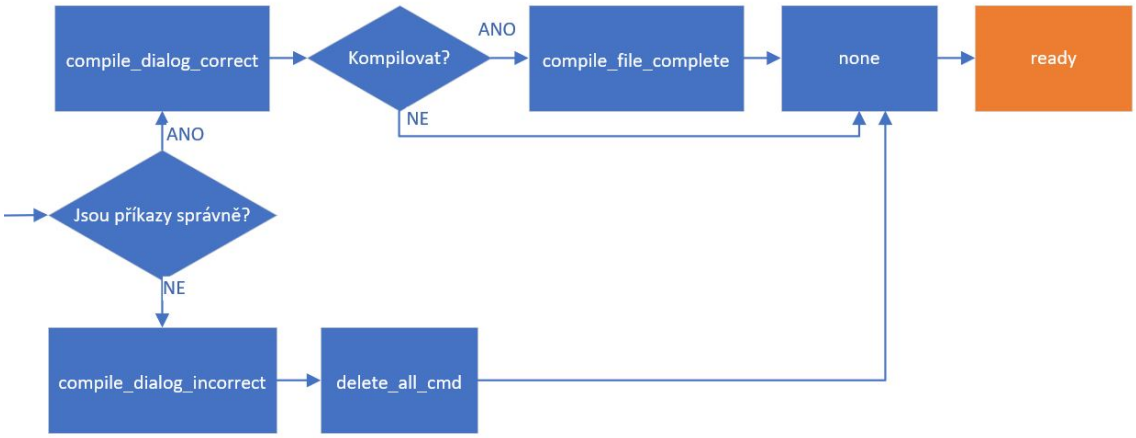
Obrázek 6.4: Stavový diagram načtení textového řetězce ze souboru.

Nejprve se po aktivaci příznaku *OpenF_btn*, signalizující stisk tlačítka pro kompilaci, načte textový řetězec ze souboru. Obsah souboru se uloží do proměnné *Text* typu *STRING [1600]*. Po uzavření souboru se přejde do stavu *compile_file_string*.



Obrázek 6.5: Stavový diagram kompilace textového řetězce.

Ve stavu *compile_file_string* se postupně prochází řetězec v proměnné *Text*. Přeskočí se prázdné znaky a do proměnné *String_cmd* se uloží řetězec, zakončený mezerou nebo novým řádkem. Následně se řetězec *String_cmd* porovná se známou sadou příkazů. Pokud není nalezena shoda, kompilace se ukončí. Narazí-li se na příkaz *batteryCheck*, který nemá číselný argument, uloží se, přejde se zpět do stavu *compile_file_string* a program pokračuje v kompilaci. Má-li nalezený příkaz číselnou hodnotu, přejde se do stavu *compile_file_val*, kde je podle typu příkazu převeden potřebný počet číselných argumentů. V tomto stavu je kontrolováno, zdali se nenachází index pole *Text* u konce jeho délky. Pokud ano, přechází se do stavu *compile_file_check*.



Obrázek 6.6: Stavový diagram kontroly příkazů.

V této fázi je kontrolováno, zdali mají příkazy správné číselné hodnoty. Narazí-li se na neplatnou hodnotu, zobrazí se ve stavu *compile_dialog_incorrect* dialog, informující o typu chyby (viz obr. 5.19/str. 36), a zkompilovaný seznam se následně

smaže. Pokud je vše korektní, zobrazí se ve stavu *compile_dialog_correct* dialog (viz obr. 5.18/str. 35), kde uživatel souhlasí s kompilací.

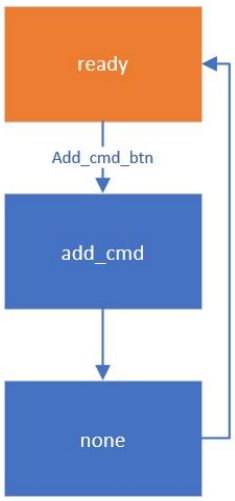
6.4 Práce s příkazy

V grafickém editoru je možné s příkazy pracovat přímo. Je možné je přidávat, dodatečně upravovat a mazat z boxu Sekvence. Všechny příkazy jsou uchovávány v poli struktury *Cmd_list[0..50]*, která obsahuje proměnné *cmd*, *val* a *val2*. Pole obsahuje celkem 51 prvků, ale počet příkazů je omezen na 50. Důvodem je, aby byl poslední, tedy 50. prvek, vždy v defaultní hodnotě (tj. *cmd* = *nop*; *val* = *0*; *val1* = *0*). To umožňuje, aby při mazání příkazů bylo využito jednoduchého posunu celého pole a tím i zachováno jejich pořadí. Proměnná *cmd* je výčtového typu, který definuje druh příkazu.

Další důležité pole proměnných pro grafický editor jsou pole *USINT GroupBox_Style[0...9]* a *Cmd_Style[0..50]* – definující barvu příkazu, *BOOL GroupBox_visible[0..9]* – ovládající viditelnost položky příkazu ve vizualizaci, *USINT Cmd_number[0..9]* – definující zobrazené číslo příkazů a *STRING Cmd_string_GB[40][0..9]* a *Cmd_string[40][0..50]* – uchovávající zobrazený text příkazu. Každý příkaz má těmito poli jednoznačně definovaný typ, hodnotu, pozici a barvu při zvýraznění. Proměnných pro chod grafického editoru je samozřejmě mnohem víc, ale pro základní představu práce s příkazy tento výčet stačí.

6.4.1 Přidání příkazu

Pro přidání nového příkazu po stisku tlačítka „Add“ (viz obr. 5.7/str. 29) slouží následující triviální sled stavů.

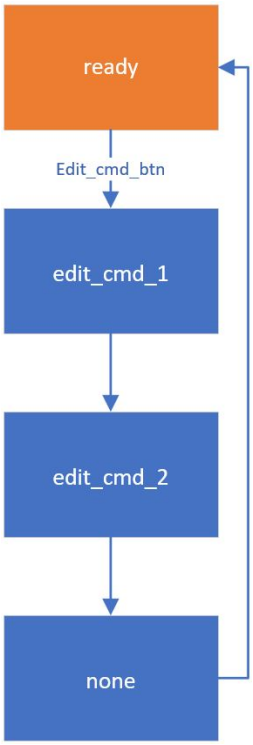


Obrázek 6.7: Stavový diagram přidání příkazu.

Ve stavu *add_cmd* se zkopíruje typ a hodnota příkazu, která je nastavena uživatelem ve vizualizaci, do první volné pozice v listu. Inkrementuje se celkový počet příkazu a nastaví se příslušná viditelnost položek v listu.

6.4.2 Editace příkazu

Při stisku tlačítka „Vybrat“ se vykoná následující sled stavů.



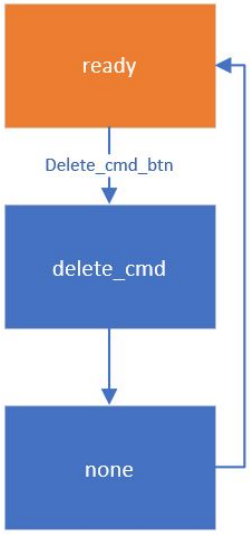
Obrázek 6.8: Stavový diagram editace příkazu.

Stav *edit_cmd_1* zabezpečí, že se nástrojový panel přenastaví do podoby editace příkazu a vybraný příkaz je zvýrazněn žlutě. Hodnoty v nástrojovém panelu jsou nastaveny na hodnoty vybraného příkazu.

Ve stavu *edit_cmd_2* se obsluhuje stisk tlačítek „Upravit“, „Přidat před“ a „Hotovo“. Při stisku tlačítka „Upravit“ se zkopírují hodnoty příkazu v nástrojovém panelu na pozici vybraného a ukončí se režim editace. Při stisku tlačítka „Přidat před“ se mechanicky posune pole listu před vybraným příkazem o index výše a na vzniklou volnou pozici se zkopírují údaje z nástrojového panelu. Režim editace je zachován na stávajícím příkazu a může se tedy pokračovat v jeho editaci. Stiskem tlačítka „Hotovo“ se režim editace ukončí.

6.4.3 Smazání příkazu

Při stisku tlačítka „Smazat“, umístěného v položce příkazu, se příkaz na dané pozici vymaže. To vykonává stav *delete_cmd*, který podle indexu stisknutého tlačítka spočítá, jaký příkaz se má v listu vymazat. Pole za tímto příkazem se posune o index níž, čímž se příkaz smaže.



Obrázek 6.9: Stavový diagram smazání příkazu.

6.5 Vykonávání sekvence

Pokud chce uživatel spustit naprogramovanou sekvenci pohybu, stiskne tlačítko *Play* na spodní straně grafického editoru (viz obr. 5.10/str. 31). Tím se přejde do stavového automatu obsluhujícího vykonávání příkazů. Jeho zobrazení ve stavovém diagramu je pro svou složitost výrazně zjednodušeno.

V prvním stavu *execute_initial* se vytvoří kopie pole příkazů *Cmd_list* v proměnné *Temp_cmd_list*. Z té se poté vykonává pohybová sekvence a jsou v ní přepisovány hodnoty opakování příkazu *jump*. Tento jednoduchý koncept umožňuje existenci vnořených cyklů. Po dokončení nebo ukončení vykonávání sekvence zůstává původní list příkazů nezměněn. Vytvoří se též kopie pole *Cmd_string* do pole *Temp_cmd_string*. Při vykonávání sekvence je u příkazu *jump* zobrazen údaj o zbývajícím počtu opakování. To je řešeno tím, že se přepisuje zobrazovaná hodnota v poli *Cmd_string*. V poli *Temp_cmd_string* je uchován původní text příkazů, který je posléze načten zpět.

Ve stavu *execute_step* se zjišťuje, zdali je aktivováno tlačítko *Pause* a zvýrazňuje se aktivní příkaz na zeleno. Je-li stisknuto, přejde se do stavu *execute_pause*, pokud

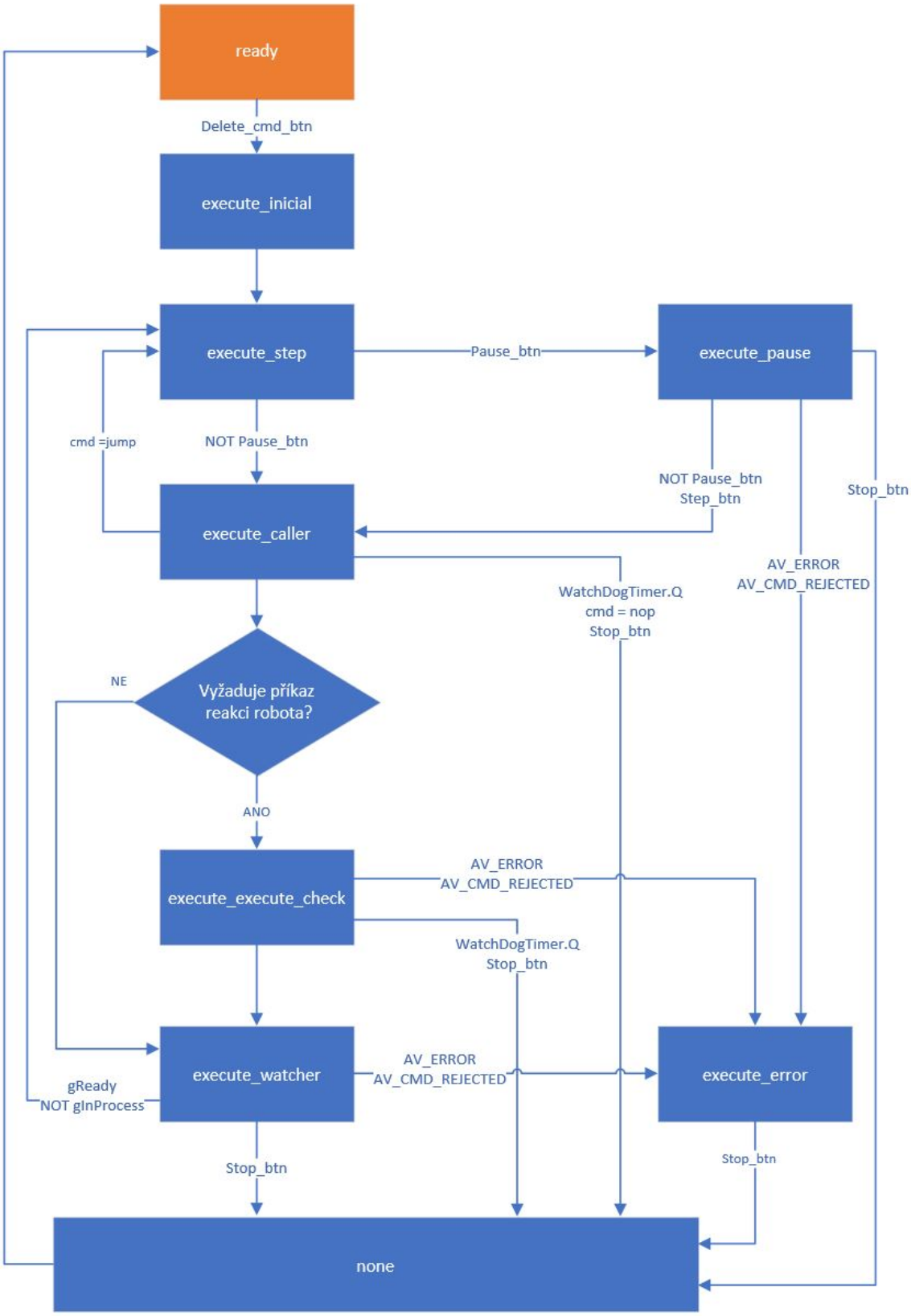
ne, přejde se do stavu *execute_caller*.

Ve stavu *execute_caller* se volají aktivní příkazy, tedy přesněji, odesílají požadavky na jejich vykonání. Zapíší se hodnoty jejich argumentů a aktivují se příslušné proměnné v globální struktuře, ze které je proces *AvatarCOM* registruje a následně vykonává. Pokud je právě vykonávaným příkazem příkaz *waiting*, aktivuje se pouze časovač s požadovaným časem a přechází se přímo do stavu *execute_watcher*, kde probíhá kontrola vykonání příkazů. Příkaz *jump* je zde rovnou obsloužen (tedy posunut index aktivního příkazu a dekrementována hodnota počtu opakování) a přechází se do stavu *execute_step*. Pokud má robot na příkaz nějak reagovat, přechází se do stavu *execute_execute_check*.

Pokud robot příkaz registruje a začal jej vykonávat, aktivuje příznak *gInProcess* (informuje a vykonávání nějaké úlohy) a deaktivuje *gReady* (informuje o připravenosti robota přijímat příkazy). Na to stav *execute_execute_check* reaguje přechodem do stavu *execute_watcher*. Jestliže robot do 5 s nezareaguje, vyhodnotí se to jako chyba, je zobrazeno informující okno (viz obr. 5.12/str. 31) a ukončí se vykonávání. To je možné ukončit i stisknutím tlačítka *Stop*.

Stav *execute_watcher* kontroluje, zda se požadovaný příkaz dokončil. Podmínka dokončení se u jednotlivých příkazů liší. Je-li aktivním příkazem příkaz *jump*, čeká se na dokončení časovače *Waiting_timer*. Pokud je příkazem *EbatteryCheck* a nabíjení je aktivováno (napětí pokleslo pod nastavenou mez), očekává se dosažení nastavené hodnoty napětí pro ukončení nabíjení. Ve chvíli, kdy je hodnoty dosaženo, odešle se požadavek na ukončení nabíjení a očekává se dokončení pohybu vyjetí z nabíječky. U ostatních příkazů je požadován aktivní příznak *gReady* a neaktivní *gInProcess*. Pokud je příkaz dokončen, pokračuje se do stavu *execute_step*. Nastane-li nějaká chyba, proces *AvatarCOM* spadne do stavu *AV_ERROR* (při neurčité poloze či chybě pohonů atp.) nebo *AV_CMD_REJECTED* (pokud je požadován pohyb na současnou polohu). Pokud je v nějakém z těchto dvou stavů, přejde se do stavu *execute_error* a položka v seznamu právě vykonávaného příkazu se zbarví červeně. Zde se pak vyhodnocuje stisk tlačítka *Stop* pro ukončení. Ve stavu *execute_watcher* je možné seznamem rolovat pomocí šipek nebo mít aktivováno tlačítko *Track* (viz obr. 5.11/str. 31), které posouvá seznam automaticky.

Pokud je stisknuto tlačítko *Pause*, stav *execute_step* přechází do stavu *execute_pause*. V tomto stavu je vykonávání pozastaveno a čeká se na reakci uživatele, uplynulý čas však pozastaven není. Pozastavený příkaz je zvýrazněn oranžově (viz obr. 5.15/str. 32). Uživatel může stisknout tlačítko *Step* a tím vykonat pozastavený příkaz, muže tlačítko *Pause* deaktivovat a tím opustit stav *execute_pause* a pokračovat ve vykonávání sekvence, nebo vykonávání ukončit stiskem tlačítka *Stop*. I ve stavu *execute_pause* je kontrolován stav robota a tím registrována jeho možná chyba.



Obrázek 6.10: Stavový diagram vykonávání sekvence.

6.6 Ostatní stavy

Mezi nezařazené stavové funkce patří rolování seznamu ve stavu *roll_list*, který je vykonán při posouvání seznamu příkazů pomocí šipek. Podle požadovaného směru rolování mění index prvního zobrazeného příkazu. Zobrazovaný seznam je aktualizován akcí *ListSyn*.

Dále je zde stav pro vymazání všech příkazů *delete_all_cmd*. Ten přepíše všechny prvky pole *Cmd_list* na defaultní hodnoty (tj. *cmd = nop, val = 0* a *val2 = 0*) a zneviditelní zobrazené prvky boxu „Sekvence“ na straně vizualizace.

Patří sem i v každém diagramu uváděné stavy *none* a *ready*. Jak už bylo jednou zmíněno, ve stavu *none* se deaktivují všechny booleovské příznaky stisku tlačítek. Uživatelem stisknuté tlačítko během vykonávání některé funkce je tedy po jejím vykonání ignorováno. To vylučuje např. dvojité stisky nebo akumulaci požadavků. Jsou zde také deaktivovány požadavky na zobrazení dialogů a message boxů, a zastaven časovač *StopWatch_timer*, měřící dobu vykonávání sekvence. Všechny FB jsou deaktivovány a následně volány. Je zde také aktualizován list příkazů akcí *ListSyn* a viditelnost položek nástrojového panelu akcí *PropertySet*.

Stav *ready* obsahuje rozvětvenou podmínku, ošetřující stisk tlačítek. Uživatel kliknutím na tlačítko aktivuje událost (event) vizualizace, která nastaví booleovskou proměnnou na OPC UA serveru. Zde jsou tyto proměnné vyhodnocovány. Stisk každého tlačítka vyvolá nějaký sled z výše uvedených stavů stavového automatu.

7 Závěr

Všechny cíle, jež si tato diplomová práce kladla, byly splněny. Byl vytvořen software, který umožňuje servisnímu technikovi ovládat robota pomocí pohybových sekvencí. Tyto sekvence lze tvořit jak z vytvořeného rozhraní vizualizace, tak i kdekoli jinde pomocí sady příkazů jednoduchého programovacího jazyka. Stačí napsanou sekvenci vložit do textového editoru vizualizace a po její kompilaci jí jednoduše spustit, nebo s ní dále pracovat. K vytvořenému HMI byl napsán manuál, který uživatele seznámí s funkcionalitou Servisního testeru. Manuál je uveden v příloze diplomové práce.

Při tvorbě vizualizace Servisního testeru byl kladen důraz na jednoduchost a názornost. Tvorba vizualizace byla v některých ohledech omezena možnostmi vývojového prostředí AS a jeho komponenty mapp View. Všechny limitace však šly do jisté míry obejít a jejich vliv nakonec úroveň výsledku nijak nesnížil. Například příkazy, přidávané do listu grafického editoru, budí dojem plně dynamického vytváření prvků, což ovšem vizualizace nepodporuje. S trochou vynaloženého úsilí se však podařilo vytoužené podoby chování dosáhnout. Neustálé ukládání souborů při práci v textovém editoru působí trochu neohrabaně. Je to však nezbytné, protože obsah grafického prvku TextPad nelze získat za daných okolností jiným způsobem. Soubory se při laxnosti uživatele mohou v paměti PLC kupit a smazat je lze pouze z FTP serveru.

Pro tvorbu pohybové sekvence přímo z rozhraní robota byl vytvořen grafický editor, ve kterém jde snadno a intuitivně vytvořit požadovanou trajektorii. Příkazy jsou přehledně zobrazeny v seznamu, ve kterém s nimi lze dále pracovat. Velmi dobrým dojmem působí barevné zvýraznění příkazů, které uživatele elegantně informují o tom, co se s daným příkazem děje. Vytvořený textový editor spolu s archivací souborů umožňuje servisnímu technikovi snadnou a rychlou tvorbu trajektorií.

Navržený jednoduchý programovací jazyk, popisující pohybové sekvence, umožňuje uživateli vytvořit libovolnou trajektorii. Díky vnořeným cyklům a možnosti automaticky nabít robota při poklesu pod definovanou úroveň napětí baterie, může být robot testován takřka neomezenou dobu.

Vše se dá zlepšovat a vyvíjet, a tato práce není výjimkou. Šlo by například trajektorii pohybu graficky zobrazovat a předpočítávat její délku. Vhodné by bylo též implementovat kontrolu nastavené cílové pozice tak, aby odpovídala skladovému

rastru. V současné době je v úvahu také rozšíření o jakousi černou skříňku, která by zapisovala vykonané příkazy robota a ukládala je pro možnost jejich pozdější analýzy. To by v případě, kdyby software plánovače způsobil nějakou nehodu, umožnilo zpětně analyzovat jeho chování a sjednat případnou softwarovou opravu.

S výsledkem práce jsem spokojený. Veškeré požadavky se podařilo splnit a funkcionalita v rozhraní robota působí dobrým dojmem. Věřím, že svým přínosem pomůže urychlit vývoj robota, a doufám, že se nejednou bude servisnímu technikovi hodit.

Literatura

[1] Zásadní problémy ve skladové logistice a jak z nich ven. Logistická akademie [online]. Ostrava-Poruba: LOGISTICKÁ AKADEMIE, 2017 [cit. 2020-04-01]. Dostupné z: https://www.logistickaakademie.cz/blog/diskutovana-temata/zasadni-problemy-ve-skladove-logistice-a-jak-z-nic

[2] Kardex Remstar Solutions Solve Warehouse Overcrowding. Kardex-remstar [online]. Praha: Kardex [cit. 2020-04-01]. Dostupné z: https://www.kardex-remstar.cz/cz/novinky-cz/news-detail/article/kardex-remstar-solutions-solve-warehouse-overcrowding.html

[3] Stacker cranes for miniload warehouses. Jungheinrich [online]. Jungheinrich, 2020 [cit. 2020-04-01]. Dostupné z: https://www.jungheinrich.cn/en/systems/automated-storage-and-retrieval-systems/automatic-small-parts-storage-1-/stacker-cranes-for-miniload-warehouses

[4] System. AutoStore [online]. AutoStore, 2020 [cit. 2020-04-01]. Dostupné z: https://autostoresystem.com/system/

[5] Robotický sklad s nižší obrátkou. Logistika [online]. Economia, 2017 [cit. 2020-04-01]. Dostupné z: https://logistika.ihned.cz/c1-65955110-roboticky-sklad-s-nizsi-obratkou

[6] Systematic: Archivace dokumentů [online]. Praha: Systematic [cit. 2020-04-01]. Dostupné z: https://www.systematic.cz/

[7] Mapp Technology. B&R Industrial Automation [online]. Roswell: B&R Headquarters, 2020 [cit. 2020-04-01]. Dostupné z: https://www.br-automation.com/en-us/products/software/mapp-technology/

[8] COMPANY B&R. TM600: Introduction to Visualization. 2016/10/10.

[9] COMPANY B&R. TM611: Working with mapp View. 2017/07/17.

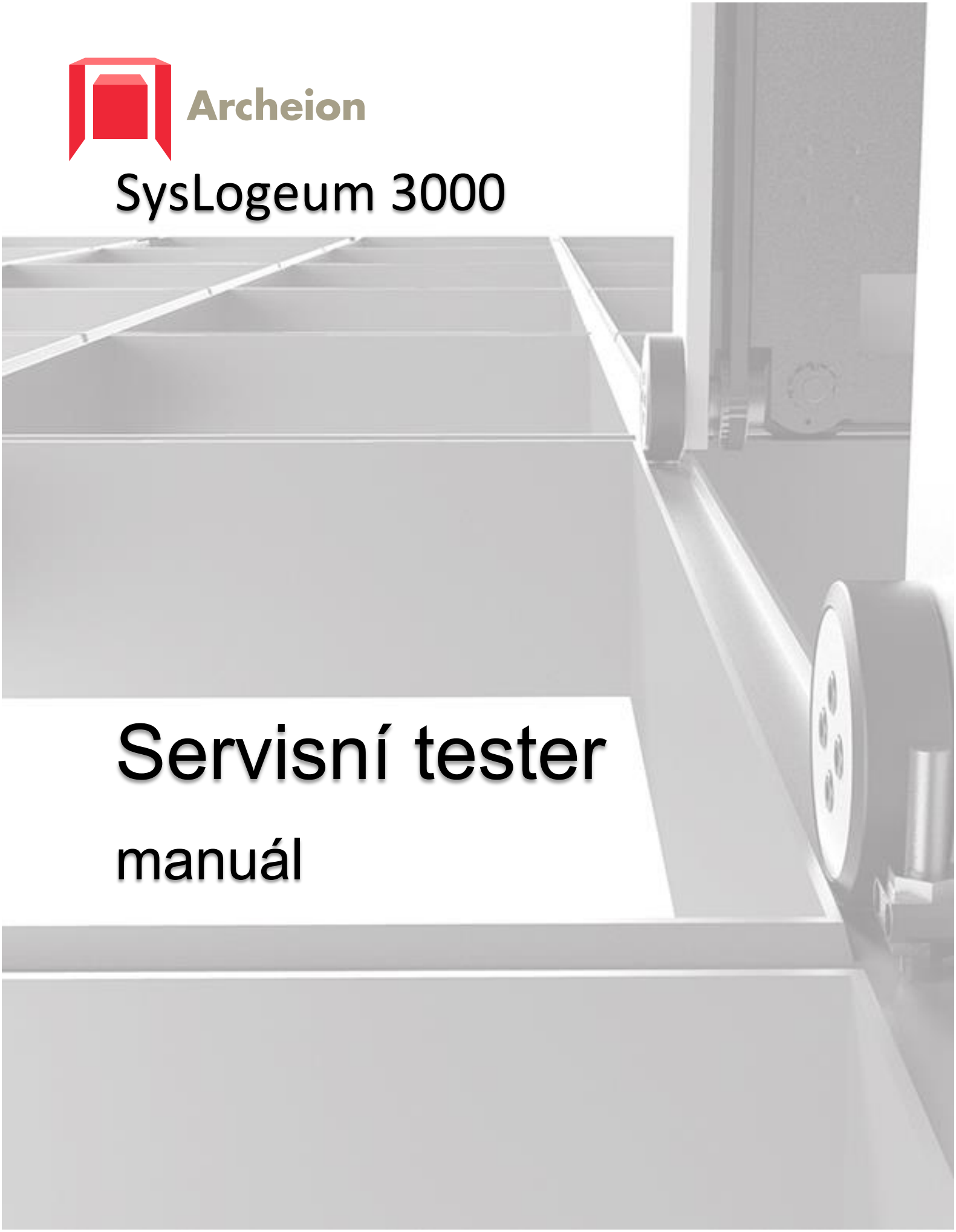
[10] ŠMEJKAL, Ladislav a Marie MARTINÁSKOVÁ. PLC a automatizace. 1. díl. Praha: BEN, 1999. ISBN 80-8605- 58-9

[11] JOHN, Kharl-Heinz; TIEGELKAMP, Michael. IEC 61131-3 Programming Industrial Automation Systems : Concepts and Programming Languages, Requirements for Programming Systems, Decision – Making Aids. 2nd

A Servisní tester - manuál

Archeion

SysLogeum 3000



Servisní tester

manuál

Obsah

Základní informace

Servisní tester je nástroj, určený pro testování funkcí robota. Umožňuje vytvářet a editovat trajektorie (pohybové sekvence), které má robot vykonávat.

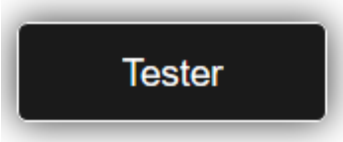
Vytvořené trajektorie lze posléze libovolně spouštět a krokovat.

Oprávnění a přístup

Rozhraní servisního testeru se nachází ve vizualizaci řídícího systému robota. Ta je přístupná pod IP adresou daného stroje na portu 81.

Odkaz pro přístup na stránku servisního testeru se nachází na dolní navigační liště.

Stránka se servisním testerem je přístupná pod navigačním tlačítkem „***Tester***“ (Obr. 1).



Obr. 1 Tlačítko pro vstup do servisního testeru

Přístup k rozhraní servisního testeru mají pouze uživatelé s rolí „***service***“ a „***supervisor***“.

Bez patřičného oprávnění je navigační tlačítko neaktivní (Obr. 2).

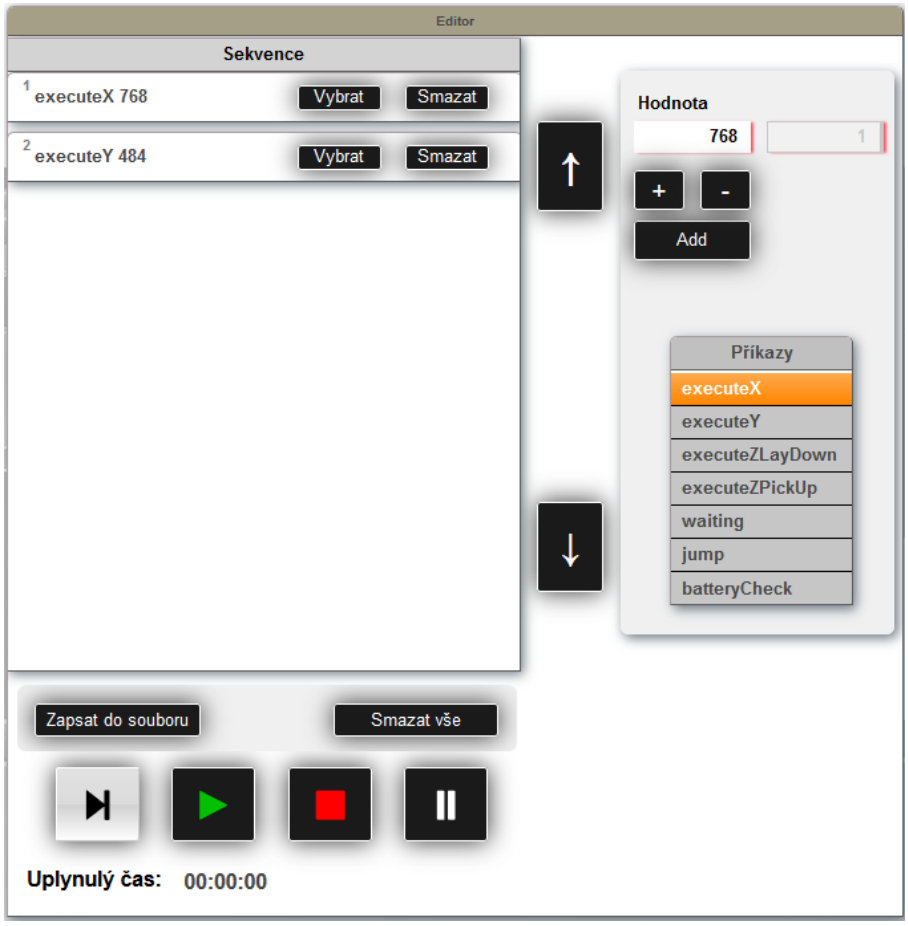
Obr. 2 Navigační tlačítko při nedostatečném oprávnění

Popis rozhrání

Rozhrání servisního editoru obsahuje:

- **Grafický editor**
- **Textový editor**
- **Ovládání Avataru**
- **Procesní proměnné**

Grafický editor



Obr. 3 Grafický editor

Grafický editor (Obr. 3) je základním nástrojem servisního testeru. Jeho pomocí lze vytvářet a editovat aktuální sekvenci příkazů. Dále pak lze tuto sekvenci spustit, pozastavit, zastavit či krokovat.

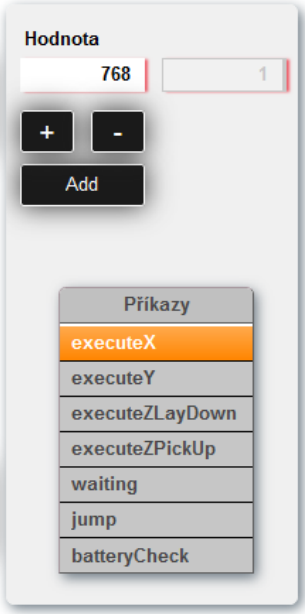
Aktuální sekvence příkazů je zobrazena v boxu „*Sekvence*“. Je-li příkazů více, než je možné zobrazit v seznamu, lze seznam posouvat šipkami (Obr. 4).

Maximálně může sekvence obsahovat 50 příkazů.

Obr. 4 Rolovací šipky

Práce s příkazy

Příkaz se přidávají a editují z nástrojového panelu (Obr. 5).



Na spodní straně panelu je seznam dostupných příkazů. Po kliknutí na požadovaný příkaz jsou na horní straně zobrazeny hodnoty parametrů. Příkaz může mít jednu, dvě, anebo žádnou číselnou hodnotu. Příkazy „*executeX*“ a „*executeY*“ mají navíc tlačítka pro inkrement skladového rastru (tzv. grid).

Po stisku tlačítka „*Add*“ je příkaz přidán do sekvence.

Obr. 5 Nástrojový panel

Hodnota příkazu musí splňovat určité podmínky, jinak příkaz nelze přidat. Je-li hodnota příkazu zadána špatně, je zobrazeno chybové hlášení.

Může se jednat o tyto chyby:

- Hodnota je záporná
- Cíl skoku příkazu „*jump*“ odkazuje na neexistující příkaz

V boxu „***Sekvence***“ je zobrazen aktuální sled příkazů. Každý příkaz má vlastní položku s ovládacími prvky (Obr. 6).

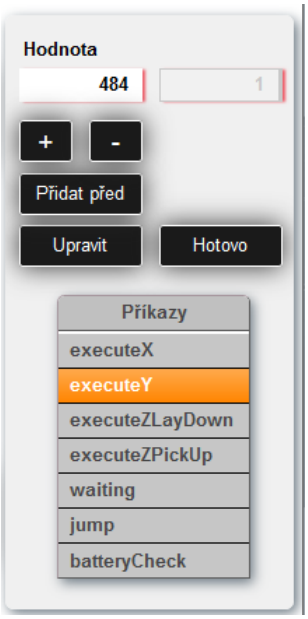
Obr. 6 Příkaz v listu

V levém horním rohu je číslo pořadí příkazu, což je důležitý údaj pro cíl skoku příkazu „***jump***“. Dále je zobrazen název příkazu a číselná hodnota (pokud je přítomna). Každý příkaz je možné vybrat (stiskem tlačítka „*Vybrat*“) a smazat (stiskem tlačítka „*Smazat*“).

Je-li příkaz vybrán, zvýrazní se žlutě (Obr. 7).

Obr. 7 Zvýraznění vybraného příkazu

Poté je možné z nástrojového panelu s příkazem dále pracovat (Obr. 8).



Automaticky je v seznamu vybrán zvolený příkaz a vyplněna jeho aktuální hodnota.

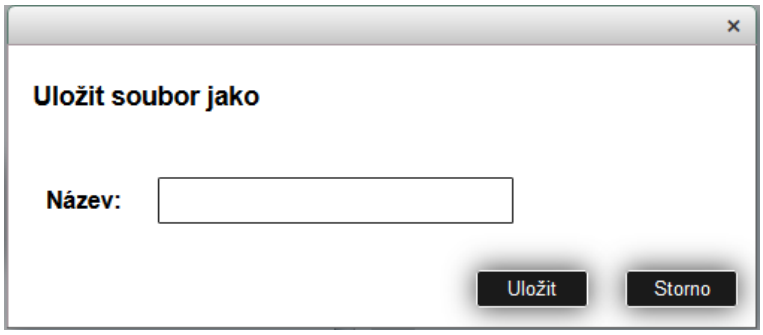
Provedené úpravy příkazu lze aplikovat pomocí tlačítka „*Upravit*“. Nejsou-li požadovány žádné úpravy, editace příkazu se ukončí tlačítkem „*Hotovo*“.

Před vybraný příkaz je možné vložit další pomocí tlačítka „*Přidat před*“.

K uložení aktuální sekvence příkazů do souboru slouží tlačítko „*Zapsat do souboru*“ (Obr. 3). Po jeho stisku je zobrazen dialog pro zadání názvu souboru (Obr. 9).

Obr. 8 Nástrojový panel při editaci

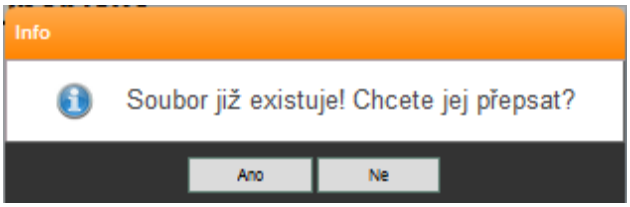
Název je zadáván bez přípony typu souboru. Ta je přidána automaticky na „.txt“.



Obr. 9 Ukládací dialog

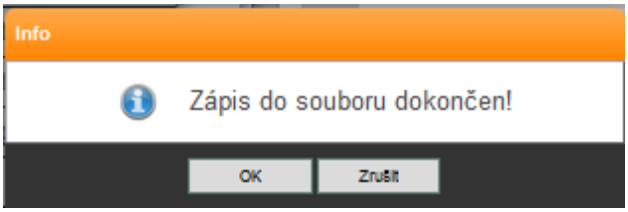
Soubory jsou ukládány na vnitřní úložiště PLC do adresáře „/Sequencer“.

Existuje-li již soubor se zadaným názvem, může jej uživatel přepsat (Obr. 10).



Obr. 10 Dialog při existenci souboru stejného názvu

Po úspěšném dokončení ukládání je zobrazen informující dialog (Obr. 11).



Obr. 11 Potvrzení zápisu do souboru

List sekvence příkazů lze vymazat stiskem tlačítka „*Smazat vše*“ (Obr. 3).

Seznam příkazů

Sekvence může být složena z následujících příkazů. Každý příkaz má jiný parametr.

Obr. 12 Seznam příkazů

executeX **–** příkaz pro pohyb v ose X, má parametr cílové polohy v mm; hodnota nesmí být záporná

executeY **–** příkaz pro pohyb v ose Y, má parametr cílové polohy v mm; hodnota nesmí být záporná

executeZLayDown **–** příkaz pro položení uchopené krabice, má parametr hloubky místa položení v mm; hodnota nesmí být záporná

executeZPickUP **–** příkaz pro uchopení krabice, má hodnotu hloubky krabice v mm; hodnota nesmí být záporná

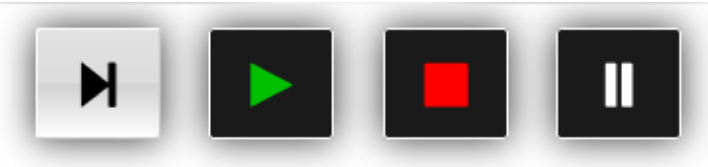
waiting **–** příkaz pro vložení časového zpoždění, má hodnotu v ms; hodnota nesmí být záporná

jump **–** příkaz skoku na jiný příkaz, má hodnotu „*Cíl*“ a „*Opakování*“; hodnota nesmí být záporná, hodnota parametru „*Cíl*“ musí odkazovat na existující příkaz, hodnota parametru „*Opakování*“ musí být kladná a nenulová

batteryCheck **–** příkaz pro kontrolu stavu baterie, bez parametru; hodnota napětí, kdy se aktivuje a ukončí nabíjení, je definována v boxu „Informace o baterii“ (Obr. 26).

Spuštění a ovládání sekvence.

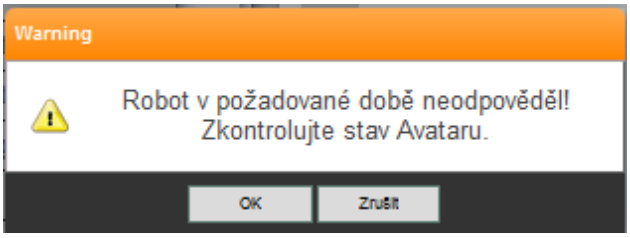
V dolní části grafického editoru jsou ovládací tlačítka (Obr. 13).



Obr. 13 Ovládání běhu programu

Po stisku tlačítka „*Play*“ (na 2. pozici, Obr. 13) je aktuální sekvence příkazů spuštěna a robot jí začne vykonávat.

Je nutné, aby robot byl v provozním stavu. Pokud z nějakého důvodu robot neodpoví, objeví se chybové hlášení (Obr. 14).



Obr. 14 Chybové hlášení při nereagování robota

Aktuálně vykonávaný příkaz je v seznamu zvýrazněn zeleně (Obr. 15).

Obr. 15 Zvýraznění vykonávaného příkazu

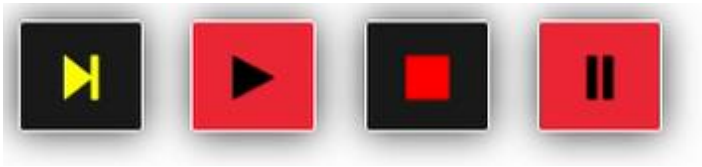
Během vykonávání programu je vedle boxu „Sekvence“ zobrazeno tlačítko (Obr. 16) pro sledování aktuálně vykonávaného příkazu. Je-li sekvence příkazů delší, než je možné najednou zobrazit, je při aktivování tlačítka seznam automaticky rolován tak, aby byl právě vykonávaný příkaz vždy viditelný.

Obr. 16 Tlačítko pro automatické rolování seznamu.

Vykonávání sekvence je možné pozastavit stiskem tlačítka „*Pause*“ (na 4. pozici, Obr. 13). Po stisku je dokončen právě vykonávaný příkaz a vykonávání následujícího příkazu je pozastaveno. Takto pozastavený příkaz je označen oranžově (Obr. 17).

Obr. 17 Zvýraznění pozastaveného příkazu

Po stisku tlačítka „*Pause*“ je aktivní tlačítko „*Step*“ (na 1. pozici, Obr. 18). Jeho stiskem je vykonán aktuálně pozastavený příkaz.



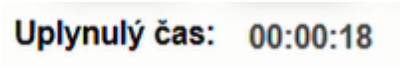
Obr. 18 Ovládací tlačítka při pozastavení programu

Stiskem tlačítka „*Stop*“ (na 3. pozici, Obr. 18) je vykonávání zastaveno a je možné sekvenci znovu editovat.

V případě chyby je příkaz, který chybu způsobil, zvýrazněn červeně a program dále nepokračuje (Obr. 19).

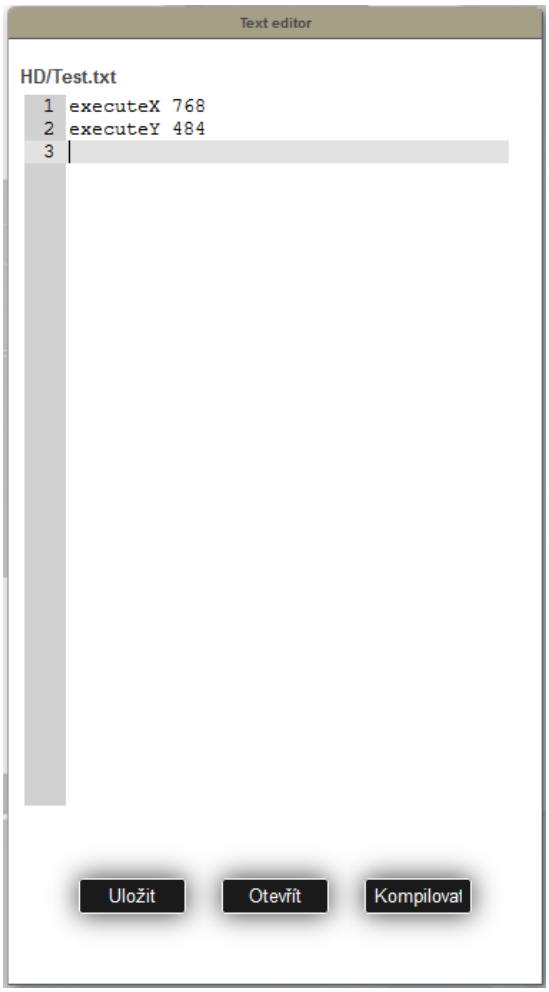
Obr. 19 Zvýraznění příkazu při chybě

Čas testu je měřen a zobrazen na levé spodní straně grafického editoru (Obr. 20). Hodnota je zachována i po zastavení či dokončení testu.



Obr. 20 Údaj o uplynulém času testu

Textový editor



Obr. 21 Textový editor

Textový editor (Obr. 21) slouží pro ruční psaní sekvence příkazů.

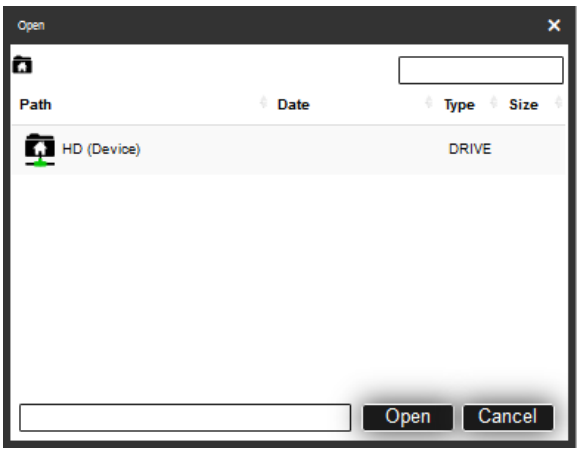
Pro kompilaci textové sekvence slouží tlačítko „*Kompilovat*“.

Tlačítko „*Kompilovat*“ je aktivní pouze tehdy, je-li soubor uložen.

Soubor se ukládá stiskem tlačítka „*Uložit*“. Následně je zobrazen souborový průzkumník.

Uložené sekvence je možné znovu otevřít stiskem tlačítka „*Otevřít*“. Poté je zobrazen souborový průzkumník.

V souborovém průzkumníku je dostupná pouze jedna položka s názvem „*HD*“ (Obr. 22). Jedná se o interní úložiště v paměti PLC v adresáři „*/Sequencer*“.

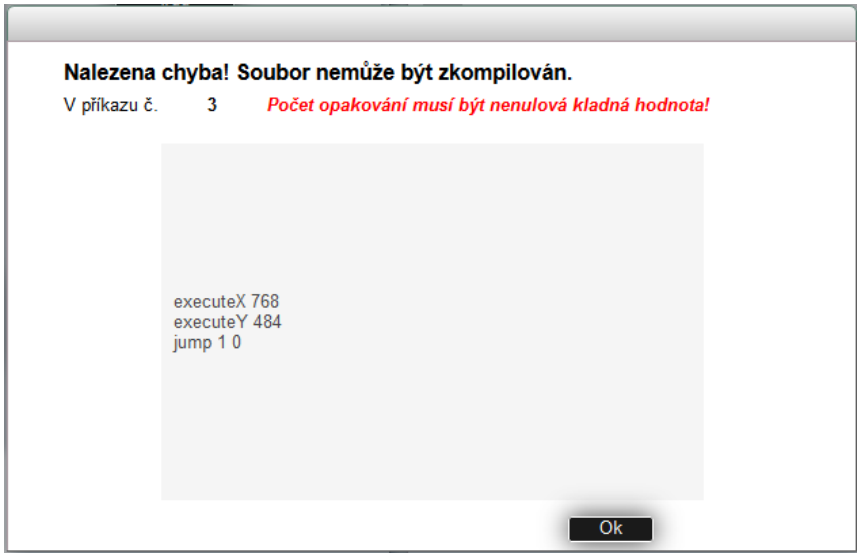


Obr. 22 Souborový průzkumník

Při kompilaci textové sekvence příkazů se mohou vyskytnout následující chyby:

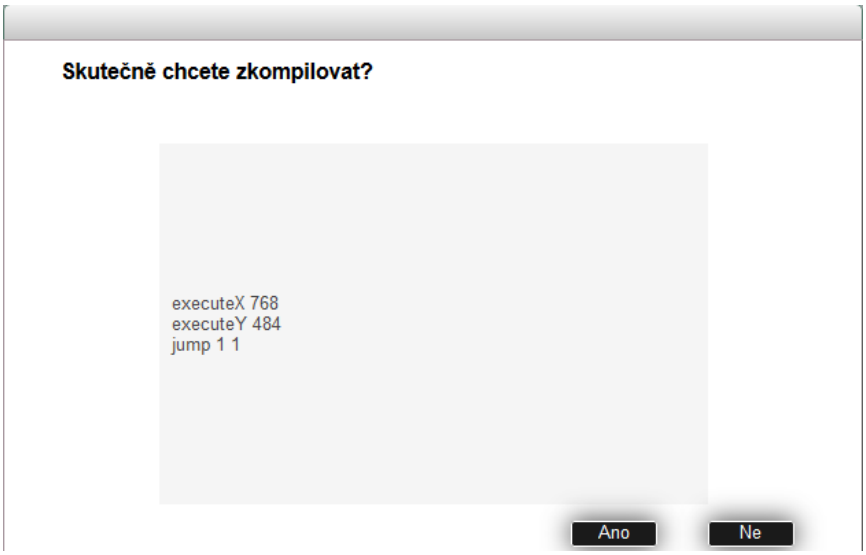
- Neznámý příkaz
- Hodnota příkazu je záporná
- Cíl příkazu „*jump*“ odkazuje na neexistující příkaz
- Počet opakování příkazu „*jump*“ je záporné číslo nebo nula
- Bylo dosaženo maximálního počtu příkazů (tj. 50).

Po kompilaci je zobrazen dialog. Ten v případě chyby obsahuje informaci o jakou chybu se jedná spolu s číslem chybného příkazu (Obr. 23).



Obr. 23 Dialog při chybě během kompilace

Pokud je kompilace v pořádku, může uživatel potvrdit překlad (Obr. 24).

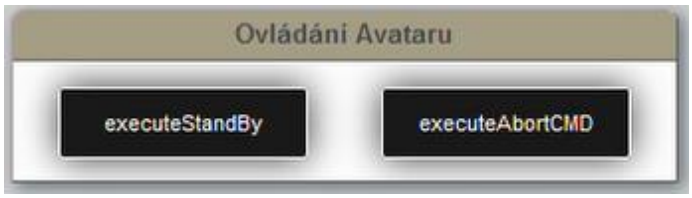


Obr. 24 Dialog po bezchybné kompilaci

Dialog zobrazuje příkazy po jejich překladu. V tuto chvíli je možné výsledek překladu zkontrolovat. Po úspěšné kompilaci je sekvence zobrazena v listu grafického editoru.

Ovládání Avataru

Na stránce servisního testeru je možné zadávat základní příkazy řídícímu systému robota, tzv. Avataru. Box s příkazy se nalézá v pravém horním rohu (Obr. 25).



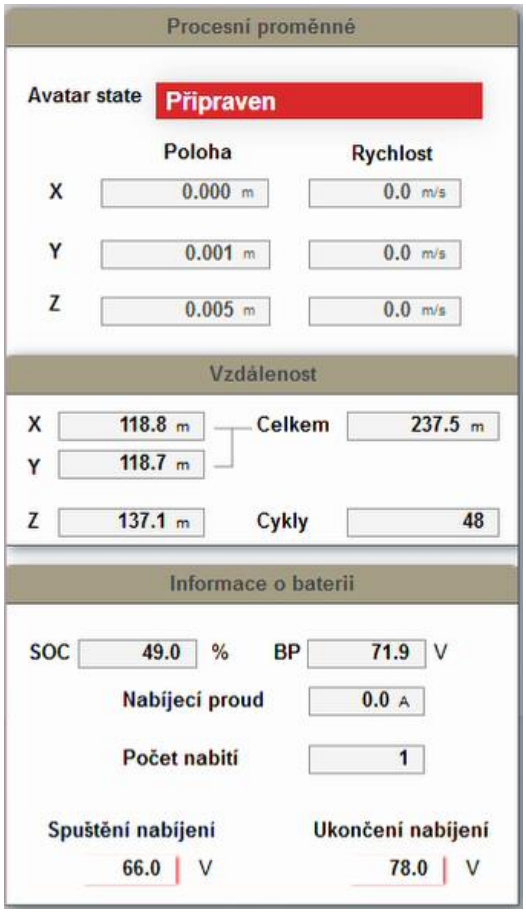
Obr. 25 Ovládání Avataru

Tlačítko „*executeStandBy*“ uvede robota do režimu „*Stand by“*. Pokud se v tomto režimu již nachází, lze ho stiskem tlačítka uvést do režimu „*Připraven*“.

Tlačítko „*executeAbortCMD*“ slouží k uvedení robota, při chybovém stavu „*Příkaz zamítnut*“, do stavu „*Připraven*“.

Procesní proměnné

Na pravé straně jsou v boxu „Procesní proměnné“ zobrazeny základní informace o robotovi (Obr. 26).



Obr. 26 Procesní proměnné

Je zde uvedena informace o aktuálním stavu robota v kolonce *„Avatar state“*.

Aktuální poloha os X, Y a Z a jejich rychlosti.

Informace o ujeté vzdálenosti jednotlivých os. Celková vzdálenost os X a Y.

Osa Z má zobrazen údaj o uskutečněném počtu cyklů (tj. zvednutí a položení).

Informace o baterii obsahuje údaj o nabití akumulátoru v % a aktuální napětí článku.

Nabíjecí proud v Ampérech a počet nájezdů na nabíječku.

Nakonec je možné nastavit úroveň napětí, při kterém se během příkazu *„batteryCheck“* spustí (*„Spuštění nabíjení“*) nebo ukončí (*„Ukončení nabíjení“*) nabíjení.