České vysoké učení technické v Praze
Fakulta strojní
Ústav mechaniky, biomechaniky a mechatroniky
Obor: Robotika a výrobní technika

Simulační model automatického chaotického skladu a jeho digitální dvojče

DIPLOMOVÁ PRÁCE

Vypracoval: Bc. Marek Vlček
Vedoucí práce: Ing. Martin Nečas, MSc., Ph.D.
Rok: 2023

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Vlček** Jméno: **Marek** Osobní číslo: **484110**

Fakulta/ústav: **Fakulta strojní**

Zadávající katedra/ústav: **Ústav mechaniky, biomechaniky a mechatroniky**

Studijní program: **Robotika a výrobní technika**

Specializace: **Robotika**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Simulační model automatického chaotického skladu a jeho digitální dvojče

Název diplomové práce anglicky:

Pokyny pro vypracování:

1. Seznamte se s problematikou simulačních modelů a problematikou tvorby digitálních dvojčat
2. Vytvořte strukturu popisu vnitřního stavu skladu a propojte ji s databází skladu
3. Vytvořte 3D vizualizační model v jednoduché a komplexní podobě v prostředí SimScape
4. Integrujte vizualizační model s řídícím algoritmem skladu, vytvořte digitální dvojče.
5. Kriticky zhodnoťe dosasažené výsledy s ohledem na budoucí využití.

Seznam doporučené literatury:

[1] GARCIA-ALFARO, Joaquin a Mariana SEGOVIA. Design, Modeling and Implementation of Digital Twins [online]. 20 July 2022 [cit. 2023-04-28]. Dostupné z: https://doi.org/10.3390/s22145396
[2] SARGENT, Robert. VERIFICATION AND VALIDATION OF SIMULATION MODELS [online]. January 2011 [cit. 2023-04-28]. Dostupné z: doi:10.1109/WSC.2010.5679166
[3] M. LAW, Averill. Simulation Modeling and Analysis. Fifth edition. Tucson, Arizona, USA: McGraw-Hill Education, 2015. ISBN 978-0-07-340132-4.

Jméno a pracoviště vedoucí(ho) diplomové práce:

Ing. Martin Nečas, MSc., Ph.D. odbor mechaniky a mechatroniky FS

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **18.10.2023** Termín odevzdání diplomové práce: **22.01.2024**

Platnost zadání diplomové práce: _____

Ing. Martin Nečas, MSc., Ph.D.	prof. Ing. Michael Valášek, DrSc.	doc. Ing. Miroslav Španiel, CSc.
podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Diplomant bere na vědomí, že je povinen vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

. _____ Datum převzetí zadání

_____ Podpis studenta

© ČVUT v Praze, Design: ČVUT v Praze, VIC

Prohlášení

Prohlašuji, že jsem svou diplomovou práci vypracoval samostatně a použil jsem pouze podklady (literaturu, projekty, SW atd.) uvedené v přiloženém seznamu.

V Praze dne .....

.....
Bc. Marek Vlček

Poděkování

Děkuji Ing. Martinu Nečasovi, MSc., Ph.D. za veškerý věnovaný čas, užitečné rady a návrhy. Dále bych chtěl poděkovat své rodině za trpělivost a velkou podporu během celého studia.

Bc. Marek Vlček

Název práce:
Simulační model automatického chaotického skladu a jeho digitální dvojče

Autor: Bc. Marek Vlček

Druh práce: Diplomová práce

Vedoucí práce: Ing. Martin Nečas, MSc., Ph.D.

Abstrakt: Tato práce se zabývá vytvořením simulačního modelu automatického chaotického skladu. V rámci práce je vytvořen popis vnitřního stavu skladu, který je propojen s databází, 3D vizualizační parametrický model s automatickým generováním desek, generátor trajektorií včetně řídicího algoritmu skladu a grafický interface ovládající celý simulační model.

Klíčová slova: Automatický chaotický sklad, Digitální dvojče, Simscape, Matlab, Parametrický model

Title:
Simulation model of an automatic chaotic warehouse and its digital twin

Author: Bc. Marek Vlček

Druh práce: Masters's thesis

Supervisor: Ing. Martin Nečas, MSc., Ph.D.

Abstract: This work deals with the creation of a simulation model of an automatic chaotic warehouse. Within the project, a description of the internal state of the warehouse is created, which is connected to a database. Additionally, there is a 3D visualization parametric model with automatic board generation, a trajectory generator including a control algorithm for the warehouse, and a graphical interface controlling the entire simulation model.

Key words: Automatic Chaotic Warehouse, Digital Twin, Simscape, Matlab, Parametric Model

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Logistika a skladování se v dnešním světě stávají klíčovými součástmi moderního průmyslu a je kladen čím dál větší důraz na efektivní způsob skladování, rychlost naskladnění a vyskladnění a také na on-line monitorování skladových zásob. Z toho důvodu jsou v průmyslu čím dál častěji využívány automatické sklady s různými typy skladování materiálu. Jedním typem je chaotické skladování, které má řadu výhod oproti klasickému způsobu skladování a je čím dál častěji využíváno v průmyslu, zejména kvůli optimalizaci dostupného prostoru.

Digitální dvojče představuje inovativní koncept, pomocí kterého je možné propojit fyzický svět a digitální technologie. V dnešním průmyslu jsou digitální dvojčata často nezbytnou součástí nových projektů, protože umožňují vizualizaci pracovních procesů, naprogramování a vyzkoušení daného systému v off-line režimu a také monitorování výrobních a technologických procesů v reálném čase.

Motivace

Motivací této práce je vytvoření simulačního modelu automatického chaotického skladu, který bude sloužit k otestování správné funkce řídicího systému.

Cíle práce

Vlastní práce se zabývá naplněním následujících cílů:

1. Seznámit se s problematikou simulačních modelů a problematikou tvorby digitálních dvojčat.
2. Vytvořit strukturu popisu vnitřního stavu skladu a propojit ji s databází skladu.
3. Vytvořit 3D vizualizační model v jednoduché a komplexní podobě v prostředí Simscape.
4. Integrovat vizualizační model s řídícím algoritmem skladu, vytvořit digitální dvojče.
5. Kriticky zhodnotit dosažené výsledky s ohledem na budoucí využití.

Kapitola 1

Automatické sklady

Automatické chaotické sklady jsou dnes v průmyslu nezbytnou součástí logistických řetězců. Ve velkém je využívá například Amazon. Oproti klasickému způsobu skladování, kdy každá položka má předem dané umístění, je chaotický způsob skladování založen na umístění zboží kamkoliv, kde je momentálně volné místo. Díky tomu lze dosáhnout efektivnějšího využití skladovacího prostoru.

1.1 Historie automatických skladů

První automatický skladovací systém AS/RS (Automated Storage/Retrieval System) vznikl v 50. letech 20. století v Německu a byl určen ke skladování a vyhledávání velkého objemu knih. Byl nainstalován v roce 1962 v nakladatelství Bertelsmann firmou Demag. Zatímco u klasického skladu vyžadovaly jednotlivé skladové pozice alokaci velkého množství prostoru, zejména kvůli omezenému dosahu obsluhy, tak u automatického skladu došlo k efektivnějšímu využití skladovacích ploch. [1, 2]

Následující desetiletí zůstal původní návrh firmy Demag v podstatě nezměněný. Technologie automatického skladu se globálně rozšířila a byla využívána především pro skladování v omezených prostorech s nutností nahradit drahou pracovní sílu. [1]

K vylepšení automatických skladů došlo až v 80. letech s příchodem lepší výpočetní techniky, která umožnila řešit náročnější úkoly ve skladování. V roce 1974 začly být využívány čárové kódy, což ulehčilo ukládání informací o skladovaném zboží. O rok později byl na trh uveden první systém pro správu skladu v reálném čase. [1, 2]

1.2 Princip chaotického skladování

Již název chaotické skladování odkazuje na způsob ukládání předmětů do skladu. Jedná se o neuspořádané, náhodné a časově proměnlivé skladování. V momentě vstupu zboží do skladu je mu přiřazena jedna z volných pozic a tato pozice je následně uložena do databáze obsahující další dostupné informace o zboží. [3]

U chaotického skladování je důležitý software, který je schopen reagovat na aktuální poptávku a dle toho zvolit vhodné dostupné místo. [4]

1.2.1 Výhody a nevýhody chaotického skladování

Hlavní výhodou chaotického způsobu skladování je **úspora logistických ploch**, s čímž souvisí snížení nákladů na skladování. U chaotického skladování využíváme veškerou volnou kapacitu skladu a také není nutné přidělovat každému výrobku konkrétní pozici. [5] Amazon tvrdí, že využitím chaotických skladů je společnost schopna skladovat dvakrát větší množství zboží než při užití klasického způsobu skladování. [4]

Rychlá dostupnost zboží je výhodou, ale zároveň také komplikací pro řídicí systém skladu. Volné pozice je vhodné volit s ohledem na kvantitu vyskladňování daného výrobku v čase. [5]

Jednoduchost chaotického skladování je další výhodou. Není nutné zaučovat nové zaměstnance, kde se nachází různé oddíly, protože skladování je náhodné. Z důvodu jednoduchosti také eliminujeme možnost smíchání různých druhů zboží, například z důvodu chybného manuálního vložení skladové položky. [3, 4]

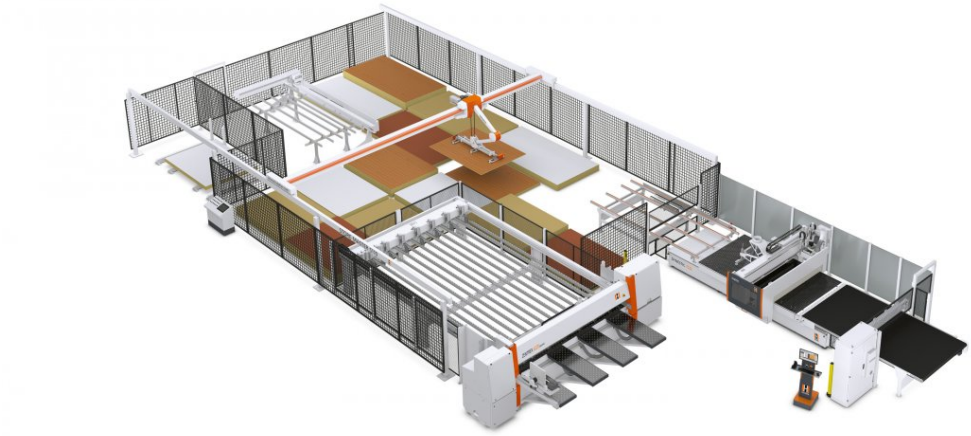
Mezi nevýhody patří **vysoká počáteční investice a obtížný restart systému v případě jeho selhání** (ztráta informace o aktuálním stavu). Chaotické skladování není vhodné pro všechny typy výrobků, problémem může být například část chlazeného a nechlazeného zboží. [6]

1.3 Aktuální nabídka na trhu

Firma HOUFEK a.s. je významný výrobce dřevoobráběcích strojů v Evropě a jeho cílem je vytvořit vlastní automatický chaotický sklad. Při vytváření vlastního řešení, je důležité zmapování aktuální situace na trhu. Proto je následující podkapitola věnována představení existujících průmyslových řešení, konkrétně horizontálním způsobům skladování.

1.3.1 Automatický sklad od firmy HOLZHER

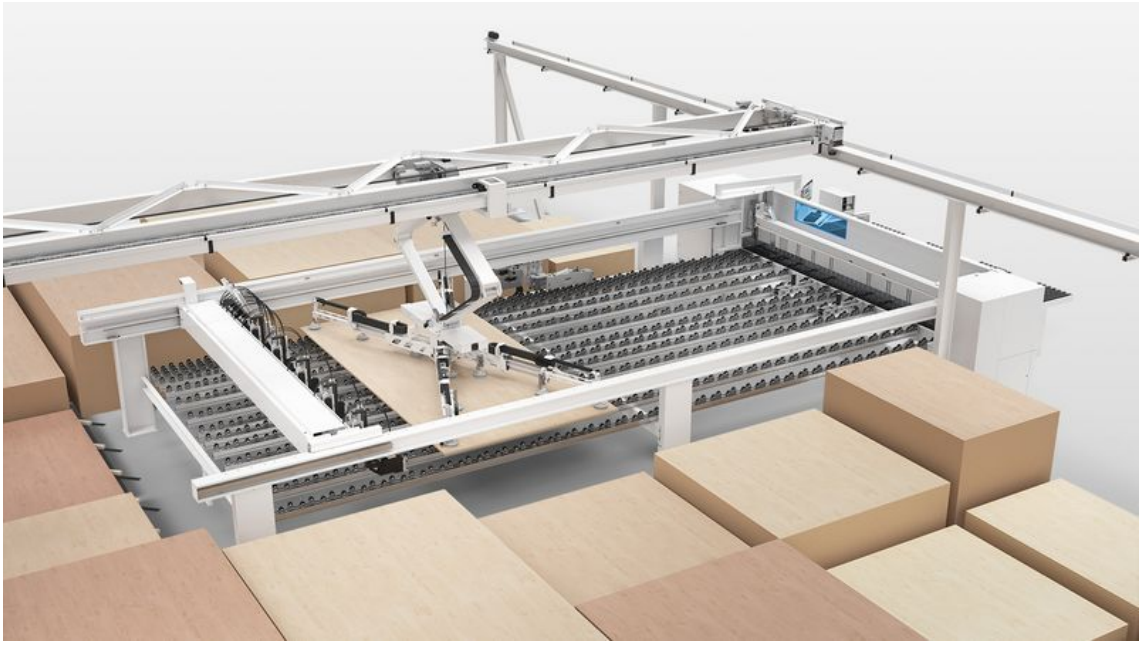
Automatický sklad od firmy HOLZHER nabízí plně automatické řízení skladu včetně evidence zbytků. Samotný manipulátor nabízí otáčení o 90° a hlídání oddělení desek od sebe. Firma také vyvinula vlastní inteligentní software včetně rozhraní pro usnadnění řízení. [7]



Obrázek 1.1: Automatický sklad od firmy HOLZHER [7]

1.3.2 Automatický sklad od firmy HOMAG

HOMAG je německá firma v dřevozpracujícím průmyslu. Nabízí automatické chaotické sklady různých velikostí. Maximální váha desky je 250 kilogramů. Firma dále nabízí rychlé a jednoduché kontaktování podpory přímo pomocí servisní aplikace. [8]



Obrázek 1.2: Automatický sklad od firmy HOMAG [8]

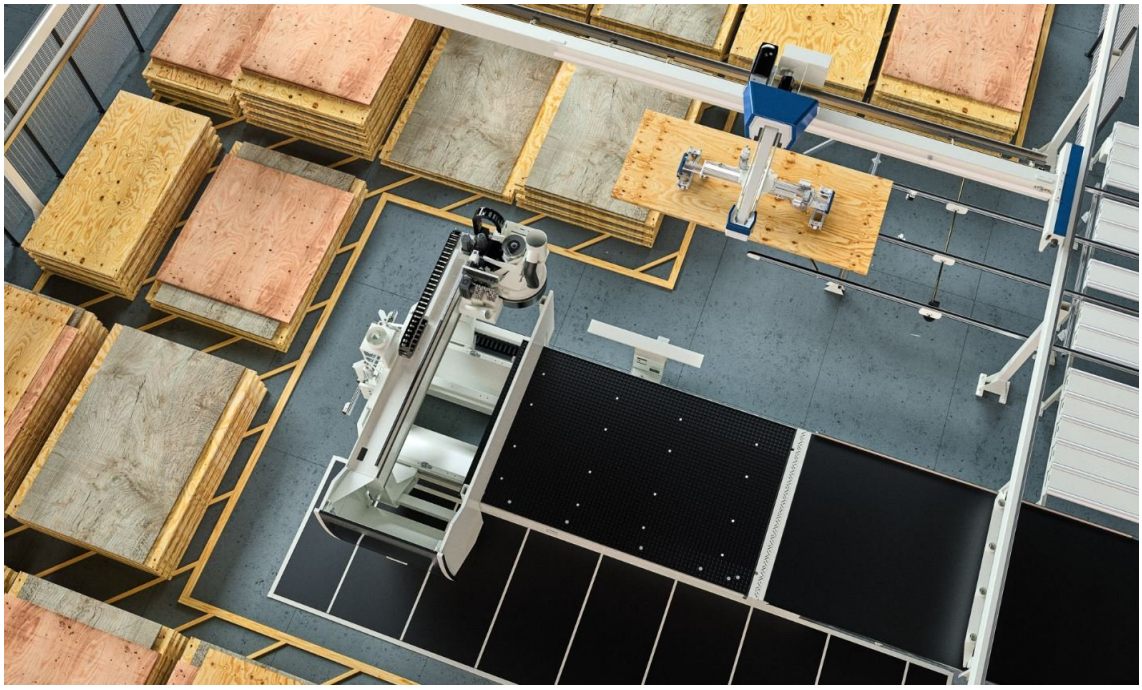
1.3.3 Automatický sklad od firmy DUIVESTEIN TECHNIEK

DUIVESTEIN TECHNIEK je nizozemská firma, která se zaměřuje zejména na automatické dopravníky a robotické sklady. Jejich 3-osý skladovací automat se chlubí vysokým výkonem a značnou flexibilitou. Firma Duivestein vyvinula v rámci chaotického skladu vlastní software DWMS (Duivestein Warehouse Management System) pro správu skladu. Samotný manipulátor je otočný o 90°a jeho součástí je detekční systém hmotnosti (desky). [9]

Obrázek 1.3: Automatický sklad od firmy DUIVESTEIN TECHNIEK [9]

1.3.4 Automatický sklad od firmy GRUNDNER

GRUNDNER je rakouská firma, jejíž specializace je výroba automatických skladů pro různé materiály. Jejich řídicí systém nabízí jak uspořádané, tak chaotické skladování. Firma nabízí také více druhů manipulátorů. [10]



Obrázek 1.4: Automatický sklad od firmy GRUNDNER [10]

1.3.5 Shrnutí aktuální situace na trhu

Aktuálně se na trhu nachází několik výrobců automatických skladů pro dřevozpracující průmysl. Většina průmyslových řešení je podobných, dokáže manipulovat s deskami do délky 5600 mm, šířky 2200 - 2600 mm a výšky stohu 2000 - 2500 mm. U většiny výrobců je minimální tloušťka desky 10 - 12 mm. Rozměry celkového skladu jsou volitelné až do délky 120 m, zatímco šířka je většinou do 12 m. Maximální váha desky se pohybuje od 250 kg do 350 kg. [7, 8, 9, 10]

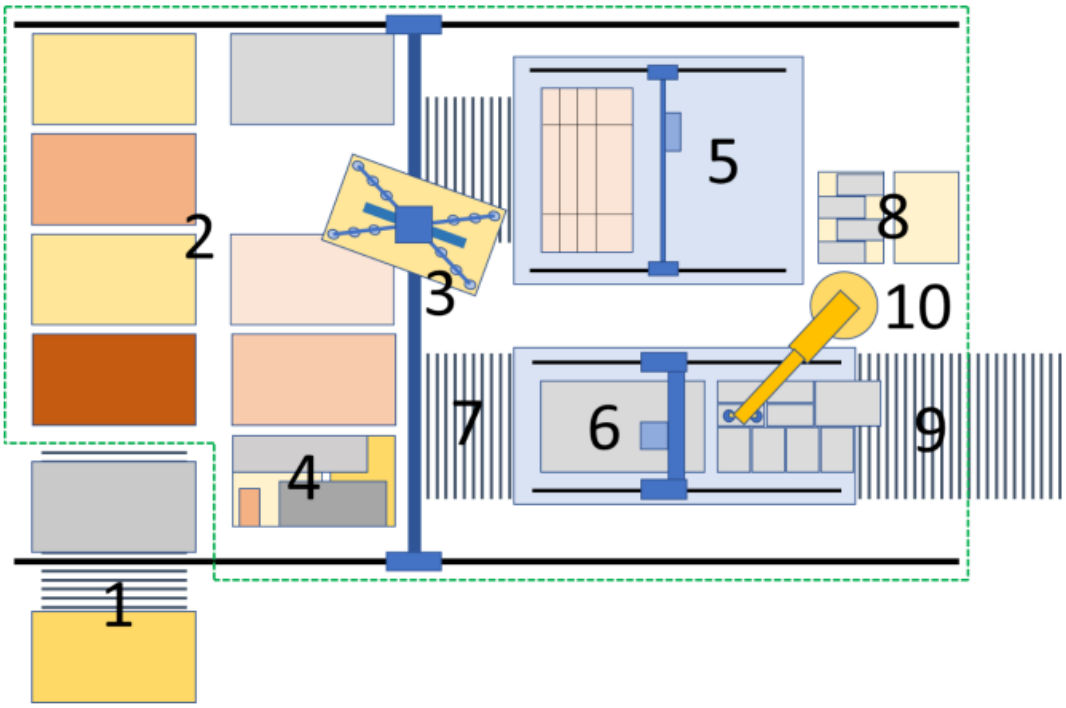
1.4 Automatický chaotický sklad firmy HOUFEK a.s.

Inspirací pro automatický skladový systém HOUFEK byla situace na aktuálním trhu. Nejdříve byl vypracován rozsáhlý průzkum trhu, v němž byly porovnány existující řešení. A následně započala tvorba vlastního automatického skladu s chaotickým způsobem skladování. [11]

Nejdříve bylo vytvořeno více variant automatického chaotického skladu. První varianta počítala jen se samotným automatickým chaotickým skladem. U druhé varianty byl požadavek na práci se zbytky, tudíž v rámci chaotického skladu zde musí být i pozice pro zbytky a manipulátor musí být schopný zbytky uchopit a přemístit zpět do skladu. Třetí varianta je nejkomplexnější. Počítá s uložením zbytků zpět do skladu a také s výstupním robotem, který přemisťuje hotové výrobky na palety. [11]

Na základě tří různých variant, zmíněných v předešlém odstavci, vzniklo ideové

schéma automatického skladu, které je vidět na obrázku 1.5. Číslo 1 označuje vstupní pozici skladu, ze které manipulátor 3 přesouvá desky do skladovací části 2. Oproti některým výrobcům chce automatický sklad firmy HOUFEK a.s. pracovat se zbytky a uskladňovat je. Tudíž součástí skladovacích pozic jsou i pozice pro zbytky 4. Čísla 5 a 6 označují pozice CNC stroje, kam bude manipulátor přemisťovat požadované desky. Součástí konceptuálního schématu je také robot 10, který ukládá vyrobené dílce na palety. Pozice 7 a 9 reprezentují válečkové dráhy. [11]



Obrázek 1.5: Ideové schéma automatického skladu [11]

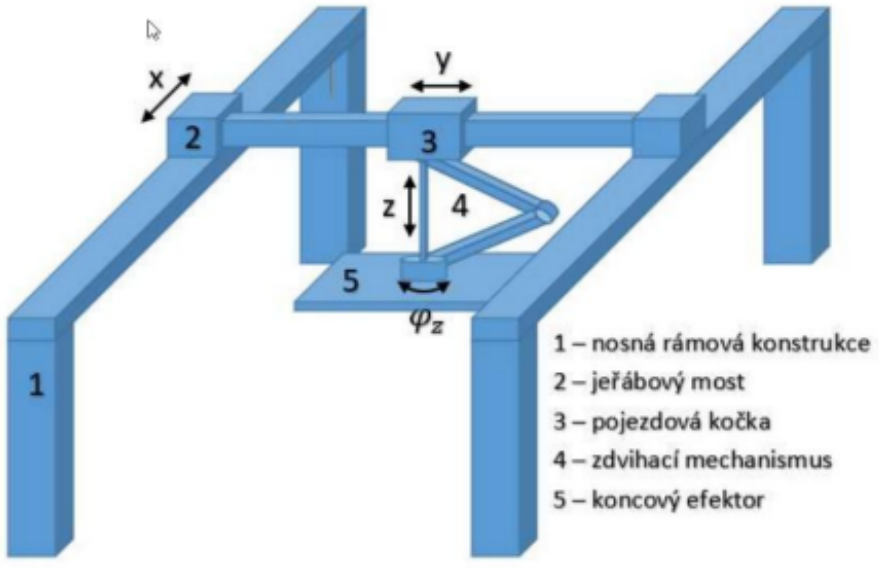
Před návrhem a realizací mechanických částí bylo nutné stanovit parametry, které má automatický chaotický sklad splňovat. Návrhové parametry jsou uvedeny v tabulce 1.1. Většina parametrů konkrétního návrhu bude odvozena z požadavků odběratele, avšak v přehledu jsou také uvedena omezení, která musí být respektována. Mezi omezení patří minimální a maximální tloušťka desky, hmotnost desky, šířka a výška skladu a maximální rozměr desky.

Název parametru	Min	Max	Typicky
Rozměr skladu x	-	-	20 m
Rozměr skladu y	-	12 m	-
Rozměr skladu z	3 m	4 m	-
Hmotnost desky	-	66 kg	200 kg
Rozměry desky	-	-	2000 x 2700 mm
Tloušťka desky	3 mm	100 mm	-
Rychlost osy x	70 m/s	80 m/s	-
Rychlost osy y	50 m/s	70 m/s	-
Rychlost osy z	30 m/s	50 m/s	-

Tabulka 1.1: Parametry automatického skladu

1.4.1 Mechanická konstrukce skladu

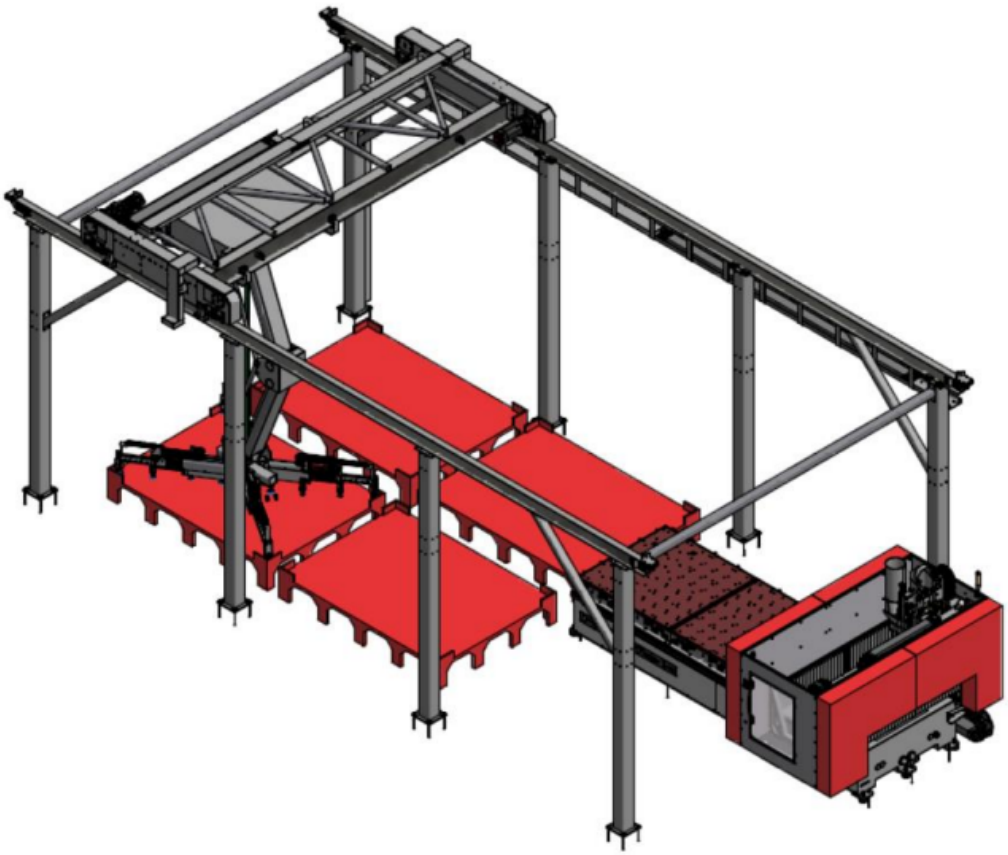
Portálový manipulátor na obrázku 1.6 je základem automatického skladu. Jde o kartézský manipulátor, který umožňuje pohyb ve všech třech osách a u koncové části je možnost otáčení kolem osy z. K zajištění rovnoběžnosti mezi pojezdovou kočkou 3 a koncovým efektorem 5 je použit paralelogram. [11]



Obrázek 1.6: Ideové schéma automatického skladu [11]

1.4.2 Řešení automatického skladu

Na obrázku 1.7 je již hotové základní řešení automatického chaotického skladu od firmy HOUFEK a.s.. Jde o komplexní zařízení se čtyřmi skladovými pozicemi, z kterých jedna může sloužit k navážení materiálu a další ke skladování zbytků. Konfigurace jednotlivých pozic je flexibilní. Dále je součástí automatického skladu také CNC obráběcí centrum. [12]



Obrázek 1.7: Ideové schéma automatického skladu [12]

Upnutí deskového materiálu zajišťuje systém 12+1 přísavek s vlastním zdrojem tlakové energie. Přídavné samostatně ovladatelné přísavky dále umožňují "odlepení" desky od ostatních desek kvůli zamezení nežádoucího pohybu spodních desek. [12]

Pro pohyb skladu byl zvolen pohonový systém od firmy FANUC. Jako řídicí systém manipulátoru byl zvolen řídicí systém FANUC pro CNC ve variantě FS 31i-B5 Plus.[12]

Kapitola 2

Digitální dvojče

Digitální dvojče je virtuální reprezentací fyzického objektu, poskytuje monitorování, vizualizaci, umožňuje předpověď, optimalizaci, testování, údržbu atd. Koncept digitálního dvojčete je úzce spojován s průmyslem 4.0. Mezi virtuálním a fyzickým modelem probíhá výměna dat v reálném čase, kde samotná data jsou sbírána řadou senzorů, aktuátorů a dalších technologických prostředků. [13, 14]

2.1 Historie digitálních dvojčat

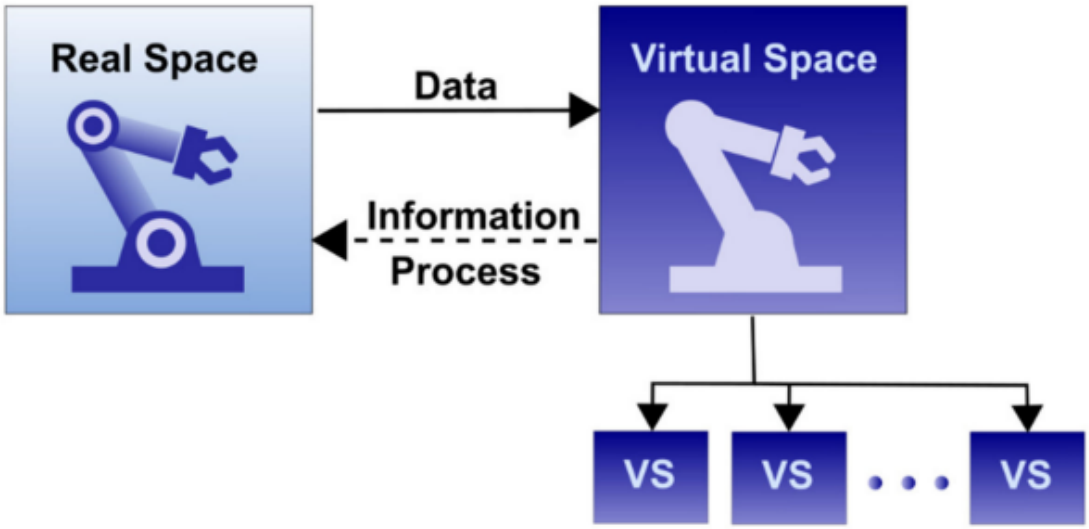
V roce 1991 představil David Gelernter software pod názvem "Mirror Worlds", který na základě údajů z fyzického světa napodobuje realitu. Samotný koncept digitálního dvojčete se ale váže až k práci pod vedením Michaela Grievese, který roku 2002 zveřejnil práci na téma řízení životního cyklu produktu. Původní model byl označován jako "Mirrored Spaces Model" a byl tvořen třemi klíčovými prvky: reálný prostor, virtuální prostor a výměna dat mezi virtuálním a reálným prostorem. [13]

Roku 2006 byl název modelu "Mirrored Spaces Model" vydaný Grievesem změněn na "Information Mirroring Model". Nově byl reálný model propojen s více virtuálními modely pro testování alternativních nápadů či designu. V tomto novém modelu byl kladen důraz zejména na obousměrný datový tok mezi dvěma prostory. Na obrázku 2.1 je tento model zobrazen. [13]

Samotný název digitální dvojče se poprvé objevil roku 2010 v technologickém plánu NASA. Ideu digitálního dvojčete ale NASA používalo již dříve v rámci programu Apollo, kdy došlo k sestavení dvou identických vesmírných vozidel a cílem bylo vzájemné zrcadlení. Vytvořit simulované prostředí k modelování a testování možných řešení bylo nutné kvůli explozi kyslíkové nádrže, ke které došlo dva dny po startu. Díky simulacím a následné realizaci nejvhodnější z nich bylo možné vytvořit improvizovaný čistič vzduchu pouze z materiálů dostupných na vesmírné lodi. Simulace dále byla použita i pro bezpečný návrat posádky zpět na Zemi. [13, 15]

Dále koncept digitálního dvojčete užívaly vzdušné síly Spojených států k návrhu, údržbě a predikci fyzikálních a mechanických vlastností letadel a predikci únavových trhlin v konstrukci. [13]

Koncept digitálního dvojčete je též hojně využíván při návrhu a konstrukci mechatronických zařízení s možností otestovat funkcionalitu vyvíjeného zařízení formou simulačních experimentů. [13]



Obrázek 2.1: Information Mirroring Model vytvořený Michaelem Grievesem [13]

2.2 Definice digitálního dvojčete

Digitální dvojče je obecně reprezentováno třemi primárními komponenty. Fyzikální realitou, virtuální realitou a propojením mezi nimi, které slouží k výměně dat. Součástí virtuální reality je mimo uživatelského rozhraní, softwaru a virtuálních senzorů, také digitální model, jenž je co nejvíce shodný s fyzickým modelem. Správně vytvořené digitální dvojče může sloužit k prozkoumávání nových řešení, předpovídání chování v neznámých stavech, optimalizaci, predikci potřebné údržby, off-line programování a mnoho dalším věcem. [13, 16]

2.2.1 Fyzická realita

Fyzickou realitou rozumíme to, co zahrnuje celou známou, ale i neznámou část systému. Skládá se z fyzického systému, fyzického prostředí a fyzikálních procesů.

Fyzický systém představuje soubor vzájemně propojených prvků, které vzájemně interagují. Prvky mohou být fyzické objekty či procesy. Jde například o jednoduché prvky, stroje nebo celé výrobní linky či haly. Klíčovým prvkem fyzického systému je jeho ohraničení, které je jasně definováno prostorem a časem, což nám umožní přesné vymezení součástí daného systému. U digitálních dvojčat je fyzický systém základním prvkem pro modelování a simulaci v digitálním světě. Digitální reprezentace fyzického systému umožní monitorování jeho stavu, simulování jeho chování a predikci chování v neznámých stavech. [13]

Fyzické prostředí je prostředí, kterým je fyzický systém obklopen a ovlivňován. Mezi fyzickým systémem a prostředím dochází k interakci prostřednictvím fyzikálních procesů. Mezi fyzické prostředí řadíme vlivy jako je teplota, vlhkost, vibrace, záření, proudění a mnoho dalších. U digitálního dvojčete je velice důležité určit, které vlivy fyzikálního prostředí jsou pro daný proces důležité. Tyto vlivy jsou běžně monitorovány v reálném prostředí řadou senzorů a posílány do virtuální reality k simulování, předpovídání a rozhodování. [13]

Fyzikální procesy popisují chování fyzického systému ve svém fyzickém prostředí. Jsou důležité zejména pro modelování chování systému ve virtuálním světě. Jedná se například o proces svařování, kování, frézování, 3D tisku, inicializace, balení a nespočetně dalších. Jednotlivé fyzikální procesy jsou často optimalizovány a vylepšovány ve virtuálním prostředí. [13]

2.2.2 Virtuální reprezentace

U virtuální reality je klíčovým prvkem kvalitní simulační model. Ten je založen na fyzikálních principech, často v součinnosti s daty získanými pomocí fyzikálních měření na reálných objektech. Je možné mít více virtuálních systémů, kde každý má jinou úroveň detailů v závislosti na tom, co simuluje. Výsledná simulace tedy může být složena z výstupů více virtuálních systémů. [13]

Idealizovaná reprezentace fyzické reality znamená nutnost stanovit cíl naší idealizace. Kromě toho je potřeba zvážit důležitost detailů naší virtuální reality pro funkční simulaci fyzického modelu a schopnost správně se rozhodovat. Idealizace celkově zahrnuje datové modely a modely chování systému často založené na fyzikálních zákonech. Důležité je porovnání hodnot z fyzické reality a virtuální reprezentace pro rozhodnutí, zda je naše idealizace dostatečně přesná pro naše účely. Na obrázku 2.2 je znázorněn proces přizpůsobování virtuální reprezentace realitě. [13]

Obrázek 2.2: Proces přibližování virtuální reprezentace fyzické realitě [13]

Stavy a parametry systému jsou důležitou součástí modelování digitálních dvojčat. Stavy jsou v čase vyvíjející aspekty systémů na základě předchozích stavů a vstupů. Parametry jsou informace, které popisují chování systému v závislosti na různých vstupech a stavech. Na základě správně zvolených parametrů a stavového popisu lze odhadovat chování systému při různých vstupech. [13]

Virtuální systém je základní součást virtuální reprezentace, obsahuje data a modely z fyzického systému o různých detailech. Úroveň detailů je důležitá pro správné

rozhodování a chování systému. Virtuální systém může obsahovat více různých modelů, které spolu mohou více či méně interagovat. [13]

Virtuální prostředí je snaha o virtuální reprezentaci fyzikálního prostředí opět s určitou mírou detailů. Podobně jako u výše zmíněných částí virtuální reality je důležité zvolit správné množství detailů k dostatečné idealizaci fyzikálního prostředí. [13]

Virtuální procesy se snaží přiblížit s různou přesností chování reálným fyzikálním procesům. Nejčastěji se setkáváme se sestavením matematického modelu, který odpovídá fyzikálním procesům. Vztah mezi vstupem a výstupem je tedy dán fyzikálními zákony, jde např. o dynamiku, kinematiku a plno dalších. Dále je možné tvořit vztah mezi vstupem a výstupem na základě sbíraného množství dat, v tomto případě je užíváno strojové učení nebo umělá inteligence. [13]

2.2.3 Výměna dat mezi virtuální reprezentací a fyzickou realitou

Třetím důležitým prvkem je výměna dat mezi virtuálním světem a realitou. Jedná se o obousměrnou výměnu dat: fyzicko-virtuální a virtuálně-fyzickou.[13]

Fyzicko-virtuální spojení je propojení mezi fyzickou realitou a virtuální reprezentací. Propojení je důležité k udržení virtuální reprezentace ve stejném stavu jako je fyzický systém. V prvním kroku je potřeba shromáždit relevantní data, která zároveň používáme ve virtuálním systému. Dále je potřeba tato data zpracovat tak, aby je bylo možné použít ve virtuální reprezentaci, např. změnu napětí vyhodnotit jako změnu deformace. Poslední částí fyzicko-virtuálního spojení je aktualizace dat ve virtuálním systému. Nejdříve se aktualizují naměřené stavy a z nich jsou poté zjištěny a aktualizovány neměřené stavy. Celý proces udržuje konzistenci mezi fyzickou a virtuální realitou. [13]

Virtuálně-fyzické spojení je spojení zpět z virtuální reprezentace do fyzické reality. Toto propojení uzavírá smyčku a je důležitou součástí digitálního dvojčete. V rámci spojení se posílají důležité informace a rozhodnutí, které byly vytvořeny ve virtuálním světě. Posílaná rozhodnutí jsou realizována ve fyzické realitě a ovlivňují tak fyzické procesy. [13]

2.2.4 Shrnutí prvků digitálních dvojčat

Digitální dvojče tedy není jen pasivním modelem, ale umí aktivně reagovat na změny v reálném světě a poskytuje cenné informace z hlediska rozhodování. V praxi to znamená, že dochází k aktualizaci a vývoji na základě dostupných informací z reality. Tímto způsobem dochází k dynamickému vývoji celého systému digitálního dvojčete včetně optimalizace a monitorování procesů, což přináší značné výhody při plánování a řízení procesů.

V tabulce 2.1 je shrnutí charakteristických prvků digitálních dvojčat, které byly

popsány výše. [13, 16]

Prvek	Popis
Fyzická realita	
Fyzický systém	Část fyzické reality vybraná pro modelování, která zahrnuje všechny relevantní vzájemně interagující části, které tvoří jednotný celek.
Fyzické prostředí	Vnější vlivy ovlivňující chování fyzického systému.
Fyzikální procesy	Procesy, které vykonává fyzický systém a způsobují změny stavů systému.
Virtuální reprezentace	
Virtuální systém	Data a výpočetní modely fyzického systému na vybrané úrovni abstrakce.
Virtuální prostředí	Virtuální reprezentace fyzického prostředí.
Virtuální procesy	Virtuální reprezentace odpovídajících fyzikálních procesů, používaná k simulaci.
Výměna dat	
Fyzicko-virtuální spojení	Způsob, jakým jsou sbírány, interpretovány, komunikovány a využívány informace z fyzického systému k aktualizaci stavů a parametrů virtuální reprezentace.
Virtuálně-fyzické spojení	Způsob, jakým jsou rozhodnutí přijímána a následně realizována ve fyzickém systému.

Tabulka 2.1: Popis prvků digitálního dvojčete [13]

2.3 Charakteristické vlastnosti digitálního dvojčete

Každé digitální dvojče má jiné vlastnosti a parametry, ale většina z nich má společné charakteristické znaky.

Vysoká věrnost virtuální reprezentace: Jednou z charakteristických vlastností digitálních dvojčat je vysoká věrnost virtuální reprezentace. Čím přesnější digitální model je použit, tím přesněji je možné napodobit a následně simulovat fyzický systém. Tato charakteristika tedy zdůrazňuje, že virtuální model, systém a prostředí by se měly co nejvíce podobat fyzickému. K vysoké míře věrnosti digitálního dvojčete je nutné fyzický systém dostatečně monitorovat. [14, 17]

Dynamika: Fyzický systém se v čase mění, tudíž musí být digitální dvojče dynamicky se měnící systém v čase. Základem je bezproblémová výměna dat a rychlá odezva virtuálního systému. [14]

Automatický vývoj: Digitální dvojče se po čas životního cyklu neustále vyvíjí. Veškeré změny v jednom z dvojčat se okamžitě odráží v tom druhém. Digitální dvojče se na základě svého fyzického dvojčete optimalizuje a adaptuje. [14]

Multifyzikálnost a víceúrovňovost: Multifyzikálnost označuje kombinaci různých fyzikální parametrů. Modely obsahují geometrické údaje, modely dynamiky, pružnost, termodynamiky, ale i vyhodnocování napětí, únav a dalších vlastností. Digitální dvojče se co nejblíže snaží napodobit jeho fyzikální protějšek. Víceúrovňovostí je myšleno více druhů různých detailů. Pro některé fyzikální modely není například potřeba přesná geometrie, ale u jiných může být nutností znát například drsnost či tolerance. [14]

Multidisciplinárnost: Multidisciplinárností je myšleno propojení více různých odvětví. V rámci průmyslu 4.0 u digitálních dvojčat propojujeme více oborů jako jsou: informatika, strojírenství, elektrotechnika, elektronika, mechatronika a další. [14]

2.4 Úroveň integrace

2.4.1 Digitální dvojče

U digitálních dvojčat existuje obousměrné propojení mezi fyzickým dvojčetem a jeho virtuálním protějškem v reálném čase. Samotné digitální dvojče simuluje, umožňuje různé optimalizace a analýzy, prediktivní údržbu, plánování výroby, ale také poskytuje on-line zpětnou vazbu fyzické verzi. Digitální dvojče, na rozdíl od digitálního stínu a digitálního modelu, ovlivňuje fyzicky existující protějšek. [14, 18]

Digitální dvojčata jsou využívána ve výrobních závodech, zdravotnických zařízeních, infrastruktuře či chytrých městech, např. Singapur, Dubaj. [18]

2.4.2 Digitální stín

Oproti digitálnímu dvojčeti dochází u konceptu digitálního stínu k posílání dat pouze z fyzické vrstvy do digitální. Data se v reálném čase odrážejí v digitálním stínu. Pokud se chtějí využít informace získané z digitálního stínu, je nutné to udělat manuálně. Digitální stín bývá často v podobě pouze matematického modelu bez vizualizace. [14, 18]

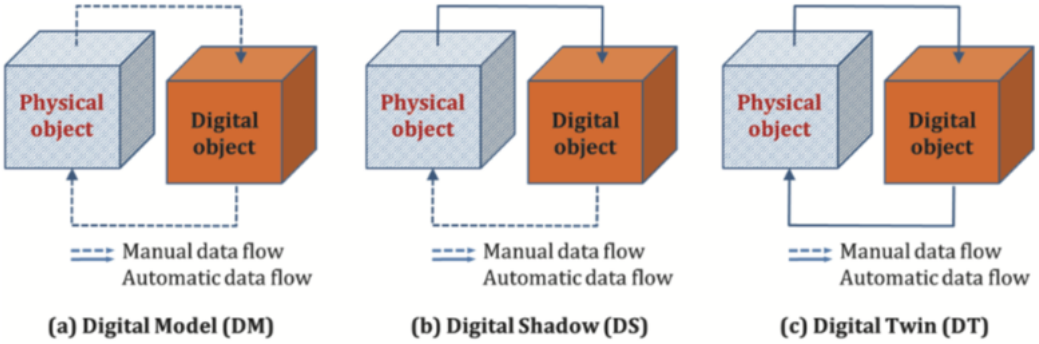
Digitální stín se nejčastěji používá ve výrobních závodech, kde slouží k monitorování, analyzování, zálohování nebo také k identifikování problematických míst z hlediska výrobních časů. [18]

2.4.3 Digitální model

U digitálního modelu je úroveň integrace nejnižší, data je možné obousměrně posílat pouze manuálně. Digitální model tedy není schopen odrážet stav a chování

fyzického modelu v reálném čase. Digitální model je k vidění nejčastěji v podobě 3D modelů, různých simulací či matematických algoritmů. Umožňuje vizualizaci, simulaci, optimalizaci, testování a další. [14, 18]

Digitální model se používá především pro vizualizaci modelů, je používán u architektonických projektů, strojírenských modelů či návrhů a dalších. V dnešní době se často využívá propojení s virtuální realitou, kdy je možné projít například celý výrobní závod prostřednictvím 3D virtualizace. [18]



Obrázek 2.3: Úrovně integrace fyzického modelu a virtuálního protějšku [19]

2.5 Hierarchie digitálních dvojčat

Z hlediska hierarchie lze dělit digitální dvojčata na tři různé úrovně. Na obrázku 2.4 jsou tyto úrovně znázorněny.

2.5.1 Úroveň jednotek (Unit Level)

Jedná se o nejmenší jednotku ve výrobě, pro kterou je vytvořené digitální dvojče. Může se jednat například o vybavení, materiál či vliv prostředí. [14]

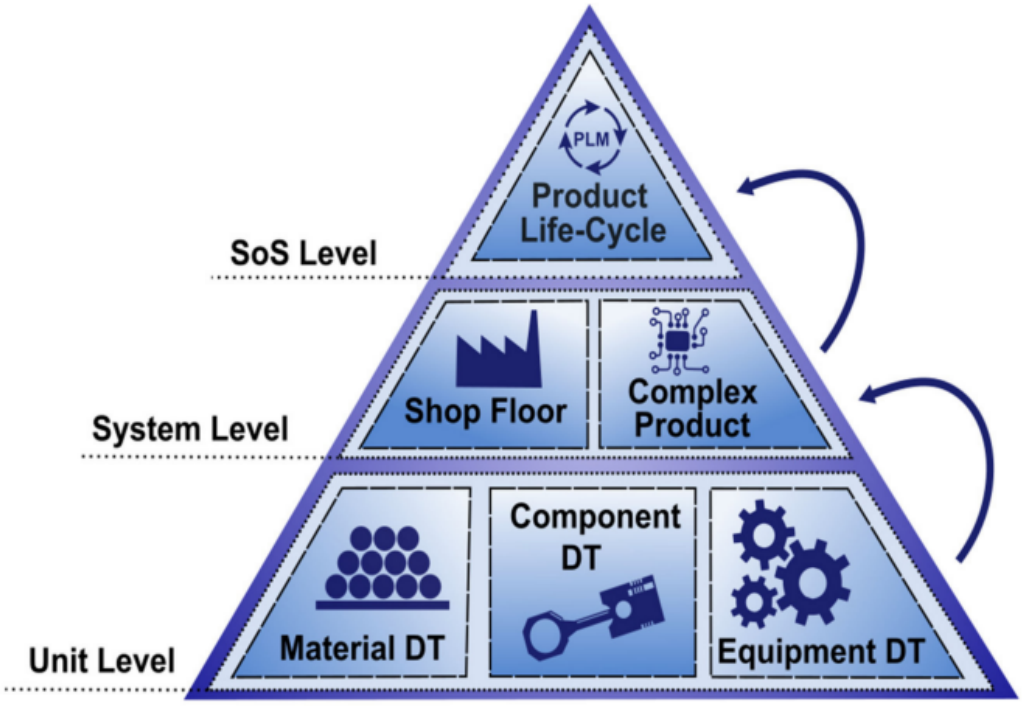
2.5.2 Úroveň systému (System Level)

Spojením více jednotkových digitálních dvojčat vznikne digitální dvojče na systémové úrovni. Jedná se o konečné výrobky, výrobní linku, dílnu či celou továrnu. [14]

2.5.3 Úroveň systému systémů (System of Systems Level)

Digitální dvojče System of Systems vznikne spojením více digitálních dvojčat jednotlivých systémů. Používá se zejména pro vytvoření spolupráce mezi více podniky či odděleními v jedné firmě. Digitální dvojče System of Systems integruje různé fáze

životního cyklu produktu, což zahrnuje návrh, výrobu, distribuci, servis a údržbu. Tím se zajišťuje celková koordinace a optimalizace procesů napříč celým životním cyklem produktu. [14]



Obrázek 2.4: Hierarchie digitálních dvojčat [14]

2.6 Výhody a nevýhody digitálních dvojčat

Hlavním důvodem proč jsou digitální dvojčata považována za klíčový prvek průmyslu 4.0, jsou jejich výhody uvedené v následující subkapitole.

2.6.1 Výhody digitálních dvojčat

Rychlost prototypování a přepracování:

Před realizací samotného pracoviště nebo při změně stávajícího pracoviště je možné vyzkoušet koncept simulačně. Může se jednat o testování v době návrhu produktu, optimalizaci stávajícího řešení nebo hledání nových řešení. Díky prototypování v digitální podobě lze také dosáhnout materiálových úspor v podobě absence tvorby fyzických vývojových prototypů. [14]

Efektivní využití nákladů:

Testování nového prototypu či přepracování stávajícího řešení ve virtuální podobě může ušetřit čas i finance. V případě přepracování stávajícího řešení není nutné zastavit celou linku, ale stačí nejdříve přepracovanou linku otestovat a naprogramovat

v digitální podobě a teprve poté nové řešení realizovat. [14]

Předpovídání problémů a plánování:

Díky výměně dat mezi fyzickým a digitálním dvojčetem v reálném čase je možné předpovídat problémy. V závislosti na předpovědi chybových stavů jde plánovat výrobu dopředu tak, aby chyby v systému narušovaly plynulý běh výrobního procesu v co nejmenší míře. [14]

Optimalizace řešení a zlepšená údržba:

Optimalizace na digitálním dvojčeti umožňuje vyzkoušet přepracované řešení, např. nově naprogramované trajektorie, komunikaci s PLC a další. Na základě simulace je možné předem určit výrobní čas. Součástí digitálních dvojčat je často predikovaná údržba, která monitoruje fyzické zařízení a na základě vyhodnocení příchozích dat ze senzorů je schopna určit čas, kdy je potřeba daný díl/část vyměnit. [14]

Ovládání na dálku a bezpečnost:

Vzdálený přístup a řízení je další výhodou digitálních dvojčat, který umožňuje ovládat fyzický protějšek skrze uživatelské rozhraní. Naše fyzické dvojče je po celou dobu monitorováno a může být upraveno i pro bezkontaktní ovládání skrze jeho digitální dvojče. Tato výhoda byla využívána například v době COVID-19. Snaha o vzdálený přístup k fyzickému dvojčeti je rovněž v extrémních a nebezpečných pracovních podmínkách. Digitální dvojčata jsou v tomto případě hojně využívána, protože zásadně snižují riziko zranění obsluhy. [14]

Dokumentace a komunikace:

V digitálních dvojčatech je možné se velice snadno dostat k veškeré dokumentaci o fyzickém dvojčeti. Dále mohou digitální dvojčata synchronizovat data z různých databází a aplikací a udržovat tato data na jednom místě. V dnešní době je často možné řešit závady skrze on-line servis, který se napojí přímo na fyzické dvojče skrze on-line prostředí. [14]

Školení:

U digitálních dvojčat lze dělat bezpečná školení obsluhy. Nastavením různých stavů se obsluha může vyškolit na řešení různých situací. Zároveň se také eliminuje možnost poškození fyzického dvojčete při školení. [14]

2.6.2 Nevýhody digitálních dvojčat

Vysoké náklady na implementaci:

Počáteční investice do digitálního dvojčete je vysoká. Je nutné pořídit výkonný hardware, specializovaný software pro tvorbu a správu digitálních dvojčat a nástroje pro speciální analýzu dat. Proto se u menších podniků s digitálními dvojčaty nesetkáváme . [20]

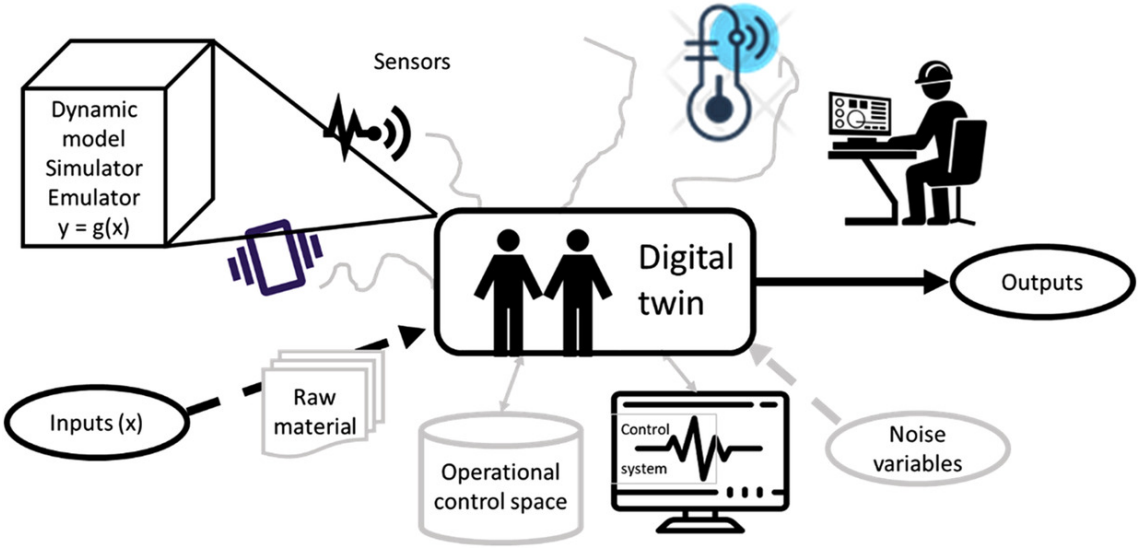
Nutná technická odbornost:

Pro nastavení a následný provoz je potřeba kvalifikovaných profesionálů. Nedosta-

tečná technická odbornost na trhu práce může vést k nedostatku personálu a být tak značnou překážkou. [20]

Riziko úniku dat:

Velký problém u digitálních dvojčat představuje možný únik dat. Společnosti potřebují mít zabezpečení na vysoké úrovni, aby nedošlo k úniku citlivých dat. [20]



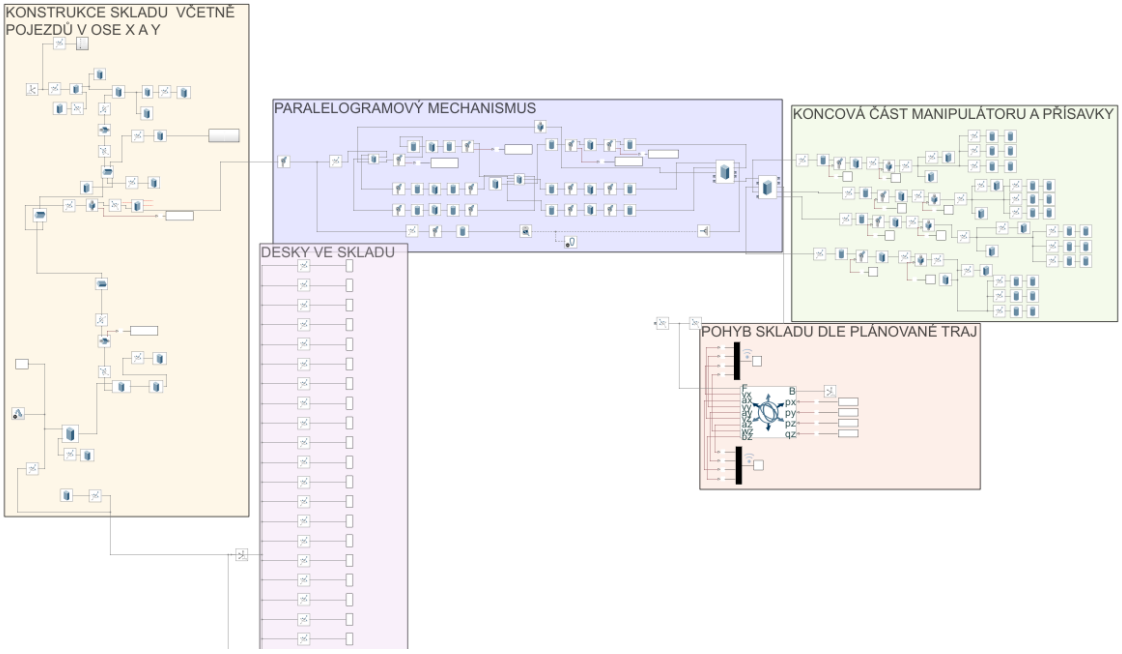
Obrázek 2.5: Digitální dvojče v průmyslu 4.0 [22]

Kapitola 3

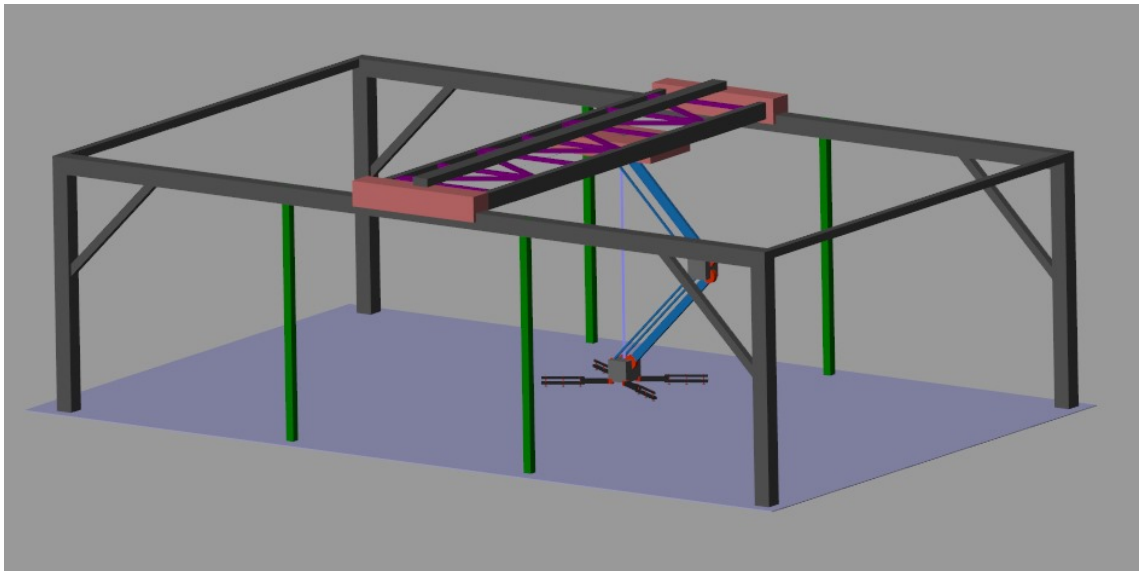
3D model automatického skladového systému

Digitální model automatického chaotického skladu byl vytvořen v prostředí Simulink pomocí knihovny Simscape Multibody. Simulink je součástí programu Matlab, ve kterém jsou napsané veškeré kódy pro ovládání digitálního modelu. Proto je tato kapitola věnována popisům jednotlivých částí parametrického modelu v Simulinku.

Samotný model vychází z bakalářské práce s názvem Manipulátor pro automatický sklad deskového materiálu od Bc. Jakuba Švadleny [21]. V práci uvedený model byl předělán dle nových požadavků: namísto nůžkového mechanismu byl vytvořen paralelogramový mechanismus včetně výsuvných přísavek na konci, bylo přidáno natočení mechanismu, došlo k upravení na komplexnější a realističtější parametrický model a také vzniklo automatické generování bloků v Simscape z prostředí Matlab. Na obrázku 3.1 je zobrazen celý vytvořený model v prostředí Simscape a na obrázku 3.2 je tento model vizualizován.



Obrázek 3.1: Model automatického skladu v prostředí Simscape

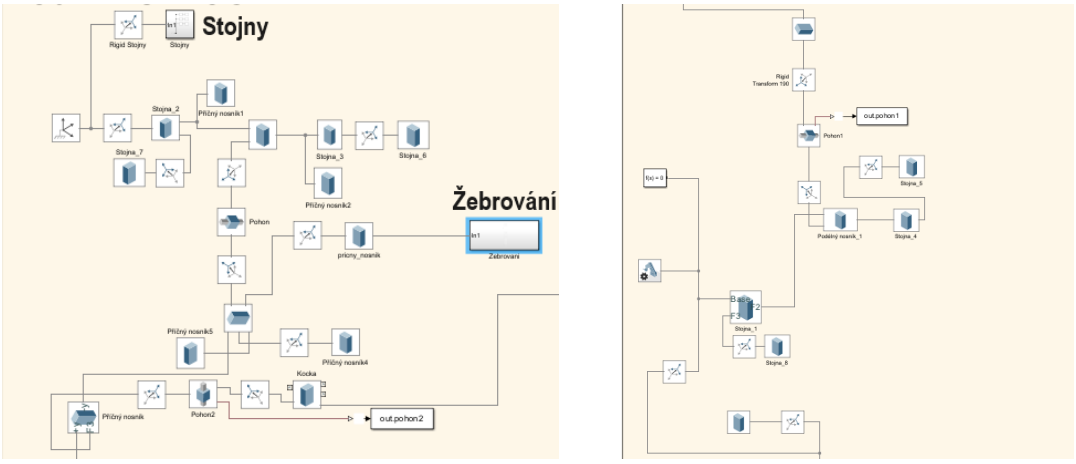


Obrázek 3.2: Vizualizace skladu

3.1 Konstrukce skladu a pojezd v ose x a y

Konstrukce automatického skladu je kompletně parametrická. Rozměrové parametry jsou vypočítány v Matlabu v závislosti na aktuální velikosti skladu a jsou uloženy do workspace pro přímé využití modulem Simscape. Na obrázku 3.3 je zobrazeno Simscape schéma modelované konstrukce skladu.

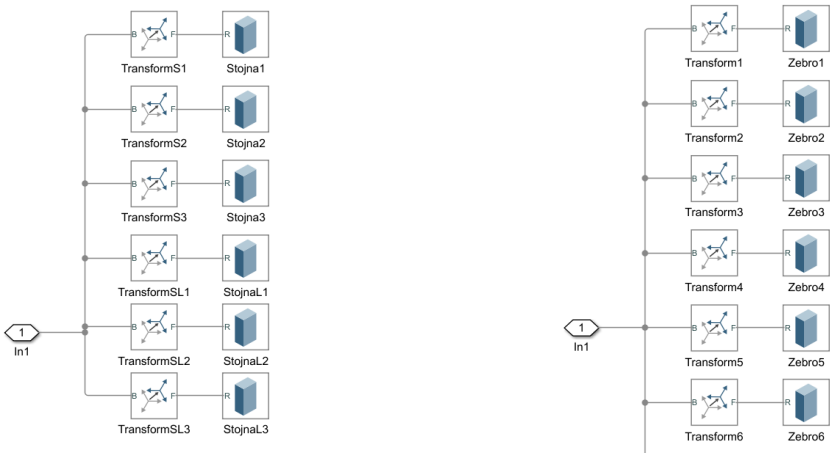
Součástí konstrukce jsou i dva generované subsystémy. První, pojmenovaný stojny,



Obrázek 3.3: Simscape model konstrukce

automaticky v závislosti na velikosti skladu vytváří stojny a je zobrazen na obrázku 3.2 zeleně. Druhý, pojmenovaný žebrování, dovytváří realističtější podobu příčníku skladového manipulátoru - na obrázku 3.2 fialově. Na obrázku 3.4 je zobrazena struktura automaticky generovaných bloků z Matlabu, kde se těmto blokům nastavují rozměry, umístění a další parametry.

Počet vygenerovaných bloků vždy závisí na rozměrech aktuálního skladu. Oba subsystémy obsahují pouze bloky "Brick Solid" a "Rigid Transform". První jmenovaný vytváří pevné těleso se zadanými parametry. "Rigid Transform" pak určuje polohu, kde se v modelu konkrétní těleso nachází a jak je natočené.

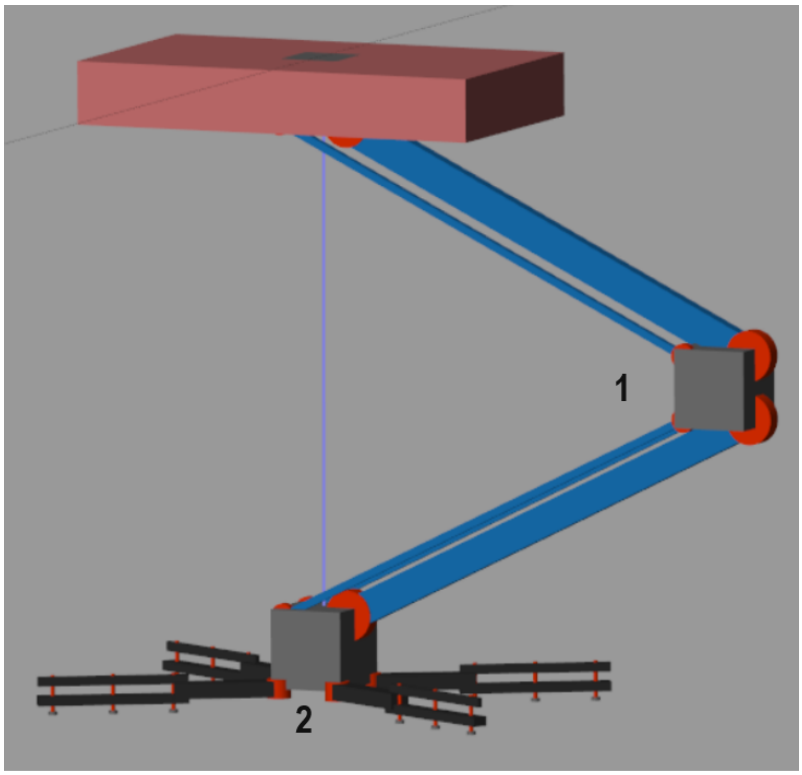


Obrázek 3.4: Subsystémy vytvořené pomocí skriptu v Matlabu

U konstrukce skladu jsou monitorovány polohy obou pohonů a následně je koncová poloha využita jako počáteční podmínka pro další simulaci. Pro monitorování pohonů je potřeba v příslušném kloubu zapnout měření pozice. Průběh pozice je exportován do Matlabu pomocí bloku "To Workspace". V Matlabu je nutné vybrat pouze poslední pozici a tu následně uložit pod proměnnou, která je u příslušného pohonu nastavena jako počáteční hodnota.

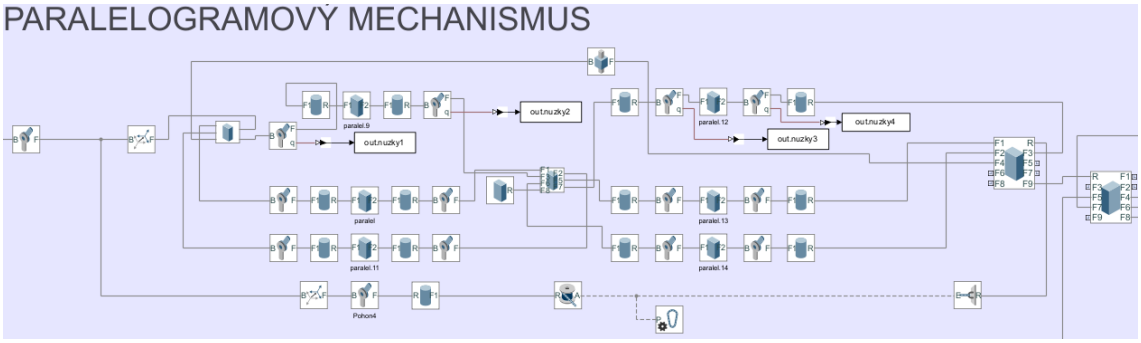
3.2 Paralelogramový mechanismus manipulátoru a přísavkový systém

Oproti starému modelu byl změněn mechanismus manipulátoru. Původní nůžkový mechanismus byl přepracován na paralelogramový mechanismus, který je zobrazen na obrázku 3.5. Manipulátor (1) umožňuje vykonávat zdvih desky ve svislém směru a natáčení desky kolem svislé osy. Dále je možné variabilně nastavovat orientaci čtyř ramen s pneumatickými přísavkami (2) a teleskopicky tato ramena podle velikosti manipulovaného materiálu vysouvat. Orientaci a vysunutí lze nastavit v uživatelském rozhraní, které bude ukázáno v další kapitole. Na obrázku 3.6 je Simscape model paralelogramu a na obrázku 3.7 je model přísavkového systému umístěného na spodní části manipulátoru.

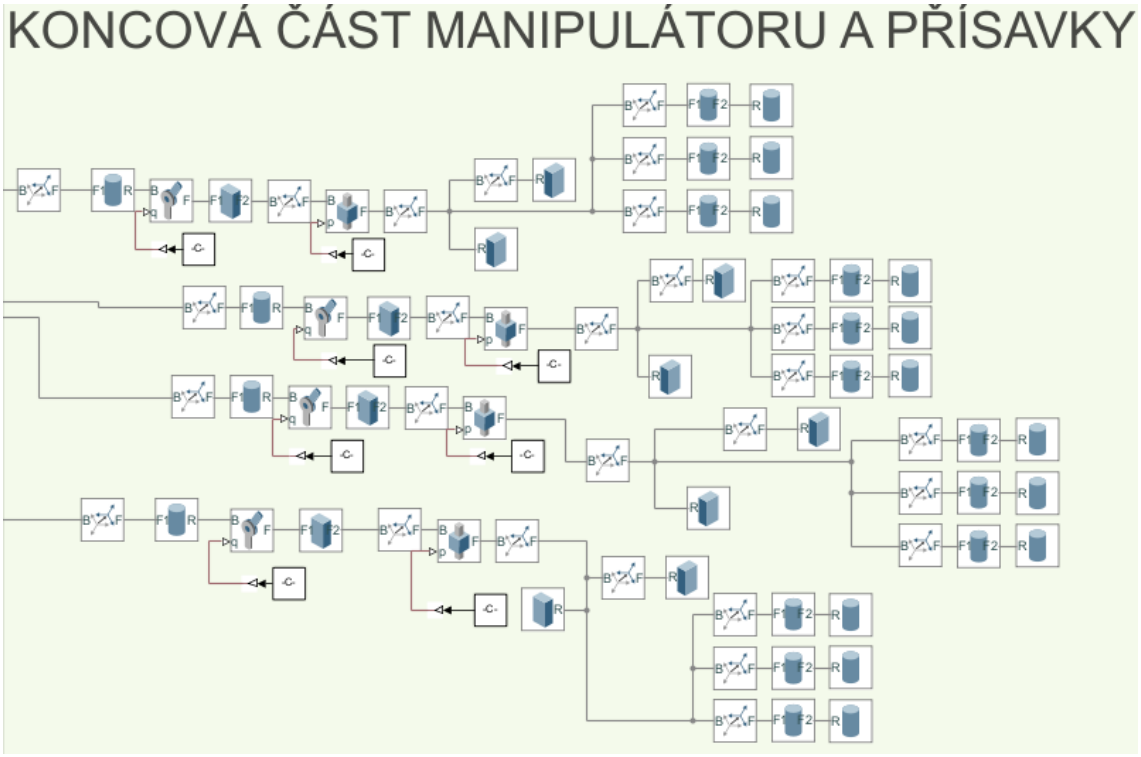


Obrázek 3.5: Vizualizace paralelogramového mechanismu s přísavkami

V paralelogramu jsou monitorována a ukládána čtyři natočení kloubů, což slouží k zapamatování posledního natočení a následného použití této hodnoty jako počáteční polohy pro další pohyb. Teoreticky by stačilo ukládat pouze hodnotu natočení pohonu pro zajištění svislého pohybu pomocí navíjení řemenu, ale v Simscape tato počáteční podmínka nebyla dostatečná.



Obrázek 3.6: Simscape model paralelogramu



Obrázek 3.7: Simscape model přísavkového systému

U přísavkového systému je hodnota nastavitelná v uživatelském rozhraní braná jako počáteční a aktuální zároveň. Obecně celý přísavkový systém není řízen z bloku "Bushing Joint" v Simscape jako paralelogram a pojezdy umístěné přímo na konstrukci, ale je ovládán přímo hodnotami z Matlabu.

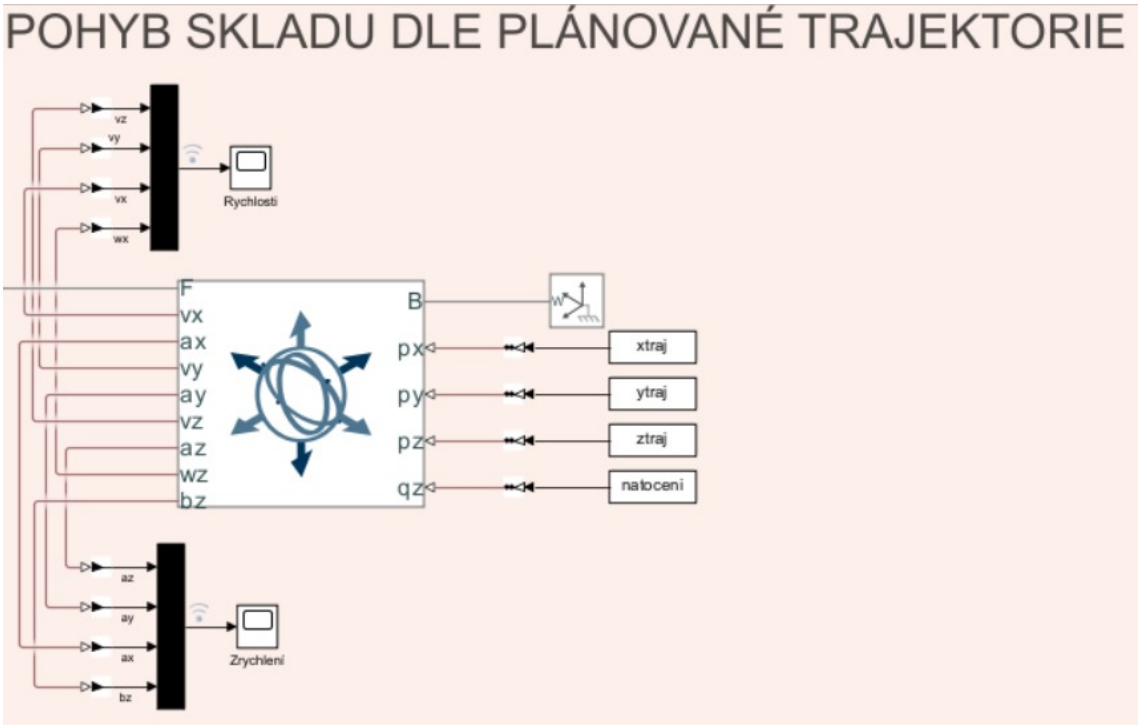
3.3 Řízení dle naplánované trajektorie

Pro implementování naplánované trajektorie do našeho simulačního modelu je možné použít například "Bushing Joint", který je zobrazen v tomto řešení na obrázku 3.8. Jedná se o vazbu se šesti stupni volnosti, která umožňuje pohyb ve všech třech osách

a také rotaci kolem nich. Vstupem jsou časově závislé souřadnice x, y, z a natočení kolem osy z.

Vazba je umístěna v koncovém bodě celého manipulátoru a počítá inverzní kinematickou úlohu. Následně Simscape použije vypočítané natočení a posuvy v daných kloubech. Simscape dopočítává polohy pro klouby, u kterých je nastavený pohyb i moment/síla jako "automaticky vypočítaný". U ostatních, které nefungují jako aktuátory, je pouze nastavené automatické dopočítání pohybu.

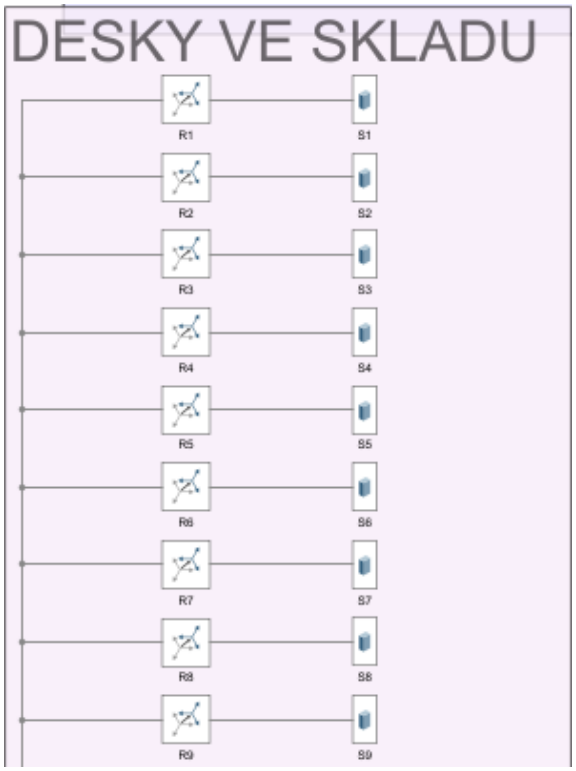
Součástí řídicího bloku jsou grafy vykreslující dosažené rychlosti a zrychlení v každém směru a natočení kolem osy z.



Obrázek 3.8: Řízení dle vytvořené trajektorie v Simscape

3.4 Automaticky generované desky ve skladu

Pro vytvoření věrného modelu, který reprezentuje fyzickou realitu a zároveň aktuální stav skladu, bylo nutné vytvořit skript pro automatické generování desek ve skladu. Algoritmus, kterému se práce bude věnovat později, vytváří desku na základě parametrů uložených v interní databázi. Každý "Brick Solid" označený na obrázku 3.9 jako S1, S2, ..., Sn představuje jednu desku aktuálně naskladněnou ve skladu. Rozměry, hustota a materiál jsou určeny typem desky z databáze. O správné umístění se stará "Rigid Transform", který je na obrázku 3.9 pojmenován jako R1, R2, ..., Rn, a obsahuje polohu a natočení.



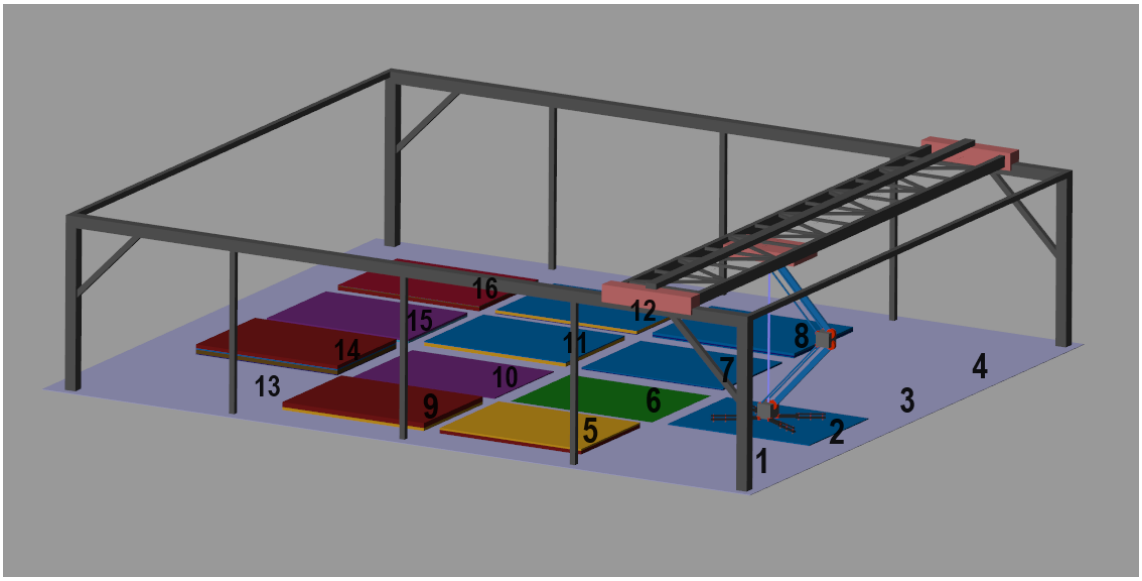
Obrázek 3.9: Automaticky generované desky v Simscape

Pokud dochází při simulaci k přemístění desky, tak dojde k odpojení této desky od "Rigid Transform" a napojení přímo na koncový bod manipulátoru. Po ukončení přesunu se opět připojí daná deska reprezentovaná "Brick Solid" k "Rigid Transform" a přepíšou se hodnoty určující polohu.

3.5 Vizualizace vytvořeného skladu

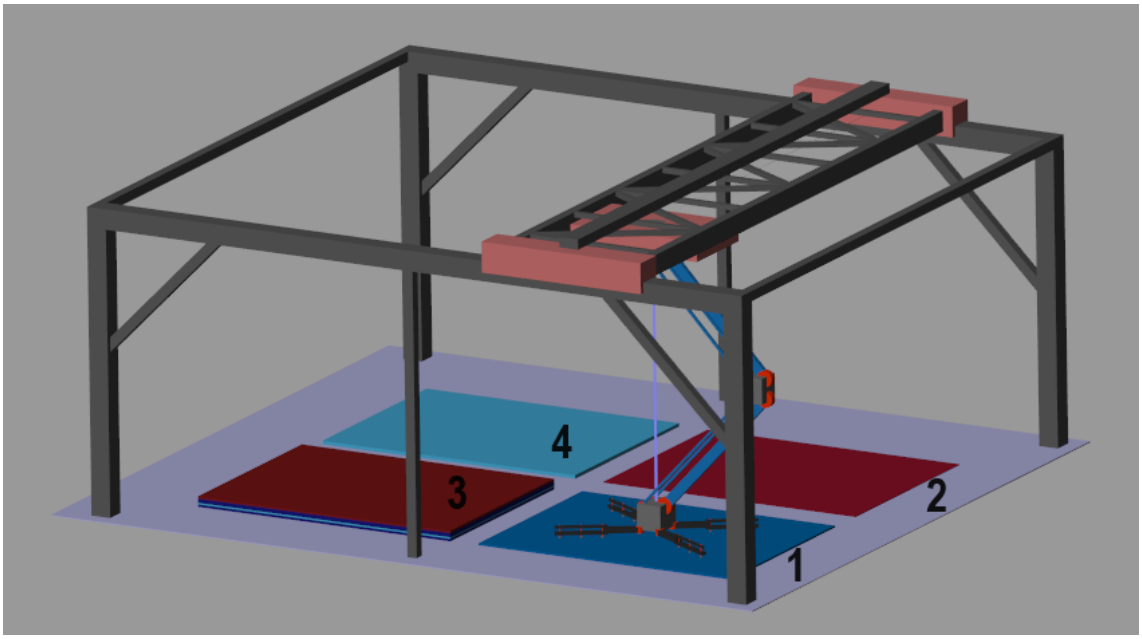
Na následujících obrázcích je zobrazena vizualizace celého simulačního modelu automatického skladu umožnujícího provádět virtuální ověřování vyvíjených řídicích algoritmů pro realizaci skladového procesu (naskladňování, vyskladňování, přemisťování, ...).

Obrázek 3.10 zachycuje skriptem automaticky vygenerovaný sklad se 12 skladovacími pozicemi (5-16), 2 vstupními pozicemi (1, 2), kde dochází k naskladnění a 2 výstupními pozicemi (3, 4), kde se desky zpracovávají.



Obrázek 3.10: Sklad 4x4

Na obrázku 3.11 je sklad 2x2, kde jsou pouze 2 skladovací pozice (3, 4), 1 naskladňovací (1) a 1 vyskladňovací (2).



Obrázek 3.11: Sklad 2x2

Kapitola 4

Software automatického skladu

Veškeré ovládání Simscape modelu probíhá skrze uživatelské rozhraní, které je naprogramované v Matlabu a bude popsáno v další kapitole. V Matlabu jsou dále vytvořeny skripty, které interface dle požadavků volá. Do Simscape modelu je vhodné zasáhnout pouze v případě chyby, jinak je celý tento model ovládán přímo z Matlabu včetně generování a odebírání funkčních bloků.

Součástí práce jsou funkce pro generování desek jak náhodně, tak automaticky z databáze či funkce pro napojení na databázi včetně čtení a zápisu aktuálních dat. Dále obsahuje funkce pro manuální přeskladnění dle jednoznačného ID a TYP ID (bude vysvětleno níže) nebo z konkrétní pozice na jinou. Kromě toho jsou součástí i funkce pro automatické přeskladnění dle naplánovaných pohybů, pro konfiguraci nového rozložení a rozměrů skladu, funkce pro definování nových typů desek a také pro vymazání všech desek ve skladu. Nejpodstatnější funkce budou popsány v další subkapitole, ostatní jsou součástí komentovaného kódu v příloze.

4.1 Interní databáze v Matlabu

Součástí této práce bylo vytvoření interní databáze, ve které jsou uvedeny všechny důležité parametry pro Simscape model.

Informace o každé uložené desce ve skladu jsou uloženy v Matlabu v maticové podobě v proměnné matice_desek, která je zobrazena na obrázku 4.1. Každá deska uložená ve skladu má své jedinečné ID, které je na prvním místě matice_desek, pomocí něhož jsme schopni monitorovat, jak dlouho ve skladu daná deska je. Zároveň slouží ve vizualizaci pro přenesení desky, protože podle jednoznačného ID je identifikován, v předchozí kapitole zmíněný, "Brick Solid" (S1, S2, ..., S**ID**) a jemu příslušný "Rigid Transform" (R1, R2, ..., R**ID**).

Ve druhém, třetím a čtvrtém sloupci matice je uvedena pozice desky. Jedná se o informaci, která je využívána v bloku "Rigid Transform" pro umístění ve skladu. Fyzikální vlastnosti a rozměry jsou definovány pro blok "Brick Solid". Každé desce se nastavuje délka, šířka a výška (5., 10., a 11. sloupec), materiál (6. sloupec, je určen číslem a dle materiálu se nastavuje barva) a hustota (8. sloupec).

Jedním z návrhových požadavků byla možnost sledovat dobu skladování desek. V 9. sloupci s názvem "čas" je pomocí funkce posixtime vytvořen časový záznam naskladnění desky. Posixtime slouží k převodu časových údajů na tzv. POSIX, což je způsob reprezentace času v sekundách od ledna 1970. [23]

ID	pozice x	pozice y	pozice z	výška	materiál	typ_ID	hustota	čas	délka	šířka
1	0	1.2000	0.0655	0.0150	10	16	840	1.7035e+09	2.7000	2
2	0	-1.2000	0.0325	0.0650	8	18	830	1.7035e+09	2.7000	2
3	-3	1.2000	0.0570	0.0300	3	3	780	1.7035e+09	2.7000	2
4	-3	-1.2000	0.0325	0.0650	8	18	830	1.7035e+09	2.7000	2
5	-3	1.2000	0.0210	0.0020	1	13	830	1.7035e+09	2.7000	2
6	0	-1.2000	0.0725	0.0150	2	2	700	1.7035e+09	2.7000	2
7	0	1.2000	0.0040	0.0080	7	6	720	1.7035e+09	2.7000	2
8	-3	-1.2000	0.0825	0.0350	6	11	850	1.7035e+09	2.7000	2
9	-3	1.2000	0.0320	0.0200	1	19	700	1.7035e+09	2.7000	2
10	-3	1.2000	0.0100	0.0200	1	19	700	1.7035e+09	2.7000	2
11	0	1.2000	0.0230	0.0300	3	3	780	1.7035e+09	2.7000	2
12	0	1.2000	0.0480	0.0200	1	19	700	1.7035e+09	2.7000	2
13	0	-1.2000	0.1550	0.0700	2	14	780	1.7035e+09	2.7000	2
14	-3	-1.2000	0.1925	0.0150	6	7	780	1.7035e+09	2.7000	2
15	-3	1.2000	0.0840	0.0200	1	19	700	1.7035e+09	2.7000	2
16	0	1.2000	0.1050	0.0200	1	1	850	1.7035e+09	2.7000	2
17	0	1.2000	0.0840	0.0220	8	9	750	1.7035e+09	2.7000	2
18	0	-1.2000	0.1000	0.0400	4	8	700	1.7035e+09	2.7000	2

Obrázek 4.1: Interní databáze v Matlabu, ve které jsou uloženy veškeré informace o naskladněných deskách

Mimo jednoznačné ID je v práci použito ještě TYP_ID, které definuje parametry společné deskám stejného typu. Při naskladňování desek je volen pouze typ desek (v podobě TYP_ID), který udává výšku, hustotu, materiál, délku a šířku desky. V externí databázi jsou pod typem desky uloženy ještě další informace, které však pro digitální model nejsou potřeba. Na obrázku 4.2 je vidět TYP_ID a vlastnosti, které mu jsou přiděleny. Například pro typ desky 1 (TYP_ID) je výška desky 0.02 m, hustota 850 kg/m^3, materiál 1 (v databázi označený celým číslem), délka desky 2.7 m a šířka 2 m.

typ_ID	výška	hustota	materiál	délka	šířka
1	0.0200	850	1	2.7000	2
2	0.0150	700	2	2.7000	2
3	0.0300	780	3	2.7000	2
4	0.0250	830	10	2.7000	2
5	0.1000	680	5	2.7000	2
6	0.0080	720	7	2.7000	2
7	0.0150	780	6	2.7000	2
8	0.0400	700	4	2.7000	2

Obrázek 4.2: Parametry přidělené různým typům desek (TYP_ID)

Pro snadné monitorování stavu skladu, zjednodušené vyhledávání desek ve stozích a také kvůli napojení na externí databázi existuje ještě pro každý stoh matice,

ve které je uloženo pořadí desek naskladněných na sobě. Tyto matice jsou uloženy v buňce (cell), která je o velikosti 1xN, kde N je počet stohů ve skladu včetně vstupních (naskladňovacích) a výstupních pozic. Na prvních místech jsou vždy VSTUPNÍ pozice, následují VÝSTUPNÍ pozice a poté teprve skladovací. Na obrázcích 4.3 a 4.4 jsou buňky schematicky uvedeny. Pozice obsahuje jednoznačné ID a pozice_typ_ID TYP_ID značící typ desky. Zatímco v první buňce, která obsahuje jednoznačné ID, se žádná čísla neopakují, tak v druhé buňce se opakovat mohou, protože ve skladu může být naskladněno více desek stejného typu.

Pozice

Vstupní p.	Výstupní p	Pozice 0	Pozice 1	Pozice 2	Pozice 3
13	14	[7;11;12;1	[2;6]	[10;5;9;3]	[4;8]

Pozice typ ID

Vstupní p.	Výstupní p	Pozice 0	Pozice 1	Pozice 2	Pozice 3
10	10	[6;3;19;16	[18;2]	[19;13;19;3	[18;11]

Obrázek 4.3: Sklad 3x2, jedna vstupní, jedna výstupní a 4 skladovací pozice

Pozice

Vstupní 1	Vstupní 2	Výstupní 1	Výstupní 2	Pozice 0	Pozice 1	Pozice 2	Pozice 3
17	18	19	[]	[3;6;15]	[16;8;11;7;	[1;14;5]	[2;10;4;13]

Pozice typ ID

Vstupní 1	Vstupní 2	Výstupní 1	Výstupní 2	Pozice 0	Pozice 1	Pozice 2	Pozice 3
1	12	7	[]	[19;8;6]	[13;16;9;15	[14;15;9]	[7;10;1;14]

Obrázek 4.4: Sklad 4x2, dvě vstupní, dvě výstupní a 4 skladovací pozice

4.2 Generování bloků do Simscape z Matlabu

V diplomové práci je potřeba automaticky generovat Simscape bloky z Matlab skriptu přímo do Simulink modelu. Vytváření bloků a také subsystémů je vyžadováno jak v případě přidání nových desek do skladu, tak při generování náhodného skladu nebo při konfiguraci nového skladu.

V následujícím odstavci bude představen zdrojový kód, který se stará o generování bloků "Rigid Transform" a "Brick Solid", nastavení příslušných parametrů z interní databáze a následné propojení mezi sebou a s modelem skladu. Při napojení na externí databázi dojde nejdříve k převedení informací do interní databáze a následně k vyčtení parametrů z ní.

K přidání bloků do modelu je využívána funkce add_block, která umožňuje vložit do Simulink prostředí kopii jakéhokoliv prvku z knihovny. Důležité je každý nový blok unikátně pojmenovat, také proto je v našem skladu využito jedinečné ID. V této diplomové práci je používána funkce add_block k vytvoření "Brick Solid" reprezentující desku a "Rigid Transform" definující její polohu. [24]

Vytvořeným blokům je nutné nastavit parametry a umístění v Simulink modelu pro přehlednost. K tomu je užita funkce set_param, která umožňuje nastavit parametry existujícím prvkům v modelu. Nejprve je potřeba definovat cestu ke konkrétnímu bloku, pro který se budou definovat parametry. Pro těleso "Brick Solid" se nastavují postupně rozměry, hustota, barva a umístění bloku v Simulink prostředí. V parametrech bloku "Rigid Transform" je nutné definovat polohu desky a pro přehlednost pozici bloku v Simulink modelu. [25]

V poslední části se vytvořené bloky propojí a napojí se přímo na existující "World Frame" pomocí funkce add_line. Je nutné specifikovat názvy bloků a jejich porty, které se budou spojovat. [26]

```
% Pridani Brick Solid pojmenovaneho dle jednoznacneho SID
    (S1, S3...)
add_block("sm_lib/Body Elements/Brick Solid", [model_path
    '/S' num2str(ID)]);
% Nastaveni parametru Brick Solid, rozmery, hustota,
   barva, umisteni
set_param([model_path '/S' num2str(ID)], 'BrickDimensions
   ', sprintf('[%f %f %f]', matice_desek(i,10),
   matice_desek(i,11), matice_desek(i,5)), 'position',
   [560  -125+65*i  580  -85+65*i], '
   GraphicDiffuseColor', sprintf('[%f %f %f]',
   brick_colour(1), brick_colour(2), brick_colour(3)), "
   Orientation", "left" ,'Density', num2str(matice_desek(
   i,8)));
% Pridani Rigid Transform pojmenovaneho dle jednoznacneho
    RID (R1, R3...)
add_block("sm_lib/Frames and Transforms/Rigid Transform",
    [model_path '/R' num2str(ID)]);
% Nastaveni parametru polohy x, y, z
set_param([model_path '/R' num2str(ID)], '
   Translationmethod', 'Cartesian', '
   TranslationCartesianOffset', sprintf('[%f %f %f]',
   matice_desek(i,2), matice_desek(i,3), -3+matice_desek(
   i,4)), 'position', [400  -125+65*i  440  -85+65*i],
    "Orientation", "right");
% Propojeni R a S
add_line(model_path, ['S' num2str(ID) '/RConn1'], ['R'
   num2str(ID) '/Rconn1']);
% Propojeni World Frame a R
add_line(model_path, ['R' num2str(ID) '/Lconn1'], 'World
   Frame/Rconn1', 'autorouting', "on");
```

Pomocí funkce delete_line a delete_block je možné automaticky mazat bloky a propojení mezi nimi. Je toho využíváno při vyskladnění desky, vymazání celého skladu a také při přenášení desek, protože nejdříve je odstraněno propojení s příslušným blokem "Rigid Transform", poté je deska připojena přímo na konec manipulátoru

a po přenesení desky je přepsán "Rigid Transform" jí příslušející a dochází k jejich opětovnému propojení. [27, 28]

```
delete_line(model_path, ['S' num2str(p) '/RConn1'], ['R'
   num2str(p) '/Rconn1']);
ziskani parametru propojeni z R, neslo to zde naprimo
   jako vys
h = get_param([model_path '/R' num2str(p)],'LineHandles')
   ;
vymazani propojeni R a WorldFrame
delete_line(h.LConn);
vymazani bloku S
delete_block([model_path '/S' num2str(p)]);
%vymazani bloku R
delete_block([model_path '/R' num2str(p)]);
```

4.3 Generování trajektorií pomocí funkce contopptraj

Pro generování trajektorie u automatického chaotického skladu byla použita funkce contopptraj. Funkce tvoří mezi zadanými body (waypoints) časově optimální trajektorii. Vytváří body popisující cestu pomocí pozice q, rychlosti qd a zrychlení qdd ve vzorcích času t. Tato trajektorie je omezena limity rychlosti (vellim) a limity zrychlení (accellim). [29]

Tato funkce je využita pro tvorbu tří lineárních pohybů. První fáze pohybu se skládá z vertikálního zdvihu v ose z do výšky o offset větší, než je výška nejvyššího stohu ve skladu. Dále následuje přejezd na požadovanou pozici v x a y beze změny v ose z a ve třetí části pohybu následuje vertikální pohyb dolů v ose z nad požadovanou desku. Generátor trajektorie má následující programovou strukturu:

```
function GenerovaniTrajektorie()
    global  robot natoceni amax vmax pocp endp pocp2
       endp2 xtraj ytraj ztraj natoceni konec_time

    % vypocitani trajektorie
    waypoints= [pocp(1) pocp2(1) ;pocp(2) pocp2(2) ;pocp
       (3) pocp2(3); 0 0 ];
    vellim=[-vmax vmax; -vmax vmax; -vmax vmax; -vmax
       vmax];
    accellim=[-amax amax; -amax amax; -amax amax; -amax
       amax];
    [trajektorie,qd,qdd,t] = contopptraj(waypoints,vellim
       ,accellim);
    waypoints= [pocp2(1) endp2(1) ;pocp2(2) endp2(2) ;
       pocp2(3) endp2(3); 0 0];
    [trajektorie2,qd2,qdd2,t2] = contopptraj(waypoints,
       vellim,accellim);
```

```
    waypoints= [endp2(1) endp(1) ;endp2(2) endp(2) ;endp2
       (3) endp(3); 0 0 ];
    [trajektorie3,qd3,qdd3,t3] = contopptraj(waypoints,
       vellim,accellim);

    % prepocitani casu
    t2=max(t)+t2;
    t3=max(t2)+t3;

    % spojeni dilcich trajektorii
    xtraj = [[t t2 t3]', [trajektorie(1,:) trajektorie2
       (1,:) trajektorie3(1,:)]'];
    ytraj = [[t t2 t3]', [trajektorie(2,:) trajektorie2
       (2,:) trajektorie3(2,:)]'];
    ztraj = [[t t2 t3]', [trajektorie(3,:)-robot.vyska_z
       trajektorie2(3,:)-robot.vyska_z trajektorie3(3,:)-
       robot.vyska_z]'];
    natoceni = [[t t2 t3]', [trajektorie(4,:)
       trajektorie2(4,:) trajektorie3(4,:)]'];

    % hledani vetsiho rozdilu nez dve sekundy mezi dvemi
       po sobe jdoucimi
    podmet_k_vymazani = find(diff([t t2 t3]) > 2);

    % pripadne odstraneni nesmyslne hodnoty
    t(podmet_k_vymazani + 1) = [];
    xtraj(podmet_k_vymazani + 1, :) = [];
    ytraj(podmet_k_vymazani + 1, :) = [];
    ztraj(podmet_k_vymazani + 1, :) = [];
    konec_time=max(t);

    %cas delky simulace
    konec_time=max(t3);
end
```

Funkce byla do Matlabu přidána teprve ve verzi R2022b a není ještě plně doladěná. Například u krátkých pohybů (o centimetry) funkce občas z neznámých důvodů vygenerovala sice správnou trajektorii, ale na konec této trajektorie přidala desítky sekund bez jakéhokoliv přejezdu. Tudíž docházelo k tomu, že po dokončení pohybu sklad čekal další desítky sekund bez jakékoliv změny, než došlo ke spuštění další simulace. Chyba se ve výstupu z generátoru trajektorií projevila velkým skokem dvou po sobě jdoucích časů (20 a více sekund). Z toho důvodu byla chyba vyřešena tím, že mezi dvěma po sobě jdoucími hodnotami času může být maximální rozdíl 2 s. Na posledním řádku obrázku 4.5 je chyba před přidáním hlídání podmínky 2 s vidět. [29]

Výstupem je trajektorie v ose x (xtraj), y (ytraj), z (ztraj) a natočení kolem osy z (natoceni). Na obrázku 4.5 jsou zobrazeny výstupy z funkce contopptraj.

xtraj		ytraj		ztraj		natoceni	
čas(s)	x(m)			čas(s)	z(m)	čas(s)	natoceni(rad)
2,545	3	2,545	-1,2	2,545	-2,999995	2,545	0
2,547	3	2,547	-1,2	2,547	-2,999999	2,547	0
2,549	3	2,549	-1,2	2,549	-2,999900	2,549	0
2,552	3	2,552	-1,2	2,552	-3,000000	2,552	0
86,55	3	86,55	-1,2	86,55	-3,000000	86,55	0

Obrázek 4.5: Vytvořená trajektorie včetně zmíněné chyby na posledním řádku

4.4 Skript pro automatické přenášení desek

Tato diplomová práce navazuje na diplomovou práci kolegy Ing. Jana Hrnčíře, který vytvořil optimalizační algoritmus pro přemísťování desek v automatickém chaotickém skladu. [30] V důsledku přímé návaznosti na jeho práci bylo nutné dodržet jím zvolené značení pozic. Vstupní a výstupní pozice jsou označeny záporným číslem, kdy vstupní pozice má vždy vyšší záporné číslo než pozice výstupní (např. vstupní -2, -3, výstupní -4, -5), skladovací pozice začínají od 0.

Jedná se komplexní a složitý skript, který je ve své úplnosti uveden v příloze. Zde jsou popsány pouze jeho nejvýznamnější funkcionality. Níže je uvedena část kódu sloužící k přeformátování dat pro realizaci automatického pohybu. Vstupní matice na obrázku 4.6 vlevo se označí jako A a číslo A(1,1) se uloží jako "pocatek".

Je-li "pocatek" kladný, tak se k této hodnotě pouze přičte 1 a délka IO_ID, ve které jsou uloženy hodnoty označující vstupní a výstupní pozice, např. $IO\_ID = [-2 - 3]$. Pokud bude v tomto případě např. pocatek=0, tak výsledná hodnota vstupující do automatického pohybu bude 3. Tudíž bude odkazovat na $pozice\{3\}$, ze které bude zjištěno jedinečné ID vrchní desky. Pomocí ID se následně vyčtou všechny uložené parametry této desky ve skladu.

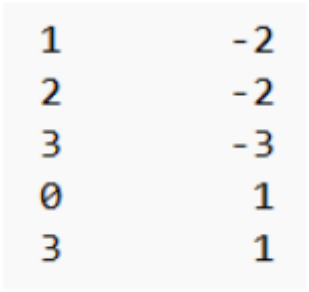
Pokud je "pocatek" záporný, tak víme, že se jedná o vstupní nebo výstupní pozici, protože jsou vždy označené záporným číslem. Výše bylo uvedeno, že vstupní a výstupní pozice jsou vždy na začátku buňky "pozice" a "pozice_typ_ID" obsahující matice, které ukazují pořadí desek na daném místě. Pokud budou hodnoty z matice A správně předpřipraveny, je možné odkázat na konkrétní matici v buňce 1xN, se kterou je spojena poloha daného stohu. Pokud bude hodnota pocatek=-2, tak po proběhnutí níže zmíněným kódem bude získáno číslo 1, tudíž bude odkazovat na $pozice\{1\}$.

Stejně budou upravena data pro koncové pozice.

```
%predpripraveni dat
if pocatek>=0
   pocatek=pocatek+length(IO_ID)+1;
else
 for o=1:length(IO_ID)
     if pocatek==IO_ID(o)
```

```
            pocatek=IO_ID(o)+abs(IO_ID(o))+o;
        else
            pocatek=pocatek;
        end
    end
end
```

Na pravé části obrázku 4.6 jsou již upravená data převedena do čitelnější podoby pro obsluhu.



```
1     -2
2     -2
3     -3
0      1
3      1
```

Počáteční pozice	Koncová pozice
Pozice 1	Vstupní 1
Pozice 2	Vstupní 1
Pozice 3	Výstupní 1
Pozice 0	Pozice 1
Pozice 3	Pozice 1

Obrázek 4.6: Sekvence pohybů dle návrhu optimalizačního algoritmu [?] nalevo a identická sekvence upravená v Matlabu

Z předpřipravených dat lze vyčíst x a y souřadnici pozice, kde je deska uložena a z matice_desek je možné zjistit souřadnici z těžiště vrchní desky. Dále následuje automatický pohyb nad tuto desku, aby mohla být uchopena. K zapnutí simulace by mělo dojít po vytvoření trajektorie příkazem SimOut a čas trvání by měl být nastaven dle času dojezdu do koncové polohy získaného z generování trajektorie. [31]

```
GenerovaniTrajektorie();
simOut = sim('
   RobotickyManipulator_Simscape_predkloubemnavic', '
   StartTime', num2str(0), 'StopTime', num2str(konec_time
   ));
pause(konec_time-1.5);
```

Po dojetí nad desku je potřeba ji uchopit. Simulace uchopení desky je realizována pomocí funkce add_line. Nejprve je odstraněno spojení mezi deskou a "Rigid Transform" jí příslušící a následně desku reprezentovanou "Brick Solid", která je následně spojena s koncovým bodem manipulátoru. Poté se vytvoří trajektorie a proběhne simulace přenesení desky z počáteční pozice na koncovou.

```
if deska==0
elseif deska==1
    delete_line(model_path, ['S' num2str(ID_deska_poc) '/
       RConn1'], ['R' num2str(ID_deska_poc) '/Rconn1']);
```

```
    add_line(model_path, ['S' num2str(ID_deska_poc) '/
       RConn1'], ['Rigid/Rconn1']);
end

GenerovaniTrajektorie();
simOut = sim('
   RobotickyManipulator_Simscape_predkloubemnavic' ,'
   StartTime', num2str(0), 'StopTime', num2str(konec_time
   ));
pause(konec_time-1.5);
```

Deska musí být po jejím přemístění opět spojena s "Rigid Transform", u kterého je nutné změnit parametry popisující umístění vůči "World Frame". Dále je potřeba změnit aktuální stav skladu, z původního stohu se deska odebere a bude přidána na poslední místo matice reprezentující koncový stoh. Současně je třeba proběhlou změnu zachytit v matici matice_desek.

```
if deska==0
elseif deska==1
    % pridani desky do stohu, kam jsme prenesli desku
    pozice{konec}=[pozice{konec};ID_deska_poc];
    pozice_typ_ID{konec}=[pozice_typ_ID{konec};
       matice_desek(ID_deska_poc,7)];
    pozice_time{konec}=[pozice_time{konec};matice_desek(
       ID_deska_poc,9)];
    % odebrani desky do stohu, odkud jsme ji prenaseli
    if length(pozice{pocatek})>1
        pozice{pocatek}=pozice{pocatek}(1:end-1,1);
        pozice_typ_ID{pocatek}=pozice_typ_ID{pocatek}(1:
           end-1,1);
        pozice_time{pocatek}=pozice_time{pocatek}(1:end
           -1,1);
    else
        pozice{pocatek} = [];
        pozice_typ_ID{pocatek}=[];
        pozice_time{pocatek}=[];
    end
    % odpojeni od manipulatoru, pripojeni na Rigid
       transform a upraveni pozice prenesene desky
    delete_line(model_path, ['S' num2str(ID_deska_poc) '/
       RConn1'], ['Rigid/Rconn1']);
    add_line(model_path, ['S' num2str(ID_deska_poc) '/
       RConn1'], ['R' num2str(ID_deska_poc) '/Rconn1']);
    matice_desek(ID_deska_poc,2:4)=[endp(1), endp(2),
       vyska_end_set];
    set_param([model_path '/R' num2str(ID_deska_poc)], '
       TranslationCartesianOffset', sprintf('[%f %f %f]',
        matice_desek(ID_deska_poc,2), matice_desek(
```

```
        ID_deska_poc,3), -3+vyska_end_set) );
end
```

Tento skript je v programu použit jak pro plně automatické přeskladnění dle sekvence na obrázku 4.6, tak pro manuální vyskladnění dle ID nebo TYP_ID. Také je možné manuální přeskladnění při výběru počáteční a koncové pozice.

4.5 Propojení a komunikace Matlabu s Pythonem

Digitální model automatického skladu funguje ve dvou režimech.

První režim funguje bez napojení na externí databázi a všechna data se ukládají pouze do interní databáze. Při řízeném vypnutí skladu, se veškeré údaje definující jeho stav v okamžiku vypnutí uloží a při zapnutí dojde k načtení posledního stavu. V případě vymazání všech desek ve skladu již není možné se k původnímu stavu vrátit.

Druhý režim je při zapnuté externí databázi. Čtení a zápis do databáze probíhá skrze Python, který voláme externě z Matlabu. Jednou z výhod připojení na databázi je možnost načíst i starší archivované verze skladu.

Python skript vyvolaný z Matlabu realizuje postupně tři operace: načtení stavu skladu, načtení uložených TYP_ID a zapisování aktuálního stavu do databáze po každém přesunu. Propojení je založené na vytvoření dočasného Python skriptu, do kterého se napíší požadované příkazy. Ty se budou lišit u všech tří operací. Vytvořený dočasný skript se následně spustí v příkazovém řádku. Pokud jsou data čtena, je potřeba po získání dat výstup z Pythonu zpracovat do proměnných v Matlabu a vybrat pouze hodnoty, které jsou potřeba do digitálního modelu. V případě zápisu je nutné data zkonvertovat do požadovaného formátu, ve kterém lze data zapsat do interní databáze. Níže je uvedena funkce v Matlabu, která získává informace o aktuálním stavu skladu.

```
function sklad= getStorage()
try
    % vytvoreni docasneho python skriptu
    tempScriptPath = 'C:\Users\Marek\Desktop\diplomka_V2\
       temp_script.py';

    % importovani vystupu do pythonu, pripojeni na
       databazi, ziskani info
    pythonCommands = [
        "import hfdb.StorageData as storage", ...
        "import json", ...
        "storage.connect()", ...
        "sklad = storage.getStorage()", ...
        "result = {'Sklad': sklad}", ...
        "print(json.dumps(result))", ...
    ];
```

```
    fid = fopen(tempScriptPath, 'w');
    fprintf(fid, '%s\n', pythonCommands{:});
    fclose(fid);

    % spusteni pythonu z prikazoveho radku
    commandLine = ['python "', tempScriptPath, '"'];
    [status, result] = system(commandLine);

    % zpracovani vystupu do matlabu
    jsonData = jsondecode(result);
    % uprava a vybrani potrebnych dat
    sklad=jsonData.Sklad;
    sklad=sklad(:,3:end);

    % smazani docasneho souboru
    delete(tempScriptPath);

catch exception
    disp(getReport(exception));
end
end
```

Obdobně jako u funkce getStorage() je možné získat informace o uložených TYP_ID a také zapsat aktuální stav skladu do databáze. Obě funkce jsou součástí přílohy.

4.6 Uživatelské rozhraní pro ovládání digitálního modelu v Matlabu

Uživatelské rozhraní bude detailně popsáno v další kapitole. V této kapitole bude popsán kód, který rozhraní vytváří, a budou charakterizovány nejpoužívanější funkce ve vytvořeném uživatelském rozhraní.

Prvním krokem je vytvoření samotného okna, ze kterého je možné digitální model ovládat. Nastaví se velikost okna a pozice, kde se bude okno otevírat.

```
%Vytvoreni noveho okna pro hlavni menu, pojmenovaneho
   Automaticky sklad
mainFig = uifigure('Name', 'Automaticky sklad');
% Nastaveni pozice x,y a velikost okna
mainFig.Position(1) = 100;
mainFig.Position(2) = 200;
mainFig.Position(3) = 600;
mainFig.Position(4) = 600;
% barva pozadi
mainFig.Color = '#DAE6FA';
```

V rámci implementace uživatelského rozhraní v Matlabu byly použity různé ovládací prvky, které umožňují interakci uživatele s aplikací. Nejčastěji použitým ovládacím prvkem jsou tlačítka, která jsou integrována do hlavního okna uživatelského rozhraní. Níže je ukázka použitého tlačítka s popisem "Naskladnění desek", které při stisknutí spustí funkci "NaskladneniDesek".

```
% tlacitko naskladneni desek
naskladneniDesekButton = uibutton(mainFig, 'Text','
   Naskladnenidesek', 'Position', [50, 330, 200, 30]);
% akce pri stistknuti tlacitka
naskladneniDesekButton.ButtonPushedFcn = @(~,~)
   OtevritNaskladneniDesek();

function OtevritNaskladneniDesek()
    % spusti funkci naskladneni desek
    NaskladneniDesek();
end
```

Dalším použitým prvkem jsou slidery, pomocí kterých se dá nastavit natočení vysouvatelných ramen držící přísavky a jejich vysunutí.

```
% popisek slideru
uilabel(mainFig, 'Text', 'Nastaveni vysunuti prisavek', '
   Position', [355, 240, 200, 20]);
% vytvoreni posuvneho pole (slideru), nastaveni jmena,
   pozice..
slider = uislider(mainFig, 'Position', [330, 230, 200,
   3]);
% nastaveni rozsahu
slider.Limits = [0 0.2];
% nastaveni pocatecni hodnoty slideru
slider.Value = 0.2;
slider.MajorTicks = 0:0.05:0.2;
% akce v pripade zmeny hodnoty na slideru
slider.ValueChangedFcn = @(~, event) UlozitDoWorkspace('
   Vysunuti', event.Value);

 % funkce, ktera ulozi hodnotu do workspace
function UlozitDoWorkspace(nazev, hodnota)
    assignin('base', nazev, hodnota);
end
```

V implementaci ovládání stavu databáze byl využit přepínač (switch), který umožňuje přepínání mezi aktivním a neaktivním stavem databáze. V situaci, kdy je databáze vypnutá, není umožněno provádět načítání uložených typů desek z databáze, stejně tak není možné načíst aktuální stav skladu z databáze. Aktualizace stavu skladu probíhá pouze v interní databázi v Matlabu. Pokud dojde k zapnutí/vypnutí databáze, tak se všechny desky ve skladu vymažou a je potřeba je znovu vygenerovat/načíst. Důležité je to zejména kvůli možným nesrovnalostem s TYP_ID, které

se u režimu "Databáze ON" a "Databáze OFF" liší.

```
% Vytvoreni prepinace s dvema polohami
databazovyPrepinac = uiswitch(mainFig, 'Position', [110,
   30, 200, 30], 'Items', {'Databaze OFF', 'Databaze ON'
   });
    % Nastaveni akce pri zmene stavu prepinace
databazovyPrepinac.ValueChangedFcn = @(src, event)
   ZmenaDatabazovehoRezimu(src);

function ZmenaDatabazovehoRezimu(src)
    % Funkce, ktera se spusti pri zmene stavu prepinace
    switch src.Value
        case 'Databaze ON'
            evalin('base', 'databaze = 1;');
            typ_ID=getDesks();
            vymazanidesek;
        case 'Databaze OFF'
            evalin('base', 'databaze = 0;');
            load typy_desek;
            vymazanidesek;
    end
end
```

Při vypnuté databázi je možné si definovat vlastní typ desky (zvolit rozměry, hustotu, materiál, ...). Proto je potřeba vytvořit v uživatelském rozhraní editovatelnou tabulku, do které lze vepsat vlastní hodnoty. Naopak při zapnuté databázi nesmí být možné definovat vlastní typ desky. Tudíž TYP_ID musí být pouze k zobrazení. Pokud bude v níže zobrazeném kódu změněna vlastnost "ColumnEditable" na false, tak bude tabulka otevřena pouze v režimu pro náhled.

```
% Vytvoreni tabulky v uzivatelskem rozhrani
tableUI = uitable(fig, 'Data', deskData, 'ColumnName', {'
   typ_ID', 'vyska', 'hustota', 'material', 'delka', '
   sirka'}, 'ColumnEditable', [true, true, true, true,
   true, true], 'ColumnFormat', {'numeric', 'numeric', '
   numeric', 'numeric', 'numeric','numeric'}, 'Position',
    [20, 90, 360, 470], 'RowName', [], 'CellEditCallback'
   , @aktualizovatData);

tableUI.ColumnWidth = {50, 60, 60, 50, 60, 60};
```

Do uživatelského rozhraní byl začleněn notifikátor, který indikuje stav skladu prostřednictvím své barvy. Pokud svítí červeně, indikuje to, že sklad je aktuálně v režimu zaneprázdněn. Tento stav signalizuje probíhající operace, jako je přeskladňování desek, generování nových desek do skladu nebo další probíhající procesy. Naopak zeleně svítící indikátor signalizuje sklad, který je připraven na další operace.

```
% ukazal stavu skladu
```

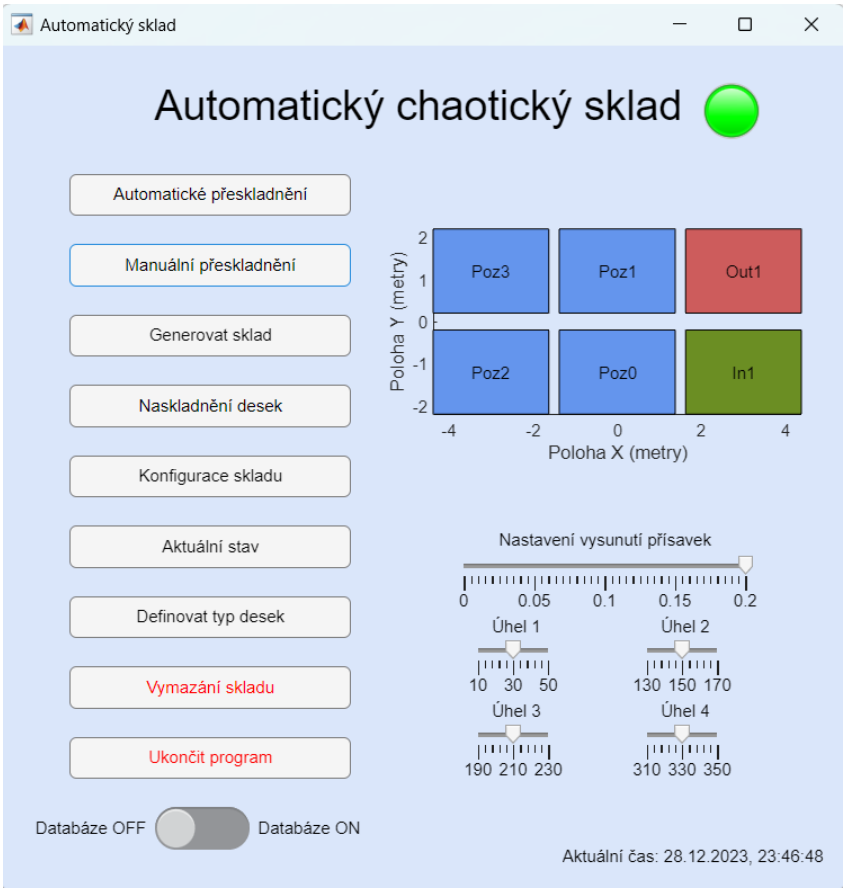
```
ukazatelBehu= uilamp(mainFig, 'position',  [500, 535,
   400, 40]);
% nastaveni barvy notifikacni diody
set(ukazatelBehu, 'Color', [1, 0, 0]);
drawnow;
```

Kapitola 5

Uživatelské rozhraní

V této kapitole bude popsáno grafické uživatelské rozhraní, které slouží k ovládání digitálního modelu automatického skladu. Na obrázku 5.1 je zobrazeno hlavní okno uživatelského rozhraní.

Grafické rozhraní lze otevřít spuštěním skriptu HlavniOkno.m v Matlabu. Současně s otevřením hlavního okna rozhraní dojde též ke spuštění Simscape modelu automatického skladu a dojde k načtení a vykreslení poslední uložené verze (k ukládání dochází automaticky po jakékoliv akci ve skladu).

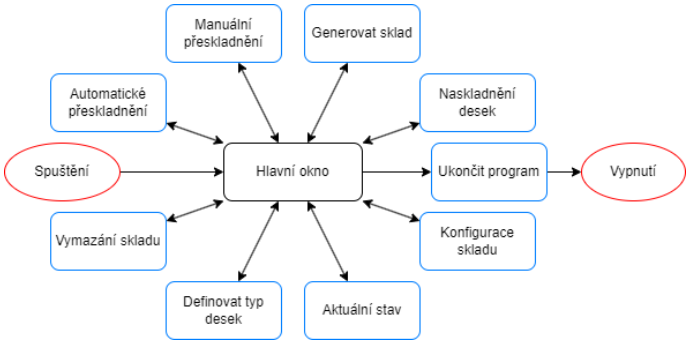


Obrázek 5.1: Hlavní okno uživatelského rozhraní

Z hlavního okna uživatelského prostředí lze spustit automatické nebo manuální přeskladnění. Dále je možné náhodně vygenerovat desky (pro testování) či z databáze načíst stav skladu. Pro naskladnění desek do skladu je vytvořené samostatné tlačítko, které otevře další okno, kde lze navolit typ desek a jejich počet. Konfigurace skladu se použije pouze v situaci, kdy je zapotřebí vytvořit sklad o jiné velikosti nebo s jiným rozložením. V případě potřeby vidět aktuální stav skladu je zde tlačítko, které otevře nové okno, ve kterém se zobrazí aktuálně naskladněné desky na každé pozici. Pokud není systém napojen na externí databázi, tak je možné definovat vlastní typ desky kliknutím na tlačítko "Definovat typ desek". V případě, že je připojení k databázi aktivní, tak je možné pouze nahlédnout na již definované typy desek. Poslední dvě tlačítka slouží k vymazání celého skladu a ukončení programu, které zavře uživatelské rozhraní.

Mimo funkční tlačítka zde jsou ještě "slidery", pomocí nichž se nastavuje vysunutí přísavek a natočení ramen, na kterých jsou přísavky umístěny.

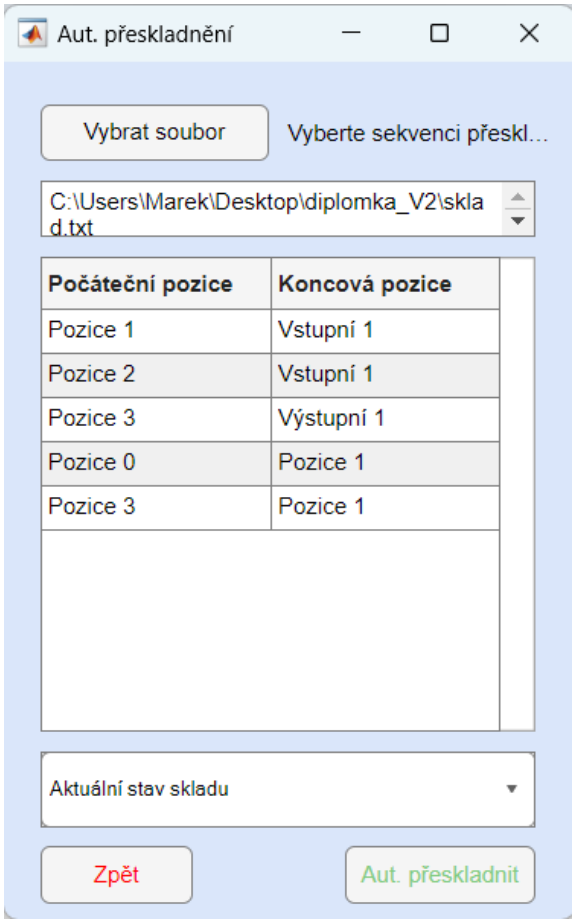
Pro přehlednost je vedle názvu umístěn indikátor aktuálního stavu skladu a pod ním je graf, který zobrazuje aktuální velikost skladu a rozložení pozic.



Obrázek 5.2: Stavový diagram hlavního menu

5.1 Automatické přeskladnění

Automatické přeskladnění je proces, během něhož skladový manipulátor provede posloupnost operací přeskládání na základě předem definované sekvence pohybů. Po kliknutí na automatické přeskladnění v hlavním okně se otevře nové okno, které je zobrazeno na obrázku 5.3.



Obrázek 5.3: Okno k automatickému přeskladnění

Funkcionalita **automatického přeskladnění** je implementována v programu v několika krocích, které jsou zobrazeny ve stavovém diagram na obrázku 5.4 a popsány níže.

1. **Výběr souboru:** Uživatel musí nejprve vybrat platný soubor obsahující sekvenci přeskladnění. Soubor obsahuje dva sloupce, v prvním je informace o počáteční pozici, odkud je třeba desku vzít. V druhém je číslo označující pozici, kam bude deska z počátečního umístění přesunuta.

2. **Zpracování souboru:** Vybraný soubor je načten a automaticky zpracován tak, aby bylo možné sledovat sekvenci pohybů, kterou sklad provede.

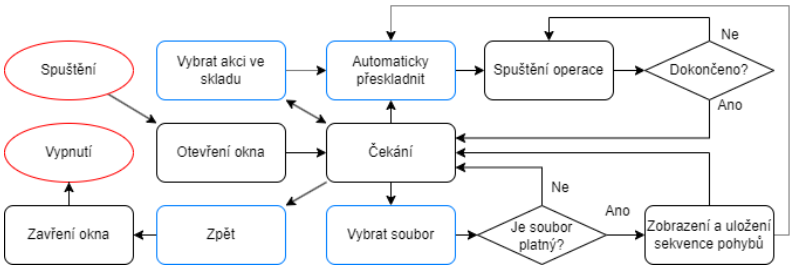
3. **Výběr možnosti:** Uživatel má možnost se rozhodnout, zda chce sekvenci pohybů aplikovat na standardně nastavený aktuální stav skladu nebo jinou možnost z výběru (vhodné zejména pro testování) :

    - **Aktuální stav skladu:** Sklad provede operace na základě aktuálního stavu skladu.
    - **Vymazat a generovat desky:** Celý sklad je vymazán a poté jsou náhodně vygenerovány nové desky ve skladu.

- **Generovat nové desky:** K existujícím deskám ve skladu jsou přidány náhodně vygenerované nové desky.
- **Načíst desky z databáze:** Desky jsou načteny z databáze.

4. **Spuštění operací:** Operace přemístění desky je provedena na základě vstupních informací ze sekvence přeskladnění. Robotický manipulátor se pohybuje a manipuluje s deskami podle specifikací daných v sekvenci. Níže bude funkce spouštějící přeskladnění z počáteční na koncovou pozici představena blíže.

5. **Konec:** Proces automatického přeskladnění končí po provedení všech operací definovaných v sekvenci.



Obrázek 5.4: Stavový diagram automatického přeskladnění

Funkce **"Spuštění operace"** zahájí proces přesunu desky ve skladu. Součástí jsou také automatické přesuny z aktuální pozice na souřadnice, kde má být deska uchopena, protože tyto informace nejsou obsaženy ve výstupu od kolegy Ing. Jana Hrnčíře. [30] Níže je detailně tato funkce popsána a stavový diagram je vykreslen na obrázku 5.5.

1. **Načtení proměnných:** Na samotném začátku je potřeba funkci zpřístupnit globální proměnné a také načíst její vstupní parametr, který obsahuje počáteční a koncovou pozici.

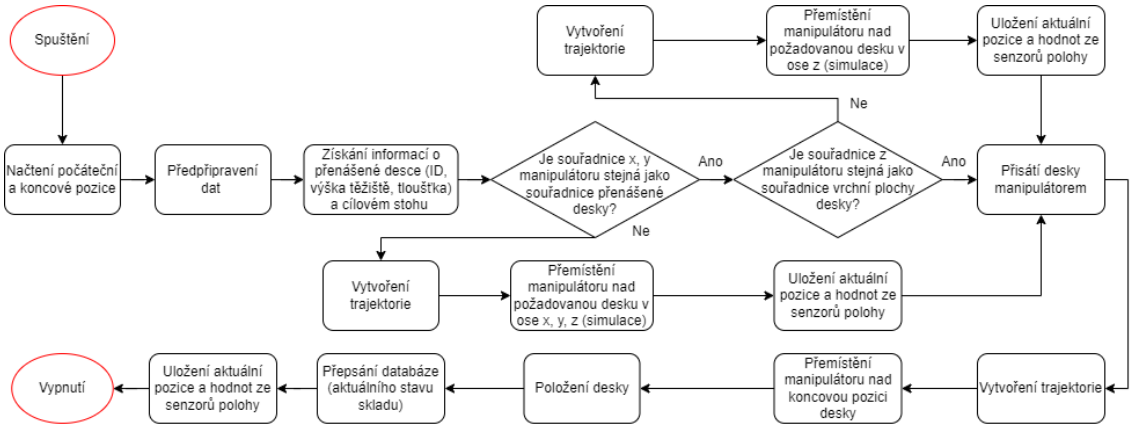
2. **Příprava dat:** Vzhledem k tomu, že je k dispozici pouze informace v podobě čísla udávající počáteční a koncový stoh, tak je potřeba zjistit z interní databáze několik informací. Výšku stohu, ze kterého má být deska odebrána a také výšku koncového stohu, kam bude deska přenesena, tloušťku a ID desky, která bude přemístěna.

3. **Vytvoření trajektorie:** Pokud je známá výška počátečního a koncového stohu a také tloušťku desky, jsou známy všechny potřebné informace k vytvoření trajektorie.

4. **Odsimulování pohybu:** Po vytvoření trajektorie pohybu dojde k odsimulování pohybu manipulátoru ve skladu.

5. **Uložení dat:** Po ukončení simulace jsou odečteny koncové hodnoty ze senzorů (natočení, posuv), uložena aktuální pozice souřadnic x, y, z a koncové hodnoty ze senzorů, které jsou využity jako počáteční podmínky pro další simulaci a je přepsán stav skladu v interní databázi.

6. **Uložení Simscape modelu:** Na závěr dojde k uložení změn v Simulink modelu a uložení dat do externí databáze, pokud je zapnutá.



Obrázek 5.5: Stavový diagram spuštění operace

5.2 Manuální přeskladnění

Cílem kódu je umožnit operátorovi manuálně přeskladnit nebo vyskladnit desky ve skladu. To zahrnuje výběr konkrétní desky na základě ID nebo TYP_ID a následně přesunutí této desky na vybranou výstupní pozici. Při volbě TYP_ID je zvolena vždy nejdostupnější deska, tj. deska zvoleného typu, pro kterou je potřeba při vyskladnění vykonat nejméně pohybů. Při výběru desky je možné zvolit pouze aktuálně naskladněné desky. Dále je možné manuální přemístění desky z jedné pozice na druhou (přeskladnění).

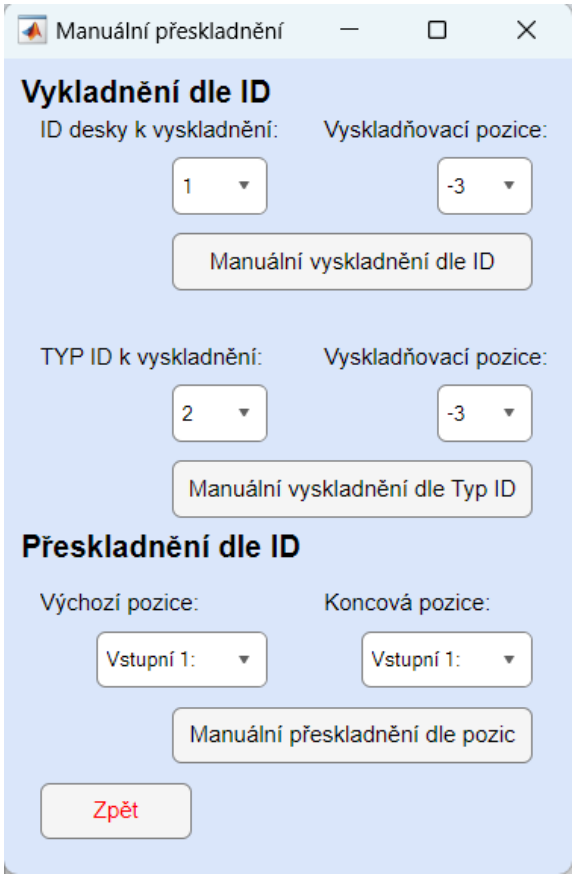
Celé manuální ovládání skladu probíhá skrze okno na obrázku 5.6, které se aktivuje kliknutím na "Manuální přeskladnění" v hlavním menu.

Stavový diagram popisující funkci **manuální přeskladnění** je vyobrazen na 5.7 a základní funkce jsou popsány níže.

1. **Vyskladnění desky podle ID:** Operátor může vybrat konkrétní desku z rozbalovacího seznamu dle ID a také určit, na jakou pozici se má tato deska vyskladnit. Pokud nad vyskladňovanou deskou jsou desky jiné, dojde k náhodnému přemístění těchto desek na jiné skladovací pozice. O samotný pohyb, uložení informací do databází a "přisátí" desky se stará funkce "Spuštění operace".

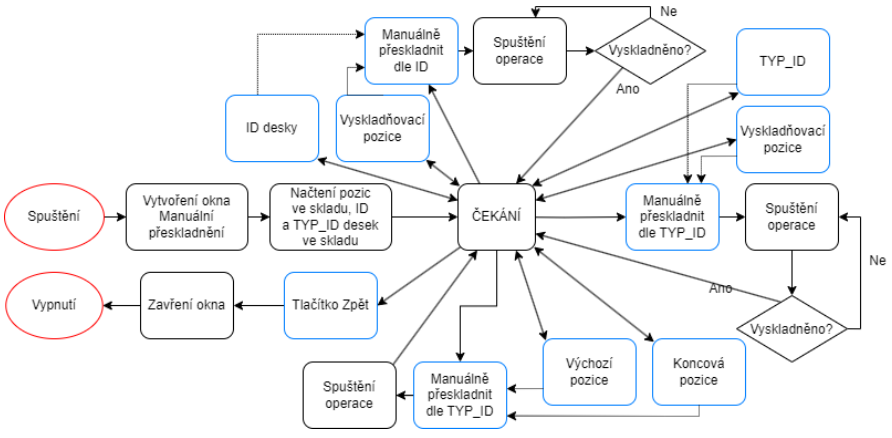
2. **Vyskladnění desky podle TYP_ID:** Operátor může vybrat typ desky z rozbalovacího seznamu dle TYP_ID a určit, na jakou pozici se má tento typ desky vyskladnit. Automaticky je v kódu vybrána deska, která pro vyskladnění vyžaduje nejméně pohybů. Pokud nad vyskladňovanou deskou jsou desky jiné, dojde k náhodnému přemístění těchto desek na jiné skladovací pozice.

O samotný pohyb, uložení informací do databází a "přisátí" desky se stará funkce "Spuštění operace".



Obrázek 5.6: Okno sloužící k manuálnímu přeskladnění a vyskladnění desek

3. **Manuální přeskladnění:** V manuálním režimu je dále možné přeskladnit desku z jedné pozice na jakoukoliv jinou pozici. V tomto případě dochází k výběru počáteční a koncové pozice. O samotný pohyb, uložení informací do databází a "přisátí" desky se stará funkce "Spuštění operace".

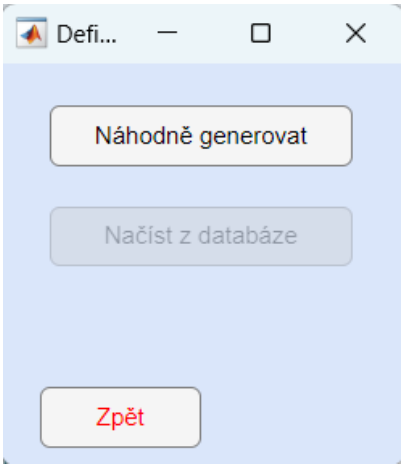


Obrázek 5.7: Stavový diagram manuálního přeskladnění

5.3 Generování skladu

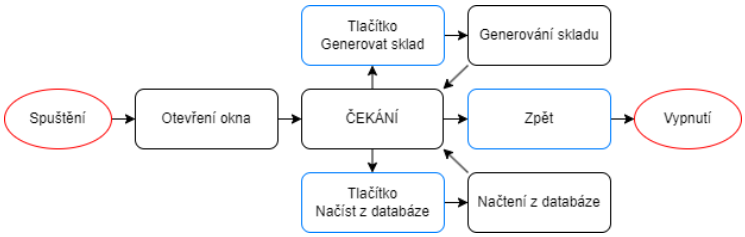
Funkce generování skladu slouží zejména pro testování algoritmu při různých stavech skladu. Při kliknutí na tlačítko generování skladu se otevře okno 5.8 a objeví se možnosti "Náhodně generovat" a "Načíst z databáze", pokud je systém připojen k externí databázi. Při náhodném generování dojde k vytvoření N (v závislosti na velikosti skladu) různých desek, jejichž parametry byly voleny z uložených TYP_ID. Při připojené databázi je TYP_ID načteno z externí databáze, jinak je bráno z interní paměti v Matlabu.

Při načtení stavu skladu z databáze jsou automaticky použita TYP_ID uložená v databázi a dojde k načtení skladu z posledního zápisu v databázi.



Obrázek 5.8: Generování skladu s nepřipojenou ext. databází

Stavový diagram pro generování desek ve skladu je na obrázku 5.9.



Obrázek 5.9: Stavový diagram pro generování desek ve skladu

Tlačítko "Náhodně generovat" spouští funkci **generování skladu**, která je popsána stavovým diagram na obrázku 5.10.

1. **Načtení proměnných:** Kód začíná zpřístupněním globálních proměnných této funkci, aby z nich funkce mohla informace nejen číst, ale i zapisovat.

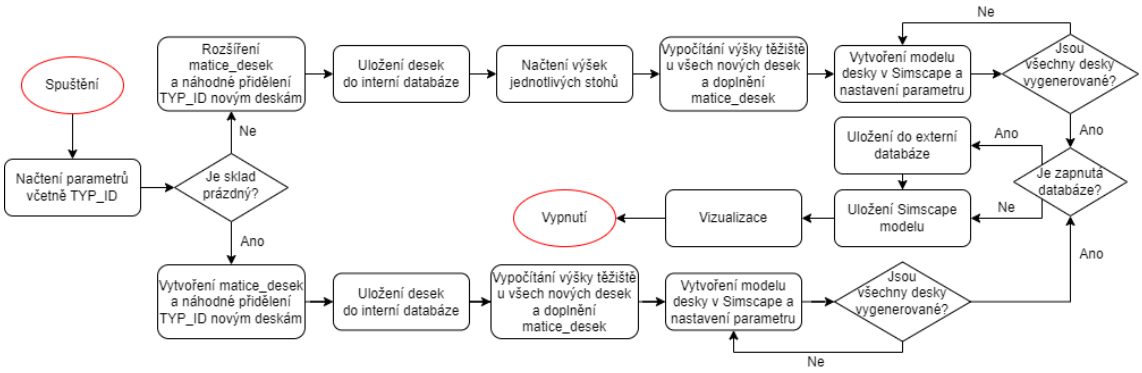
2. **Zjištění počátečního stavu:** Existují dva možné počáteční stavy. Pokud je sklad prázdný, počáteční výšky všech stohů jsou nulové a značení desek dle

jedinečného ID začíná od 1. Druhým možným stavem je dogenerování desek do již naplněného skladu. Poté je potřeba zjistit výšky všech stohů, tloušťku desky na vrchu stohu a začít číslovat desky od nejvyššího již použitého ID.

3. **Generování parametrů desek:** Když je sklad prázdný, kód vytvoří matici desek (matice_desek) a umístí desky na náhodné skladovací pozice. Každé desce je přiřazeno jedinečné ID a jsou definovány jejich vlastnosti, jako je tloušťka, materiál, hustota, délka a šířka. Pokud sklad prázdný není, dojde k rozšíření matice desek o počet generovaných desek a dále je proces stejný.

4. **Vytvoření modelů desek:** Desky jsou následně přidány do simulačního modelu v Simulinku. Každá deska je reprezentována pomocí bloku "Brick Solid" a "Rigid Transform" v Simscape modelu.

5. **Uložení desek do interní a externí databáze:** Do interní databáze jsou informace o deskách ukládány v průběhu a na konci dojde k uložení do souboru, aby bylo možné po znovuotevření skladu pokračovat přesně tam, kde se skončilo. Pokud je navíc zapnutá databáze, dojde k uložení aktuálního stavu skladu do externí databáze.



Obrázek 5.10: Stavový diagram pro náhodné generování desek ve skladu

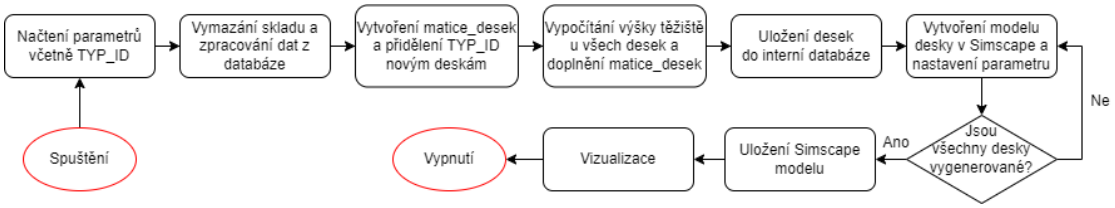
Při zapnuté databázi a kliknutí na tlačítko "Načíst z databáze" se spustí funkce **načtení skladu z databáze**, která je schematicky zobrazená na obrázku 5.11.

1. **Načtení dat z databáze a proměnných:** Nejprve je potřeba pomocí funkce getStorage(), která je popsána v kapitole o propojení Matlabu s Pythonem, načíst data z externí databáze. Zároveň je třeba zpřístupnit této funkci proměnné v Matlabu pro čtení a zápis.

2. **Zapsání parametrů desek:** Při načítání desek z databáze se vychází vždy z prázdného skladu. Funkce vytvoří matici desek (matice_desek) a dle získaných informací z databáze přidělí do této matice hodnoty (ID, TYP_ID, hustota, materiál, ...).

3. **Vytvoření modelů desek:** Desky jsou následně přidány do simulačního modelu v Simulinku. Každá deska je reprezentována pomocí bloku "Brick Solid" a "Rigid Transform" v Simulink modelu.

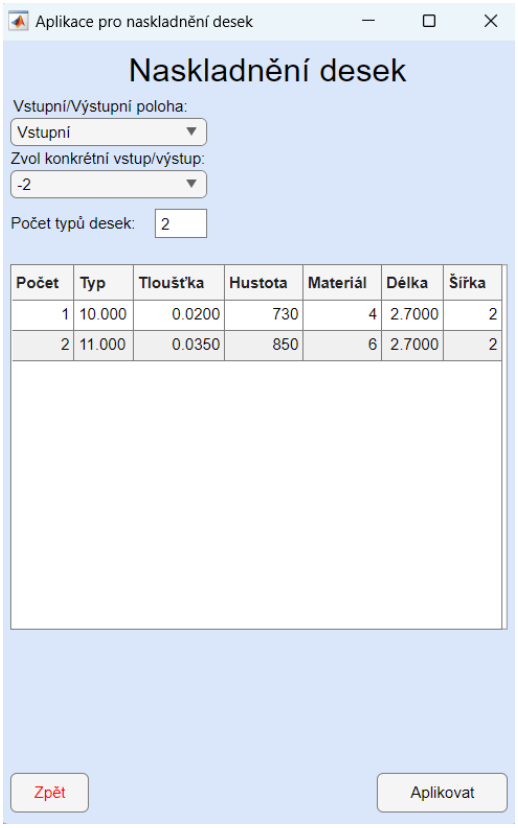
4. **Uložení desek do interní databáze:** Do interní databáze jsou informace o deskách ukládány v průběhu a na konci dojde k uložení do souboru, aby bylo možné po znovuotevření skladu pokračovat přesně tam, kde se skončilo.



Obrázek 5.11: Stavový diagram pro načtení desek z externí databáze

5.4 Naskladnění desek

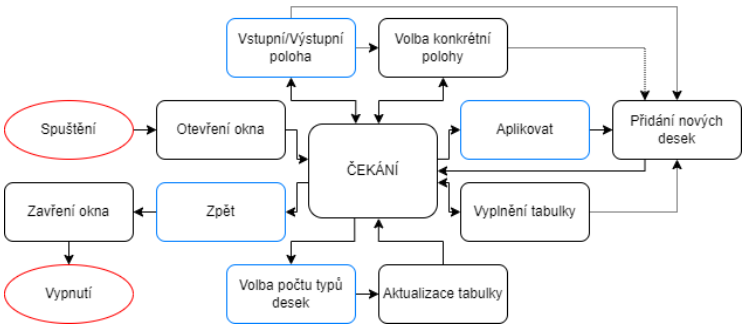
Při aktivaci pokynu "Naskladnění desek" v hlavním menu se otevře okno, které je vyobrazeno na obrázku 5.12. Slouží k naskladnění desek do skladu, což bude u fyzického dvojčete jediná možná cesta, jak umístit desky do skladu.



Obrázek 5.12: Okno pro naskladnění desek

Při naskladnění desek je nutné vybrat konkrétní vstupní, případně výstupní pozici, a poté zvolit počet typů desek. Následně se aktualizuje tabulka a je možné vyplnit počet desek a typ desek, které chceme naskladnit. Ostatní parametry, jako je

tloušťka, hustota a další, jsou pouze informativní a nedají se měnit. Na obrázku 5.13 je znázorněn stavový diagram naskladnění desek.



Obrázek 5.13: Stavový diagram pro naskladnění desek do skladu

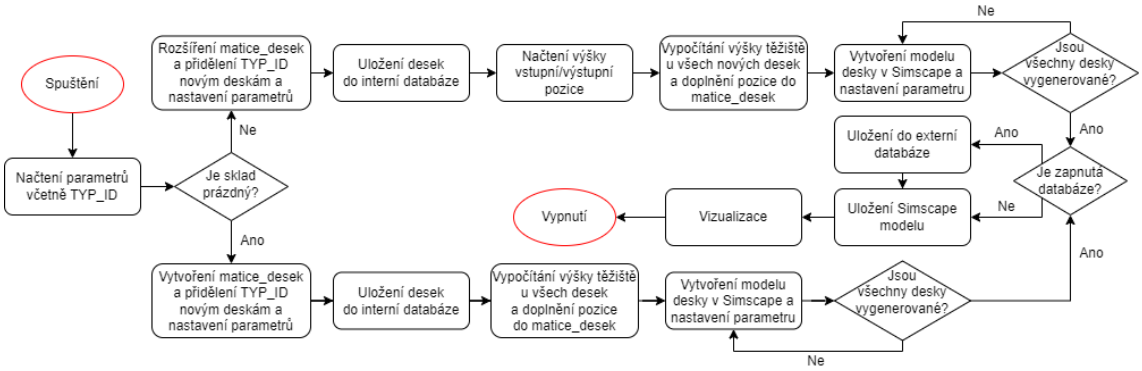
O samotné vytvoření desek a uložení těchto desek do interní a případně i externí databáze se stará funkce **přidání nových desek**, která je schematicky znázorněna na obrázku 5.14 a popsána níže.

1. **Načtení a příprava dat:** Funkce přijímá dva vstupní argumenty: data z tabulky z obrázku 5.12 a informaci o poloze, kam budou desky naskladněny. Data z tabulky je potřeba upravit, protože jsou ve formě "String" a je nutné je převést do numerické podoby. Dále jsou zpřístupněny potřebné globální proměnné.

2. **Zapsání parametrů desek:** Nejprve dojde k rozšíření matice desek, do které zapíšeme naskladněné desky a jejich parametry. Poté dojde k vypočtení souřadnice z těžiště pro každou naskladněnou desku.

3. **Vytvoření modelů desek:** Nově naskladněné desky se vytvoří v prostředí Simulink.

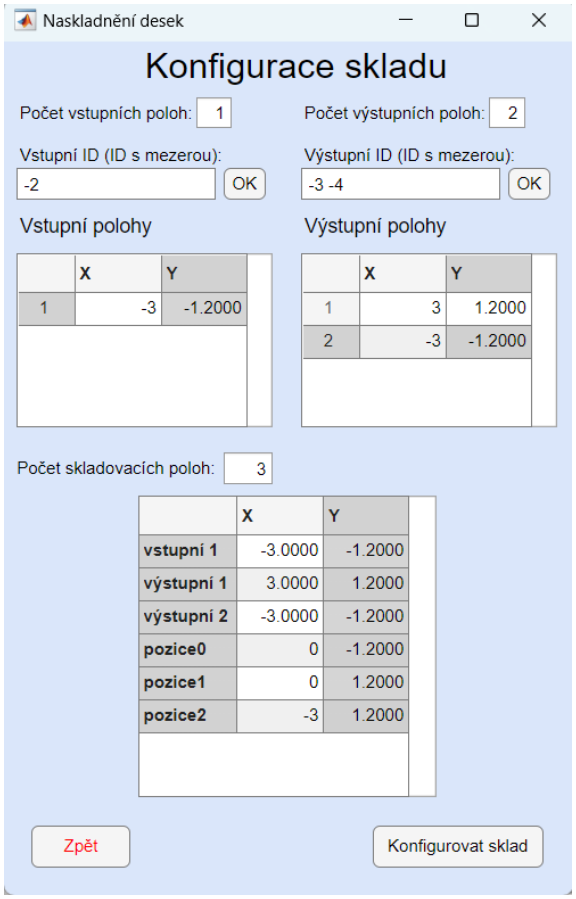
4. **Uložení do interní a případně externí databáze:** Na konci funkce se informace o všech deskách uloží do interní databáze a ta se poté uloží do souboru v počítači. Pokud je zapnuté připojení na externí databázi, tak dojde k uložení naskladněných desek i do externí databáze.



Obrázek 5.14: Stavový diagram pro naskladnění desek do skladu

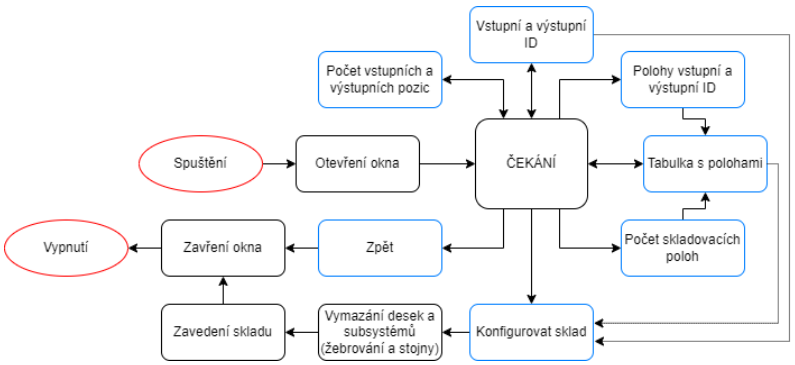
5.5 Konfigurace skladu

Konfigurace skladu je funkce, která otevře nové okno (obrázek 5.15) a je používána pouze v případě vytváření nového skladu (změna v rozložení pozic nebo ve velikosti skladu).



Obrázek 5.15: Okno pro konfiguraci skladu

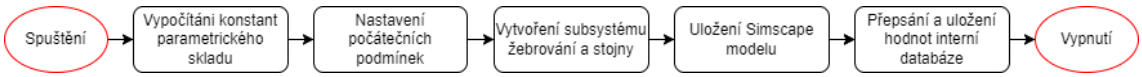
Nejprve je nutné vyplnit počet vstupních a výstupních pozic, zadat ID těchto pozic (navázání na diplomovou práci Ing. Jana Hrnčíře [30]) a vyplnit pozice x a y těžiště vstupních a výstupních pozic. Souřadnice těžiště vstupních a výstupních pozic se automaticky přepíšou na začátek celkové tabulky, dle které bude následně vytvářen nový parametrický sklad. Dále je nutné zvolit počet skladovacích poloh a dovyplnit souřadnice skladovacích míst. Stavový diagram je znázorněn na obrázku 5.16.



Obrázek 5.16: Stavový diagram pro konfiguraci skladu

Stisknutím tlačítka "Konfigurovat sklad" dojde k vymazání všech desek ve skladu a také k odstranění subsystému žebrování a výztužných stojen v modelu skladu. Poté je spuštěna funkce **zavedení skladu**, která je znázorněna na obrázku 5.17 a popsána níže.

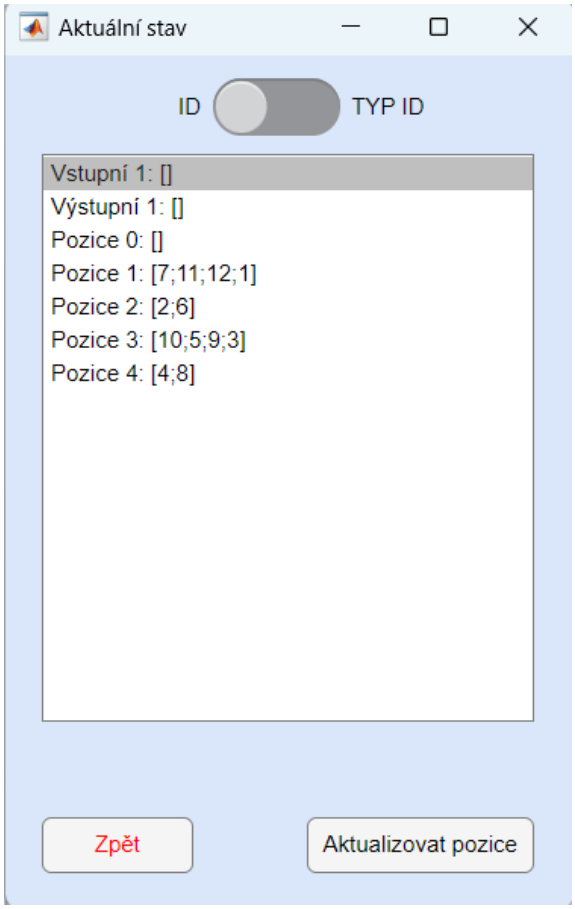
1. **Načtení a příprava dat:** Funkce přijímá jako vstupní argument data z tabulky z obrázku 5.15. Data z tabulky je potřeba upravit, protože jsou ve formě "String" a je nutné je převést do numerické podoby. Dále jsou zpřístupněny potřebné globální proměnné a určeny počáteční podmínky a konstanty pro parametrický model.
2. **Vytvoření subsystémů parametrického modelu:** Dojde k vytvoření subsystému "žebrování" a "stojny" a k vygenerování bloků dle velikosti skladu.
3. **Uložení hodnot do interní databáze:** Na konec se přepíše a uloží interní databáze.



Obrázek 5.17: Stavový diagram pro zavedení nového skladu

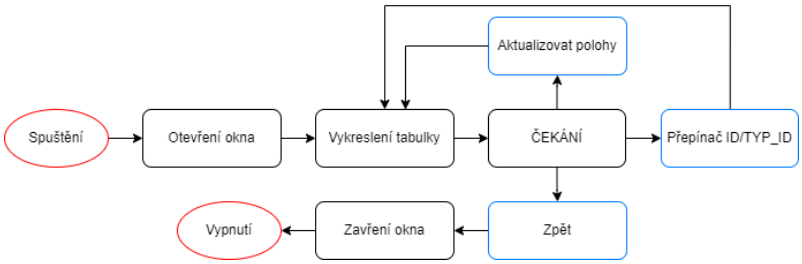
5.6 Aktuální stav skladu

Z hlavního okna lze nahlédnout na aktuální stav skladu, který se otevře v novém okně při kliknutí na "Aktuální stav" a zobrazí momentálně naskladněné desky ve skladu. Desky je možné sledovat pomocí jedinečného ID nebo TYP_ID. Na obrázku 5.18 je zobrazeno okno s aktuální stavem skladu.



Obrázek 5.18: Okno zobrazující aktuální stav skladu

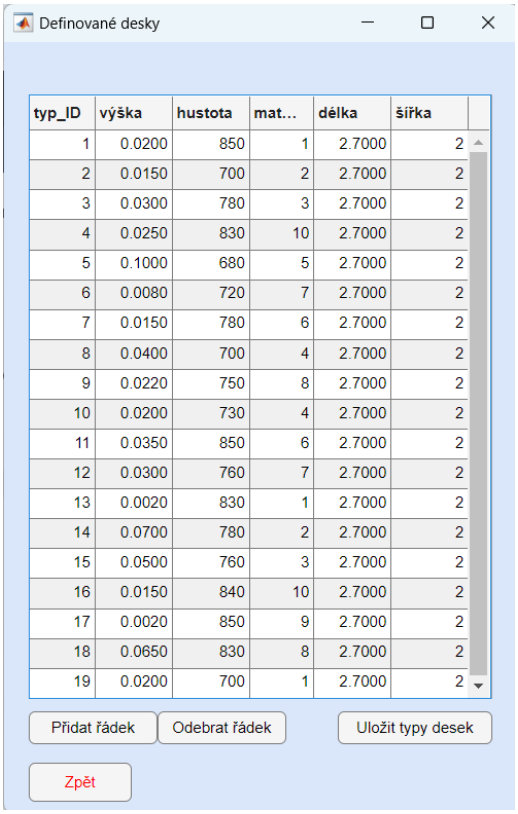
Stavový diagram zobrazený na obrázku 5.19 je pro pro aktuální stav skladu jednoduchý. Okno slouží pouze k vykreslení tabulky zobrazující aktuálně naskladněné desky dle TYP_ID a ID, mezi kterými se dá přepínat pomocí přepínače. Součástí okna je tlačítko "aktualizovat polohy", které slouží ke znovu vykreslení tabulky (např. po dokončení pohybu ve skladu).



Obrázek 5.19: Stavový diagram pro aktuální stav skladu

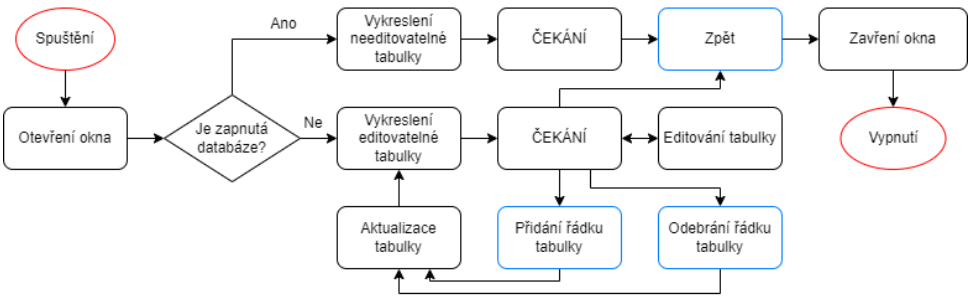
5.7 Definování/Zobrazení typu desek

Okno s tabulkou na obrázku 5.20 vyobrazuje definované typy desek (TYP_ID). Je-li externí databáze vypnutá, tak je možné přidat/odebrat řádek z tabulky a celou tabulku volně editovat.



Obrázek 5.20: Okno pro definování/zobrazení typu desek

Jak již bylo zmíněno výše, rozlišují se dva stavy: zapnutá a vypnutá externí databáze. Pokud je databáze zapnutá, není možné tabulku jakkoliv měnit, slouží pouze k zobrazení nadefinovaných desek. Je-li databáze vypnutá, objeví se pod tabulkou navíc tlačítka, která umožňují přidat a odebrat řádky z tabulky a celou tabulku uložit do interní databáze. Stavový diagram je popsán na obrázku 5.21.



Obrázek 5.21: Stavový diagram pro definování/zobrazení typu desek

5.8 Vymazání desek

Funkce vymazání desek, která je přímo implementována v hlavním okně uživatelského rozhraní, má za úkol vymazat veškeré desky ve skladu. Slouží k vymazání desek při testování automatického skladu nebo pro odstranění chybových stavů. Po vymazání všech desek dojde k uložení stavu do interní a případně i externí databáze.

Obrázek 5.22: Stavový diagram pro vymazání desek ve skladu

Kapitola 6

Externí databáze

Tato kapitola popisuje interní aplikační strukturu pro fungování databází spojených s automatickým skladem založenou z hlediska bezpečnosti, jednoduchosti obsluhy a snadné rozšiřitelnosti na virtualizační technologii kontejneru (Docker – PostGreSQL – pgAdmin - Python3). Realizovaný software byl vytvořen ve spolupráci s ing. Pavlem Bastlem PhD.

6.1 Implementace software

Software automatického skladu je komplexní systém, který je systémově rozdělen do dvou základních úrovní s následující funkcionalitou:

- Ovládací, horní část software, zajišťuje úlohy, které nevyžadují odezvu v reálném čase, ale jsou nezbytné pro integraci celého systému a integraci s vnějším okolím nebo nadřazeným systémem.
- Řídicí část zajišťuje řízení automatického skladu a jeho součástí, přičemž může být vyžadována odezva v reálném čase. V rámci struktury řídicího softwaru automatického skladu jsou některé interní části integrovány s ovládací částí softwaru.

V rámci projektu byla řešena ovládací část softwaru, přičemž jednotlivé řídicí části jsou nadále řešeny v rámci zprovoznění jednotlivých fyzických částí automatického skladu. Ovládací část je řešena za pomoci virtualizace s využitím různorodého vývojového prostředí a jazyků pro implementaci. Virtualizace využitá pro nasazení je rozdělena do dvou rovin:

- Virtualizace na hardware úrovni využívá privátního cloudového řešení OpenSTACK.
- Virtualizace na software úrovni využívá software kontejnerů, konkrétně prostředí Docker.

Tato struktura byla využita s ohledem na několik aspektů provozu software:

- Hardware virtualizace odděluje jednotlivé části aplikace na úrovni hardware a umožňuje úplné oddělení více částí software a zajišťuje dobře definované podmínky interakce mezi nimi. Hardware virtualizace vyžaduje nezanedbatelnou režii výkonu pro běh jednotlivých virtualizovaných serverů a není vhodné, aby byl provozován samostatný server pro každý běžící proces.

- Software virtualizace umožňuje vytvoření kontejnerů pro jednotlivé samostatně běžící procesy bez potřeby větší režie. Zároveň umožňuje oddělit prostředí jednotlivých aplikací, které je možné provozovat nezávisle, resp. umožňuje nezávislé spouštění a ukončování těchto aplikací. Toto řešení není ovšem imunní proti případnému pádu virtualizovaného serveru a není vhodné pro oddělení větších samostatných celků software.

Obecně virtualizace nabízí mnoho výhod a u systémů většího rozsahu je nutná. Zde jsou některé jmenovány:

- Oddělení běhu složitějších aplikací v nezávislém prostředí, čímž dojde ke zvýšení bezpečnosti jak z hlediska možnosti kontroly přístupu a komunikace, tak z hlediska potenciálních problémů při běhu jednotlivých aplikací.

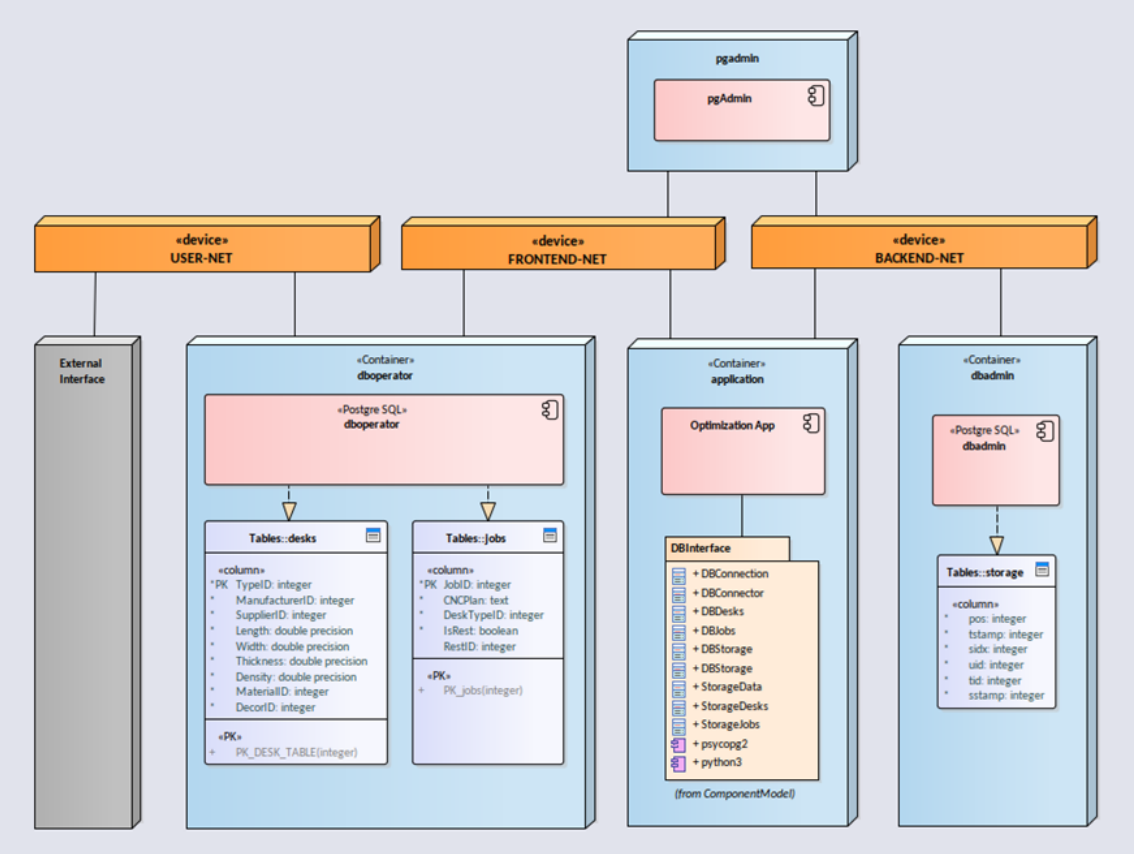
- Snadnější údržba rozsáhlejšího software systému. Jednotlivé části lze nezávisle spouštět a vypínat, což umožňuje snadnou údržbu a upgrade softwaru bez narušení funkčnosti ostatních částí.

- Snadná rozšiřitelnost takto rozsáhlého systému. Systém lze snadno rozšířit, a to pomocí nových software modulů běžících ve vytvořeném virtuálním prostředí a jejich začleněním do komunikační struktury.

- Snadná implementace integrace celého systému do vnějšího okolí vytvořením specializovaných modulů s důrazem na bezpečnost celého software systému. Systém běží v samostatných strukturách a není ovlivněn potenciálním pádem integrační vrstvy.

6.2 Implementace prostředí databází

Databáze jsou v rámci projektu využívány pro perzistentní uchování běhových dat celého systému. Vzhledem k potřebám perzistence bylo vytvořeno prostředí pro jednotlivé aplikace, které přístup k těmto datům vyžadují. Těmito aplikacemi jsou především optimalizační algoritmy nebo software komponenty digitálního dvojče. Vytvořené prostředí je univerzálně využitelné i pro jiné aplikace do budoucna. Celá struktura prostředí je na obrázku 6.1.



Obrázek 6.1: Struktura kontejneru pro komunikace s databázemi

Části prostředí se dělí do celkem tří kategorií:

- Jednotlivé kontejnery tvořící prostředí pro běh jednotlivých software komponent.
- Síťové prostředí umožnující komunikaci. Komunikace může být nastavena s ohledem na bezpečnost a ostatní aspekty běhu aplikací.
- Struktura úložiště, které umožňuje zachovat perzistenci dat jednotlivých databází i během jejich restartu nebo odstávky, a tím i zotavení systému v případě nenadálých pádů.

V této struktuře je vytvořeno několik kontejnerů, které mají svou specifickou funkci:

- **dboperator**, kontejner, který zajišťuje běh databáze pro uchování dat výrobního materiálu (dále desks), kterým jsou v našem případě desky pro výrobu nábytku. Toto lze samozřejmě upravit pro libovolný typ materiálu. Dále zabezpečuje databázové tabulky pro uchování dat pro jednotlivé výrobní úlohy (dále jobs).
- **dbadmin**, databáze běžící v samostatném kontejneru a uchovávající tabulky dat databáze pro aktuální stav automatického skladu a jeho historii, přičemž historii skladu lze dynamicky nastavovat. Tato funkcionalita kromě jiného slouží i pro případné zotavení skladu, pokud dojde k nenadálé situaci nebo pro inspekci jednotlivých stavů. Pro implementaci databáze byla zvolena aplikace

Postgre SQL.

• **app** je kontejner pro konkrétní aplikaci, v našem případě především pro optimalizaci chování a pohybu automatického skladu. Těchto kontejnerů může být nasazeno i několik pro různé aplikace, přičemž je zachován přístup k datům uloženým v databázi.

• **PgAdmin** je samostatný kontejner, který zajišťuje běh serveru, který je používán pro nezávislou administraci obou nasazených databázových serverů.

• **ExternalInterface** je komponenta, která není v rámci projektu realizována, ale ukazuje připojení vnějšího prostředí do stávající struktury.

Jednotlivé kontejnery jsou potom propojeny celkem třemi sítěmi:

• **FRONTEND-NET** je síť, která propojuje aplikace s databázemi desek (desks) a úloh ke zpracování (jobs). Obě tyto databáze musejí být dostupné jak aplikacím, tak vnějším uživatelům.

• **BACKEND-NET** je síť, která připojuje data běhového prostředí a musí být skryta vnějším uživatelům, aby nedocházelo k nežádoucím interakcím.

• **USER-NET** je síť, která je vytvořena pro připojení vnějšího prostředí. Tímto způsobem síť poskytne přístup k datům v databázích desks a jobs, ale neumožní přímý přístup k aplikacím systému pro vnějšího uživatele.

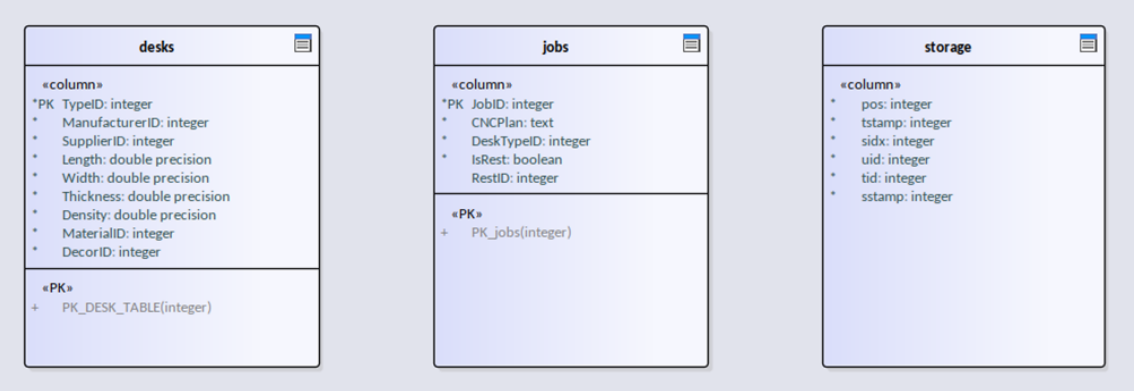
Zároveň je nasazen dohledový systém nad databázemi, kterým je software balík pgAdmin, který v případě nutnosti umožňuje provádět zásahy do těchto databází. Tímto jsou také definovány rozdílné úlohy jednotlivých uživatelů struktury: externí uživatel, správce systému prostředí.

Tabulky jednotlivých databází byly implementovány na základě požadavků aplikací. Jejich struktury jsou patrné z obrázku 6.2. Byly implementovány tyto tabulky:

• **Desks** uchovávající data o jednotlivých typech desek používaných pro výrobu.

• **Jobs** uchovávající informace o jednotlivých výrobních úlohách, které se mají v rámci automatického skladu provádět.

• **Storage** uchovávající informace o aktuálním stavu skladu v perzistentní formě a zároveň informace o historii stavu skladu.



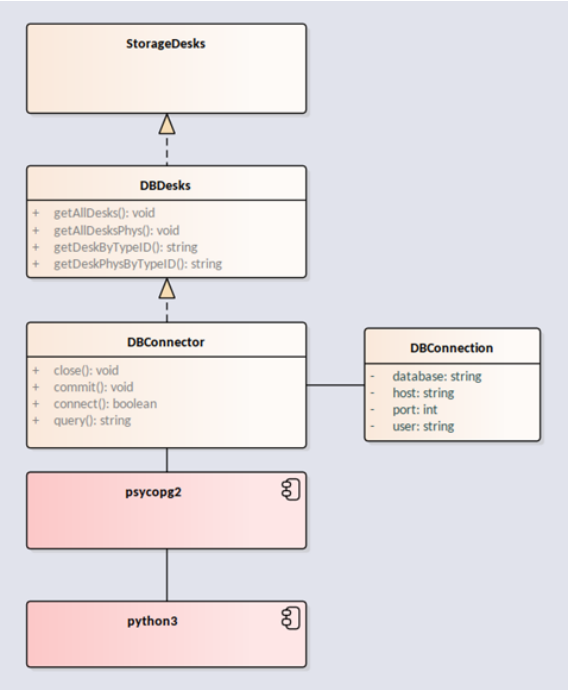
Obrázek 6.2: Implementace tabulek jednotlivých databází

Databáze **desks** uchovává především informace o jednotlivých typech desek, které jsou využívány pro výrobu. Každý typ desky má jednoznačný identifikátor (TypeID), který je využit jako primární klíč vyhledávání. Dále jsou uloženy informace o fyzikálních parametrech (rozměry, hustot – plošná nebo objemová) a informace o materiálu a povrchu.

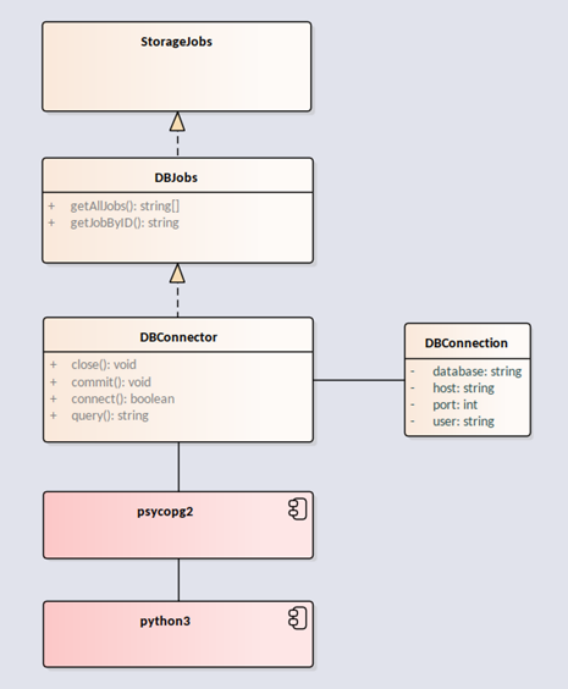
Databáze **jobs** obsahuje informace o jednotlivých výrobních úlohách, které jsou v rámci skladu zpracovávány, zde je primárním klíčem jednoznačný identifikátor úlohy. Dále je uchovávána informace o nářezovém plánu (CNCPlan). Tato informace je uchovávána ve formě ssh klíče, který jednoznačně určuje nářezový plán, resp. určuje jméno podadresáře, kde jsou data k danému plánu uchovávána. Tento přístup byl zvolen s ohledem na různorodost dat, která je třeba uchovat, a to jak do počtu souborů, tak formátu dat, která jsou závislá na typu připojených obráběcích strojů. Tímto uspořádáním je umožněna maximální flexibilita celého systému.

Databáze **storage**, která uchovává data aktuálního stavu automatického skladu, a to v perzistentní formě, čímž umožňuje zotavení po restartu (chtěném nebo nechtěném) a umožňuje jeho kontrolované odstavení a údržbu. Zároveň je uchovávána informace o historii automatického skladu, která umožňuje buď hlubší analýzu, třeba v rámci pokročilých aplikací, nebo analýzu v případech nenadálých vnějších událostí.

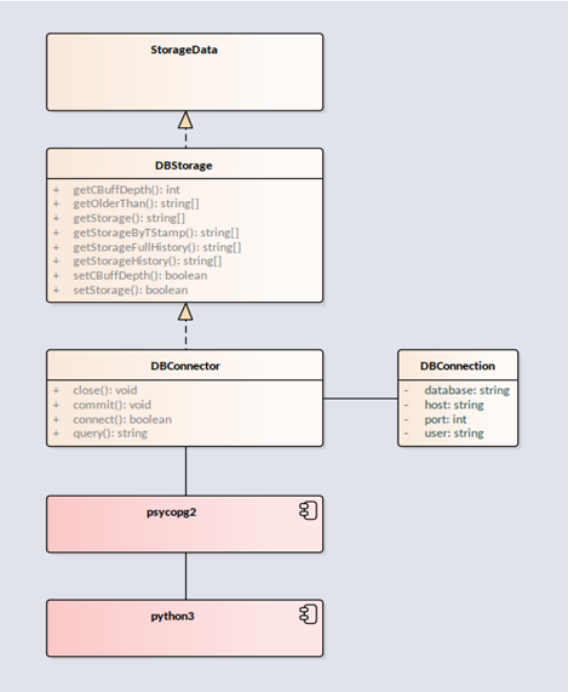
V rámci projektu byl realizován balík software pro transparentní manipulaci s daty v jednotlivých databázích. Za tímto účelem bylo vytvořeno specifické API pro jednotlivé tabulky na základě jak uchovávaných dat, tak požadavků ze strany aplikací, a to především optimalizačních úloh automatického skladu. Jednotlivé implementace, rozhraní (API) i struktura jednotlivých komponent jsou zobrazeny na obrázcích 6.3, 6.4 a 6.5. Komponenty využívají programovací jazyk Python3 a knihovnu pro komunikace s databází psycopg2. Ve struktuře je dále implementována třída, která definuje připojení ke zvolené databázi (DBConnection) a také třída, která realizuje komunikaci s databází na základě těchto informací (DBConnector). Tyto dvě třídy jsou poměrně obecné a jsou sdíleny s API pro všechny tři vytvořené databáze. Dále je vytvořena specifická nadřazená třída, která obsahu specifické API pro jednotlivé databáze (DBDesks, DBJobs a DBStorage).



Obrázek 6.3: Implementace rozhraní do databáze desek



Obrázek 6.4: Implementace rozhraní do databáze úloh



Obrázek 6.5: Implementace rozhraní do databáze stavu skladu

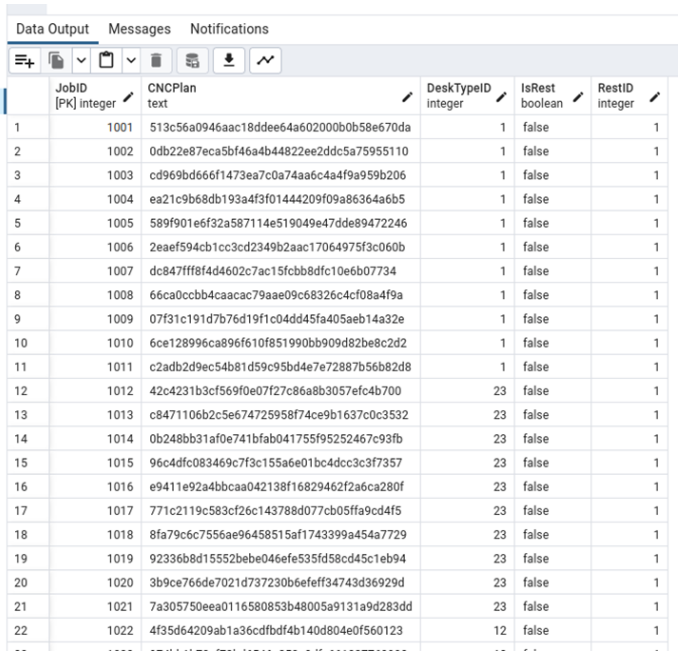
Dále jsou vytvořena specifická rozhraní do jednotlivých již běžících Postgre SQL serverů s informací o konkrétním připojení (StorageDesks, StorageJobs a Storage Data). Tímto způsobem je realizováno transparentní prostředí pro komunikaci s jednotlivými databázemi, přičemž nevyžaduje od programátora aplikace žádné specifické znalosti o struktuře síťového propojení a ostatních strukturách prostředí běhu aplikace.

Pro příklad jsou dále na obrázcích 6.6, 6.7 a 6.8 zobrazeny tabulky jednotlivých databází tak, jak byly naplněny daty pro vývoj aplikačních software a testování.

Data Output | Messages | Notifications

	TypeID [PK] integer	ManufacturerID integer	SupplierID integer	Length double precision	Width double precision	Thickness double precision	Density double precision	MaterialID integer	DecorID integer
1	1	1	1	2000	1200	18	15	102	10010
2	2	1	1	2000	1200	18	15	102	10010
3	3	1	5	2000	1200	18	15	102	10010
4	4	1	5	2000	1200	18	15	102	10010
5	5	1	5	2000	1200	18	15	102	10010
6	6	1	5	2000	1200	18	15	102	10010
7	7	2	5	2000	1200	18	15	102	10010
8	8	2	5	2000	1200	18	15	102	10010
9	9	2	5	2000	1200	18	15	102	10020
10	10	2	5	2000	1200	18	15	102	10020
11	11	2	5	2000	1200	18	15	102	10020
12	12	2	5	2000	1200	18	15	102	10020
13	13	1	7	2000	1200	18	15	102	10020
14	14	1	7	2000	1200	18	15	102	10020
15	15	1	7	2000	1200	18	15	103	10020
16	16	1	7	2000	1200	18	15	103	10020
17	17	1	7	2000	1200	18	15	103	10020
18	18	1	7	2000	1200	18	15	103	10020
19	19	3	7	2000	1200	18	15	103	10020
20	20	3	7	2000	1200	18	15	103	10020
21	21	3	7	2000	1200	18	15	103	10020
22	22	3	7	2000	1200	18	15	103	10020

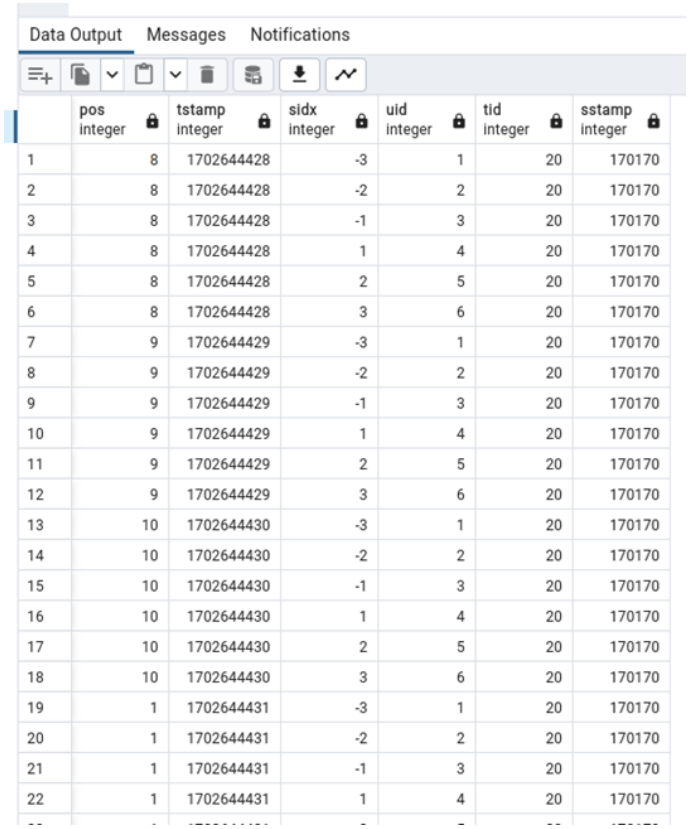
Obrázek 6.6: Ukázka implementace databáze desek v prostředí pgAdmin



Data Output | Messages | Notifications

	JobID [PK] integer	CNCPlan text	DeskTypeID integer	IsRest boolean	RestID integer
1	1001	513c56a0946aac18ddee64a602000b0b58e670da	1	false	1
2	1002	0db22e87eca5bf46a4b44822ee2ddc5a75955110	1	false	1
3	1003	cd969bd666f1473ea7c0a74aa6c4a4f9a959b206	1	false	1
4	1004	ea21c9b68db193a4f3f01444209f09a86364a6b5	1	false	1
5	1005	589f901e6f32a587114e519049e47dde89472246	1	false	1
6	1006	2eaef594cb1cc3cd2349b2aac17064975f3c060b	1	false	1
7	1007	dc847fff8f4d4602c7ac15fcbb8dfc10e6b07734	1	false	1
8	1008	66ca0ccbb4caacac79aae09c68326c4cf08a4f9a	1	false	1
9	1009	07f31c191d7b76d19f1c04dd45fa405aeb14a32e	1	false	1
10	1010	6ce128996ca896f610f851990bb909d82be8c2d2	1	false	1
11	1011	c2adb2d9ec54b81d59c95bd4e7e72887b56b82d8	1	false	1
12	1012	42c4231b3cf569f0e07f27c86a8b3057efc4b700	23	false	1
13	1013	c8471106b2c5e674725958f74ce9b1637c0c3532	23	false	1
14	1014	0b248bb31af0e741bfab041755f95252467c93fb	23	false	1
15	1015	96c4dfc083469c7f3c155a6e01bc4dcc3c3f7357	23	false	1
16	1016	e9411e92a4bbcaa042138f16829462f2a6ca280f	23	false	1
17	1017	771c2119c583cf26c143788d077cb05ffa9cd4f5	23	false	1
18	1018	8fa79c6c7556ae96458515af1743399a454a7729	23	false	1
19	1019	92336b8d15552bebe046efe535fd58cd45c1eb94	23	false	1
20	1020	3b9ce766de7021d737230b6efeff34743d36929d	23	false	1
21	1021	7a305750eea0116580853b48005a9131a9d283dd	23	false	1
22	1022	4f35d64209ab1a36cdfbdf4b140d804e0f560123	12	false	1

Obrázek 6.7: Ukázka implementace databáze úloh v prostředí pgAdmin



Data Output | Messages | Notifications

	pos integer	tstamp integer	sidx integer	uid integer	tid integer	sstamp integer
1	8	1702644428	-3	1	20	170170
2	8	1702644428	-2	2	20	170170
3	8	1702644428	-1	3	20	170170
4	8	1702644428	1	4	20	170170
5	8	1702644428	2	5	20	170170
6	8	1702644428	3	6	20	170170
7	9	1702644429	-3	1	20	170170
8	9	1702644429	-2	2	20	170170
9	9	1702644429	-1	3	20	170170
10	9	1702644429	1	4	20	170170
11	9	1702644429	2	5	20	170170
12	9	1702644429	3	6	20	170170
13	10	1702644430	-3	1	20	170170
14	10	1702644430	-2	2	20	170170
15	10	1702644430	-1	3	20	170170
16	10	1702644430	1	4	20	170170
17	10	1702644430	2	5	20	170170
18	10	1702644430	3	6	20	170170
19	1	1702644431	-3	1	20	170170
20	1	1702644431	-2	2	20	170170
21	1	1702644431	-1	3	20	170170
22	1	1702644431	1	4	20	170170

Obrázek 6.8: Ukázka implementace databáze stavu skladu v prostředí pgAdmin

Závěr

V první části diplomové práce je přiblížena problematika automatických skladů. Je zde popsána historie automatických skladů, princip chaotického skladování a zmapování aktuální nabídky na českém i zahraničním trhu.

Navazující kapitola popisuje historii vzniku konceptu digitálního dvojčete a definici digitálního dvojčete. Poté jsou uvedeny jejich charakteristické vlastnosti a porovnány různé úrovně integrace digitálních dvojčat. Závěrem je provedeno srovnání jejich výhod a nevýhod.

Třetí kapitola popisuje vznik 3D simulačního modelu automatického skladu, včetně detailního popisu fungování jednotlivých částí jeho realizace. Původní jednoduchý 3D model byl změněn do komplexního parametrického modelu, který automaticky generuje subsystémy do prostředí Simulink. Vzniklé subsystémy a detailnější model manipulátoru přispívají k přesnějšímu vyobrazení reálného skladu. Tím lze považovat cíl vytvoření 3D vizualizačního modelu v jednoduché i komplexní podobě za splněný.

Ve čtvrté kapitole je řešeno softwarové řízení skladu, včetně implementace základního GUI a je popsána interní databáze, ve které jsou uloženy veškeré informace o naskladněných deskách, velikosti skladu, očíslování jednotlivých pozic a pořadí desek ve stozích. Jsou zde popsány skripty generující Simscape bloky do prostředí Simulink, funkce vytvářející trajektorii pohybu ve skladu a řídicí algoritmus skladu. Součástí řídicího algoritmu je funkce, která propojuje vnitřní stav skladu (interní databáze) s externí databází. Tato funkce umožňuje čtení i zápis do databáze a je zde detailně popsána. V této části je popsáno splnění cíle integrování vizualizačního modelu s řídícím algoritmem skladu a cíle propojení vnitřního stavu skladu s databází.

V další části práce bylo vytvořeno uživatelské rozhraní pro ovládání simulačního modelu. Díky komplexnímu GUI rozhraní není nutné spouštět jednotlivé skripty a zadávat parametry přímo v Matlabu. Uživatelské rozhraní zobrazuje aktuální rozložení skladu a umožňuje sledovat aktuálně naskladněné desky. Z hlavního menu grafického interface je možné zapínat automatické přeskladnění a manuální vyskladnění konkrétní desky či typu desky, která je nejsnadněji vyskladnitelná. Také lze načíst stav skladu z externí databáze či náhodně rozmístit desky ve skladu. Na vstupní a výstupní pozice skladu je možné skrze rozhraní naskladnit typy desek definované v databázi. Pro testování je zde funkce, která automaticky vytvoří simulační model skladu dle počtu požadovaných pozic a jejich souřadnic.

Na závěr je popsána externí databáze, která vznikla ve spolupráci s Ing. Pavlem Bastlem PhD. Je zde uvedena struktura softwaru automatického skladu a popsány jednotlivé části databáze.

V diplomové práci byly splněny všechny stanovené cíle.

Seznam použitých zdrojů

[1] AUTOSTORE. He Ultimate Beginner's Guide to AS/RS. Online. Dostupné z: https://www.autostoresystem.com/insights/automated-storage-and-retrieval-systems-ultimate-guide. [cit. 2023-11-24].

[2] HOPSTACK. Evolution of Warehousing Systems: History and Timelines. Online. Dostupné z: https://www.hopstack.io/blog/evolution-warehousing-systems-history-timelines. [cit. 2023-11-25].

[3] INTERLAKE Mecalux. Chaotic warehousing: pros and cons. Online. Dostupné z: https://www.interlakemecalux.com/blog/chaotic-storage-advantages#how-does-location-management-work-in-a-chaotic-or-random-placement-warehouse. [cit. 2023-12-01].

[4] WASP. Why Chaotic Storage Is Perhaps the Best Inventory Management System. Online. Dostupné z: https://www.waspbarcode.com/buzz/why-chaotic-storage-is-perhaps-the-best. [cit. 2023-12-01].

[5] TOMAN, Pavel. Aby byl chaos ve skladu přínosem, musí jej řídit výkonný počítač a sofistikovaný software. Lidské chyby jsou pak eliminovány. Online. Dostupné z: https://logistika.ekonom.cz/c1-66856630-kdyz-je-chaos-ve-skladu-zadouci. [cit. 2023-12-03].

[6] BERGLER. Chaotic storage. Online. Dostupné z: https://industrieservices.de/en/company/lexicon/chaotic-storage. [cit. 2023-12-03].

[7] PILART. Automatický chaotický sklad STORE MASTER 5110. Online. Dostupné z: https://www.pilart.cz/produkt/Automaticky-chaoticky-sklad-STORE-MASTER-5110-1276/. [cit. 2023-12-06].

[8] EPIMEX. HOMAG STORETEQ P-500. Online. Dostupné z: https://epimex.cz/drevoobrabeci-stroje/skladove-a-dopravni-systemy/automaticky-chaoticky-sklad-plosneho-materialu-homag-storeteq-p-500/. [cit. 2023-12-06].

[9] DUIVESTEIN. MD-SV FLAT STORAGE SYSTEM. Online. Dostupné z: https://www.duivestein-techniek.nl/en/warehouse-systems/md-series/md-sv-flat-storage-system. [cit. 2023-12-06].

[10] Pillart. Inteligentní logistický a skladovací systém grunder. Online. Dostupné z: https://www.pilart.cz/produkt/inteligentni-logisticky-a-skladovaci-system-GRUNDNER-1264/. [cit. 2023-12-06].

[11] Jan Ledr a Martin Nečas. Odborná zpráva o postupu prací a dosažených výsledcích za rok 2021, Chytrý automatický sklad.

[12] Jan Ledr a Martin Nečas. Odborná zpráva o postupu prací a dosažených výsledcích za rok 2022, Chytrý automatický sklad.

[13] VANDERHORN, Eric a MAHADEVAN, Sankaran. Digital Twin: Generalization, characterization and implementation. Online. 2021. Dostupné z: https://doi.org/https://doi.org/10.1016/j.dss.2021.113524. [cit. 2023-12-08].

[14] Maulshree Singh, Evert Fuenmayor, Eoin P. Hinchy, Yuansong Qiao, Niall Murray a Declan Devine. Digital Twin: Origin to Future. Applied System Innovation. 2021,4 (2), DOI 10.3390/asi4020036.

[15] https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8901113

[16] Z. Wang, 'Digital Twin Technology', Industry 4.0 - Impact on Intelligent Logistics and Manufacturing. IntechOpen, Mar. 18, 2020. doi: 10.5772/intechopen.80974.

[17] Schweiger, L., Barth, L. Properties and Characteristics of Digital Twins: Review of Industrial Definitions. SN COMPUT. SCI. 4, 436 (2023). https://doi.org/10.1007/s42979-023-01937-4

[18] MARTINESCU, Livia. Exploring the concepts of digital twin, digital shadow, and digital model. Online. 2023. Dostupné z: https://oxfordinsights.com/insights/exploring-the-concepts-of-digital-twin-digital-shadow-and-digital-model/ [cit. 2023-12-12].

[19] GRIEVES, Michael. Digital Model, Digital Shadow, Digital Twin. Online. 2023. Dostupné z: https://www.researchgate.net/publication/369830792_Digital_Model_Digital_Shadow_Digital_Twin. [cit. 2023-12-12].

[20] ASPIRIN YOUTHS. Advantages and Disadvantages of Digital Twin Technology. Online. Dostupné z: https://aspiringyouths.com/advantages-disadvantages/digital-twin-technology/. [cit. 2023-12-15].

[21] ŠVADLENA, Jakub. MANIPULÁTOR PRO AUTOMATICKÝ SKLAD DESKOVÉHO MATERIÁLU. Bakalářská práce. Praha, Česká republika: České vysoké učení technické v Praze, 2022. Dostupné z:https://dspace.cvut.cz/handle/10467/102812?show=full.

[22] S. KENETT, Ron a BORTMAN, Jacon. The digital twin in Industry 4.0: A wide-angle perspective. Online. Dostupné z: https://doi.org/https://doi.org/10.1002/qre.2948. [cit. 2023-12-15].

[23] Convert MATLAB datetime to POSIX time. Online. In: . Dostupné z: https://www.mathworks.com/help/matlab/ref/datetime.posixtime.html. [cit. 2024-12-15].

[24] Add block to model. Online. In: . Dostupné z:https://www.mathworks.com/help/simulink/slref/add_block.html.[cit. 2024-01-12].

[25] Set Simulink parameter value. Online. In: . Dostupné z:https://www.mathworks.com/help/simulink/slref/set_param.html[cit. 2024-12-15].

[26] Add line to Simulink model. Online. In: . Dostupné z:https://www.mathworks.com/help/simulink/slref/add_line.html[cit. 2024-12-18].

[27] Delete line from Simulink model. Online. In: . Dostupné z:https://www.mathworks.com/help/simulink/slref/delete_line.html[cit. 2024-12-18].

[28] Delete blocks from Simulink system. Online. In: . Dostupné z:https://www.mathworks.com/help/simulink/slref/delete_block.html?searchHighlight=delete_block&s_tid=srchtitle_support_results_1_delete_block[cit. 2024-12-20].

[29] Generate trajectory subject to kinematic constraints. Dostupné z:https://www.mathworks.com/help/robotics/ref/contopptraj.html[cit. 2024-12-20].

[30] HRNČÍŘ, Jan. OPTIMÁLNÍ ŘÍZENÍ TOKU MATERIÁLU V CHAOTICKÉM SKLADU S VÝROBNÍMI STROJI. Diplomová práce. Praha, Česká republika: České vysoké učení technické v Praze, 2023. Dostupné z:https://dspace.cvut.cz/handle/10467/111499?show=full.

[31] Simulate Simulink model. Dostupné z:https://www.mathworks.com/help/simulink/slref/sim.html.[cit. 2024-12-20]