Automatická registrační linka

Jaroslav Maixner

Bakalářská práce
2024

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav automatizace a řídicí techniky

Akademický rok: 2023/2024

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Jaroslav Maixner**
Osobní číslo:	**A21205**
Studijní program:	**B0714A150006 Aplikovaná informatika v průmyslové automatizaci**
Specializace:	**Inteligentní systémy s roboty**
Forma studia:	**Kombinovaná**
Téma práce:	**Automatická registrační linka**
Téma práce anglicky:	**The Automatic Registration Line**

Zásady pro vypracování

1. Vypracujte literární rešerši na téma Registrace a ověřování vstupního materiálu na vstupu technologického procesu.
2. Popište stávající stav a postup registrace materiálu do skladu v typových firmách.
3. Navrhněte automatizovaný způsob registrace a štítkování materiálu (s vyloučením pracovníka – lidské chyby).
4. Navržený způsob realizujte v praxi.
5. Vyřešte komunikaci registrační linky s nadřazeným systémem.

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

1. SMITH, Ben. Beginning JSON. Springer, 2015. ISBN 978-1-4842-0203-6.
2. KLEMENT, Milan. TECHNOLOGIE POČÍTAČOVÝCH SÍTÍ. 2. aktualizované vydání. Univerzita Palackého v Olomouci, 2019. ISBN 978-80-244-5580-8.
3. NOVOTNÝ, František, Vlastimil HOTAŘ, Marcel HORÁK, Marie STARÁ a Michal STARÝ. ÚVOD DO AUTOMATIZACE A ROBOTIZACE VE STROJÍRENSTVÍ. Technická univerzita v Liberci, 2020. ISBN 978-80-7494-545-8.
4. KOBRLE, Pavel a Jiří PAVELKA. Elektrické pohony a jejich řízení. 3. přepracované vydání. nakladatelství ČVUT, 2016.
5. ŠMEJKAL, Ladislav a MARTINÁSKOVÁ, Marie. PLC a automatizace. Praha: BEN – technická literatura, 1999. ISBN 8086056589.
6. VÍTEK, Martin a OLEHLA, Miroslav. Programovatelný PLC dle IEC 61 131. Technická Univerzita v Liberci, 2006. Dostupné také z: https://dspace.tul.cz/handle/15240/2323.

Vedoucí bakalářské práce: **Ing. Tomáš Sysala, Ph.D.**
Ústav automatizace a řídicí techniky

Datum zadání bakalářské práce: **8. prosince 2023**
Termín odevzdání bakalářské práce: **27. května 2024**

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky
DĚKAN
L.S.

doc. Ing. Jiří Vojtěšek, Ph.D. v.r.
děkan

prof. Ing. Vladimír Vašek, CSc. v.r.
ředitel ústavu

Ve Zlíně dne 8. prosince 2023

Prohlašuji, že

- beru na vědomí, že odevzdáním bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně;
- byl jsem seznámen s tím, že na moji bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – bakalářskou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně, dne

Jaroslav Maixner, v.r.
podpis studenta

ABSTRAKT

Sledování toku materiálu v průmyslovém procesu je dnes běžnou praxí. K tomu je třeba mít definovaný způsob označení jednotlivých sledovaných dílu. Pokud výrobní proces vyžaduje velké množství nakupovaných komponentů, je vysoce nepravděpodobné že všichni dodavatelů budou mít stejný formát označení materiálu který je v procesu použitý. Je tedy vhodné při příjmu materiálu provést jeho registraci a přeznačení na vnitropodnikový standard se kterým počítají výrobní linky.

Klíčová slova: automatizace, sledování materiálu, ověřovací proces

ABSTRACT

Tracking material flow in an industrial process is now common practice. For this, it is necessary to have a defined method of labelling of the individual parts to be tracked. If a manufacturing process requires a large number of purchased components, it is highly unlikely that all suppliers will have the same format for labelling the material that is used in the process. It is therefore useful to register and relabel the material during receive to an in-factory standard that the production lines expect.

Keywords: automation, material tracking, verification process

Prohlašuji, že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

Prohlašuji, že při tvorbě této práce jsem použil nástroj generativního modelu AI Microsoft Copilot; https://copilot.microsoft.com/ za účelem úpravy vět a hledání dostupných zdrojů. Po použití tohoto nástroje jsem provedla kontrolu obsahu a přebírám za něj plnou zodpovědnost.

OBSAH

ÚVOD

Jako téma bakalářské práce jsem zvolil možnost použít aktuálně řešený projekt ve svém zaměstnání.

Jedná se o registrační linku do skladu, konkrétně tedy na vstupu technologického řetězce u zákazníka K2 Machine s.r.o., který si nepřeje být jmenován. Firma K2 Machine s.r.o. byla zákazníkem oslovena, aby navrhla a zrealizovala automatizaci registrace materiálu, protože dosavadní systém v procesu obsahuje lidské chyby.

Původní stav registrace probíhal ručně. Po vybalení materiálu z krabice byl každý díl pracovníkem načten čtečkou kódů. Tato data byla odeslána do nadřazeného systému a pokud bylo vše v pořádku, vytiskl se štítek. Štítek byl pracovníkem následně nalepen na zpracovávaný díl. Tento systém přináší možnost lidské chyby, kdy jsou načteny špatné kódy. Samotný dodavatelský štítek jich obsahuje několik a je potřeba načíst ty správné, což vede ke zdržení a prodloužení procesu. V horším případě nový vytištěný štítek nalepí pracovník na jiný díl, než který skenoval a na který patří. Tyto nedokonalosti procesu vedou k velkým prodlévám a problémům v dalších úsecích procesu výroby, kde jsou tyto díly používány.

Požadavek na navrženou linku je tedy z procesu načítání a štítkování plně odstranit lidský faktor a plně ho zautomatizovat. Registrační linka tedy byla navržena tak, aby splňovala požadavky zákazníka, eliminovala zásah člověka do samotného čtení a lepení štítku, a přidala verifikační proces. Rovněž bylo nutné v kooperaci s IT oddělením zákazníka navrhnout a realizovat přenos dat s nadřazeným systémem.

Mechanická konstrukce linky, elektro projekt, program pro kamerový systém na hledání dílů a čtení kódů byly realizovány jednotlivými odděleními firmy K2 Machine s.r.o.. Řídicí programy pro PLC, robota a HMI panel, které popisuje tato práce byly realizován mou osobou.

I. TEORETICKÁ ČÁST

1 REGISTRACE A OVĚŘENÍ MATERIÁLU V TECHNOLOGICKÝCH PROCESECH

1.1 Just in Time

„*Efektivní řízení toku materiálů a komponentů ve výrobních a montážních linkách je klíčem k efektivní výrobě. V optimálním dodavatelském řetězci jsou všechny materiály a součásti přijímány včas.“* [18]

Díky Just in Time jsme schopni ušetřit náklady na skladovací prostory, minimalizovat vázaný kapitál firem v nakoupených komponentech a surovinách potřebných ve výrobě, minimalizovat ztrátové dopady, kdy se při vývoji mění a dolaďují návrhy (požadovaná změna ve výrobě se projeví rychle). Protože tento princip zahrnuje minimální skladovací prostory ve výrobě, je potřeba pečlivě sledovat toky materiálu a jeho skladové zásoby. To je jeden z několika důvodů, proč je registrace materiálu při vstupu do výrobního procesu (do skladu) tak důležitá. Je potřeba vědět jaký materiál v omezeném skladovacím prostoru máme, jaký budeme potřebovat a jaký je potřeba objednat.

1.2 Industry 4.0

Industry 4.0 ovlivňuje výrobní prostředí při plánování operací. Namísto plánování podle předpovědí, se zaměřuje na změny v reálném čase. Celý výrobní proces se tak snaží být inteligentní a samo optimalizační. To znamená že se výrobní proces dokáže přizpůsobit nečekaným změnám například v náhlém navýšení objednávky. [19]

Jedním z prvků, které zahrnujeme pod tento pojem je traceabilita ve výrobě. Jinými slovy sledování toků materiálů napříč výrobním procesem, rozpracovaných dílů, hotových dílů a vůbec všeho co vstupuje do výrobního procesu. Díky důkladnému sledování toků materiálu můžeme najít úzká hrdla ve výrobním procesu, optimalizovat je a zefektivnit. Dalším důvodem, proč je traceabilita důležitá pro mnoho výrobců jsou reklamace a neshodné díly. Pokud vznikne reklamace na produkt, nebo je špatný díl odhalen již během výrobního procesu, lze díky historickým datům dílu dohledat příčinu problému, šarže nebo konkrétní balení použitých komponentů, operátory strojů, kteří díl vyrobily a další informace (samozřejmě vždy závisí na míře sledovaných informací).

Just in time i Industry 4.0 jako aspekty plánování a vyhodnocení výrobního procesu jsou závislé na sledování toku materiálu. Nejběžnějším typem označení materiálu jsou čárové

kódy, DMC či QR kódy, které při stejné velikosti obsáhnou mnohem více informací. Při jejich použití se při manipulaci s materiálem tyto kódy načítají pomocí čteček čárových kódů a tyto informace jsou předány dál podnikovému řídicímu systému, který změny zaeviduje a případně odešle zpětnou vazbu, zda operace proběhla v pořádku. Dalším používaným způsobem jsou RFID tagy, kde je princip práce s materiálem stejný, jen je použitý jiný způsob čtení obsažených dat. RFID tagy mají výhodu v kapacitě informací, kterou oproti čárovým kódům dokáží pojmou nebo také v životnosti v náročných podmínkách. Na rozdíl od tištěného štítku, který se může poškodit, může být RFID tag skrytý a chránění před okolními vlivy. Nevýhodou pak mohou být zvýšené náklady, protože tištěný štítek bude vždy levnější než čip.

Pokud pro sledování materiálu zvolíme čárové kódy, můžeme u vstupního materiálu od různých dodavatelů narazit na různé typy kódů s různými obsahy. Pokud je daný materiál používán na větším množství pracovišť, může se toto ukázat jako problém. Všechny pracoviště musí umět zpracovat informace všech dodavatelů. Mnohem efektivnější je tento problém řešit ihned při dodávce materiálu a jeho zaskladnění. Dodaný materiál se přeštítkuje unifikovaným štítkem, který je posléze používaný v celém výrobním procesu. To přináší zjednodušení jednotlivých stanic, protože ty vždy dostanou typově stejný štítek. Všechny dodavatelské štítky poté musí znát pouze vstupní stanice.

2 CO JE PLC

„Programovatelný logický kontrolér neboli PLC je dnes všudypřítomný v každém druhu procesního a výrobního průmyslu. PLC byly původně navrženy jako náhrada elektromechanických reléových systémů s cílem zjednodušit návrh a úpravy řídicího systému. Namísto přepojování velkého bloku relé umožňuje rychlé modifikace z PC nebo programovacího zařízení.“ [1]

PLC je určeno k vykonávání řídicích funkcí procesu nebo strojního zařízení. Mezi výhody jeho použití patří modularita, snadná rozšiřitelnost, možnost programovat v různých jazycích, sjednocené funkce a funkční bloky mezi různými výrobci, podpůrné funkce a funkční bloky jak od výrobců PLC, tak od výrobců komponentů. To vše umožnuje rychle a efektivně vytvářet nové technologické prvky a modifikovat nebo rozšiřovat stávající.

„Prvním zástupcem PLC byl Modicon model 087, který byl uveden v roce 1968-69. Z pohledu dnešní doby se nejednalo o nijak výkonné zařízení. Nicméně v kontextu tehdejší relátkové logiky byl tento počin velice významným krokem vpřed, který ukázal budoucí směr automatizace a jejího řízení.“ [2]

2.1 Dělení PLC

Jedním ze základních dělících aspektů, nikoli však jediným, je dělení na modulární kompaktní systémy.

2.1.1 Kompaktní systémy

Tyto PLC mají prakticky vždy integrované výpočetní CPU s pamětí pro vykonávání programu spolu s několika I/O a není tedy pro základní automatizaci již nic potřeba. Tyto PLC lze takto rozšiřovat o další I/O karty, ale obvykle bývá celkový počet omezen. [15]

Stejně tak programové vybavení těchto PLC bývá ochuzené o pokročilejší funkce, jakými jsou například řízení servopohonů (především jejich typ a počet), pokročilé PID regulátory a podobně. Mezi jejich zástupce patří například řada S7-1200 od společnosti Siemens nebo řada NX1P2 od společnosti Omron.

Obrázek 1 Kompaktní PLC S7-1200 [3]

Obrázek 2 Kompaktní PLC NX1P [4]

2.1.2 Modulární systémy

Modulární typ se liší od kompaktního především samostatnou CPU jednotkou pro vykonávání programu, která v základu neobsahuje žádné I/O. Na druhou stranu poskytují větší možnosti rozšíření, řízení a programového vybavení. Pokud má výrobce v nabídce modulární PLC, nabízí velkou paletu doplňujících karet, která prakticky vždy přesahuje nabídku pro kompaktní verze. Zástupce od společnosti Siemens může být řada S7-1500, nebo řada NJ od společnosti Omron.

Obrázek 3 Modulární PLC S7-1500 [5]

Obrázek 4 Modulární PLC NJ5 [6]

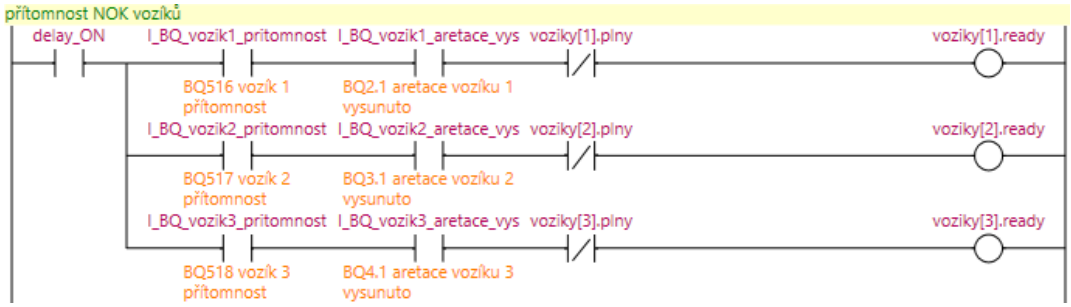
2.2 Programovací jazyky pro PLC

„V normě IEC 61131-3 jsou zařazeny čtyři programovací jazyky. Jejich sémantika a syntaxe je normou přesně definována. Jsou to: LD, FBD, IL, ST. Zvládnutím těchto jazyků se nám otevírá cesta pro rychlou realizaci požadované aplikace pomocí hardware různých výrobců, kteří přijali tuto normu. Jako další jazyky se uvádějí SFC, CFC, nejsou však zařazeny přímo mezi jazyky, ale jsou to tzv. společné prvky tvořící nadstavbu pro strukturování celé aplikace.“ [7]

2.2.1 LD – ladder diagram

„Žebříkové diagramy se používají k formulaci logických výrazů PLC v grafické podobě. Používají symboly k reprezentaci podmíněných, vstupních a výstupních výrazů. Žebříková

schémata jsou podobná reléovým řídicím obvodům a používají se kvůli jejich snadnému programování ve srovnání s textovými programovacími jazyky“.[8]



Obrázek 5 Ukázka LD diagramu

2.2.2 ST – structured text

ST se řadí mezi textové jazyky. Jeho syntaxe se podobá programovacím jazykům Pascal nebo Delphy. Je tedy více podobný programovacím jazykům, které známe z PC.

```
// uprava pro čtení QTY .. kamera vrátí LOT i QTY stejný kod, pro scenu 22 se hledá "xxx qty xxxxx"
tmpQty := inst_sapCom.i_label_lot;
inst_sapCom.i_label_qty := '';
// nalezení 1 mezery
startChar := FIND(tmpQty, ' ');
// odmazání znaků před qty
tmpQty := DELETE(inst_sapCom.i_label_lot,startChar,1);
// nalezení druhé mezery
startChar := FIND(tmpQty, ' ');
// odmazání QTY od zbytku
tmpQty := LEFT(tmpQty,startChar);
// přepis qty
inst_sapCom.i_label_qty := tmpQty;
```

Obrázek 6 Ukázka ST kódu

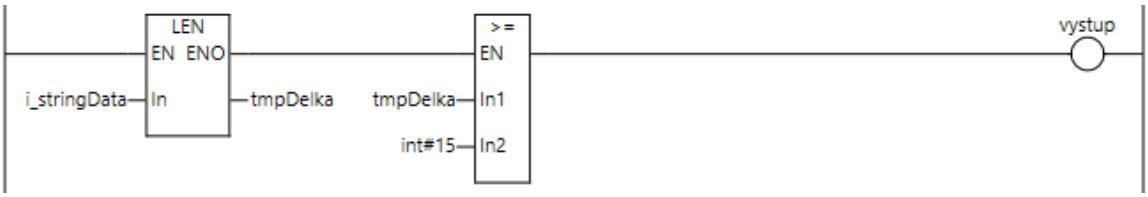
2.2.3 Srovnání LD a ST

Registrační linka popsaná v této práci je naprogramována kombinací těchto dvou jazyků, tedy LD a ST. V dnešních PLC jsou oba tyto jazyky plnohodnotné a zaměnitelné, což znamená, že lze dosáhnout stejné funkcionality kódu bez ohledu nato, který zvolíme. Nicméně jsou oblasti, pro které se hodí více ST či LD.

Jazyk ST je vhodnější pro použití všude tam, kde je potřeba práce s datovým typem STRING nebo tam, kde je potřeba použití smyček typu FOR či WHILE. Operace s textem i smyčky lze realizovat také v jazyce LD, ale přehlednější a jednodušší je použití jazyka ST, jak je ukázáno níže. V příkladu je znázorněn rozdíl zápisu ST a LD pro jednoduchou funkci. Požadavkem je, aby funkce sepnula výstup, pokud má textový řetězec délku 15 či více znaků

(jednoduchá validace například ze čtečky čárových kódů, pokud známe nejkratší možnou délku kódu).

Při použití jazyka LD je nejprve nutné získat délku řetězce pomocí funkce LEN. Tuto hodnotu je potřeba uložit do dočasné proměnné a poté použít funkci pro porovnání dvou čísel.



Obrázek 7 Příklad LD

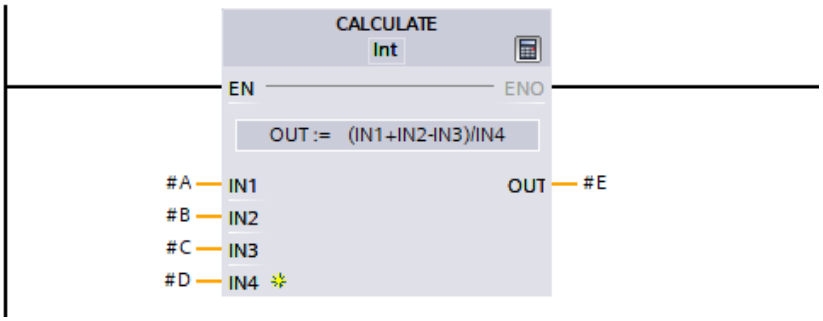
ST umožnuje přímý zápis výsledku jedné funkce ihned do vstupu další, tak jako například v C#. Opět musíme získat délku řetězce pomocí funkce LEN, ale není potřeba jí uchovávat v žádné proměnné a ihned celé volání použít jako jednu stranu porovnání.

```
1 vystup := LEN(i_stringData) >= 15;
```

Obrázek 8 Příklad ST

Je tedy patrné, že pro práci s textovými řetězci umožnuje ST jednoduší a přímější zápis. Stejně tak je tomu u matematických výpočtů. Například E = (A + B - C) / D. Takovýto výraz v ST lze zapsat tak, jak je naznačeno. Při použití LD je nutné sečíst A+B a uchovat výsledek. Od něho odečíst C a opět uchovat výsledek a poté vydělit D.

Nicméně toto už je závislé i na výrobci PLC. Použitý systém OMRON na registrační lince nemá v LD jiné řešení než toto. Naproti tomu například PLC Siemens umožnuje takový zápis zjednodušit pomocí interní funkce CALCULATE. Je tedy vidět, že se zápis může lišit i při použití stejného jazyka napříč různými výrobci PLC systému.



Obrázek 9 Funkce calculate

3 ŘÍZENÍ POHONŮ

„Elektrický pohon je systém vytvořený z vhodné kombinace elektrických zařízení pro elektromechanickou konverzi energie a pro vytváření, přenos a zpracování řídicích signálů, které tuto elektromechanickou konverzi řídí. Vstupní řídicí signály jsou určeny pro provoz nebo nadřazený řídicí, regulační nebo automatizační prvek a výstupní veličiny jsou parametry mechanického pohybu.“ [9]

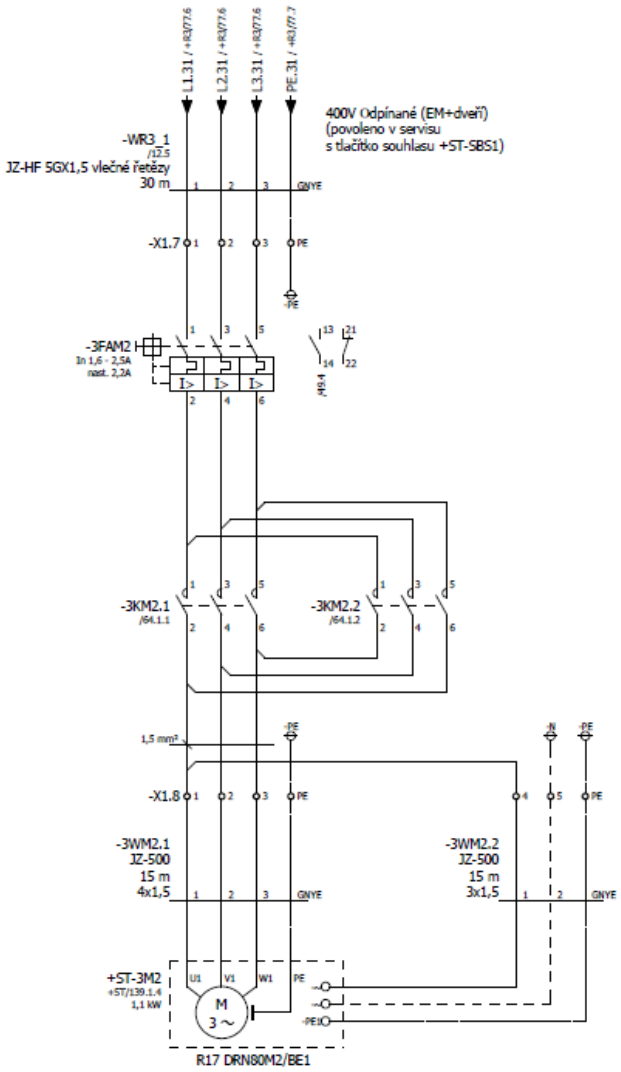
Řízení obecně lze rozdělit na dva druhy

1. Ovládání. Zásah do řízeného procesu nemá většinou přímou zpětnou vazbu a nesrovnává se ihned, zda byla reakce systému dle očekávání. Na základě znalosti systému je předpokládáno, že daný řídicí pokyn vykonal požadovanou akci. Příkladem mohou být vnější dopravníky, které jsou na registrační lince. Na základě senzorů PLC vydá pokyn k rozběhnutí dopravníku do frekvenčního měniče. Přitom nemá zpětnou informaci, zda se dopravníky opravdu roztočili.
2. Regulace. Při regulaci dochází k okamžitému vyhodnocení regulované veličiny a přizpůsobení zásahu regulátoru tak, aby výsledné hodnota odpovídala požadované. Příkladem může být 3D manipulátor registrační linky, jehož jednotlivé osy jsou osazeny servopohony a řízeny jako polohová osa se zpětnou vazbou polohy. PLC tedy v každou chvíli jejího pohybu kontroluje reálnou polohu s předpokládanou a provádí nutné korekce tak, aby se pohon udržel v požadovaných parametrech.

3.1 Možnosti připojení asynchronního motoru k PLC

3.1.1 Připojení pomocí stykače

Tento způsob je nejjednodušším, nejlevnějším, ale také nejvíce omezeným z pohledu řízení pro asynchronní pohony. Řídicí systém dokáže poslat pouze signál chod nebo stop, případně opačný směr otáčení za použití reverzačního zapojení dvou stykačů. Není tu žádná zpětná vazba, zda motor pracuje správně. Nejsme schopni nastavit, ani regulovat otáčky nebo rozběhový a doběhový čas.



Obrázek 10 Příklad zapojení se stykačem

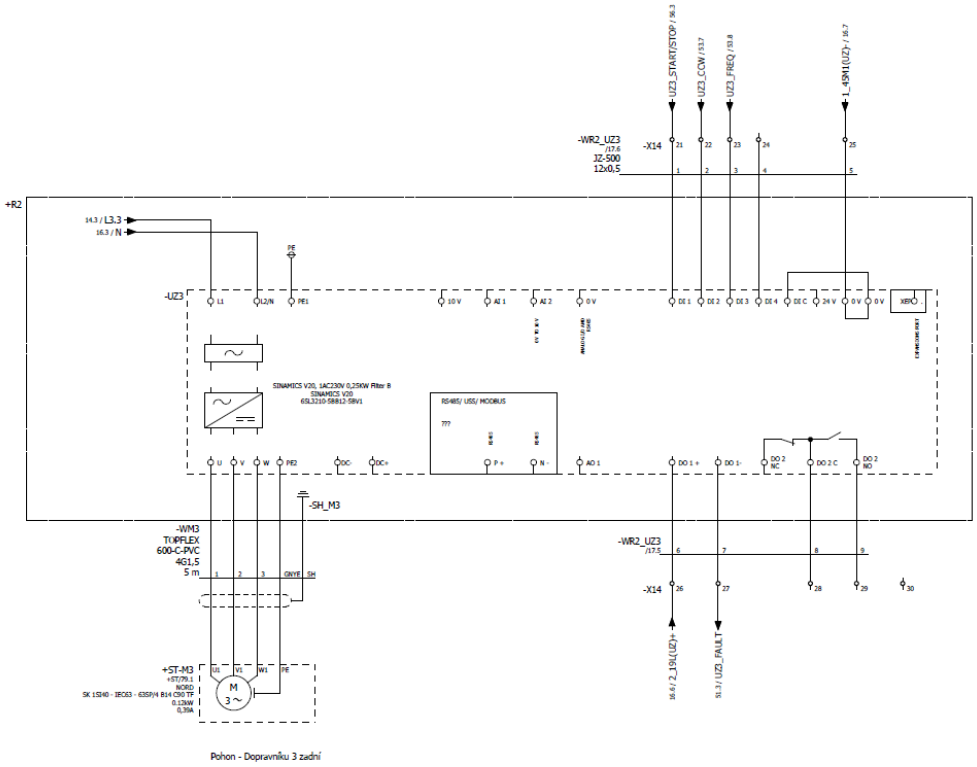
3.1.2 Připojení pomocí frekvenčního měniče

Tento způsob umožní rozšířené nastavení parametrů provozu asynchronního pohonu. Vyšší řady dnešních frekvenčních měničů obsahují i vstupy pro enkodéry a umožnují tak přesnou regulaci pohonu. Připojení k řídicímu systému lze realizovat dvěma způsoby.

1. Pomocí digitálních vstupních a výstupních signálů, případně analogových signálů pro přenos rychlosti, proudu nebo rozběhového času atd.. Máme možnost řízení rychlosti otáčení, směru, rozběhových doběhových časů, zpět do řídicího systému odesílání poruchových stavů a mnoho dalšího. Tímto způsobem jsou na registrační lince, kterou se zabývá tato práce, připojeny vnější dopravníky 1, 2 a 3. Na obrázku 11 je schéma zapojení těchto tří

pohonů a jejich signály od PLC – start/stop, reverzace CCW, výběr pevné frekvence FREQ, tak i signál zpět do PLC o chybě FAULT.

2. Pomocí sběrnice řídicího systému. Tento způsob vyžaduje, aby frekvenční měnič obsahoval patřičnou sběrnici dle použitého řídicího systému. Pokud však potřebujeme komplexní řízení pohonu, tak se jedná o nejkomplexnější způsob připojení, protože pomocí sběrnice máme přístup ke všem parametrů pomohu.



Obrázek 11 Příklad zapojení fr. měniče

3.2 Možnosti připojení servopohonu k PLC

„*Servomotory jsou speciálním typem elektromotoru navrženého se samostatnými zpětnovazebními systémy s uzavřenou smyčkou, obvykle enkodéry nebo lineárními snímači, což znamená, že poloha výstupního hřídele je neustále hlášena řídicímu systému.*“ [17]

To umožnuje přesné řízení otáček nebo polohy pohonu. Servopohony lze připojit řídicímu systému dvěma hlavními způsoby.

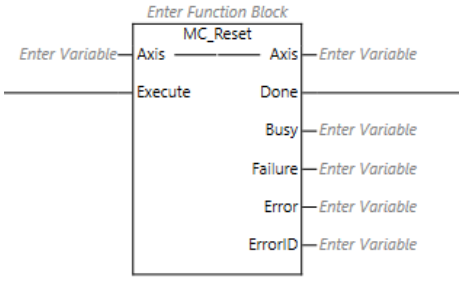
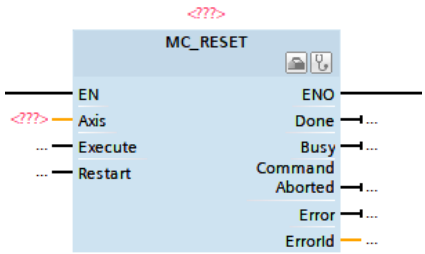
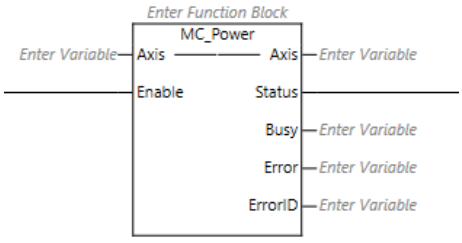
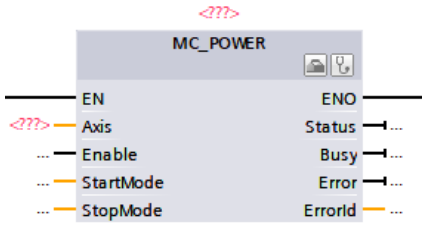
1. Binárně s přenosem řídicích pulzů do servoměniče. Podobně jako u připojení frekvenčního měniče pomocí binárních signálů, jsou přenášeny základní informace jako povolení pohonu, chyba pohonu, směr otáčení a další. Navíc

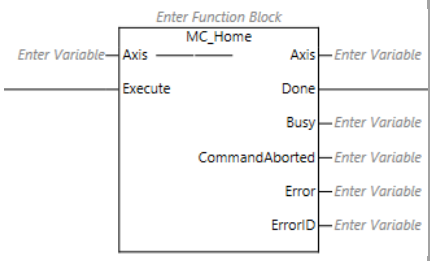
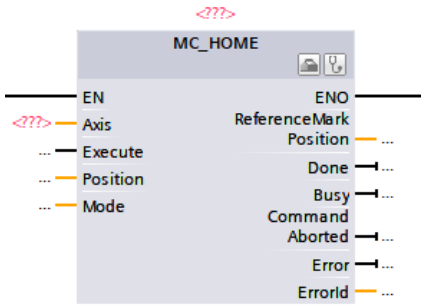
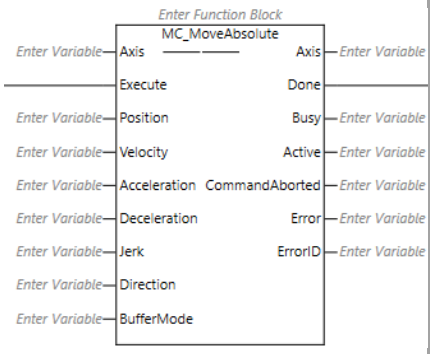
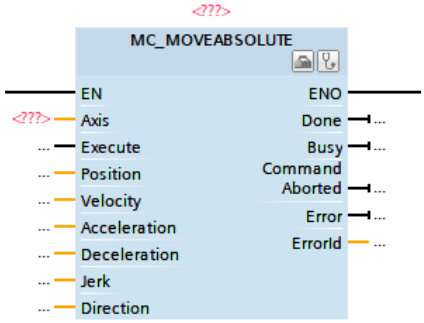
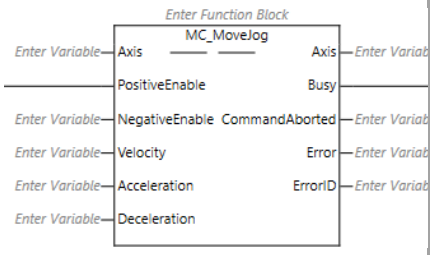
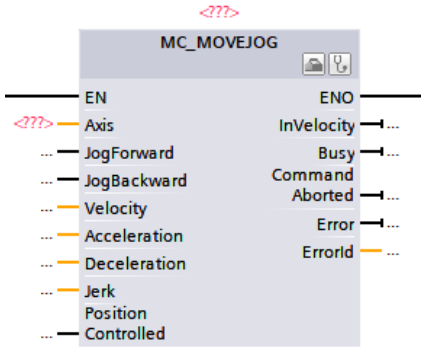
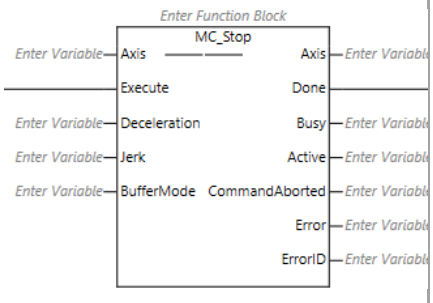
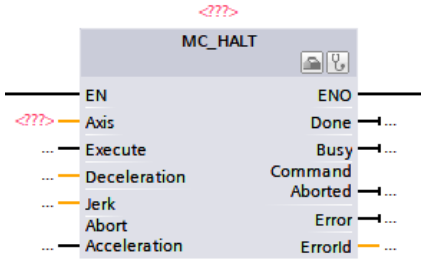
toto připojení vyžaduje pulzní kartu na straně PLC, které její pomocí vysílá počet pulzů o kolik se má servopohon pootočit. V tomto případě lze použít i jednodušší PLC, které disponuje pulzními výstupy. Po odeslání všech pulzů je do PLC binárně odeslán signál o dokončení pohybu.

2. Pomocí sběrnice. PLC i servopohon jsou propojeny průmyslovou sběrnicí, která zajišťuje přenos všech řídicích a stavových signálů. PLC má tedy celou dobu informace o poloze, otáčkách, momentu a dalších parametrech pohonu. Tento způsob je použitý na registrační lince pro připojení všech servopohonů. Protože je zde použit řídicí systém OMRON a taktéž servopohony OMRON, jsou propojeny sběrnicí EtherCat, kterou tento výrobce používá.

Pro samotné řízení servopohonu jako polohové nebo rotační osy lze použít standardizované MC (Moution Control) funkce, které jsou přímo integrované většinou výrobců řídicích systému. Níže je srovnávací tabulka 1 základních integrovaných MC funkcí PLC řady S7-1500 společnosti Siemens a NJ společnosti Omron.

Tabulka 1 Srovnání základních MC funkcí

Název funkce	Implementace Omron	Implementace Siemens
MC_reset		
MC_power		

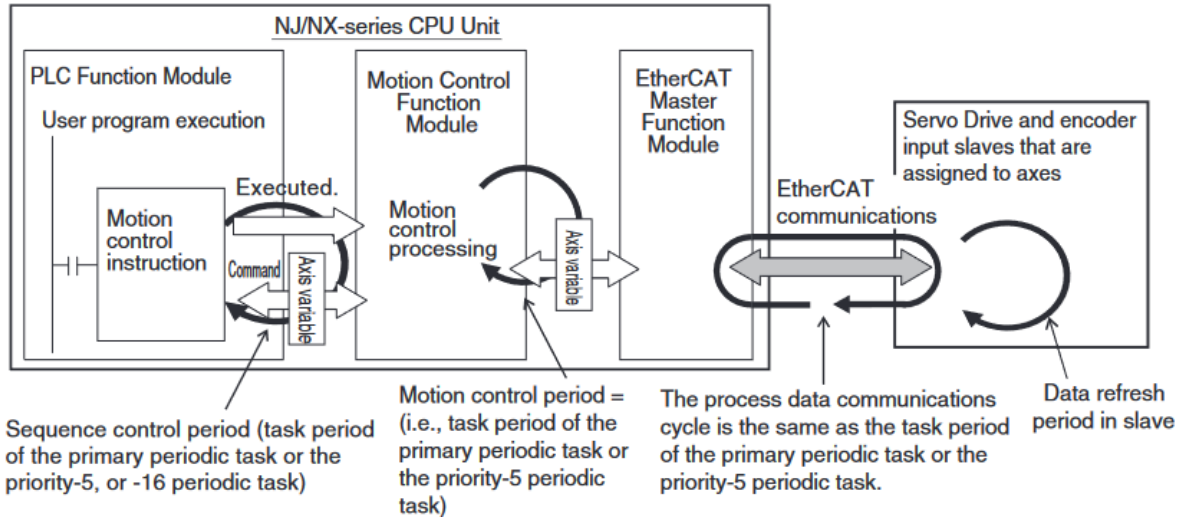
Název funkce	Implementace Omron	Implementace Siemens
MC_home		
MC_moveAbolut		
MC_moveJog		
MC_stop		

Z tabulky je patrné, že se mohou lišit vstupní i výstupní parametry funkce. Vždy závisí na implementaci daného výrobce. Nicméně by vždy měla být funkcionalita stejná, bez ohledu na výrobce. Často se setkáme s omezením MC funkcí dle konkrétní PLC (jejich výkonu a ceny), kdy například synchronizační funkce více os nebo přímo řízení kinematiky mají

pokročilejší nebo specializované řady PLC. Tento soubor MC funkcí ulehčuje práci a vytvoření řídicího programu pro servopohony, kdy se servopohon propojí řídicí sběrnicí s PLC (dle výrobce) a poté už stačí pouze volat MC funkce s referencí na fyzický pohon.

3.2.1 Přenos informace MC funkce v PLC Omron

V uživatelské části programu je řízení servopohonů realizování pomocí MC funkcí, které předávají požadavky do MC jádra. Ten na základě příkazů vygeneruje řídicí signály, které jsou poté předány jednotlivým pohonům prostřednictvím sběrnice EtherCAT. Stejnou cestou MC jádro vyčítá stavy pohonů a předává je zpět do PLC. Uživatel tedy z programu PLC nekomunikuje s pohony napřímo, ale skrze MC jádro. [17]



Obrázek 12 Přenos dat od MC funkce [17]

4 PC SÍTĚ

4.1 Přenos dat pomocí TCP/IP

4.1.1 IP – internet protokol

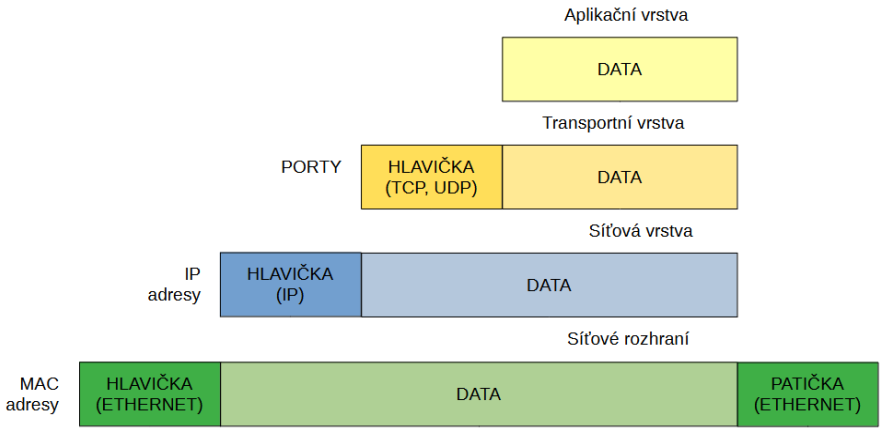
Jedná se o protokol síťové vrstvy. Tento protokol přenáší IP datagramy mezi vzdálenými počítači. Ty se skládají ze záhlaví, které nese adresu příjemce, a přenášených dat. Každé síťové rozhraní v síti má přiřazenou unikátní IP adresu, díky čemuž je zaručeno, že datagram dorazí ke správnému adresátovy.[10]

4.1.2 TCP/IP

TCP a UDP protokol se nachází v transportní vrstvě. Oba protokoly zajišťují spojení mezi dvěma aplikacemi běžícími na různých zařízeních v síti. Hlavním rozdílem TCP a UPD protokolu je ověření doručení dat. TCP protokol je takzvaně spojovaný a příjemce dat potvrzuje že data opravdu dostal. Pokud by došlo ke ztrátě dat, příjemce si požádá o opakování přenosu. UDP po odeslání dat nemá žádný kontrolní mechanizmus, zda data opravdu došla.[10]

4.1.3 Zapouzdření dat

Samotná data jsou podle obrázku 13 předána z aplikační vrstvy do transportní. V ní je přidána UTP/TCP hlavička. Takto vytvořený rámec putuje hlouběji do síťové vrstvy, která k němu přidá IP hlavičku a doplněný rámec se opět posune hlouběji na vrstvu síťového rozhraní, pomocí níž dojde k odeslání dat.



Obrázek 13 Zapouzdření dat TCP/IP

4.1.4 TCP/IP v PLC Omron NJ

Na straně PLC má většina dnešních výrobců již připravené funkční bloky, které usnadňují práci s vytvářením a obsluhou komunikací pomocí TCP/IP. V řídicích systémech Omron jsou označeny jako „Socket Service Instructions“.

Tabulka 2 Funkce pro TCP/IP PLC Omron [11]

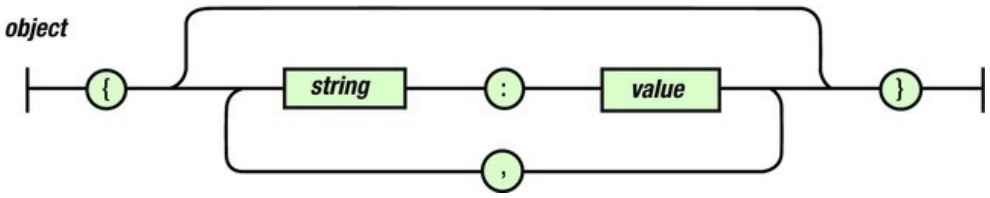
Název služby	Funkční blok	Popis
Připojení TCP soketu	SktTCPConnect	Připojení ke vzdálenému zařízení dle IP a portu
odeslání dat po TCP soketu	SktTCPSend	Odeslání dat
příjem dat po TCP soketu	SktTCPRcv	Příjem dat
uzavření TCP soketu	SktClose	Uzavření spojení
Získání stavu TCP soketu	SktGetTCPStatus	Čtení stavu spojení

4.2 Přenosový formát dat JSON

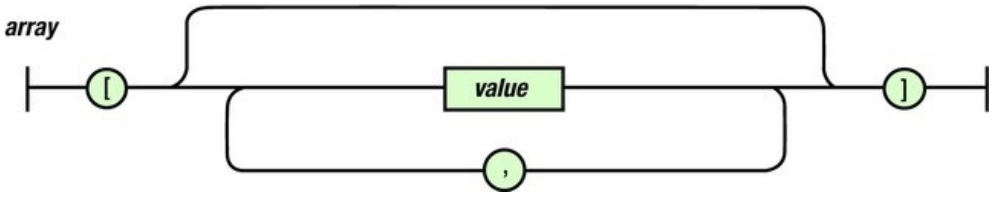
JSON je odlehčená syntaxe nezávislá na programovacím jazyku. Je odvozena od programovacího jazyka ECMAScript. JSON nijak nedefinuje, jak má být jeho validní zápis interpretován. Díky tomu lze použít více možností, jak s takovými daty pracovat a jak je interpretovat. Proto, aby výměna mezi dvěma účastníky komunikace pomocí formátu JSON byla smysluplná, je potřeba dohoda obou stran na sémantickém zápisu, aby data obě strany interpretovaly dle očekávání. [12]

„JSON je definován jako malá sada pravidel která specifikují zápis dat do struktur. JSON udává dva způsoby strukturování dat.

1. *Sbírka prvků s názvem a hodnotou – reprezentace objektu / struktury*
2. *Seřazený seznam hodnot – reprezentace pole / listu „*[13]



Obrázek 14 JSON objekt [12]



Obrázek 15 JSON pole [12]

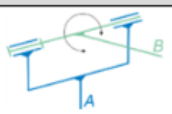
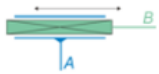
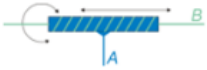
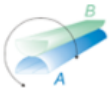
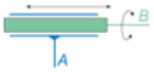
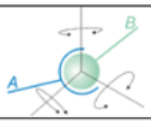
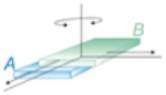
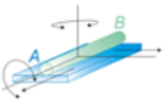
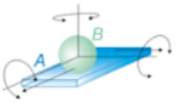
Pravidla pro převod dat do formátu JSON jsou tedy jednoduchá a lze je tedy bez větších problémů implementovat i v řídicích systémech PLC.

5 PRŮMYSLOVÝ ROBOTI A MANIPULÁTORY

Každý průmyslový robot nebo manipulátor je složen z mechanických dílů jednotlivých os a pohonů, kterými jsou tyto osy svázány. Vzájemnému pohyblivému spojení dvou mechanických prvků říkáme kinematická dvojce. *„Počet stupňů volnosti kinematické dvojice je roven počtu nezávislých posuvů a rotací, které mohou členy dvojice vůči sobě navzájem vykonávat.“* [14]

„Obecně platí, že počet stupňů volnosti je roven počtu nezávislých pohybů, které může struktura vykonávat a pro každý nezávislý pohyb musí být soustava vybavena příslušným pohonem. Nejčastěji je technická realizace provedena pomocí rotačních, resp. translačních pohybových jednotek s individuálním vnitřním pohonem, kdy každá jednotka má samostatný nezávislý pohon. Potom počet pohybových jednotek (pohonů) odpovídá počtu nezávislých souřadnic výsledné polohy koncového členu robotu a je jím určen počet stupňů volnosti.“ [14]

Během konstrukce a návrhu manipulátorů a robotů můžeme vytvářet a kombinovat kinematické dvojce z obrázku 16 tak, abychom docílili požadované funkčnosti a potřebných stupňů volnosti.

NÁZEV	SCHÉMA	POHYBLIVOST B:A => i	SYMBOL	TŘÍDA j = 6 - i
ROTAČNÍ		1°	R	5
POSUVNÁ		1°	T	5
ŠROUBOVÁ		1°	H	5
VALIVÁ		1°	V	5
VÁLCOVÁ		2°	C	4
SFÉRICKÁ		3°	S	3
ROVINNÁ		3°	F	3
VÁLEC NA ROVINĚ		4°	-	2
OBECNÁ		5°	O	1

Obrázek 16 Kinematické dvojce [14]

5.1 Manipulátory X, XY, XYZ

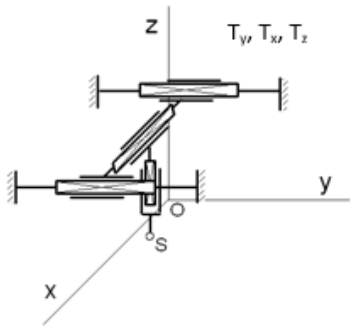
Často se setkáváme s aplikací, kde je třeba pouze translační pohyb a osy nebo osa souřadného systému je rovnoběžná s mechanickou osou – kartézský typ.

• Manipulátor X(T) - jeden translační pohyb - 1° volnosti. Takovýto manipulátor dokáže vykonávat pouze přímočarý pohyb. Navzdory omezeným možnostem pohybu najde takovýto manipulátor uplatnění například při kamerových kontrolách, kde je potřeba posuv kamery, například nutnost změny pozice pro různé typy kontroly.

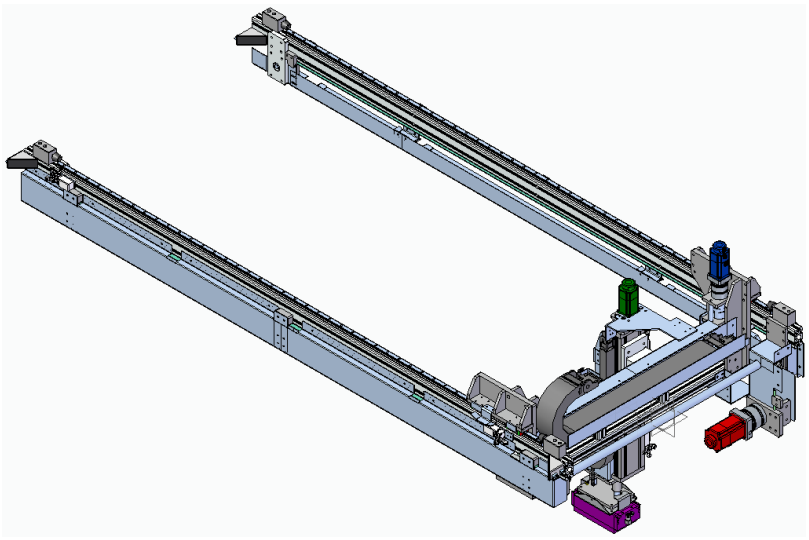
• Manipulátor XY(TT) - dva na sebe kolmé translační pohyby - 2° volnosti. V této konfiguraci již dokážeme v rovině vytvářet libovolnou křivku. Uplatnění takového manipulátoru najdeme například v CNC strojích pro řezání laserem nebo vodou.

• Manipulátor XYZ(TTT) - tři na sebe kolmé translační pohyby - 3° volnosti. Podobně jako konfigurace XY, i zde dokážeme vytvářet libovolnou křivku, ale díky ose Z nikoli v rovině, ale v prostoru. Tato konfigurace je vhodná všude tam, kde potřebujeme pracovat v prostoru. Lze tento způsob najít například na 3D tiskárnách, CNC frézách nebo manipulačních zařízení.

V rámci této práce je použitý manipulátor XYZ(TTT) pro výstupní uskladnění dílů. Jeho kinematické schéma je znázorněno níže (Obr. 17), stejně tak jeho konstrukční řešení (Obr. 18).



Obrázek 17 Manipulátor – kinematika [14]



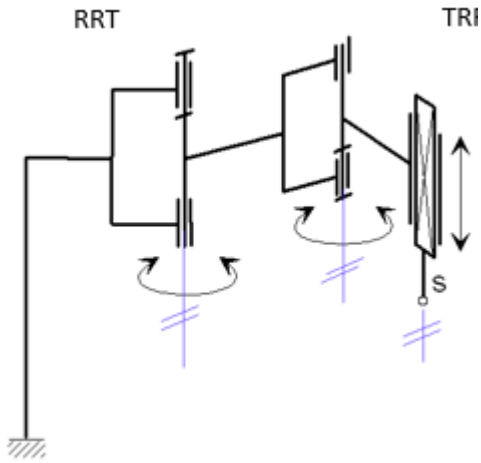
Obrázek 18 Manipulátor – realizace

5.2 Průmyslový SCARA robot

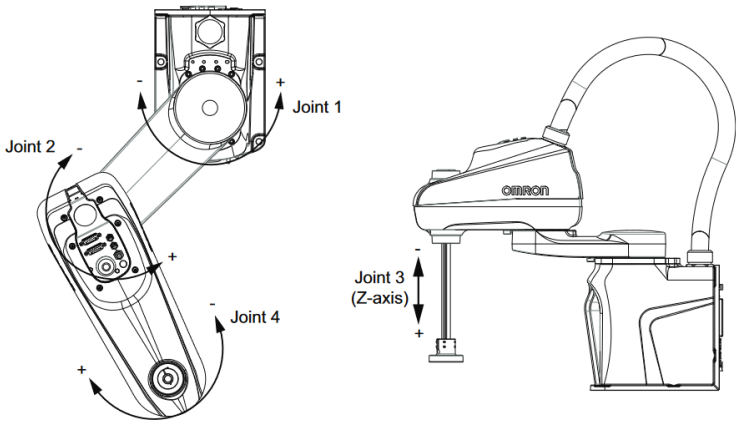
System Compliance Assembly Robot Arm – tvoří ho kinematický řetězec RRT nebo TRR. Všechny osy jsou rovnoběžné a svislé. Lze ho využít všude tam, kde je potřeba vysoká rychlost, přesnost a není potřeba rotace kolem osy X a Y. *„Struktura je výhodná pro*

montážní i technologické účely s možností přesného polohování (běžná přesnost ± 0,05 mm) a značných rychlostí v horizontální rovině (běžně 4–6 m.s-1).“ [14]

V rámci této práce je SCARA robot použitý na vstupní části, ve které nese kamerový systém a přísavky pro aplikaci štítku. Kinematický popis je na obrázku 19. Pod ním je na obrázku 20 nákres a popis os použitého robota v této práci.



Obrázek 19 SCARA – kinematika

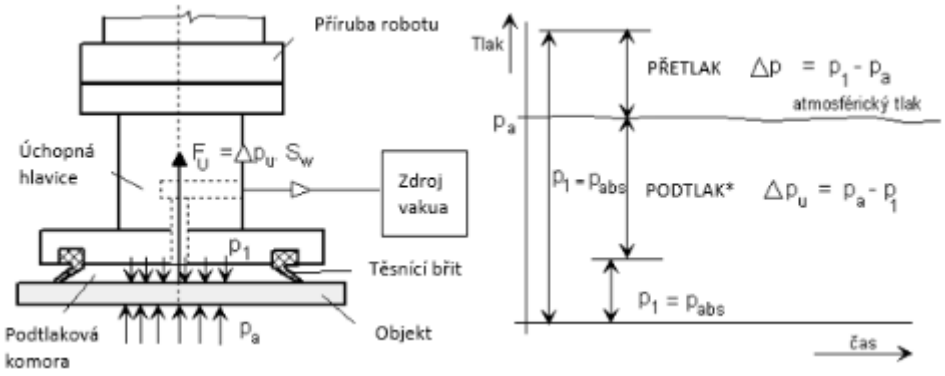


Obrázek 20 SCARA – realizace [16]

5.3 Podtlakové chapadlo

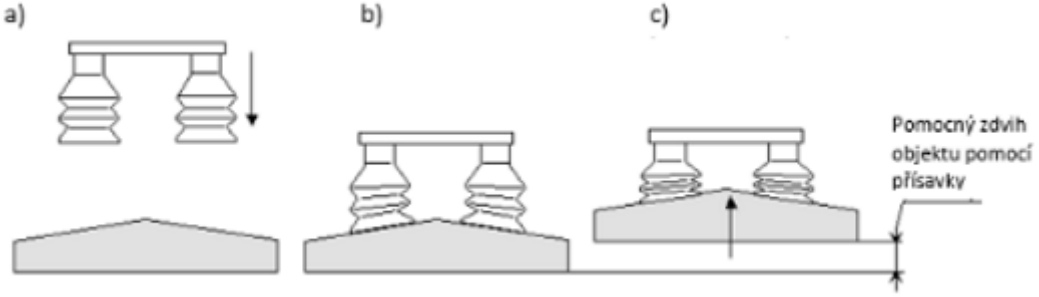
Pokud potřebujeme chytit předmět pomocí robota nebo manipulátoru a nemůžeme si dovolit předmět "obejmout" nebo na něho vytvořit boční tlak čelistí, přichází na řadu uchycení pomocí přísavek a podtlaku. Dnes se úchopové hlavice ve spojení s ejektory využívají především pro potřeby manipulačních úloh. Úchopová síla je definována rozdílem tlaku uvnitř přitisknuté přísavky a jejím okolím.[14]

„Podtlak je definován rozdílem atmosférického tlaku P_a a absolutního tlaku P_1 pod přísavkou ΔP_u= P_a − P_1, přičemž platí 0< ΔP_u<1 .V literatuře je často podtlak uváděn jako záporný přetlak ΔP_u∈ ⟨−P_a, 0⟩ a˙někdy je uváděn %. Vakuum = (ΔP_u)/P_a ∗100%.“[22]



Obrázek 21 Princip podtlakové přísavky [14]

Pro uchopení nepravidelných tvarů, respektive nerovného povrchu, lze využít takzvané skládané přísavky. Tento typ přísavek obsahuje vlnovec, který slouží jako deformační zóna a je schopný korigovat nerovnou plochu pod sebou. Při uchopení tímto typem přísavky dojde nejprve k deformaci vlnovce, který se přizpůsobí sklonu plochy objektu, který chceme zvednout. Poté dojde k zapnutí ejektoru, vytvoření vakua a přisátí k objektu.[14]



Obrázek 22 Princip uchopení přísavkou [14]

II. PRAKTICKÁ ČÁST

6 POPIS STÁVAJÍCÍHO STAVU

Stávající registrace materiálu do skladu zákazníka probíhá manuálně. Operátor vybalí přijatý kus a načte z něj požadované čárové kódy. Každý materiál má přitom jiný dodavatelský štítek a je tedy potřeba, aby operátor přesně věděl, který materiál chce přijmout a které kódy se mají skenovat. Po naskenování kódů pomocí terminálu na PC odešle požadavek do SAP systému, který provede kontrolu a v případě shody vytiskne interní štítek. Tento štítek je operátorem nalepen na předem specifikované místo.

Po operaci načtení a štítkování musí ještě proběhnout verifikace. Opět se jedná o načtení kódů. Tentokrát z vytisknutého štítku, který obsahuje kód materiálu (PN) a unikátní systémem generované číslo přidělené konkrétnímu vstupnímu kusu. Po naskenování systém kontroluje, zda skenované kódy souhlasí s těmi, které byly naposledy odeslány na tiskárnu.

Tímto principem zákazník registruje SMD kotouče se součástkami, které poté putují a osazovací linky. Z pohledu sledování materiálu se tedy nejedná o registraci konkrétního kusu dílu použitého ve výrobě ale o šarži balení. S podobným principem sledování toku materiálu ve výrobě se můžeme setkat napříč průmyslovými odvětvími. Větší komponenty nebo před montované sestavy komponentů často nalezneme s vypálenými nebo vygravírovanými čárovými kódy či DMC které nesou informace o daném výrobku. Pokud je výstupem výrobního procesu nový komponent, bývá většinou označen logem výrobce a podobným štítkem který nese informace o něm (datum výroby, výsledky testů, výrobní závod, a další).

6.1 Nevýhody stávajícího systému

Hlavní nevýhodou stávajícího systému je možnost lidské chyby a nároky na operátory. Při první operaci musí každý operátor vědět s jakým materiálem pracuje a které kódy z dodavatelského štítku potřebuje. Systém sice dokáže rozpoznat špatně naskenované kódy (kódy, které se neměly skenovat) a vrátit chybu, nicméně pokud se to děje často mohou vznikat prostoje při řešení návratové chyby ze systému. Následná operace lepení štítku opět klade nároky na operátora. Operátor musí věnovat pozornost tomu, který vstupní materiál naskenoval a jestli štítek, který se vytiskl nalepil na správný díl. Tady opět co rozdílní vstupní materiál to jiná pozice, kam je možné interní štítek nalepit.

6.2 Požadavky na zlepšení

Hlavním požadavkem je tedy automatizace procesu a vyloučení lidské chyby (minimalizovat vstup obsluhy do registračního a verifikačního procesu). Chyby lze rozdělit do dvou skupin.

- Méně závažné – chyba při prvním skenování dílu a tím způsobené prostoje při registraci.
- Závažná chyba – chyba při lepení vytisknutého štítku a následné verifikaci, kdy se operátor zmýlí a vezme jiný díl, popřípadě mu nejde přečíst kód na štítku a použije již vytisknutý štítek z vedlejšího kusu.

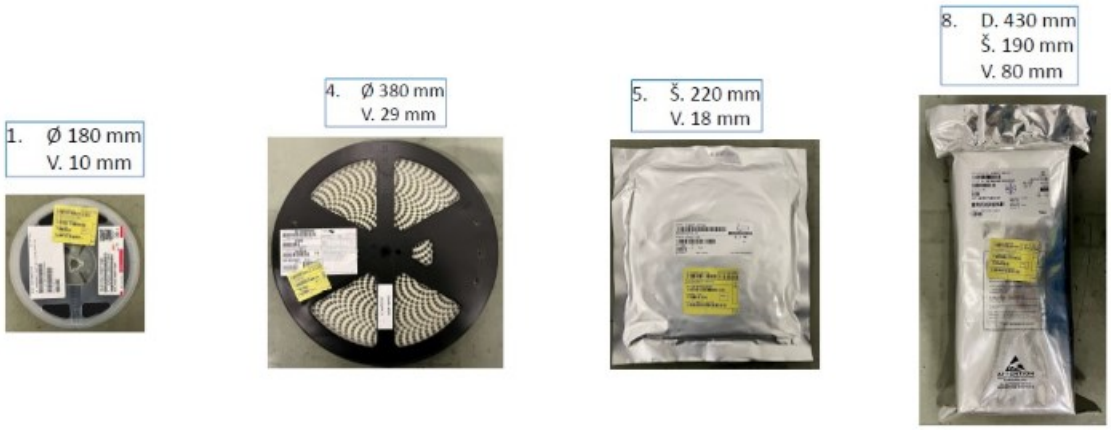
6.3 Zadané parametry pro návrh linky

Základní dělení dílů bylo specifikováno následovně:

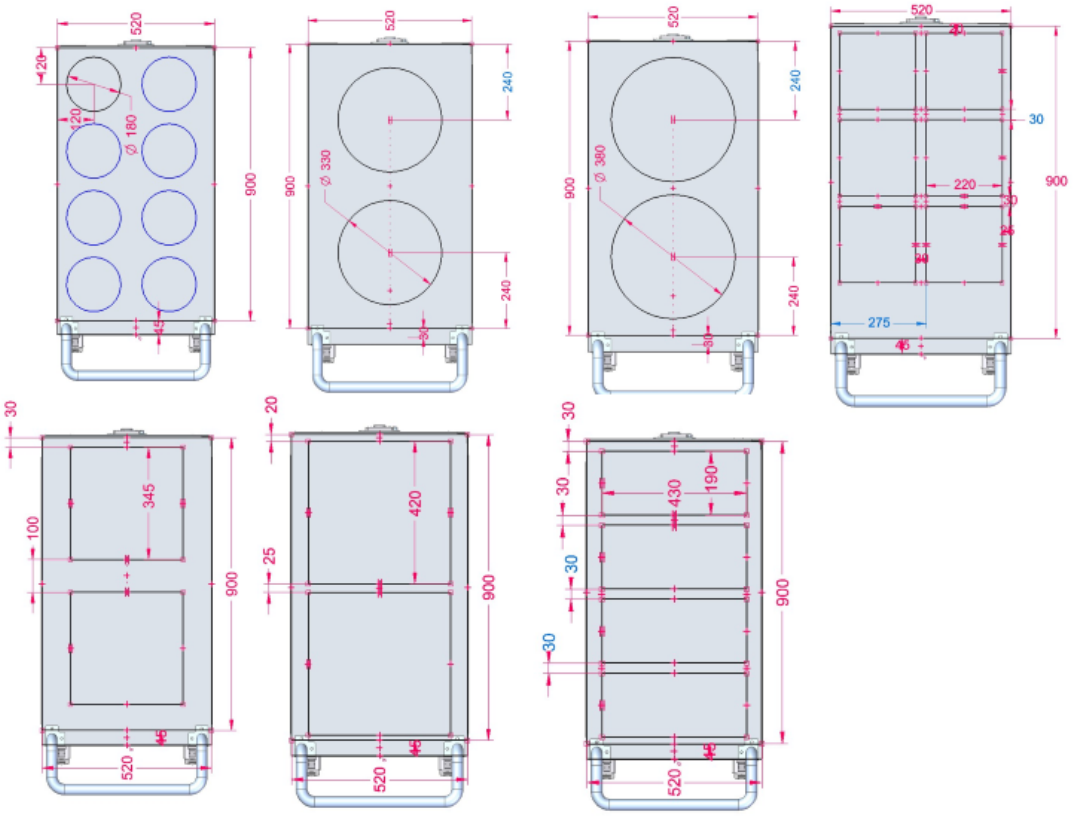
Na obrázku 23 je vidět základní portfolio (nikoli však konečné) dělené na základě mechanických rozměrů. Na jednotlivých dílech jsou vidět bílé dodavatelské štítky (jejich možné umístění pro představu, nikoli však konečné a od všech dodavatelů) a možná pozice interních žlutých štítků (opět pro představu možného umístění, pozici lze jednoduše pro každý materiál měnit, viz dále v popisu). Na následujícím obrázku (Obr. 24) je definice zaskladnění dílů do výstupních vozíků.

Dále byl specifikován předpis pro skládání jednotlivých typů materiálu do manipulačních vozíků.

1. Nemíchat různé druhy balicích jednotek a různé typy dílů.
2. Stoh balicích jednotek nesmí být vyšší než obvodový rám vozíku v= 200 mm.
3. Maximální počet balicích jednotek na sobě je 8 ks.
4. Dále jsou uvedeny předpokládané typy složí podle velikostí materiálu. Aby se využila plocha vozíku lze míchat různé velikosti materiálu do jednoho vozíku (vedle sebe, nikoli na sebe).



Obrázek 23 Základní sortiment



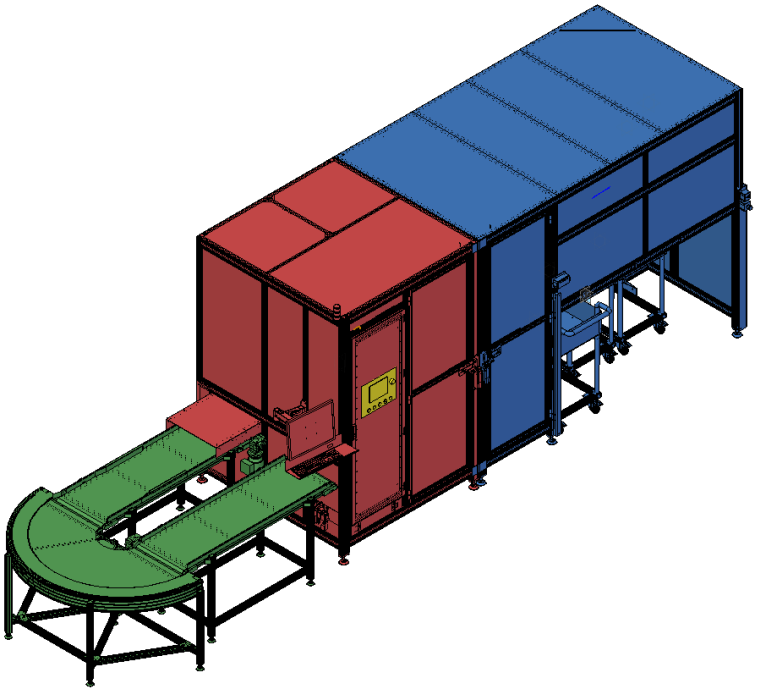
Obrázek 24 Definice složí

7 ZÁKLADNÍ POPIS NAVRHNUTÉ REGISTRAČNÍ LINKY

Registrační linka je navrhnuta jako poloautomatické zařízení. Obsluha vybaluje materiál k příjmu a registraci. Následně ho pokládá na vstupní dopravník. Samotné načtení materiálu, jeho registrace, verifikace a zaskladnění do výstupních vozíků probíhá již bez zásahu obsluhy. Ta na konci linky již pouze vyveze plný vozík s oštítkovanými a verifikovanými díly a naveze vozík prázdný.

Na obrázku 25 je celkový pohled na linku, kde jsou zvýrazněné jednotlivé části, kterými se tato práce zabývá.

- Vstupní dopravníky
    - Základní popis interakce s operátorem
    - Použité technické řešení jednotlivých dopravníku – pohony
- Kamerové navádění s robotem
    - Kamerové navádění
    - Manipulační robot
- Operátorský panel pro ovládání stroje
- Výstupní manipulátor
    - Technické řešení manipulátoru a jeho řízení
    - Virtuální zóna pro odebírání
    - NOK box
    - Výstupní vozíky



Obrázek 25 Celkový pohled na linku

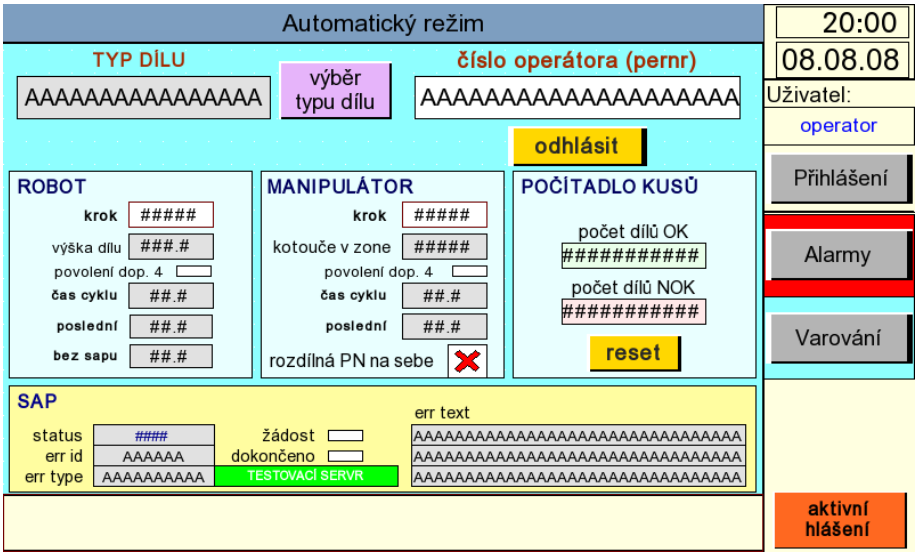
7.1 Ovládání linky

Linka je osazena operátorským panelem s tlačítky "start, stop, servis" pro základní ovládání a nastavení. Dále je na lince osazen ovládací box s tlačítky pro uvolnění výstupních vozíků. Linka obsahuje celkem tři provozní režimy a v závislosti na nastaveném režimu se mění její ovládání.

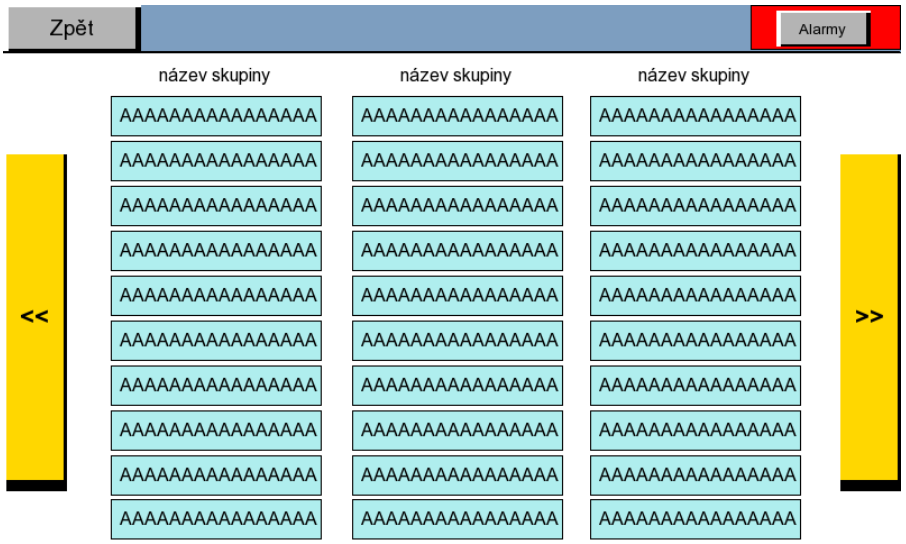
7.1.1 Provozní režimy – plně automatický

Základní režim je plně automatický. Po spuštění linky je operátor vyzván prostřednictvím operátorského panelu k její inicializaci. To znamená že proběhne kontrola snímačů a přestavěný všech akčních členů do základní polohy pro start automatického režimu. Je zkontrolována pozice robota a pokud je to možné, robot je poslán na výchozí souřadnice "HOME". Pokud je robot mimo bezpečnou zónu pro automatické polohování, je operátor vyzván, aby robota ručně posunul do předem dané zóny. Stejně jako robot je i výstupní manipulátor srovnán do výchozí polohy.

Po dokončení inicializace je na panelu zobrazena základní obrazovka automatického režimu (Obr. 26). Operátor se přihlásí osobním ID, které se používá pro komunikaci s nadřazeným systémem a navolí řadu materiálu z databáze stroje, který se bude zpracovávat (Obr. 27). K odstartování automatického režimu použije tlačítko "start".



Obrázek 26 Obrazovka automatického režimu



Obrázek 27 Obrazovka pro výběr dílu

Po spuštění dojde k rozjetí dopravníků jedna až čtyři, na kterých je hledán materiál. Operátor umisťuje materiál na vstupní dopravník. Po něm putuje pod kamerový systém, kde dojde k zastavení dopravníků pomocí snímače. Naváděcí kamera díl vyfotí a získaná data odešle do PLC. PLC vyhodnotí, zda jsou všechny parametry v toleranci pro bezpečný provoz a start robota. Pokud by například nebyl nalezen dodavatelský štítek, který se má číst, ale materiál jako takový byl na dopravníku zaměřen správně (v rámci tolerancí), je takový díl automaticky označen jako NOK a odvezen pod manipulátor, který ho odloží do NOK boxu. Pokud nastane chyba v zaměření dílu jako takového je po posunutí dílu pod manipulátor operátor vyzván k jeho odebrání, protože bez jeho souřadnic ho nelze automaticky odebrat.

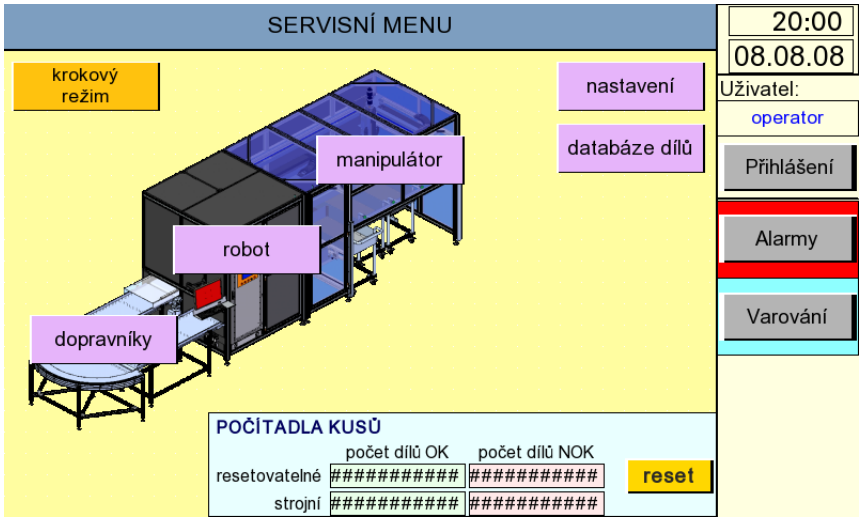
Pokud vše proběhlo v pořádku je spuštěn program robota. Ten najede nad materiál a změří jeho výšku. Poté najede čtecí kamerou, kterou má na chapadle, nad dodavatelský štítek a dojde k přečtení kódů ze štítku. Ty jsou zpracovány v PLC, upraveny do patřičného formátu a odeslány do nadřazeného systému. Pokud systém najde problém, opět je díl označen jako NOK a automaticky odstraněn. Pokud je vše v pořádku dojde k vytisknutí štítku a systém pošle do PLC data, která tiskl pro verifikaci.

Po vytištění interního štítku ho robot převezme od tiskárny a nalepí na místo určené naváděcí kamerou. Po nalepení se nad štítek najede kamerou a dojde ke čtení jeho dat. Tato data jsou odeslána do PLC, kde proběhne kontrola, zda se shodují s daty, která poslal nadřazený systém. Pokud jsou data stejná je jisté, že nedošlo k chybě tisku, ani promíchání dílů nebo štítků. Záznam o verifikaci se opět odesílá zpět do systému s časovou značkou, kdy k verifikaci došlo. Pokud by nastala chyba bude do systému odeslána chyba verifikace a díl je automaticky vyřazen do NOK. V případě že vše proběhlo v pořádku díl pokračuje jako OK.

Všechny díly, které lze zpracovat automaticky, putují po dopravníku pod manipulátor. Ten pomocí přísavky díly odebírá z dopravníku a podle příznaku OK/NOK je zakládá buď do výstupních vozíků nebo do NOK boxu. Po zaplnění výstupního vozíku je operátor upozorněn pomocí světelné signalizace, že může dojít k jeho výměně za prázdný. Linka neobsahuje kontrolu, že nově založený vozík je prázdný. Tato kontrola je na operátorovi.

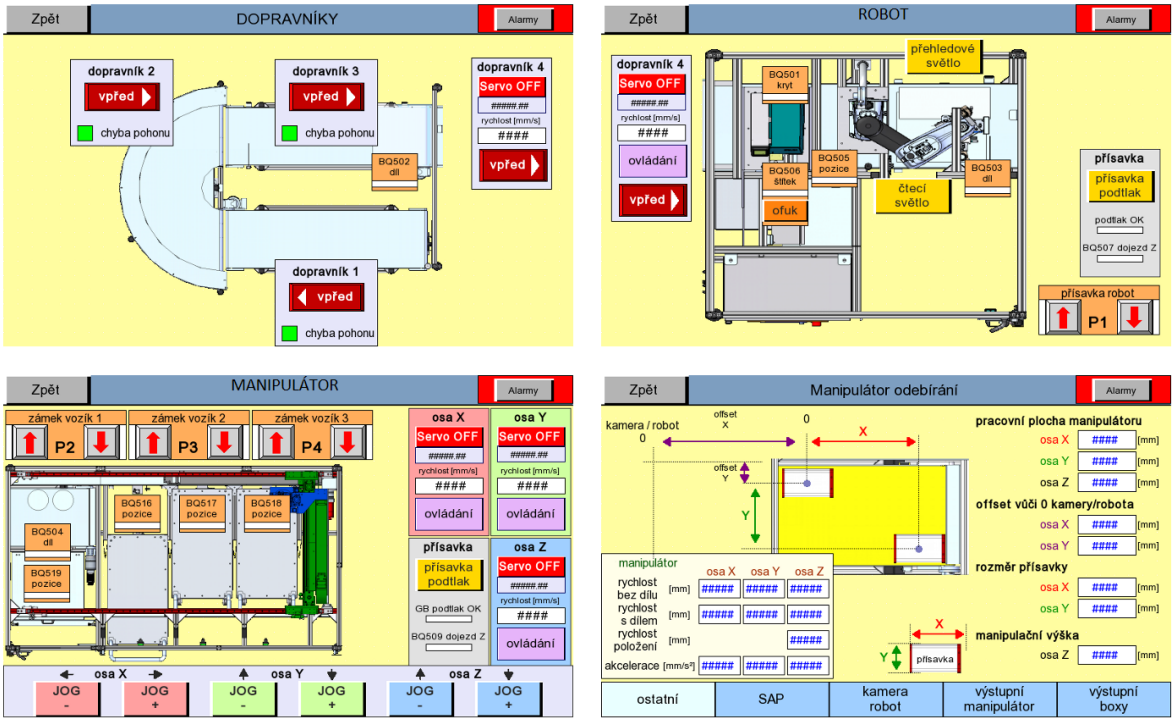
7.1.2 Provozní režimy – servisní režim

Pro potřeby seřízení a nastavení linky je implementován servisní režim. Linka se pomocí přepínacího klíčku u operátorského panelu přepne do polohy servis. Na panelu se zobrazí základní menu servisního režimu (Obr. 28).



Obrázek 28 Obrazovka servisního režimu

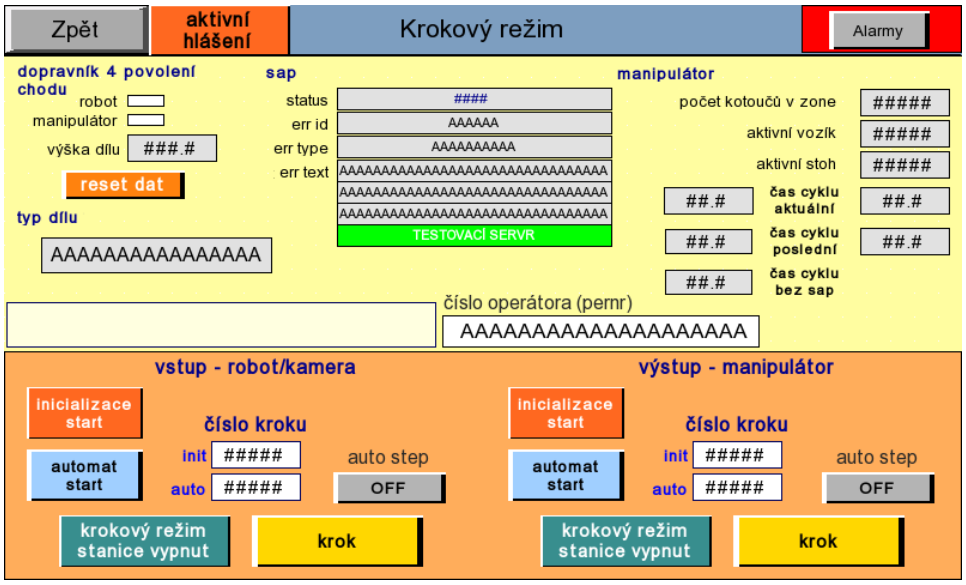
Zde je členěná struktura a servisní technik/údržbář si vybere část stroje, kterou potřebuje ovládat nebo na ní zkontrolovat senzoriku. Tento Přístup vnořování do struktury značně usnadňuje ovládání linky (Obr. 29). Dále je zde přístup k nastavení linky a k databázi materiálů, se kterými stroj umí pracovat.



Obrázek 29 Ukázka obrazovek servisního režimu

7.1.3 Provozní režimy – krokový režim

Pro potřeby seřízení a nastavení linky je implementován servisní režim. Linka se pomocí přepínacího klíčku u operátorského panelu přepne do polohy servis. Na panelu se zobrazí základní menu servisního režimu (Obr. 30).



Obrázek 30 Obrazovka krokového režimu

Tento režim je vhodný při ladění a hledání chyb v programu a pro učení nových typů materiálu do kamerového systému, protože umožnuje postupně krokovat jednotlivé pohyby a komunikace a sledovat, zda jsou správně interpretovány a vyhodnoceny. Ve srovnání s programováním například v C# se jedná o jednoduchý debug, kdy aktivací krokového režimu dojde k zapnutí připravených "breakpointů" v PLC programu.

8 ŘÍZENÍ LINKY

Protože se jedná o zakázkovou výrobu na míru, bývá ve většině případů součástí popisu a specifikace stroje od zákazníka i soupis povolených komponentů. To se týká především výrobce řídicího systému, pohonů, senzoriky, bezpečnosti a podobně. Tyto požadavky na konkrétní výrobce jednotlivých komponentů šetří zákazníkům skladové zásoby náhradních dílů pro údržbu, popřípadě různá školení údržby. Pokud má zákazník již stroje s řízením Omron, údržba má náhradní díly, umí diagnostikovat závadu, popřípadě má zákazník i SW nástroje, je tedy předpoklad, že takový systém bude zákazníkem preferovaný.

Výše uvedenými důvody byl tedy ovlivněn i výběr použitých komponentů na lince. Níže je uvedena specifikace výrobce a konkrétní zvolené typy pro pět hlavních prvků – PLC, HMI, robot, servopohony a kamerový systém.

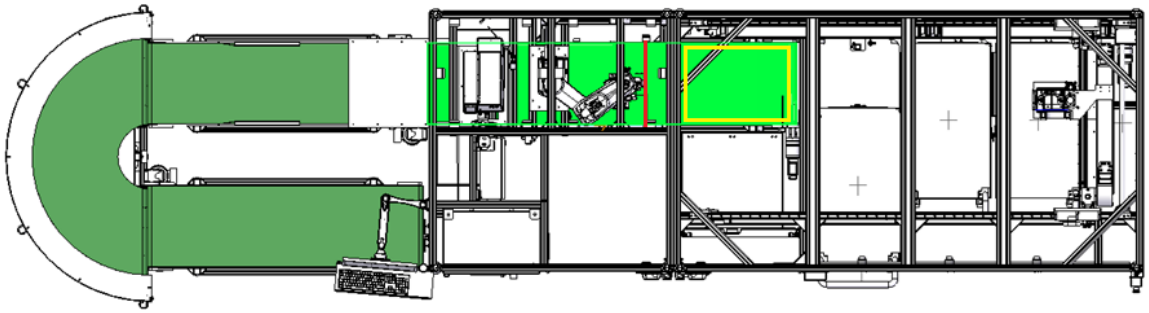
- PLC – OMRON NJ501-R300 – při výběru konkrétního PLC se přihlíželo i k použitému robotu a byla zvolena varianta s plnou robotickou integrací.
- HMI – OMRON NB7W-TW01B – jedná se sedmipalcový, dotykový panel, který slouží jako hlavní prostředek k ovládání a nastavení celé linky. Aby se sjednotilo vývojové prostředí, nabízel se panel řady NA který má i větší sw možnosti. Nicméně panel řady NB je pro linku dostačují a jedná se o levnější řešení. K PLC je připojen pomocí Ethernetu.
- Robot – OMRON RS4-2055004 - jde o SCARA robota, který byl vybrán s ohledem na potřebný dosah a nosnost. Jedná se o variantu robota bez "řízení", který vyžaduje, aby robotické jádro a kinematika běžely v PLC. Tento přístup usnadňuje programování, protože plně integruje PLC i robota do jednoho vývojového prostředí. Samotný robot poté obsahuje pouze výkonovou část pro motory a s řídicím systémem komunikuje pomocí sběrnice EtherCAT.
- Servopohony – OMRON R88D-1S – protože je použito řízení OMRON je vhodné (a v tomto případě i předepsané) použití servopohonů stejné značky. To stejné platí i pro robota. Zjednoduší se integrace, nastavení a připojení. Stejně jako u robota je použita komunikace EtherCAT.
- Kamerový systém - OMRON FH-2050 - kamerový systém byl volen s ohledem na to, že ho již zákazník využívá a má s ním zkušenosti. Ke kamerovému kontroléru jsou připojeny celkem dvě kamery. Staticky umístěná kamera FH-SMX05 pro hledání a zaměření dílu na dopravníku a kamera FH-SM12 umístěná na chapadle

robota pro čtení čárových kódů. Připojení k PLC je realizování jak pomocí EtherCAT (základní diagnostika a ovládání systému – přepnutí scény, trig kamery, hlášení o poruše atd.) tak pomocí Ethernet TCP/IP spojení, po kterém jsou přenášené načtené čárové kódy.

8.1 Vstupní dopravníky

Vstupní dopravník je jedním ze dvou bodů, s kterými operátor při běžné výrobě pracuje.

Dopravníky jsou rozděleny na dvě části. Vnější dopravník, který se skládá z dvou rovných pásů a jedné zatáčky o 180°, na obrázku 31 tmavě zelená část. Operátor umisťuje vstupní materiál na první rovnou část před zatáčku a to tak, aby se díly nedotýkaly a nepřekrývaly. Doporučená mezera pro bezproblémový provoz je cca dva centimetry. Materiál namotaný na kotouči není vázán žádným předpisem a lze jej tedy umístit na pás libovolně natočený. Balení, která mají čtvercový nebo obdélníkový tvar, již musí být umístěna rovnoběžně s dopravníkem a to tak, že delší hrana dílu je rovnoběžně s hranou dopravníku.



Obrázek 31 Vstupní dopravníky

Na konci vnějších dopravníků je kontrolní snímač, který v případě, že vnitřní dopravník (na obrázku 31 světle zelená část), není schopný díly převzít, zastaví vnější dopravníky. Vnější dopravníky jsou řízeny asynchronními motory s měniči pomocí bitové komunikace s PLC bez možnosti nastavení rychlosti. Ta je napevno nastavena v měniči. Měniče jsou použity z důvodů nastavení požadované rychlosti posuvu materiálu. Samotné řízení si vystačí pouze s dvěma signály – 1x signál z PLC "chod" a 1x signál do PLC "chod". Ačkoli je připojení minimalistické, lze jednoduše vyhodnotit i poruchový stav pohonu. Pokud je odeslán pokyn k roztočení dopravníku, měl by se zpět vrátit signál o provedení pokynu. Pokud se tak nestane do stanového času, dojde k vyhlášení poruchy daného motoru.

Vnitřní dopravník je již operátorovi nepřístupný. Pro správnou kamerovou identifikaci dílů je potřeba, aby ve směru pohybu dílů kamera viděla vždy pouze jeden díl. Z toho důvodu

má tento dopravník zvýšenou rychlost oproti vnějším, aby došlo k "odtrhnutí" dílu a vytvoření mezery mezi díly. Toto řešení je z důvodu využití délky vnějších dopravníků a možnosti skládat materiál na dopravník těsně u sebe bez ohledu na to, že by takto nebyl zpracovatelný pro kamerový systém. Potřebnou mezeru si tedy stroj tvoří sám.

8.1.1 Vnitřní dopravník, virtuální zóna

Protože je vnitřní dopravník společný pro kamerový systém s robotem i pro výstupní manipulátor, je řízen servopohonem, pomocí kterého dochází k přesnému polohování již načtených dílů.

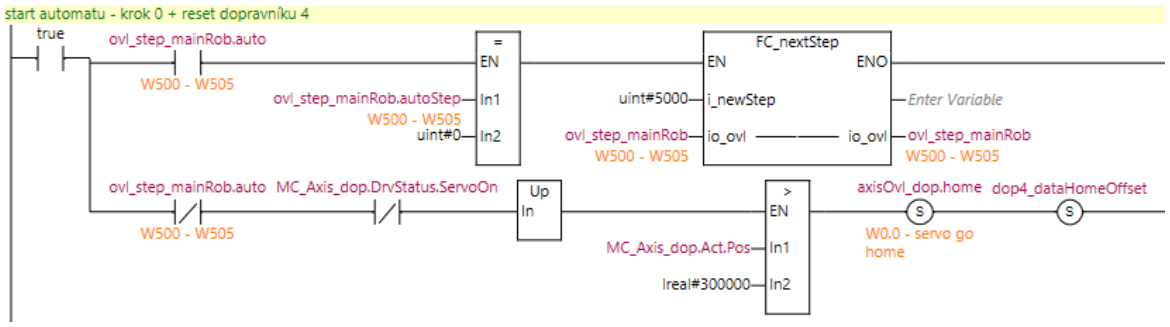
Po najetí dílu na snímač v části s robotem dojde k zastavení dopravníku a zpracování daného dílu (Obr. 33 - červená linka). Po dokončení tohoto procesu jsou data dílu (pozice dílu na dopravníku, stav dílu) uložena do virtuálního buffru dopravníku. Tento buffer má celkem 15 pozic. Mechanicky je možné mezi stanice umístit maximálně 12 dílů a 3 pozice zůstávají jako rezerva. Díky tomu zná PLC stále polohy zpracovaných a zatím ne zaskladněních dílů na dopravníku. Níže je uvedený kód, který zajištuje posuv souřadnic dílu po dopravníku.

```
// vyhodnocení pozice kotouče na dopravníku 4
procesniData.man_kotouc_zona_count := 0;
dop4_dily := 0;
FOR x := 1 TO SizeOfAry(dataDilu_pas) DO
	// kontrola zda je pole používané
	IF dataDilu_pas[x].status_dil <> c_status_dil#nul THEN
		inc(dop4_dily);
		// kontrola pozice kotouče na dopravníku
		IF (dataRet.man_pos_dop_offset_kamRob_x - dataDilu_pas[x].pozice_dopravnik_x +
		(dataDilu_pas[x].rozmer_dilu_x / 4)) <= (MC_Axis_dop.Act.Pos - dataDilu_pas[x].offset_dopravnik_x)
		THEN
			// kontrola OK / NOK dílu
			IF dataDilu_pas[x].status_dil = c_status_dil#verifikace_done THEN
				dataDilu_pas[x].status_odebrani := c_status_dil#odebrani_ok;
			ELSE
				dataDilu_pas[x].status_odebrani := c_status_dil#odebrani_nok;
			END_IF;
			dataDilu_pas[x].odebrani := TRUE;
		END_IF;
	END_IF;
	// vyhodnocení počtu kotoučů pod manipulátorem
	IF dataDilu_pas[x].odebrani THEN
		inc(procesniData.man_kotouc_zona_count);
	END_IF;
END_FOR;
```

Nejprve jsou smazány pomocné proměnné o aktuálním počtu dílů na dopravníku "**dop4_dily**" a počet kotoučů v zóně manipulátoru pro odebrání (Obr. 33 - žlutá zóna) "**procesniData.man_kotouc_zona_count**". Poté je ve smyčce prohledán buffer dopravníku "**dataDilu_pas** ", kde se kontroluje, zda je pozice buffru obsazena (status dílu není nulový

"**dataDilu_pas[x].status_dil** "). Pokud se jedná o použitou pozici je kontrolována jeho pozice s ohledem na aktuální polohu (posuv) motoru " **MC_Axis_dop.Act.Pos** ". Pokud je vypočtená pozice již v zóně manipulátoru, díl je označen pro odebrání " **dataDilu_pas[x].odebrani := TRUE** " a je započítán do odebírací zóny " **inc(procesniData.man_kotouc_zona_count)** ". S těmito informacemi poté pracuje výstupní manipulátor.

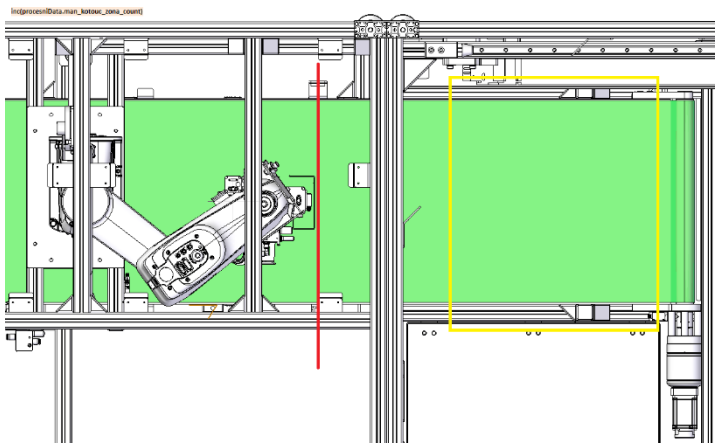
Pro uchování pozice servopohonu používá PLC datový typ LREAL. Ten má rozsah +2.22507385850721e−308 do +1.79769313486231e+308. Vzhledem k tomu, že se dopravník točí pouze jedním směrem může se stát, že proměnná "přeteče". Pokud by se to stalo během pohybu dílů do virtuální zóny, došlo by ke ztrátě pozice dílu na dopravníku. Aby se zamezilo ztrátě pozice je pozice dopravníku pravidelně resetována. Nejprve je při spuštění kontrolována pozice dopravníku " **MC_Axis_dop.Act.Pos**" (Obr. 32) . Pokud je větší než nastavená konstanta, dojde k volání kódu pro nulování pohonu a úpravě souřadnic všech dílů, které jsou zapsány v buffru dopravníku, aby se neztratila jejich pozice.



Obrázek 32 Nulování dopravníku 4 LD

ST část ošetření přetečení dopravníku 4, která posouvá všechny virtuální díly tak, aby se neztratila jejich pozice po snulování dopravníku.

```
// korekce dat na dopravníku z důvodů HOME dopravníku
IF dop4_dataHomeOffset THEN
        dop4_dataHomeOffset := FALSE;
        FOR x := 1 TO SizeOfAry(dataDilu_pas) DO
                // kontrola zda je pole používané
                IF dataDilu_pas[x].status_dil <> c_status_dil#nul THEN
                        dataDilu_pas[x].offset_dopravnik_x := dataDilu_pas[x].offset_dopravnik_x -
                        MC_Axis_dop.Act.Pos;
                END_IF;
        END_FOR;
END_IF;
```



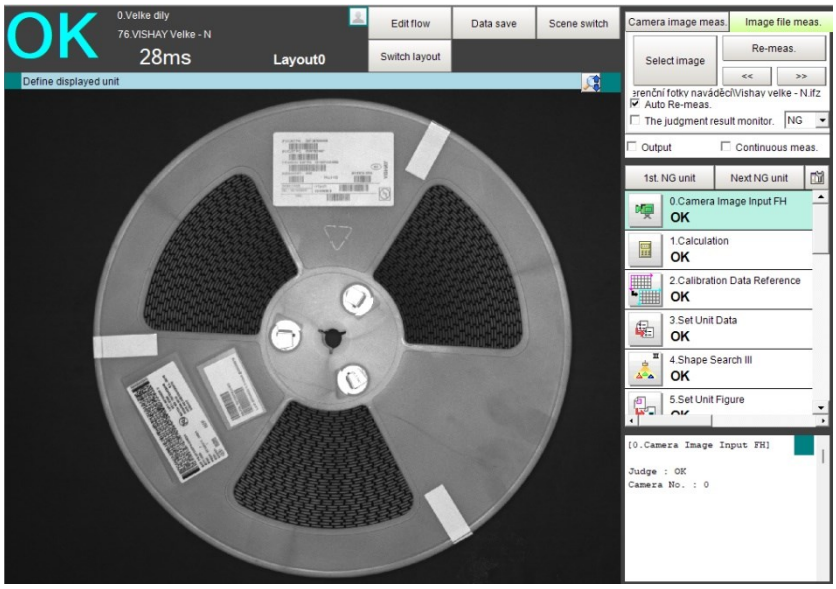
Obrázek 33 Vnitřní zóny dopravníku 4

8.2 Kamerový systém

Důležitým prvkem linky je kamerový systéme, který slouží jak pro čtení čárových kódů, tak pro nalezení dílu, štítku a místa pro lepení nového štítku. Konkrétní typ byl zvolen i s ohledem na požadavek zákazníka, který již tyto systémy používá. Jak bylo zmíněno připojení k PLC je skrze dva kanály. Jeden prostřednictvím EtherCatu, kde jsou přenášeny statusy systému, chyby, trig kamery a přepínání jednotlivých scén. Pro data získaná ze systému je použitý druhý kanál a to TCP/IP. Po tomto kanále jsou přenášeny všechny výsledky jako souřadnice a načtené kódy. Toto řešení je z důvodů omezení, které kamerový systém má respektive omezení funkcí, které čtou čárové kódy. Ty dokáží odeslat přečtený text pouze po RS232 nebo po ethernetu.

8.2.1 Naváděcí kamera

Naváděcí kamera je umístěna nad vnitřním dopravníkem. Po zastavení dílu na snímači pod kamerou dojde k vyfocení dílu na dopravníku a následnému hledání jednotlivých prvků na něm. Ukázka naváděcí scény je na obrázku níže (Obr. 34). Nejprve je nalezen díl a jeho střed vůči dopravníku. Je změřen jeho průměr (u kotoučů, u čtvercového) nebo rozměr v x a y ose u obdélníkového balení. Tyto informace se používají pro odebrání dílu výstupním manipulátorem a pro kontrolu, že vstupní materiál odpovídá rozměrově předpokládanému vstupu. Následně je zaměřen dodavatelský štítek, který je potřeba načíst, a jeho natočení na díle. Poslední údaj je pozice a natočení bodu, kam chceme, aby se nalepil tištění štítek. Informace o starém i novém štítku jsou použity pro navádění robota.



Obrázek 34 Ukázka naváděcí scény

Všechny tyto informace jsou do PLC odeslány skrze TCP/IP protokol pomocí bloku "data output". Ten data odešle jako string a mezi jednotlivé položky přidá definovaný oddělovač (v našem případě ",") a zakončení (";"). Přijatá data v PLC tedy vypadají následovně:

počet viditelných dílů,
index aktuálně odeslaného dílu,
rozměr dílu osa X,
rozměr dílu osa Y,
pozice středu osa X (pozice na dopravníku),
pozice středu osa Y (pozice na dopravníku),
pozice stítku osa X,pozice štítku osa Y,
rotace štítku,pozice volného místa osa,
pozice volného místa osa Y,
rotace volného místa;

Bez ohledu na složitost scény u některých typů (snaha sjednotit více typů do jedné scény) musí být formát výstupních dat vždy stejný, aby ho PLC dokázalo zpracovat.

Kód pro zpracování dat z přehledové kamery.

```
// data z přehledové kamery - zpracování
tmpStringFull := AryToString(cam_tcpCom.dataRecive[0], cam_tcpCom.dataReciveLength - 1);
SubDelimiter(tmpStringFull, cam_tcpCom.dataDecode, _eDELIMITER#_COMMA);
procesniData.kam_viditelneDily := real_to_uint(cam_tcpCom.dataDecode.viditelneDily);
procesniData.kam_zpracovavanyDil := real_to_uint(cam_tcpCom.dataDecode.zpracovavanyDil);
procesniData.dataDilu_zpracovavany.rozmer_dilu_x := cam_tcpCom.dataDecode.rozmer_dilu_x;
procesniData.dataDilu_zpracovavany.rozmer_dilu_y := cam_tcpCom.dataDecode.rozmer_dilu_y;
procesniData.dataDilu_zpracovavany.pozice_dopravnik_x := real_to_lreal(dataRet.kamera_offset_x +
cam_tcpCom.dataDecode.pozice_dopravnik_x);
procesniData.dataDilu_zpracovavany.pozice_dopravnik_y := real_to_lreal(dataRet.kamera_offset_y +
cam_tcpCom.dataDecode.pozice_dopravnik_y);
procesniData.dataDilu_zpracovavany.stitek_x := dataRet.kamera_offset_x + cam_tcpCom.dataDecode.stitek_x;
procesniData.dataDilu_zpracovavany.stitek_y := dataRet.kamera_offset_y + cam_tcpCom.dataDecode.stitek_y;
procesniData.dataDilu_zpracovavany.stitek_r := cam_tcpCom.dataDecode.stitek_r;
procesniData.dataDilu_zpracovavany.misto_x := dataRet.kamera_offset_x + cam_tcpCom.dataDecode.misto_x;
```

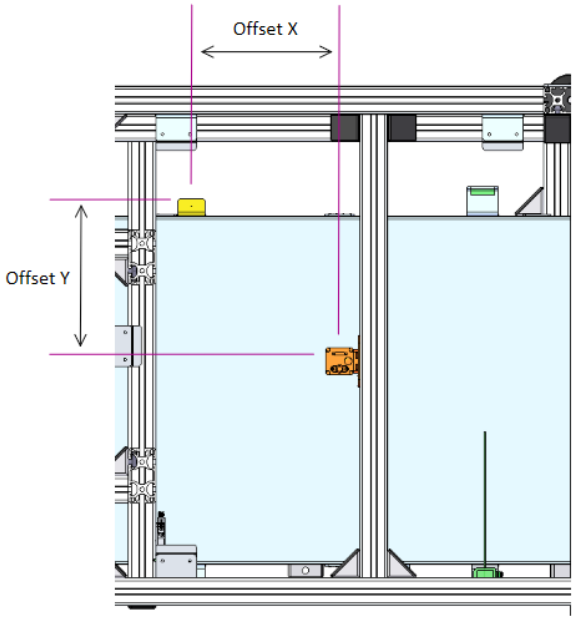
```
procesniData.dataDilu_zpracovavany.misto_y := dataRet.kamera_offset_y + cam_tcpCom.dataDecode.misto_y;
procesniData.dataDilu_zpracovavany.misto_r := cam_tcpCom.dataDecode.misto_r;
```

Po přijetí zprávy pomocí funkčního bloku SktTCPRcv je pole byte předáno dále ke zpracování. Díky stanovému formátu dat lze na straně PLC jednoduše použít vestavěnou funkci SubDelimiter. Přijaté pole je tedy převedeno na string

```
"tmpStringFull := AryToString(cam_tcpCom.dataRecive[0], cam_tcpCom.dataReciveLength - 1);"
```

a je vynechán zakončovací znak ;. Takto získaný řetězec znaků slouží jako vstupní data pro SubDelimiter. Ten podle zvoleného oddělovače rozdělí jednotlivé podřetězce (data) na samostatné proměnné "SubDelimiter(tmpStringFull, cam_tcpCom.dataDecode, _eDELIMITER#_COMMA);". Jako druhý parametr funkce je struktura dat, kam se jednotlivé oddělené prvky zapíší a automaticky se převedou na požadovaný datový typ. V níže uvedené struktuře je vidět, že se každý prvek převádí na proměnnou typu REAL. Tato interní funkce parsování řetězců s automatickým převodem datových typů zjednodušuje zpracování vstupních dat z kamerového systému.

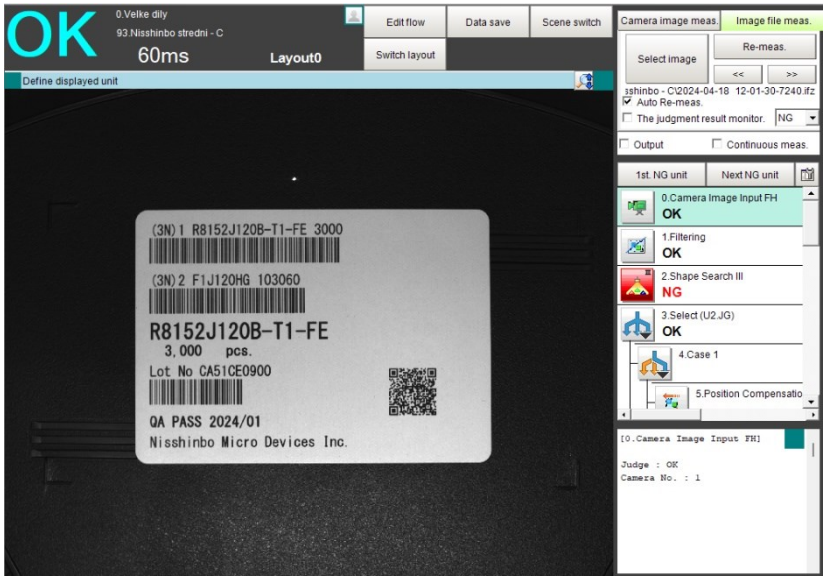
Následně se tyto převedené proměnné dle potřeby upraví a vloží se do struktury "**procesniData.dataDilu_zpracovavany**", kde jsou drženy informace o aktuálně zpracovávaném dílu. Úprava se týká souřadnic dílu, štítku a volného místa pro lepení, ke kterým jsou přičteny kalibrační offsety naváděcí kamery vůči dopravníku "**dataRet.kamera_offset_x**""**dataRet.kamera_offset_y**". Ty byly odměřeny a jsou nastaveny tak, aby například souřadnice 100X 100Y od kamery odpovídala posunutí 100mm v X a 100mm v Y od kalibračního bodu dopravníku. Na obrázku níže (Obr. 35) je půdorys dopravníku v místě zpracování. Zeleně je označen laserový reflexní snímač, který slouží k detekci dílu na dopravníku a jeho zastavení. Oranžově je zobrazena přehledová kamera. Žlutě poté nulový bod dopravníku, vůči kterému jsou stanoveny kalibrační offsety.



Obrázek 35 Offsety naváděcí kamery

8.2.2 Čtecí kamera

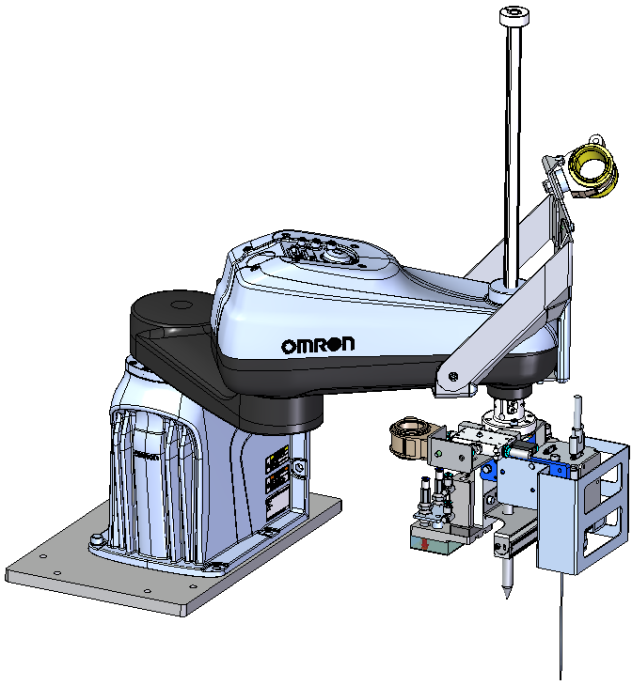
Po zaměření dílu je robot odeslán nad nalezený dodavatelský štítek, aby z něho kamera, kterou nese, mohla sejmout potřebné informace. Potřebná data, které musí systém přečíst, jsou pro všechny typy materiálu stejná – fifo, mfprn, pn, lot, qty. První čtyři jsou identifikační kódy, podle kterých lze jednoznačně daný materiál identifikovat, poslední qty je množství v konkrétním skenovaném balení.



Obrázek 36 Ukázka čtecí scény

8.3 Robot

Linka obsahuje robota, který slouží k manipulaci s částí kamerového systému pro čtení štítků a přísavky pro odebírání a lepení vytisknutého štítku. Při výběhu bylo přihlédnuto k použitému řídicímu systému a byl tedy zvolen robot od stejného výrobce. Protože nástroj robota je potřeba vždy mít rovnoběžně s dopravníkem byla zvolena kinematika SCARA. Samotná jednotka robota neobsahuje žádné řízení a celá kinematika je plně integrována v použitém PLC. V patě robota je tedy pouze výkonová část pro motory.



Obrázek 37 Použitý robot s nástrojem

Základní poloha robota je umístěna bokem mimo dopravník tak, aby robot nestínil přehledové kameře při zaměření dílu. Pro spuštění stroje je potřeba, aby robot byl v této pozici. Pro ulehčení práce operátorovi se robot dokáže automaticky dorovnat v případě, že je v definované zóně nad dopravníkem a je tedy bezpečné automatické srovnání. Pokud se v této zóně nenachází je operátor vyzván, aby robota ručně do zóny posunul.

8.3.1 Nástroj robota

Nástroj robota se skládá ze tří prvků, které jsou upevněny na čtvrté ose robota, na obrázku (Obr. 38) modře.

- Snímač pro odměření výšky materiálu – nástroj obsahuje optický difúzní snímač pro odměření výšky čteného dílu, na obrázku (Obr. 38) zeleně. Po

najetí nad díl se robot rozjede osou 3 směrem dolů k dílu a kontroluje stav snímače. Jakmile snímač sepne logický stav TRUE. Pohyb robota se zastaví. Na náběžnou hranu tohoto signálu PLC odečte pozici třetí osy.

kód robota pro pohyb osou 3

```
; měření výšky
WAIT NOT rob_bq_vyska
DO
        DRIVE 3, 3, rob_speed_z
        DECOMPOSE locpos_pole[0] = HERE
UNTIL (rob_bq_vyska) OR (locpos_pole[2] < -10)
```

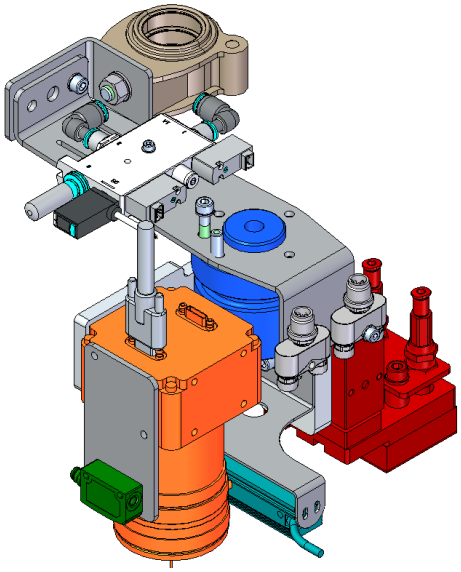
Na začátku odměření je provedena kontrola, zda je snímač v logické nule FALSE. Pokud tomu tak není PLC vyhlásí chybu. "WAIT NOT rob_bq_vyska". Poté je spuštěna smyčka DO, která posune osu 3 o tři milimetry níže a zkontroluje stav snímače na logickou hodnotu TRUE.

kód PLC pro odměření výšky

```
// odměření výšky kotouče
IF rob_step_from = 5000 THEN
        IF rob_BQ_vyska AND NOT trig_3 THEN
                procesniData.dataDilu_zpracovavany.rozmer_dilu_z :=
                dataRet.rob_offsetDopravnik_vyska -
                LREAL_TO_REAL(RC_Robot001.JointActPos.J3);
        END_IF;
END_IF;
trig_3 := rob_BQ_vyska;
```

Pokud je robot v kroku měření výšky " `IF rob_step_from = 5000 THEN`" je vyhodnocován stav snímače. Po náběžné hraně je do procesních dat uložena výška dílu, která je spočítána odečtením aktuální pozice osy tři robota od referenční výšky prázdného pásu. Z důvodů zpoždění systému (senzor, sběrnice IO-link, PLC) je odměření v toleranci +-1 až 1,5mm. To je dostačující přesnost pro kamerový systém, aby po najetí kamery byl text ostrý.

- Kamerový senzor pro čtení čárových kódů na obrázku (Obr. 38) oranžově spolu s přídavným osvětlením.
- Přísavka pro štítky s pneumatickým výsuvem obrázku (Obr. 36) červeně. Ta je určena pro odebírání štítků z tiskárny a jeho lepení na díl. Z důvodů relativně veliké chyby při odměření dílu jsou přísavky, které drží štítek umístěny na pneumatickém válci s chodem 30 mm. Ten kompenzuje možnou chybu výšky díly a umožní tak bezproblémové nalepení štítku.

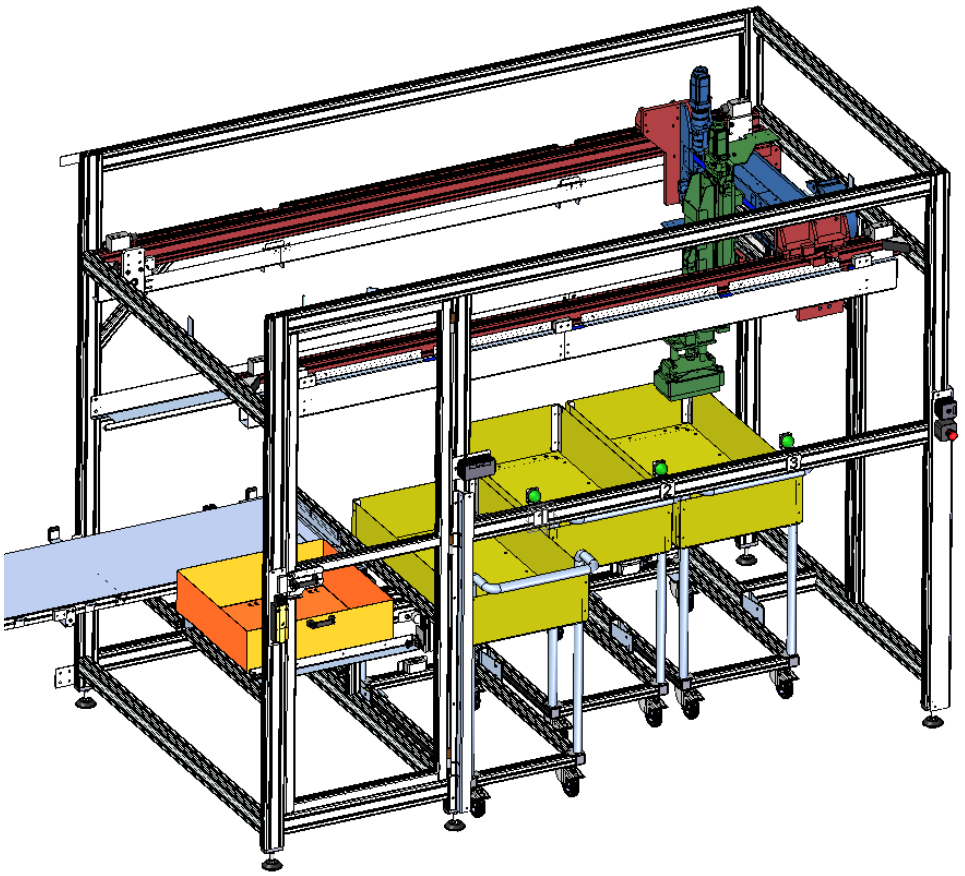


Obrázek 38 Nástroj robota

8.4 Výstupní sekce - manipulátor

Po zpracování je díl pomocí dopravníku čtyři posunutý pod výstupní manipulátor, který díl odebere a uloží, a to do OK vozíku nebo do NOK boxu podle jeho stavu. Na obrázku (Obr. 39) jsou OK vozíky znázorněny žlutě, NOK box potom oranžově. Aby se neztrácel čas dojezdem na koncový snímač dopravníku a mohlo za určitých podmínek probíhat odebrání dílu a štítkování nového současně, obsahuje dopravník virtuální zónu pro odebrání. Pokud dojde k zastavení dopravníku z důvodů najetí nového dílu pod kamerový systém a robota, zkontrolují se všechny díly ve virtuální buffru dopravníku čtyři a pokud je nějaký v zóně pro odebrání spustí se výstupní manipulátor. Manipulátor je tedy plně oddělený od sekce s robotem a je plně autonomní.

Samotný manipulátor se poté skládá ze tří lineárních os připojených na sebe (osa X – červeně, Y – modře, Z – zeleně) a přísavkové hlavy na konci osy Z, která slouží k uchopení dílu. S manipulátorem se jezdí pouze po přímých dráhách a nejsou potřeba žádné synchronizace několika os dohromady. To zjednodušuje jeho řízení, kde každý motor je ovládaný samostatně jako polohová osa.

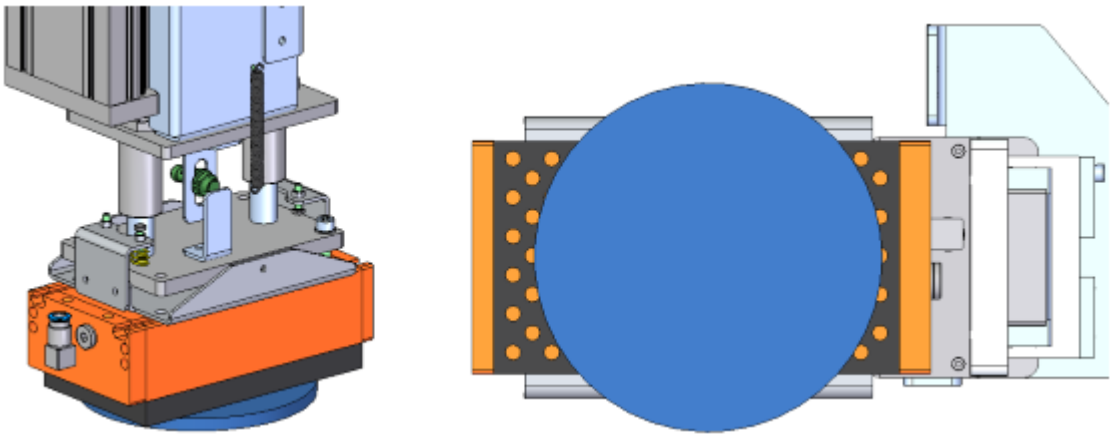


Obrázek 39 Výstupní sekce

8.4.1 Chapadlo manipulátoru

Protože vstupní díly jsou různých velikostí a mají různé prolisy a odlehčení, byla přísavka zkonstruována tak, aby dokázala spolehlivě manipulovat s jakýmkoli vstupním materiálem. Proto je použita přísavková hlava KENOS KVG120, která obsahuje pole podtlakových kanálů s automatickým uzavřením, pokud daný kanál není přiložený k dílu a nelze na něm vytvořit podtlak, jak je vidět na obrázku (Obr. 40) kde je znázorněný nejmenší díl. Kraje přísavky jsou mimo obrys dílu a kanálky, které vidíme se automaticky uzavřou, aby nedocházelo k úniku vzduchu. Celá přísavková hlava je volně uložena v ose Z a obsahuje detekční snímač pro měření výšky materiálu. Vzhledem k chybě odměření výšky zpracovávaného dílu pomocí robota s difuzním snímačem, obsahuje manipulátor snímač, který je označený na obrázku (Obr. 40) zeleně, pro detekci dotyku přísavkové hlavy s materiálem. Jede-li osa Z směrem dolů a hlava narazí na díl, dojde k zastínění indukčního snímače, což způsobí zastavení pohybu směrem dolů a odečtení hodnoty enkodéru motoru. Z této hodnoty se stejně jako u robota odečte reference na prázdný dopravník a výsledkem je přesná výška materiálu. Ta je potřebná pro správné uložení materiálu. Snímač také

zamezuje poškození manipulovaného materiálu při zakládání v případě, že by do stroje nebyl vložený prázdný vozík.



Obrázek 40 Chapadlo manipulátoru

8.5 Výstupní vozíky - složе

Zákazníkem jsou definovány výstupní složе (Obr. 24) a stejně tak požadavek, aby se maximálně využívalo místo v každém vozíku. Teoreticky dokáže linka zpracovávat rozdílné velikosti materiálu najednou. Proto je tedy možné míchat složе tak, aby byla plocha vozíku maximálně využita. Aby bylo řešení univerzální, vychází se z nejmenších dílů. Tedy maximálního počtu dílů do jedné vrstvy. Jak je ze zadání patrné jedná se tedy o matici 2 x 4. Kód pro generování složí je rozdělený na dvě části.

8.5.1 Kontrola obsazenosti vozíku

Pro jednotlivé prvky matice se zohledněním jejich velikosti. V ukázce je kód pro první čtyři prvky. Zbylé čtyři jsou kopie s jinými indexy pole. Je tedy procházeno pole 1–8 kde každý index označuje pozici v matici vozíku. Pro zjednodušení podmínek je použita reverzní logika – pozice je nejdříve zakázána a poté proběhne kontrola, zda se může povolit.

```
// ##### správa pozice vozíku a NOK #####
(*
  X
-------          pokud je stoh 0 větší v x větší než 1/2 rozměru bedny stoh 1 disable (automaticky pro další řady)
| 1 | 2 |        pokud je stoh 0 větší v y větší než 1/4 rozměru bedny stoh 2 disable (automaticky pro další sloupce)
-------          mezera mezi stohy je počítána 25mm
| 3 | 4 |
------- Y
| 5 | 6 |
-------
| 7 | 8 |
```

```
-------
*)

// vyhodnocení volné pozice pro založrení do vozíku dle obsazenosti a příchozího kotouče
FOR i := 1 TO SizeOfAry(voziky) DO
	// kontrola prvku [1]
	voziky[i].stoh[1].disable := TRUE;
	IF (voziky[i].stoh[1].rozmer_x = 0) THEN
		voziky[i].stoh[1].disable := FALSE;
	ELSE
		IF (voziky[i].stoh[1].rozmer_x = dataDilu_manipulator.rozmer_dilu_x) THEN
			IF (voziky[i].stoh[1].rozmer_y = dataDilu_manipulator.rozmer_dilu_y) THEN
				IF (voziky[i].stoh[1].pn = dataDilu_manipulator.pn OR voziky[i].stoh[1].pn = '' OR
				dataRet.vozik_stohovani_pn) THEN
					voziky[i].stoh[1].disable := FALSE;
				END_IF;
			END_IF;
		END_IF;
	END_IF;
	// kontrola prvku [2]
	voziky[i].stoh[2].disable := TRUE;
	IF (not voziky[i].stoh[1].disable) THEN
		IF (dataRet.vozik_rozmer_x - voziky[i].stoh[1].rozmer_x - dataRet.vozik_mezeraStohu >=
		dataDilu_manipulator.rozmer_dilu_x + dataRet.vozik_mezeraStohu) THEN
			IF (voziky[i].stoh[2].pn = dataDilu_manipulator.pn OR voziky[i].stoh[2].pn = '' OR
			dataRet.vozik_stohovani_pn) THEN
				voziky[i].stoh[2].disable := FALSE;
			END_IF;
		END_IF;
	ELSE
		// překopírování hodnoty z osy Y
		voziky[i].stoh[2].rozmer_y := voziky[i].stoh[1].rozmer_y;
	END_IF;
	// kontrola prvku [3]
	voziky[i].stoh[3].disable := TRUE;
	IF (dataRet.vozik_rozmer_y - (2 * dataRet.vozik_mezeraStohu) - voziky[i].stoh[1].rozmer_y >=
	dataDilu_manipulator.rozmer_dilu_y + dataRet.vozik_mezeraStohu) THEN
		IF (voziky[i].stoh[3].rozmer_x = 0) THEN
			IF (voziky[i].stoh[3].pn = dataDilu_manipulator.pn OR voziky[i].stoh[3].pn = '' OR
			dataRet.vozik_stohovani_pn) THEN
				voziky[i].stoh[3].disable := FALSE;
			END_IF;
		ELSE
			IF (voziky[i].stoh[3].rozmer_x = dataDilu_manipulator.rozmer_dilu_x) AND
			(voziky[i].stoh[3].rozmer_y = dataDilu_manipulator.rozmer_dilu_y) THEN
				IF (voziky[i].stoh[3].pn = dataDilu_manipulator.pn OR voziky[i].stoh[3].pn = '' OR
				dataRet.vozik_stohovani_pn) THEN
					voziky[i].stoh[3].disable := FALSE;
				END_IF;
			END_IF;
		END_IF;
	END_IF;
	// kontrola prvku [4]
	voziky[i].stoh[4].disable := TRUE;
	IF (not voziky[i].stoh[3].disable) THEN
		IF (dataRet.vozik_rozmer_x - voziky[i].stoh[3].rozmer_x - dataRet.vozik_mezeraStohu >=
		dataDilu_manipulator.rozmer_dilu_x + dataRet.vozik_mezeraStohu) THEN
			IF (voziky[i].stoh[4].pn = dataDilu_manipulator.pn OR voziky[i].stoh[4].pn = '' OR
			dataRet.vozik_stohovani_pn) THEN
				voziky[i].stoh[4].disable := FALSE;
			END_IF;
		END_IF;
	ELSE
```

```
            // překopírování hodnoty z osy Y
            voziky[i].stoh[4].rozmer_y := voziky[i].stoh[3].rozmer_y;
        END_IF;
END_FOR
```

1. Pozice: Nejdříve dojde k zakázání pozice " **voziky[i].stoh[1].disable := TRUE;**". Poté následuje kontrola rozměru dílu, který drží manipulátor s rozměrem, jaký už je na daném prvku uložen " **IF (voziky[i].stoh[1].rozmer_x = dataDilu_manipulator.rozmer_dilu_x) THEN IF (voziky[i].stoh[1].rozmer_y = dataDilu_manipulator.rozmer_dilu_y) THEN**" Pokud rozměr souhlasí a nebo je pozice prázdná (má hodnoty 0) je pozice opět povolena pro zaskladnění právě drženého dílu.
2. Pozice: Tady je logika podobná jako u pozice 1. Navíc se zde ale zohledňuje ještě velikost právě zpracovávaného dílu. U pozice 1 to není potřeba, protože nemůže přijít takový díl, který by nebylo možné zaskladnit. Přibyla tedy kontrola v ose X zda se díl do pozice 2 vejde vedle dílu na pozici 1 " **IF (dataRet.vozik_rozmer_x - voziky[i].stoh[1].rozmer_x - dataRet.vozik_mezeraStohu >= dataDilu_manipulator.rozmer_dilu_x + dataRet.vozik_mezeraStohu) THEN**". Tato podmínka zkontroluje, zda se držený díl vejde vedle prvního, pokud ne, nechá pozici 2 zamčenou.
3. Pozice: Tady je logika stejná jako u pozice 1 + opět kontrola zda se vejdou díly tentokrát pod sebe v ose Y.
4. Pozice: Kombinace podmínek pro pozici 2 a 3.

Celý systém kontroly místa je tedy navržen tak, že začíná na pozici 1 a postupně zaplňuje plochu vozíku. Budeme-li tedy mít pouze dva velké kotouče. Z pohledu systému se bude jednat o pozici 1 a 3. Díky tomu lze tedy na vozík umístit 2x malý kotouč na pozice 1 a 2, 1x velký kotouč na pozici 3 a 2x opět malý kotouč na pozice 5 a 6 (do jedné vrstvy).

8.5.2 Výběr pozice pro založení

První část kódu zkontrolovala jednotlivé pozice po rozměrové stránce a dala pouze informaci, který stoh lze nebo nelze použít a běží neustále na pozadí bez ohledu na to, v jakém kroku je výstupní manipulátor. Níže uvedený kód je spouštěn už jen při výpočtu souřadnice pro uložení manipulovaného dílu.

Nejprve zkontrolujeme, které pozice jsou povolené od první části kódu.

```
// výběr aktivní pozice ve vozíku
FOR i := 1 TO SizeOfAry(voziky[procesniData.vozik_aktivni].stoh) DO
        // stoh nesmí být zakázaný
        IF NOT voziky[procesniData.vozik_aktivni].stoh[i].disable THEN
```

```
                // musí být stejný typ dílu
                IF voziky[procesniData.vozik_aktivni].stoh[i].typDilu = dataDilu_manipulator.typDilu OR
                voziky[procesniData.vozik_aktivni].stoh[i].typDilu = 0 THEN
                        // musí být menší počet kusů nasobě
                        IF voziky[procesniData.vozik_aktivni].stoh[i].pocetDilu < dataRet.vozik_stohMax_dilu THEN
                                // musí být menší výška stohu
                                IF voziky[procesniData.vozik_aktivni].stoh[i].vyska_z +
                                dataDilu_manipulator.rozmer_dilu_z < dataRet.vozik_stohMax_vyska_Z THEN
                                        procesniData.stoh_aktivni := i;
                                        EXIT;
                                END_IF;
                        END_IF;
                END_IF;
        END_IF;
END_FOR;
```

Po výběru aktivní pozice se zkontroluje, zda už obsahuje nějaké díly. Pokud ano, použijí se již spočítané souřadnice. Pokud ne, spočítají se souřadnice pro danou pozici a uloží se pro další použití.

```
// pokud jsou souřadnice nulové - založení stohu dle rozměru dílu
(*
X
-------
| 1 | 2 |
-------
| 3 | 4 |
-------  Y
| 5 | 6 |
-------
| 7 | 8 |
-------
*)
IF zalozeni_okDil THEN
        IF voziky[procesniData.vozik_aktivni].stoh[procesniData.stoh_aktivni].pozice_x = 0 AND
        voziky[procesniData.vozik_aktivni].stoh[procesniData.stoh_aktivni].pozice_y = 0 THEN
                CASE procesniData.stoh_aktivni OF
                        1,3,5,7:
                                // kontrola rozměru X jestli není měnší než přísavka
                                IF dataDilu_manipulator.rozmer_dilu_x < dataRet.man_prisavkaRozmer_x THEN
                                        pos_x_zakladani := dataRet.vozik_ref_x[procesniData.vozik_aktivni] +
                                        dataRet.vozik_mezeraStohu;
                                ELSE
                                        pos_x_zakladani := dataRet.vozik_ref_x[procesniData.vozik_aktivni] +
                                        dataRet.vozik_mezeraStohu + ((dataDilu_manipulator.rozmer_dilu_x -
                                        dataRet.man_prisavkaRozmer_x) / 2);
                                END_IF;
                                CASE procesniData.stoh_aktivni OF
                                        1:
                                                pos_y_offset := dataRet.vozik_mezeraStohu;
                                        3:
                                                pos_y_offset := (dataRet.vozik_mezeraStohu * 2) +
                                                voziky[procesniData.vozik_aktivni].stoh[1].rozmer_y;
                                        5:
                                                pos_y_offset := (dataRet.vozik_mezeraStohu * 3) +
                                                voziky[procesniData.vozik_aktivni].stoh[1].rozmer_y +
                                                voziky[procesniData.vozik_aktivni].stoh[3].rozmer_y;
                                        7:
                                                pos_y_offset := (dataRet.vozik_mezeraStohu * 4) +
                                                voziky[procesniData.vozik_aktivni].stoh[1].rozmer_y +
```

```
                                voziky[procesniData.vozik_aktivni].stoh[3].rozmer_y +
                                voziky[procesniData.vozik_aktivni].stoh[5].rozmer_y;
                        END_CASE;
                        pos_y_zakladani := dataRet.vozik_ref_y[procesniData.vozik_aktivni] + pos_y_offset +
                        ((dataDilu_manipulator.rozmer_dilu_y - dataRet.man_prisavkaRozmer_y) / 2);
                2,4,6,8:
                        IF dataDilu_manipulator.rozmer_dilu_x < dataRet.man_prisavkaRozmer_x THEN
                                offsetX := ((dataRet.man_prisavkaRozmer_x -
                                dataDilu_manipulator.rozmer_dilu_x) / 2);
                        ELSE
                                offsetX := ((dataDilu_manipulator.rozmer_dilu_x -
                                dataRet.man_prisavkaRozmer_x) / 2);
                        END_IF;
                        CASE procesniData.stoh_aktivni OF
                                2:
                                        pos_x_offset := (dataRet.vozik_mezeraStohu * 2) +
                                        voziky[procesniData.vozik_aktivni].stoh[1].rozmer_x;
                                        pos_y_offset := dataRet.vozik_mezeraStohu;
                                4:
                                        pos_x_offset := (dataRet.vozik_mezeraStohu * 2) +
                                        voziky[procesniData.vozik_aktivni].stoh[3].rozmer_x;
                                        pos_y_offset := (dataRet.vozik_mezeraStohu * 2) +
                                        voziky[procesniData.vozik_aktivni].stoh[2].rozmer_y;
                                6:
                                        pos_x_offset := (dataRet.vozik_mezeraStohu * 2) +
                                        voziky[procesniData.vozik_aktivni].stoh[5].rozmer_x;
                                        pos_y_offset := (dataRet.vozik_mezeraStohu * 3) +
                                        voziky[procesniData.vozik_aktivni].stoh[2].rozmer_y +
                                        voziky[procesniData.vozik_aktivni].stoh[4].rozmer_y;
                                8:
                                        pos_x_offset := (dataRet.vozik_mezeraStohu * 2) +
                                        voziky[procesniData.vozik_aktivni].stoh[7].rozmer_x;
                                        pos_y_offset := (dataRet.vozik_mezeraStohu * 4) +
                                        voziky[procesniData.vozik_aktivni].stoh[2].rozmer_y +
                                        voziky[procesniData.vozik_aktivni].stoh[4].rozmer_y +
                                        voziky[procesniData.vozik_aktivni].stoh[6].rozmer_y;
                        END_CASE;
                        pos_x_zakladani := dataRet.vozik_ref_x[procesniData.vozik_aktivni] + pos_x_offset +
                        offsetX;
                        pos_y_zakladani := dataRet.vozik_ref_y[procesniData.vozik_aktivni] + pos_y_offset +
                        ((dataDilu_manipulator.rozmer_dilu_y - dataRet.man_prisavkaRozmer_y) / 2);
        END_CASE;
    ELSE
        pos_x_zakladani := voziky[procesniData.vozik_aktivni].stoh[procesniData.stoh_aktivni].pozice_x;
        pos_y_zakladani := voziky[procesniData.vozik_aktivni].stoh[procesniData.stoh_aktivni].pozice_y;
    END_IF;
ELSE
    pos_x_zakladani := dataRet.nok_ref_x + (dataRet.nok_rozmer_x / 2) - (dataRet.man_prisavkaRozmer_x / 2);
    pos_y_zakladani := dataRet.nok_ref_y + (dataRet.nok_rozmer_y / 2) - (dataRet.man_prisavkaRozmer_y / 2);
    IF pos_y_zakladani > 710 THEN
        pos_y_zakladani := 710;
    END_IF;
END_IF;
```

9 KOMUNIKACE S NADŘAZENÝM SYSTÉMEM

Pro fungování linky je třeba komunikovat s nadřazeným systémem který ověřuje materiál a tiskne nové štítky. Ta probíhá pomocí HTTP POST skrze TCP/IP spojení a přenosu dat ve formátu JSON. Pro každý zpracovaný díl se komunikují celkem dvě zprávy.

1. Registrace nového dílu
2. Verifikace dílu

9.1 Registrace nového dílu

Po zaměření dílu a přečtení dodavatelského štítku jsou data z něho odeslána na registraci. Nadřazený systém ověřuje, zda se daný materiál očekává (je na něho vystavena příjemka do skladu). Pokud tato kontrola dopadne s chybou je spolu s jejím popisem odesláno do PLC chybové hlášení. V případě, že je vše v pořádku, vytiskne se nový štítek pro daný díl a do PLC je odeslána zpráva s obsahem tohoto štítku pro krok verifikace.

Níže je dohodnutý formát zprávy pro registraci nového dílu. Obsahuje HTTP POST hlavičku s přihlášením "Http POST /label HTTP/1.1 Authorization: (Base64 zakódované jméno a heslo, oddělené ":")" a potom data ve formátu JSON. První tři položky jsou identifikátory načtené z dílu (pm,qty,lot). Poté následuje číslo operátora a číslo linky (pernr, wokp).

```
http
POST /label HTTP/1.1
Authorization: (Base64 zakódované jméno a heslo, oddělené ":")
{
  "pn": "C2DBYY001774",
  "qty": 2,
  "lot": "2",
  "pernr": "46412",
  "workp": "010"
}
```

V závislosti na stavu kontroly na straně SAP mohou přijít dva typy odpovědí.

OK "status: 200" - materiál lze přijmout, SAP vytiskne štítek na tiskárně, která je umístěná ve stroji a do PLC odešle jeho obsah pro kontrolu. Opět ve formátu JSON.

```
status: 200
{
  "fifo": "fifostitek",
  "mfprn": "0052758569",
  "pn": "C2DBYY001774",
  "qty": 2,
  "lot": "2",
  "msl": "nejakymsl",
  "goodsReceipt": "46412",
  "goodsDate": "20230612",
```

```
  "movement_indicator": "010",
  "timestamp": "2023-06-12T06:07:08",
  "fifo_bc": "dsad1sa1d16",
  "matnr_bc": "dsad1sa1d16",
  "qty_bc": "dsad1sa1d16",
  "lot_bc": "dsad1sa1d16"
}
```

NOK "status: 400" - materiál nelze přijmout, do PLC je odesláno chybové hlášení s popisem. Na chybnou odpověď je upozorněn operátor a je zobrazena na HMI panelu linky.

```
status: 400
{
  "errors": [
    {
      "id": "ZPLC",
      "type": "E",
      "message": "Material 666 doesn't exists"
    }
  ]
}
```

Použitý PLC systém má omezení délky datového typu string pouze na 255 znaků. Toto omezení je dostačující na dílčí části zprávy, ale nikoli na celou zprávu. Proto je celá zpráva skládána jako pole BYTE, aby se dané omezení obešlo. V tomto konkrétním případě se jedná o pole velikosti 1024 BYTE pro odeslání dat a druhé stejně veliké pro příjem. Aby bylo možné dílčí části zprávy zapisovat jako STRING a kód zůstal uživatelsky čitelný byla navržena jednoduchá funkce, která převede STRIG data na pole BYTE a přidá je do odesílacího buffru "FC_strToArryAdd".

```
Clear(tmpArray);
tmpCount := StringToAry(i_stringData, tmpArray[0]);
FOR x := 0 TO tmpCount DO
        io_sendData[io_sendDataPointr + x] := tmpArray[x];
END_FOR;
io_sendDataPointr := io_sendDataPointr + tmpCount;
```

Nejprve je smazáno pomocné pole z předchozího převodu "**Clear(tmpArray);**". Poté následuje interní funkce pro převod datového typu String na pole bytu " **tmpCount := StringToAry(i_stringData, tmpArray[0]);**", kde návratová hodnota této funkce je počet převedených znaků. Posledním krokem je nakopírování převedeného pole na konec buffru a inkrement aktuálního pointeru buffru.

Další podpůrnou funkcí, která musela být vyvinuta je kódování do BASE64, protože touto funkcí použitý PLC systém nedisponuje. Pro kódování do BASE64 je potřeba pracovat s jednotlivými bity znaků vstupem do funkce je pole BYTE a výstupem zakódovaná data opět jako pole BYTE. Při kódování jsou odebírány vstupní znaky jeden po druhém a na základě předpisu BASE64 jsou kódovány do výstupního pole, které obsahuje více znaků než vstupní.

```
// zpracování probíhá po 1 znaku
inBits_index := 0;
outChar_index := 0;
padding := 0;
clear(o_data);
Clear(inBits);
FOR x := 0 TO SizeOfAry(i_data) DO
	// kontrola zda prvek obsahuje data - pokud ne ukončení
	IF i_data[x] = 0 THEN
		IF padding <> 0 THEN
			FOR y := 1 TO padding DO
				StringToAry('=', o_data[outChar_index]);
				inc(outChar_index); // increment indexu zápisu výsledku
			END_FOR;
		END_IF;
		EXIT;
	END_IF;
	// vstupního znaku převod na bity
	tmpByteBit.uByte := i_data[x];
	FOR y := 0 TO 7 DO
		inBits[y] := tmpByteBit.uBit[y];
	END_FOR;
	inBits_index := inBits_index + 8;

	// kodování znaku
	WHILE inBits_index >= 6 DO
		tmpByteBit.uByte := 0;
		FOR y := 0 TO 5 DO
			tmpByteBit.uBit[5 - y] := inBits[inBits_index - 1 - y];
		END_FOR;
		StringToAry(MID(stdBase, UINT#1, BYTE_TO_UINT(tmpByteBit.uByte) + 1), o_data[outChar_index]);
		inc(outChar_index); // increment indexu zápisu výsledku
		// posunutí zbylích bitů a zápis nového indexu
		IF inBits_index - 6 > 0 THEN
			FOR y := 0 TO inBits_index - 6 - 1 DO
				inBits[8 + y] := inBits[y];
			END_FOR;
			inBits_index := inBits_index - 6;
			// kontrola posledního znaku a zbylích bitů
			// jsou 2 doplnění ==
			// jsou 4 doplnění =
			IF x = SizeOfAry(i_data) OR i_data[x + 1] = 0 THEN
				CASE inBits_index OF
					2:
						padding := 2;
					4:
						padding := 1;
				END_CASE;
				IF padding <> 0 THEN
					FOR y := 0 TO 5 DO
						IF y < 6 - inBits_index THEN
							inBits[y] := FALSE;
						ELSE
							inBits[y] := inBits[y + inBits_index + 2];
						END_IF;
					END_FOR;
				inBits_index := 6;
				END_IF;
			END_IF;
		ELSE
			inBits_index := 0;
		END_IF;
	END_WHILE;
```

```
END_FOR;
```

Po vytvoření těchto dvou pomocných funkcí mohla být sestavena celá zpráva. Do pomocné proměnné typu STRING se vždy zapíše část zprávy, která je poté přidána do pole odesílacího buffru - příklad dat pro přidání " `tmpString := 'POST /label HTTP/1.1$L';`" poté už následuje volání připravené funkce "`FC_strToArryAdd(tmpString, sendData, sendDataPointr);`". Ta vezme data, která se mají připojit, přidá je do buffru " `sendData`" a upraví pointer na konec buffru " `sendDataPointr`". Poté se už jen celý proces opakuje, dokud není poskládána celá zpráva k odeslání. Níže uvedený kód sestavuje celou zprávu pro registraci materiálu. Obsahuje také volání funkce pro kódování uživatelského jména a hesla pro přístup do SAP " `FC_Base64Code(userDataArray,tmpArray,codeLen);`". Stejným způsobem se skládá i zpráva pro verifikaci dat pro SAP.

```
// nový štítek - registrace
IF io_label_sendReq THEN
	io_label_sendReq := FALSE;
	// promazání proměnných
	Clear(sendData);
	Clear(reciveData);
	Clear(userDataArray);
	sendDataPointr := 0;
	// poskládání požadavku
	// hlavička
	tmpString := 'POST /label HTTP/1.1$L';
	FC_strToArryAdd(tmpString, sendData, sendDataPointr);
	// autorizace
	// jmeno:heslo
	tmpString := CONCAT(i_userName, ':');
	tmpString := CONCAT(tmpString, i_userPass);
	StringToAry(tmpString, userDataArray[0]);
	// kodování jména a hesla
	FC_Base64Code(userDataArray,tmpArray,codeLen);
	// spojení s daty
	tmpString := CONCAT('Authorization: Basic ', AryToString(tmpArray[0], codeLen));
	FC_strToArryAdd( tmpString, sendData, sendDataPointr);
	tmpString := '$R$L';
	FC_strToArryAdd(tmpString, sendData, sendDataPointr);
	// json data
	tmpString := '{';
	FC_strToArryAdd(tmpString, sendData, sendDataPointr);
	// pn
	tmpString := CONCAT('"pn": "', i_label_pn);
	tmpString := CONCAT(tmpString,'",');
	FC_strToArryAdd(tmpString, sendData, sendDataPointr);
	// uprava kodu QTY
	// uprava pro čtení spojených kodů PN QTY
	IF i_label_pn = i_label_qty THEN
		tmpQty := '';
	ELSE
		// uprava kodu QTY - odmazání Q na začátku nebo pcs na konci
		IF LEFT(i_label_qty, UINT#1) = 'Q' THEN
			tmpQty := DELETE(i_label_qty,UINT#1, UINT#1);
		ELSE
			// uprava kodu QTY - odmázání PCS na konci kodu
```

```
                    IF RIGHT(i_label_qty, UINT#3) = 'PCS' THEN
                        tmpQty := DELETE(i_label_qty,UINT#3,LEN(i_label_qty) - 2);
                    ELSE
                        tmpQty := i_label_qty;
                    END_IF;
                END_IF;
        END_IF;
        // uprava kodu QTY - ošetření oddělovače tisíců
        odd := FIND(tmpQty,'.');
        IF odd = 0 THEN
            odd := FIND(tmpQty,',');
        END_IF;
        IF odd <> 0 THEN
            tmpQty := DELETE(tmpQty,UINT#1,odd);
        END_IF;
        tmpString := CONCAT('"qty": ', tmpQty);
        tmpString := CONCAT(tmpString,',');
        FC_strToArryAdd(tmpString, sendData, sendDataPointr);
        // lot
        tmpString := CONCAT('"lot": "', i_label_lot);
        tmpString := CONCAT(tmpString,'",');
        FC_strToArryAdd(tmpString, sendData, sendDataPointr);
        // pernr
        tmpString := CONCAT('"pernr": "', i_label_pernr);
        tmpString := CONCAT(tmpString,'",');
        FC_strToArryAdd(tmpString, sendData, sendDataPointr);
        // workp
        tmpString := CONCAT('"workp": "', i_label_workp);
        tmpString := CONCAT(tmpString,'"');
        FC_strToArryAdd(tmpString, sendData, sendDataPointr);
        // zakončení
        tmpString := '}';
        FC_strToArryAdd(tmpString, sendData, sendDataPointr);
        com_sendData_req := TRUE;
END_IF;
```

9.2 Odpověď do nadřazeného systému

Pro příjem dat ze SAP musela být vyvinuta funkce na parsování dat z JSON formátu, který mě SAP vrací. Díky tomu, že bylo předem dohodnuté pořadí dat a formát JSON je standardizován, byl zvolen jednoduchý způsob parsování na základě oddělujících znaků a předem známého pořadí proměnných.

```
// snulování pomocných pozic
char_1p := 0;
char_2p := 0;

// hledání pozice znaků
FOR x := io_startPointr TO SizeOfAry(i_data) DO
        // hledání prvního znaku
        IF i_data[x] = i_char_1 AND char_1p = 0 THEN
            char_1p := x;
        END_IF;
        // hledání druhého znaku
        IF (i_data[x] = i_char_2 OR i_data[x] = 0 OR x = SizeOfAry(i_data)) AND char_2p = 0 THEN
            char_2p := x;
        END_IF;
            // "vyříznutí" požadové části řetězce
            IF char_1p <> 0 AND char_2p <> 0 THEN
```

```
                    IF char_2p - char_1p - 1 = 0 THEN
                        o_data := AryToString(i_data[char_1p + 1], char_2p - char_1p);
                    ELSE
                        o_data := AryToString(i_data[char_1p + 1], char_2p - char_1p - 1);
                    END_IF;
                    // ošetření prázdných znaků
                    o_data := TrimL(o_data);
                    o_data := TrimR(o_data);
                // increment pointru
                io_startPointr := char_2p + 1;
                EXIT;
        END_IF;
END_FOR;
```

" **FC_sap_dataSep(i_char_1, i_char_2, i_data, io_startPointr,o_data)**" funkce má tři vstupní parametry, jeden vstupně/výstupní a jeden výstupní. " **i_char_1, i_char_2**" jsou vstupní parametry, které určují, mezi jakými znaky se požadovaná data nachází. " **i_data**" je vstupní pole přijatých dat. " **io_startPointr**" je vstupně výstupní proměnná, která určuje index do přijatých dat, tedy kde předchozí parsování skončilo a kde bude následují pokračovat. "**o_data**" je potom výstupní proměnná, která obsahuje požadovaná data.

Níže je uvedená ukázka kódu pro dekódování. Nejprve se nastaví proměnné, které označují začátek a konec dat " **startChar := 16#3A; // : endCHar := 16#7B; // {** ". V tomto případě hledáme data mezi znaky: a {. Potom je zavolána funkce parsování a výsledná data jsou zpracována. Následuje opět nastavení znaků a opětovné volání funkce, dokud není celá zpráva zpracována.

9.3 Verifikace dílu

Vzhledem k tomu, že při požadavku na registraci nového dílu bylo potřeba vytvořit všechny pomocné funkce, verifikace spočívá už jenom pouze v poskládání verifikační zprávy a jejím odeslání podobně jako u požadavku na registraci. Uvedu tedy pouze problémy a odlišnosti, které bylo potřeba ošetřit.

Samotná verifikace probíhá jednoduše, a to ověřením kódů, které byly přijaty od SAP a kódů, které byly přečteny z nalepeného štítku. " **o_verify_verified := o_sap_pn = i_verify_pn AND o_sap_fifo = i_verify_fifo;**" Pokud jsou oba kódy shodné je výsledek verifikace true, pokud nejsou shodné je verifikace false. Na základě toho je potom upraven text pro JSONu pro SAP.

```
// verifikace
IF o_verify_verified THEN
        tmpString := '"verified": true,';
ELSE
        tmpString := '"verified": false,';
END_IF;
FC_strToArryAdd(tmpString, sendData, sendDataPointr);
```

Stejně jako je tomu u předchozího skládání nastavuje se pomocný string, který je potom připojen ke zprávě.

Novým prvkem je časová značka, aby bylo identifikovatelné, kdy došlo k verifikaci dílu. Je tedy do zprávy přidána v požadovaném formátu ISO_8601 - tedy 2023-01-01T12:00:00. Nejprve se vyčte aktuální čas PLC a provede se převod na strukturu, kde je každý prvek data v samostatné proměnné " **DtToDateStruct(In:=GetTime(), DateStruct=>actTime);**". U všech prvků, mimo roku, je potřeba provést kontrolu dat, zda jsou dvoumístná, pokud tomu tak není je potřeba doplnit 0. Nutnost této úpravy vzniká z toho důvodu že číslici 0–9 funkce pro převod na text převede jako 0–9 namísto požadovaného formátu 00–09.

```
// timestamp"
DtToDateStruct(In:=GetTime(), DateStruct=>actTime);
tmpString := '"timestamp": "';
tmpString := CONCAT(tmpString, UINT_TO_STRING(actTime.Year));
tmpString := CONCAT(tmpString, '-');
IF LEN(UsINT_TO_STRING(actTime.Month)) = 1 THEN
        tmpString := CONCAT(tmpString, '0');
END_IF;
tmpString := CONCAT(tmpString, UsINT_TO_STRING(actTime.Month));
```

ZÁVĚR

V rámci této bakalářské práce byl navržen, vytvořen a odzkoušen prototyp linky na automatickou registraci vstupního materiálu.

V době psaní této práce byl na lince již ukončen zkušební provoz a bylo tedy možné všechny požadavky na zlepšení procesu, které měla tato linka přinést otestovat. Stejně tak všechny popsané programové části.

Souhrn požadavků a výsledků během zkušebního provozu.

1. Eliminace zásahu obsluhy do procesu čtení a lepení štítku. – Linka splnila očekávání. Kritický bod je plně oddělen od operátorů a ti do něho nijak nezasahují.
2. Přidání ověření lepeného štítku tak, aby nedošlo k záměně. – Implementace komunikace s nadřazeným systémem je plně funkční včetně verifikačního procesu. Oproti původním předpokladům se měnil formát dat, které se komunikují a jsou potřebné pro verifikaci.

Vzhledem k tomu, že se jedná o linku stavěnou na míru, nikoli o sériovou výrobu, se dali předpokládat jisté komplikace jak při návrhu, tak při spuštění do reálného provozu. Oproti předpokladům se některé díly v rámci kamerového systému musely rozdělit na více podtypů z důvodů rychlosti vyhodnocení, a především údržbě a správě scén tak, aby byly scény co nejpřehlednější pro případné úpravy, které si provádí zákazník sám bez naší asistence. Na straně PLC toto navýšení počtu typů či podtypů znamenalo pouze rozšíření databáze z původně plánovaných 120 typů na 500.

Další problém vznikl u nestandartních balení, která ale nebyla zahrnuta v původním zadání pro návrh linky. Jedná se o díly, které mají dodavatelské štítky z výroby pod ochranou folií a jsou tedy pro kamerový systém z důvodů odlesků nečitelné. Proběhly testy těchto dílů a směrem k zákazníkovy bylo navrhnuto řešení o doplnění bočních světel, která odlesky eliminují. V době odevzdání této práce se tento bod nedořešil.

Závěrem lze říci, že všechny body zadání této práce byly splněny. Stejně tak navržená linka splnila požadavky zákazníka a prošla zkušebním provozem.

SEZNAM POUŽITÉ LITERATURY

[1] ST. MICHAEL, Stephen. *What Is a PLC? An Introduction to Programmable Logic Controllers*. Online. In: Control Automation. 2022. Dostupné z: https://control.com/technical-articles/what-is-a-plc-an-introduction-to-programmable-logic-controllers/. [cit. 2024-05-03].

[2] PETERSON, David. The Origin Story of the PLC. Online. Control Automation. 2 March 2022n. l., s. 1. Dostupné z: https://control.com/technical-articles/the-origin-story-of-the-plc/. [cit. 2024-05-03].

[3] SIMATIC S7-1200: Can be flexibly tailored to your requirements. Online. In: SIEMENS. Siemens.com. C1996 - 2024. Dostupné z: https://www.siemens.com/in/en/products/automation/systems/industrial/plc/s7-1200.html. [cit. 2024-05-11].

[4] Omron NX1P. Online. In: OMRON. Store.omron.com.au. C2024. Dostupné z: https://store.omron.com.au/series/nx1p. [cit. 2024-05-03].

[5] SIMATIC S7-1500. Online. In: SIEMENS. Siemens.com. C1996 - 2024. Dostupné z: https://www.siemens.com/global/en/products/automation/systems/industrial/plc/simatic-s7-1500.html. [cit. 2024-05-11].

[6] Omron NJ5. Online. In: OMRON. Store.omron.com.au. C2024. Dostupné z: https://store.omron.com.au/series/nj5. [cit. 2024-05-03].

[7] VÍTEK, Martin. Programovatelný PLC dle IEC 61 131. Online, diplomové práce, vedoucí Prof. Ing. Miroslav Olehla, CSc. Studentská 1402/2, 461 17 Liberec: Technická univerzita v Liberci, 2006. Dostupné z: https://dspace.tul.cz/handle/15240/148207. [cit. 2024-05-03].

[8] Ladder Logic Basics. Online. Ladder logic world. C2023. Dostupné z: https://ladderlogicworld.com/ladder-logic-basics/. [cit. 2024-05-03].

[9] KOBRLE, Pavel a PAVELKA, Jiří. *Elektrické pohony a jejich řízení*. 3. přepracované vydání. V Praze: České vysoké učení technické, 2016. ISBN 978-80-01-06007-0.

[10] KLEMENT, Milan. Úvod do problematiky počítačových sítí. Online, PDF. 2. vydání. V Olomouci: Univerzita Palackého, 2019. ISBN 978-80-244-5580-8. Dostupné z:

www.researchgate.net/publication/338363014_TECHNOLOGIE_POCITACOVYCH_SITI_-_uvod_do_problematiky_pocitacovych_siti. [cit. 2024-05-11].

[11] OMRON. Machine Automation Controller NJ-series: General-purpose Ethernet Connection Guide (TCP/IP) OMRON Corporation. PDF. C2013. Dostupné také z: https://edata.omron.com.au/eData/Networks/ConnectionGuides/P558-E1-01.pdf.

[12] ECMA INTERNATIONAL. ECMA-404, The JSON Data Interchange Syntax. 2nd Edition. 2017. Dostupné také z: https://ecma-international.org/wp-content/uploads/ECMA-404_2nd_edition_december_2017.pdf.

[13] SMITH, Ben. *Beginning JSON*. PDF. United States: Apress, 2015. ISBN 978-1-4842-0202-9. [cit. 2024-05-05].

[14] NOVOTNÝ, František; HOTAŘ, Vlastimil; HORÁK, Marcel; STARÁ, Marie a STARÝ, Michal. *ÚVOD DO AUTOMATIZACE A ROBOTIZACE VE STROJÍRENSTVÍ*. V Liberci: Technická univerzita v Liberci, 2020. ISBN 978-80-7494-545-8.

[15] ŠMEJKAL, Ladislav a MARTINÁSKOVÁ, Marie. PLC a automatizace. Praha: BEN - technická literatura, 1999. ISBN 80-860-5658-9.

[16] OMRON. I4L Robots. Online, PDF. C2022. Dostupné také z: https://assets.omron.eu/downloads/manual/en/v1/i658_i4l_series_users_manual_en.pdf.

[17] SHAWN, Dietrich. How-To: Servo Motor Control Concepts and Programming. Online. Control Automation. 27 March 2024n. l., s. 1. Dostupné z: https://control.com/technical-articles/how-to-servo-motor-control-concepts-and-programming/. [cit. 2024-05-11].

[18] POURASIABI, Hamed. Just In Time (JIT) Production and Supply Chain Management. PDF. 2012. Dostupné také z: https://www.researchgate.net/publication/341965168_Just_In_Time_JIT_Production_and_Supply_Chain_Management. International Iron & Steel Symposium.

[19] SANDERS, Adam; ELANGESWARAN, Chola a WULFSBERG, Jens. Industry 4.0 Implies Lean Manufacturing: Research Activities in Industry 4.0 Function as Enablers for Lean Manufacturing. Online. Journal of Industrial Engineering and Management. Roč. 2016, s. 816. ISSN 2013-0953. Dostupné z: http://www.jiem.org/index.php/jiem/article/view/1940/780. [cit. 2024-05-11].

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

IT	Informační technologie.
PLC	Programmable Logic Controler.
HMI	Human Machine Interface.
DMC	DataMetrix Code
QR	Quick Response code
RFID	Radio Frekquency Identification
I/O	Input/Output
CPU	Central Processing Unit
LD	Ladder Diagram
FBD	Function Block Diagram
IL	Instruction List
ST	Strukturierte Text
SFC	Sequential function chart
CFC	Continuous Function Chart
PC	Personal Computer
CW	ClockWise
CCW	Counter ClockWise
IP	Internet Protocol
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
JSON	JavaScript Object Notation
CNC	Computer Numerically Controlled
SCARA	Selective Compliant Articulated Robot Arm
PN	Part Number

OK	OK stav
NOK	Not OK stav
HOME	Výchozí pozice
ID	IDentifikace
SW	Software
HW	Hardware
RS232	sériová linka
X, Y, Z	osy kartézského souřadného systému
HTTP	HyperText Transfer Protocol.
POST	dotazovacích metod HTTP protokolu
SAP	System Analysis Program Development
MC	Motion Control
SMD	Surface Mount Device

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH