VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ
FACULTY OF MECHANICAL ENGINEERING

ÚSTAV AUTOMATIZACE A INFORMATIKY
INSTITUTE OF AUTOMATION AND COMPUTER SCIENCE

OPTIMALIZACE PÁROVÁNÍ SOUČÁSTÍ LOŽISEK
OPTIMIZING PAIRS OF BEARING COMPONENTS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. Lukáš Blahút**
AUTHOR

VEDOUCÍ PRÁCE — **prof. RNDr. Ing. Miloš Šeda, Ph.D.**
SUPERVISOR

BRNO 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA STROJNÍHO INŽENÝRSTVÍ

Zadání diplomové práce

Ústav:	Ústav automatizace a informatiky
Student:	**Bc. Lukáš Blahút**
Studijní program:	Strojní inženýrství
Studijní obor:	Aplikovaná informatika a řízení
Vedoucí práce:	**prof. RNDr. Ing. Miloš Šeda, Ph.D.**
Akademický rok:	2016/17

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma diplomové práce:

Optimalizace párování součástí ložisek

Stručná charakteristika problematiky úkolu:

Párování ložisek je komplexní proces předpřípravy pro ložiskové automaty. Skládá se z volby vhodného páru vnějšího a vnitřního kroužku ložiska na základě naměřených parametrů oběžných drah a přiřazení vhodné velikosti valivého tělíska pro dosažení požadované přesnosti. Úkolem je návrh a implementace algoritmu pro automatizované párování dvouřadových ložisek. Algoritmus musí efektivně využívat vstupní maticové zásobníky a korigovat volbu vhodných párů na základě kontroly parametrů v průběhu montáže.

Cíle diplomové práce:

1. Proveďte rozbor programovacího prostředí CODESYS (s ohledem na použitelnost v aktuální problematice/aplikaci).
2. Navrhněte algoritmus párování součástí ložiska pro dosažení požadovaných parametrů.
3. Specifikujte potřebné vstupy a výstupy.
4. Algoritmus implementujte v prostředí CODESYS.

Seznam doporučené literatury:

Manuál programu CODESYS.

ZELINKA, I., SNÁŠEL, V., ABRAHAM, A. (eds.): Handbook of Optimization. From Classical to Modern Approach. Berlin, Springer-Verlag, 2013. ISBN 978-3-642-30-503-0.

Termín odevzdání diplomové práce je stanoven časovým plánem akademického roku 2016/17

V Brně, dne

L. S.

doc. Ing. Radomil Matoušek, Ph.D. ředitel ústavu	doc. Ing. Jaroslav Katolický, Ph.D. děkan fakulty

ABSTRAKT

Táto diplomová práca sa zaoberá optimalizáciou procesu párovania súčastí ložiska. Práca obsahuje popis programovacieho prostredia Codesys, jeho štruktúry, použiteľné jazyky a niekoľko príkladov využitia. Hlavná časť práce popisuje sériu algoritmov, ktoré zabezpečujú potrebné činnosti pre jeden z dvoch spracovaných prístupov párovania. Tieto algoritmy sú následné implementované ako podprogramy v prostredí Codesys. Potrebné vstupy i výstupy sú súčasťou navrhnutých algoritmov i spracovaných podprogramov.

ABSTRACT

This diploma thesis deals with the optimization of the process of matching the bearing components. The work contains a description of Codesys programming environment, its structure, usable languages and some examples of usage. The main part of the thesis describes a series of algorithms that provide the necessary actions for one of two processed pairing approaches. These algorithms are subsequently implemented as subprograms in the Codesys environment. The necessary inputs and outputs are part of the proposed algorithms as well as processed subroutines.

KĽÚČOVÉ SLOVÁ

Algoritmus párovania, montáž ložísk, špeciálne dvojradové ložiská, Codesys, Festo

KEYWORDS

Pairing algorithm, bearing assembly, speacial doublerow bearings, Codesys, Festo

BIBLIOGRAFICKÁ CITÁCIA

BLAHÚT, L. *Optimalizace párování součástí ložisek.* Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2017. 81 s. Vedoucí diplomové práce prof. RNDr. Ing. Miloš Šeda, Ph.D..

POĎAKOVANIE

Chcem sa poďakovať vedúcemu práce prof. RNDr. Ing. Milošovi Šedovi, Ph.D., za cenné rady, odbornú pomoc a konzultácie, ktoré mi poskytol pri vypracovaní tejto diplomovej práce.

Zároveň sa chcem poďakovať firme Matador Industries a.s. za možnosť práce na tomto unikátnom projekte.

Osobitné poďakovanie patrí mojej rodine, priateľke a priateľom za podporu a pochopenie.

ČESTNÉ PREHLÁSENIE

Prehlasujem, že predkladaná diplomová práca je mojou pôvodnou autorskou prácou, ktorú som vypracoval pod vedením vedúceho diplomovej práce a s použitím uvedenej literatúry.

V Brne dňa 26. 5. 2017 ……

Bc. Lukáš Blahút

OBSAH

1 ÚVOD

Moderným trendom výrobných podnikov je orientácia na novú politiku spojenú so zavedením metód a nástrojov združených pod pojmom Industry 4.0. Štvrtá priemyselná revolúcia, ako je tento pojem mnohokrát nazývaný, je koncept zavedený len nedávno ako výsledok štúdie špičkových technológii vypracovaný pre nemeckú vládu [1]. Medzi základné prvky možno zaradiť kyberneticko-fyzické systémy, autonómne roboty, Internet vecí, spracovanie veľkých dát a cloud computing.

Zavedenie konceptu Industry 4.0 do praxe nie je jednoduchá úloha. Jedným zo základných krokov možno nazvať operačnú efektívnosť. Tento krok združuje procesy ako zefektívnenie využitia majetku, redukciu prevádzkových nákladov a zvýšenie pracovnej produktivity [2]. Pre dosiahnutie týchto cieľov je potrebná vysoká miera digitalizácie a automatizácie procesov na všetkých úrovniach.

Úvod práce bude venovaný krátkemu opisu problematiky a motivácii k riešeniu daného problému.

Nasledujúca kapitola bude teoreticky popisovať programovacie prostredie Codesys, ktoré sa používa pre programovanie hlavne priemyselných aplikácii. Súčasťou kapitoly bude popis prostredia, použiteľných programovacích jazykov a niekoľko ukážok využitia.

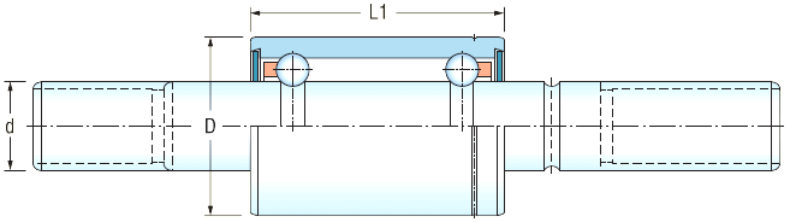
Štvrtá kapitola sa bude zaoberať návrhom samotného algoritmu párovania. Algoritmy budú navrhnuté na základe teoretických znalostí, požiadaviek zákazníka i skúsenosti v oblasti automatizácie.

Obsahom predposlednej kapitoly bude implementácia navrhnutých algoritmov v prostredí Codesys. Súčasťou programov bude aj popis potrebných vstupov a výstupov.

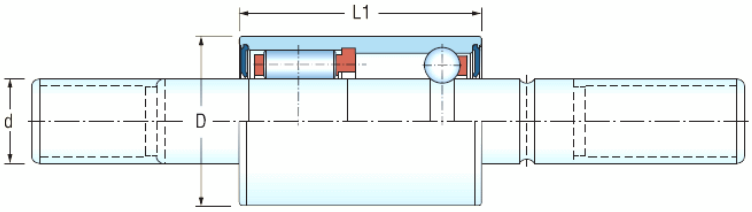
Záver práce bude venovaný dosiahnutým výsledkom a prínosom práce.

2 MOTIVÁCIA PRÁCE

Táto práca je zameraná na algoritmus automatizácie predvýrobných etáp pre montážny automat ložísk. Zmontované špeciálne dvojradové ložiská sú určené pre vodné pumpy nákladných automobilov. Ich charakter sa výrazné nelíši od typických valivých ložísk. Hlavným rozdielom medzi nimi je náhrada vnútorného krúžku za hriadeľ. Ložiská sa vyrábajú v dvoch základných prevedeniach, ktoré sa líšia v použitých valivých telieskach. Typ K má dve rady guľôčok a typ R má jeden rad guľôčok a jeden rad valčekov. Zjednodušený nákres oboch typov sa nachádza na obrázku č.1 a č. 2. Každý z typov sa vyrába v rôznych modifikáciách. Ložiská majú rovnaký menovitý priemer puzdra i hriadeľa. Modifikácie vychádzajú z použitia rôznych dĺžok puzdier a hriadeľov.



Obr. 1 - Ložisko typu "K" [3]

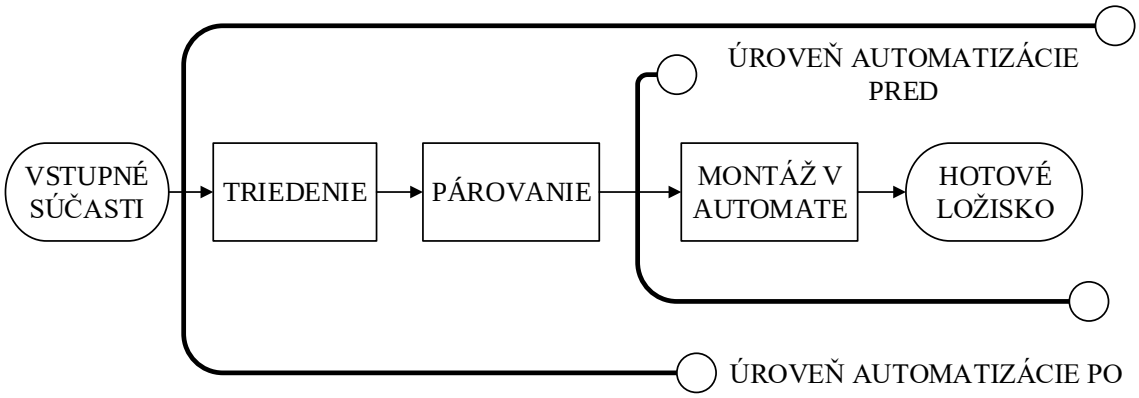


Obr. 2 - Ložisko typu "R" [3]

Aktuálna stratégia predprípravy pozostáva zo série manuálnych operácii, ktorých výsledkom sú roztriedené zostavy súčastí (puzdier a hriadeľov) podľa odchýlky od menovitého rozmeru vyjadrenej v mikrometroch. Zostavy vnútorných a vonkajších súčastí sú následne spárované podľa daného vnútropodnikového predpisu na predpísaný typorozmer guľôčky. Takto predpripravené zostavy sú následné zmontované v montážnom automate. Vzhľadom na krátky cyklový čas montážneho automatu, rádovo v sekundách, sa vyžadujú výrobné dávky vo väčších množstvách. Prezoradenie montážneho automatu na iný typorozmer ložiska zaberie niekoľko minút. Kontrola správnosti spárovaných zostáv prebieha kontrolným meraním vôli ložiska v priebehu montáže. Dielce, ktorých spárovanie je problematické (malý počet, veľké odchýlky od nominálnej hodnoty, ...) končia v sklade. Tieto súčasti sa využijú, keď ich množstvo dosiahne určitú minimálnu hodnotu alebo sa pošlú na úpravu rozmeru obežných dráh alebo sa ich montáž vykoná manuálne.

Automatizovaný proces predprípravy by mal efektívne využívať dostupné dielce a párovanie optimalizovať na základe nameraných údajov. Cieľom automatizácie je odstránenie nutnosti tvorby spárovaných zostáv súčastí. Žiadaným výsledkom je možnosť

priamo vkladať súčasti na začiatok procesu, bez akejkoľvek nutnej predprípravy. Riadenie celého procesu bude zabezpečovať hardvér od firmy Festo naprogramovaný v programe Codesys. Úroveň automatizácie pred a po aplikácii automatizovaného párovania je vidno na obrázku č.3.



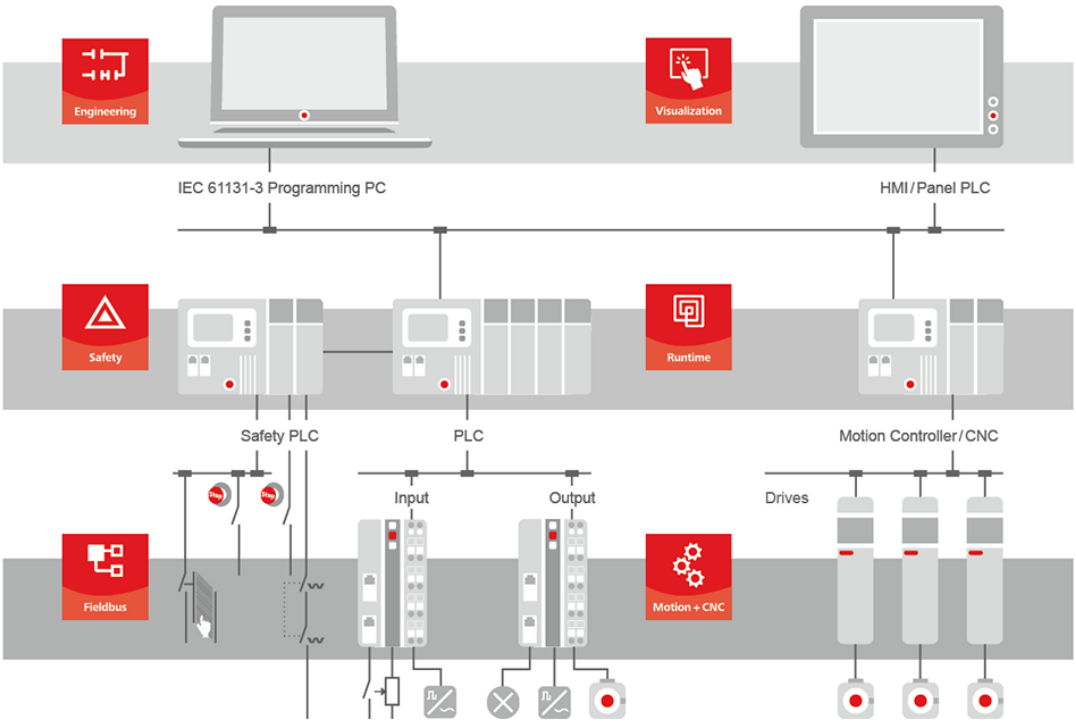
Obr. 3 - Úroveň automatizácie linky

Aby nedošlo k nedorozumeniu, je vhodné oddeliť niektoré špecifické pojmy od pojmov bežných pre technickú prax. Prvým pojmom je párovanie ložísk. V technickej praxi tento pojem predstavuje použitie ložísk s kosouhlým stykom v skupinách. Rôznou orientáciou ložísk namontovaných tesne vedľa seba je možné docieliť špecifické vlastnosti, akými sú napr. zvýšená únosnosť, schopnosť prenášať axiálne zaťaženie alebo znížená citlivosť na nesúososť [4]. Pojem párovania ložísk v tejto práci predstavuje proces predvýrobnej etapy popísaný vyššie. Tento proces sa niekedy označuje aj ako rozmeriavanie.

Druhým pojmom je montáž ložiska. Pre technickú prax predstavuje tento pojem samotný proces osadenia, príp. nalisovania ložiska do zariadenia. Pojem montáž ložiska v tejto práci predstavuje výrobný proces ložiska, teda jeho zloženie z jednotlivých komponentov.

3 CODESYS

CODESYS je programovacia platforma vyvinutá nemeckou firmou 3S-Smart Software Solutions GmbH. Skratka predstavuje výraz „Controlled Development System“, čo sa dá preložiť ako Systém riadeného vývoja. Je primárne určená pre vývoj aplikácii pre programovateľné automaty v súlade s normou IEC-61131. Program v sebe sústreďuje viaceré nástroje, ktorých kombináciou je možné dosiahnuť požadovaného cieľa v oblasti automatizácie na akejkoľvek úrovni. K programovacej platforme, ktorá umožňuje prácu s PLC a jeho programovanie, je možné pripojiť ďalšie prvky. Tie umožňujú prácu s riadiacimi systémami servopohonov, vizualizáciu pomocou veľkého množstva priemyselných panelov, komunikáciu pomocou rôznych protokolov a spravovanie bezpečnostných systémov (do úrovne bezpečnosti SIL 2/3 podľa normy IEC 61508). Organizáciu tejto platformy je možno vidieť na obrázku č.4.

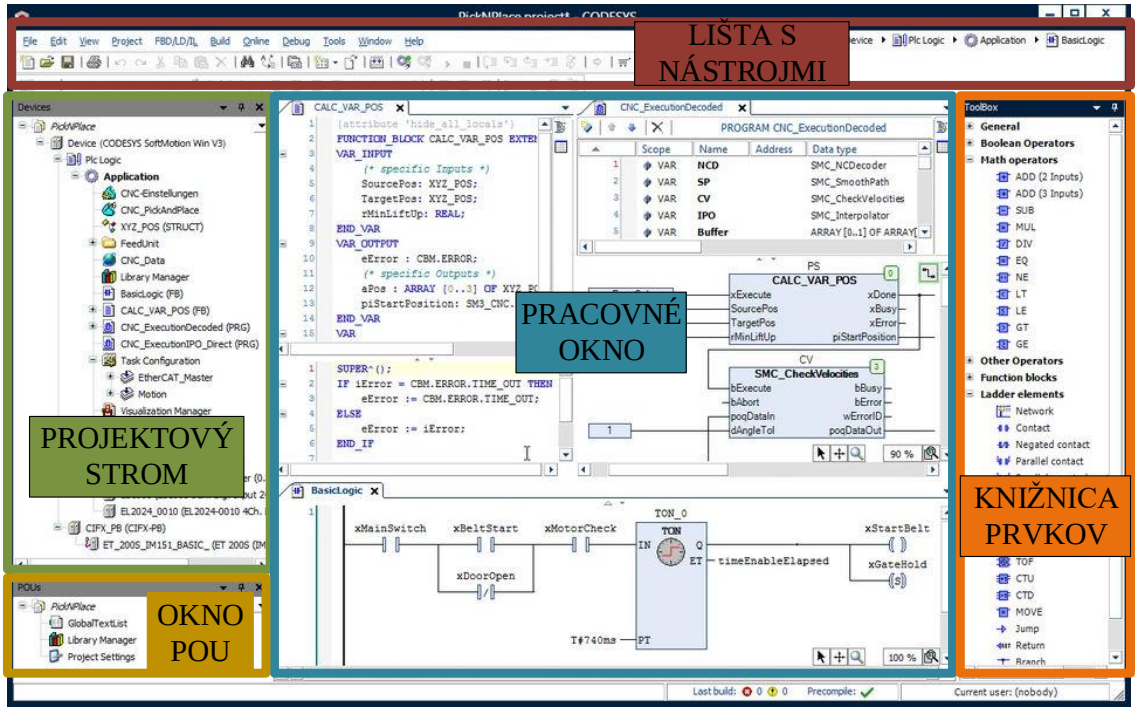


Obr. 4 - Platforma Codesysu [5]

Codesys bol certifikovaný nezávislou organizáciou PLCopen, ktorá sa aktívne podieľa na vývoji a vylepšovaní normy IEC 61131 ako celku. V niektorej literatúre sa používa staršie označenie normy IEC 1131, ktoré prešlo prečíslovaním kvôli možnosti jednotnej harmonizácie. [6]

3.1 Užívateľské prostredie

Samotné užívateľské rozhranie pripomína svojim vzhľadom niektoré iné vývojové prostredia. Obsahuje základne funkcie ako editor zdrojového kódu pre viacero jazykov, kompilér, debugger, a mnohé iné. Rozloženie okien je možno vidieť na obrázku č.5. Jednotlivé elementy je možné prispôsobiť podľa potreby užívateľa. Celkový vzhľad sa mierne zmení po prihlásení sa do online režimu, kde sa elementy pre úpravu programu skryjú a na ploche pribudnú elementy pre ladenie a kontrolu programu. Vizuálnou zmenou prejde i zobrazenie programu, kde každá premenná zobrazuje svoju aktuálnu hodnotu v textovom alebo grafickom znázornení.



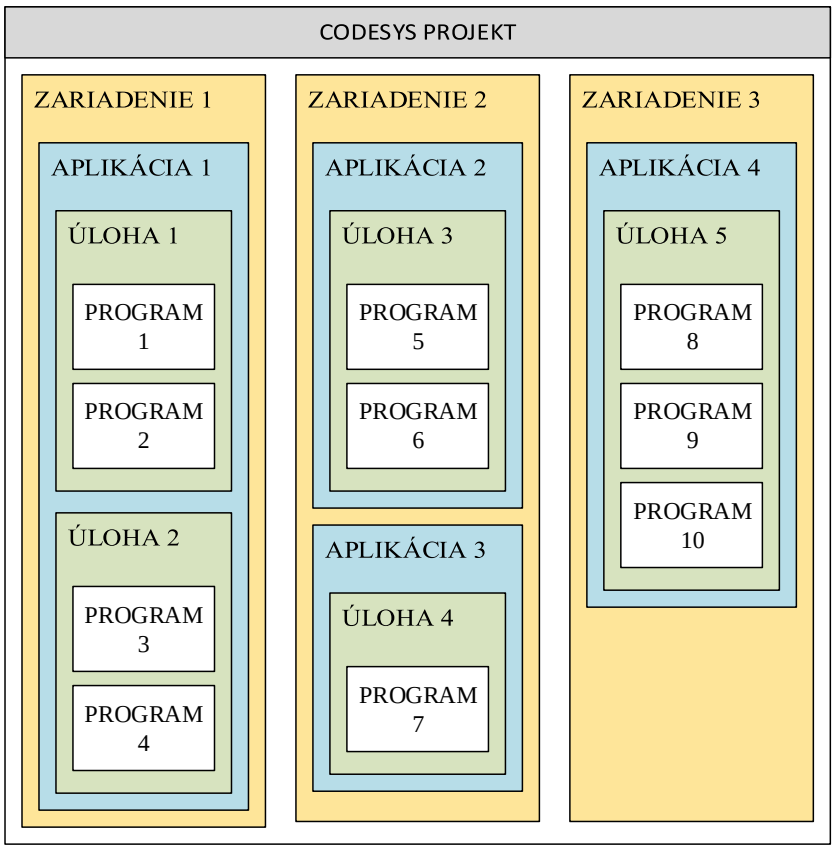
Obr. 5 - Rozloženie okien v programe Codesys [5]

3.2 Organizácia projektu

Norma IEC 61131 definuje určitú hierarchiu v zložení projektov. Najvyšším organizačným objektom je konfigurácia. Tá reprezentuje špecifický PLC systém spolu s hardvérovým usporiadaním, mapovaním pamäte či riadením spracovania zdrojov. Zdroje definujú poskytovateľa výpočtového výkonu v rámci nakonfigurovaného PLC systému V tele svojej deklarácie následne združujú úlohy. Úloha je riadiaci element, ktorý na základe určitých podnetov (cyklické opakovanie, udalosť, ...) vyvoláva samotné programy. Úlohy je možné deklarovať len v rámci zdrojov. Jedna úloha môže obsahovať viacero programov, pričom ich poradie vykonávania je definované poradím zápisu v úlohe. Poradie vykonávania úloh sa definuje v podobe priority vyjadrenej číselne [7].

Najvyšším organizačným objektov v prostredí Codesys je projekt. Ten môže obsahovať viac ako jednu konfiguráciu. Štruktúra celého projektu je symbolicky zobrazená na obrázku č.6. V rámci prostredia je štruktúra viditeľná v projektovom

strome. Codesys nazýva jednotlivé prvky Device (Zariadenie = Konfigurácia), Application (Aplikácia = Zdroje), Task (Úloha) a Program (Program).



Obr. 6 - Organizácia projektu v Codesys-e

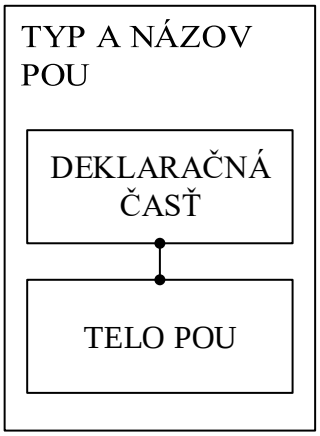
Pre spustenie programu je potrebné celý projekt vystavať (Build), prihlásiť sa do systému PLC (Login) a spustiť PLC (Run). Program sa skompiluje a nahrá do PLC v rámci procesu prihlásenia sa do systému. Pre účely testovania a simulácii je možné použiť softvérovo emulované PLC, ktoré je súčasťou prostredia.

3.3 POU – Program organization unit – Programová organizačná jednotka

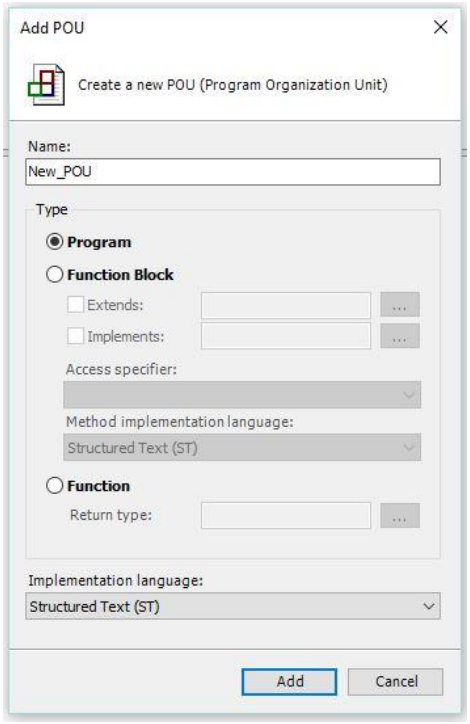
POU je súhrnné označenie blokov, z ktorých sú zložené samotné programy. Norma IEC 61131 unifikovala bloky do troch kategórii s presne stanovenými vlastnosťami – funkcie, funkčné bloky a programy [7]. Jednotky vytvára užívateľ alebo sú priamo poskytované od dodávateľa hardvéru vo forme doplnkov a knižníc. POU je vo svojej podstate zapuzdrená jednotka, čo dovoľuje jednotky kompilovať samostatne a následne ich spájať v rámci väčších projektov.

Každá jednotka pozostáva z 3 hlavných častí : typ jednotky a názov, deklaračná časť a programová časť. Štruktúra je zobrazená na obrázku č.7. Typ jednotky nadobúda jednu z troch hodnôt (FUNCTION, FUNCTION_BLOCK alebo PROGRAM). Názov jednotky musí spĺňať podmienky podobné ako názvy premenných. Deklaračná časť POU slúži na zadeklarovanie premenných potrebných pre danú POU usporiadaných do skupín

podľa použitia (vstupné, výstupné, lokálne, ...). Telo POU (tiež nazývaná inštrukčná alebo výkonová časť) je súhrn inštrukcií, ktoré sa vykonajú po zavolaní danej POU.



Obr. 7 - Zloženie POU



Obr. 8 - Okno pre vkladanie novej POU

Samotné jednotky je možné do programu vložiť pomocou jednotného okna zobrazeného na obrázku č.8. Codesys následne automaticky vytvorí predpripravenú formu danej jednotky. Jazyk novo vytvorených jednotiek sa môže líšiť od jazyka použitého v hlavnom programe.

3.3.1 Funkcie

Funkcia je najjednoduchšia forma POU. Môže mať viaceré vstupné hodnoty ale je obmedzená len na jednu výstupnú, návratovú hodnotu. Dátový typ návratovej hodnoty sa definuje pri deklarácii funkcie za jej názvom a názov funkcie slúži ako výstupná premenná. Funkcia sa deklaruje medzi kľúčové slová FUNCTION a END_FUNCTION. Funkcie nemajú žiadne statické premenné dôsledkom čoho je výstupná hodnota pri rovnakých vstupných parametroch (argumentoch) vždy rovnaká. Funkcie sú preťažiteľné (overloading – dokážu pracovať so vstupmi rôznych dátových typov) a rozšíriteľné (extensible – môžu mať premenný počet vstupov). Samotná norma definuje veľké množstvo preddefinovaných funkcií, ktoré možno v rámci štandardnej knižnice využívať. Funkcia nemôže volať funkčné bloky ani programy.

3.3.2 Funkčné bloky

Funkčný blok je POU, ktorý dovoľuje väčší počet výstupov ako jeden. Na rozdiel od funkcie, funkčný blok môže obsahovať statické premenné dôsledkom čoho nemusí byť výstupná hodnota rovnaká pri rovnakých vstupných parametroch. Rozdielnosť je spôsobená predchádzajúcimi vnútornými stavmi, ktoré si funkčný blok pamätá. Funkčný

blok sa deklaruje medzi kľúčové slová FUNCTION_BLOCK a END_FUNCTION_BLOCK.

Zo zadeklarovaných funkčných blokov sa vytvárajú inštancie podobne ako pri deklarovaní nových premenných (názov_inštancie : funkčný_blok). Tie sú pamäťovo oddelené a medzi sebou nezávislé. Pre pristupovanie k premenným (vstupy a výstupy) daných inštancií z iných POU je potrebné okrem názvu premennej definovať aj požadovanú inštanciu (názov_inštancie.premenná).

Podobne ako pri funkciách, štandardné knižnice obsahujú mnoho preddefinovaných funkčných blokov (bistabilné prvky, detekcia hrán, počítadla a časovače)

3.3.3 Programy

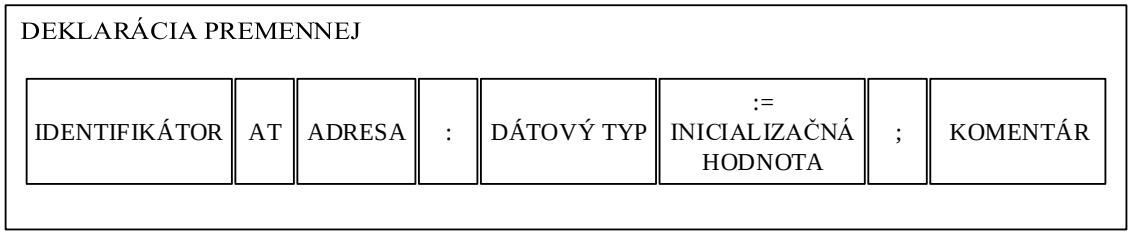
Program je najvyššia forma POU definovaný normou ako „logický súhrn prvkov programovacích jazykov a konštrukcií nutných pre myslené spracovanie signálov, ktoré sú potrebné pre riadenie stroja alebo procesu systémom programovateľného automatu“.[7] Programy nevytvárajú inštancie ako funkčné bloky, zachovávajú stavy premenných a dokážu pracovať s viac ako jednou výstupnou premennou. Program môže byť volaný len z iných programov a môže volať všetky typy POU. Program sa deklaruje medzi kľúčové slová PROGRAM a END_PROGRAM.

V rámci každého projektu je špeciálne preddefinovaná POU s názvom PLC_PRG. Tento program je volaný práve raz počas každého pracovného cyklu. PLC_PRG je považovaný za hlavný program a bez pokročilých znalostí sa ho neodporúča vymazať alebo premenovať.

Ukončenie deklarácie projektu (a aj ostatných typov POU) kľúčovým slovom END_PROGRAM sa nevyžaduje. Codesys automaticky vykonáva tento krok pri kompilácii programu.

3.4 Deklarácia a typy premenných

Premenné slúžia k uchovávaniu dát v pamäti. Každá premenná musí byť zadeklarovaná pred jej použitím v programe. Deklarácia premennej má presne definovaný tvar, ktorý je zobrazený na obrázku č.9. Povinnými údajmi sú identifikátor a dátový typ. Adresa (v tvare AT adresa), inicializačná hodnota a komentár sú údaje voliteľné.

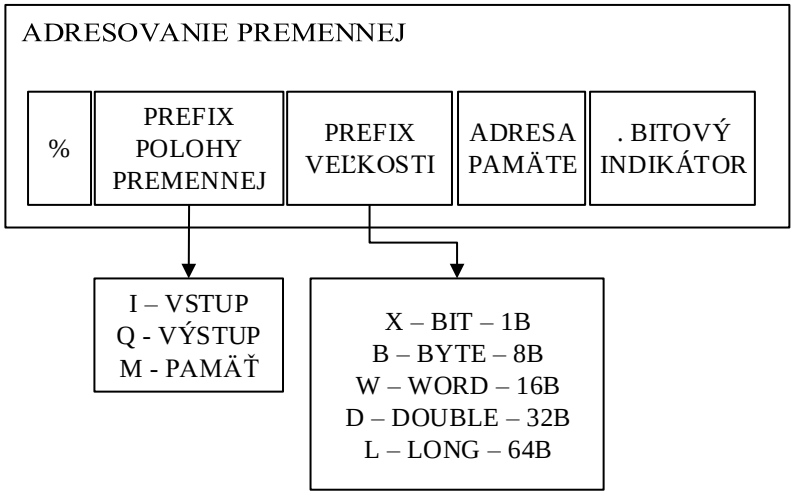


Obr. 9 - Deklarácia premennej

Pri tvorbe identifikátorov treba dodržiavať určité pravidlá. Identifikátor nie je obmedzený počtom znakov, musí začínať písmenom a nerozlišuje veľké a malé písmená.

Existuje ešte niekoľko ďalších špecifických obmedzení ako napr. obmedzenie počtu podčiarkovníkov (_) na maximálne jeden po sebe idúci znak a iné. Porušenie týchto pravidiel vedie na chybu pri kompilácii.

Rozloženie premenných v pamäti PLC ma na starosti prekladač. Táto činnosť je vykonávaná automaticky. Pre prípady, ktoré si to vyžadujú, je možné doplniť adresu do deklarácie danej premennej pridaním kľúčového slova AT spolu s adresou. Adresovanie sa riadi normou IEC 61131-3, ktorá definuje tvar adresy podľa obrázku č.10. Bitový indikátor sa používa len v prípade adresovania po bitoch. Rozsah použiteľných adries je závislý na použitom hardvéri.



Obr. 10 - Adresovanie premenných

Dátový typ premenných môže nadobúdať viac ako 24 rôznych typov. Každý typ presne definuje charakter dát a veľkosť potrebnej pamäte. Štandardné typy premenných (definované normou IEC 61131-3) sú doplnené o rozširujúce (Union, Bit, Pointer,...) a užívateľsky definované (Structure, Reference, Identifier,...). Dátové typy definované v norme IEC 61131-3 sú všeobecne známe typy premenných ako je Bool, Integer, Word či Real. Medzi menej známe typy patria typy definujúce čas a dátum (Time, Time_of_day TOD, Date a Date_and_Time DT) a špeciálny typ 64b premennej LTime, ktorý definuje čas v rozlíšení nanosekúnd (slúži pre veľmi presné časovače).

Doplnením deklarácie o špecifické kľúčové slová je možné upraviť charakter premennej. Doplnením kľúčového slova „Constant“ sa definuje konštanta, teda premenná s nemennou hodnotou. Pri použití kľúčových slov „Retain“ a „Persistent“sa premenné ukladajú na permanentnú pamäť a ich aktuálne hodnoty zostávajú dostupné aj po reštartoch (tvrdých i mäkkých) či výpadkoch napájania. Použitie týchto dvoch kľúčových slov zodpovedá určitým pravidlám a je potrebná lepšia znalosť systému.

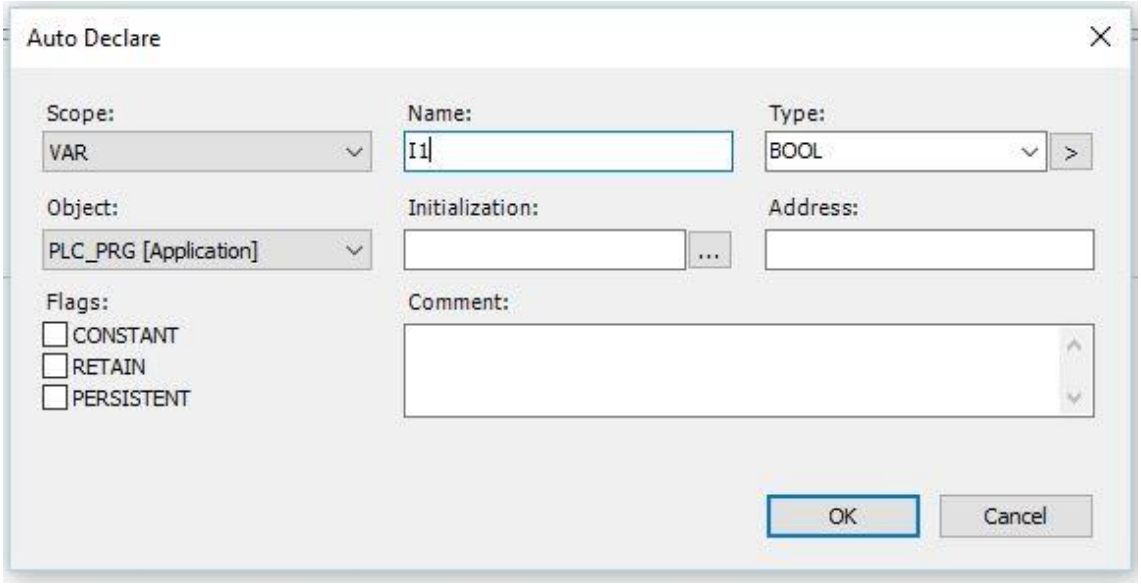
Premenné možno deklarovať viacerými spôsobmi a na viacerých miestach v programe. Deklarácia môže mať tvar textový alebo tabuľkový. Medzi oboma spôsobmi sa možno plynulo prepínať. Ukážka oboch spôsobov je zobrazená na obrázku č.11.

```
VAR
  IN1 AT %MW120 : BOOL; (*Vstup 1 typu boolean*)
  OUT1 : WORD; (*Výstup 1 typu word*)
  Message : STRING := "VZOR"; (*Textová správa*)
END_VAR
```

	Scope	Name	Address	Data type	Initialization	Comment	Attributes
1	VAR	**IN1**	%MW120	BOOL		Vstup 1 typu boolean	
2	VAR	**OUT1**		WORD		Výstup 1 typu word	
3	VAR	**Message**		STRING	"VZOR"	Textová správa	

Obr. 11 - Textová a tabuľková deklarácia premenných

Codesys sleduje zadeklarované premenné a pokiaľ sa v programe použije nová, dovtedy nezadeklarovaná premenná, vyvolá okno automatickej deklarácie. Údaje sa následne dopíšu v korektnej forme medzi deklarované premenné. Okno automatickej deklarácie je zobrazené na obrázku č.12.



Obr. 12 - Okno automatickej deklarácie

Deklarácia premenných môže byť realizovaná globálne alebo lokálne. Globálne deklarované premenné sa zapisujú do zoznamov GVL (Global variable list). Jeden projekt môže obsahovať viaceré zoznamy, častokrát za účelom zvýšenia čitateľnosti programu. Lokálne premenné sa deklarujú v deklaračnej časti príslušného POU.

3.5 Programovacie jazyky podľa normy IEC 61131-3

Norma IEC 61131 predstavuje medzinárodný štandard pre programovateľné automaty. Norma bola publikovaná v decembri 1993 medzinárodnou elektrotechnickou komisiou (skratka IEC). Prešla niekoľkými revíziami a platí jej aktuálna verzia 3.0 z roku 2013. Norma obsahuje 9 statí, ktoré popisujú súhrn požiadaviek na moderné riadiace systémy. Jej tretia časť pojednáva o syntaxi a sémantike programovacích jazykoch. Norma rozlišuje jazyky textové (IL a ST), grafické (FBD a LD) a špeciálne (SFC). CODESYS podporuje všetky tieto jazyky. Norma bola prevzatá úradom pre normalizáciu pod označením STN EN 61131-3, pričom aktuálne platí jej verzia z roku 2013 [8]. V Českej republike platí rovnaká verzia normy pod označením ČSN EN 61131-3 vydaná taktiež v roku 2013 [9].

3.5.1 IL – Instruction list – Zoznam inštrukcií

Tento textovo založený jazyk bol zrušený v poslednom vydaní normy. Je však vhodné si ho pripomenúť, nakoľko mnohé staršie aplikácie stále fungujú na programe vytvorenom v tomto jazyku. Jeho základom sú jednoduché inštrukcie, ktoré tvoria dohromady inštrukčnú sadu. Jazyk svojou štruktúrou pripomína assembler. Jednotlivé inštrukcie sa zapisujú na samostatné riadky ako kombinácia operátora (príkazu) a operandov (premenných). Vysoká kontrola nad spracovávanými inštrukciami spolu s efektívnym využívaním pamäte patria medzi hlavné výhody programov v jazyku IL. Medzi nevýhody jazyka patrí jeho náročnosť na znalosti, dĺžka programu a jeho neprehľadnosť pri zložitejších konštrukciách. Ukážka jednoduchého programu je zobrazená na obrázku č.13.

```
1  PROGRAM PLC_PRG
2  VAR
3      I1: BOOL;
4      I2: BOOL;
5      I3: BOOL;
6      Q1: BOOL;
7  END_VAR
```

```
LD   I1
AND  I2
ORN  I3
ST   Q1
```

Obr. 13 - Deklarácia premenných a ukážka programu v jazyku IL

3.5.2 ST – Structured text – Štruktúrovaný text

Jazyk štruktúrovaného textu pripomína vyššie programovacie jazyky, ktoré boli predlohou pri jeho tvorbe (Pascal, C či Ada). Jazyk dovoľuje viacnásobné vetvenie použitím príkazov If-then-else alebo Case a vytváranie iteračných slučiek pomocou príkazov For, While alebo Repeat. Programy napísané v jazyku ST sú niekoľkonásobne kratšie a ľahšie čitateľné ako pri rovnakom programe v jazyku IL. V jazyku ST je možné definovať zložitejšie funkcie, ktoré je následne možné využiť v ostatných jazykoch. Ukážka jednoduchého programu je zobrazená na obrázku č.14.

```
1  PROGRAM PLC_PRG
2  VAR
3      I1: BOOL;
4      I2: BOOL;
5      I3: BOOL;
6      Q1: BOOL;
7  END_VAR
```

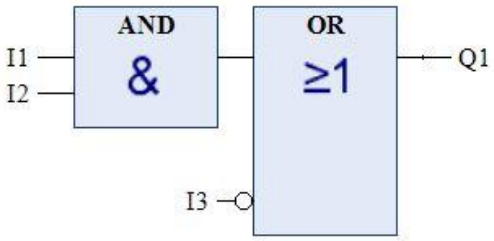
```
Q1 := (I1 AND I2) OR NOT I3;
```

Obr. 14 - Deklarácia premenných a ukážka programu v jazyku ST

3.5.3 FBD – Function block diagram – Schéma funkčných blokov

FBD patrí medzi grafické jazyky. Jeho štruktúra pripomína elektronické obvodové schémy alebo procesné diagramy. Základným prvkom jazyka sú funkčné bloky, ktoré reprezentujú jednotlivé funkcie v podobe obdĺžnikových značiek. Bloky sú poprepájané navzájom medzi sebou. Pripojenia na ľavej strane reprezentujú vstupy, zatiaľ čo pripojenia sprava slúžia ako výstupy. Spájať medzi sebou je možné len rovnaké dátové typy. Celý program sa následne vykonáva zľava doprava a zhora nadol. Ukážka jednoduchého programu je zobrazená na obrázku č.15.

```
1  PROGRAM PLC_PRG
2  VAR
3      I1: BOOL;
4      I2: BOOL;
5      I3: BOOL;
6      Q1: BOOL;
7  END_VAR
```



Obr. 15 - Deklarácia premenných a ukážka programu v jazyku FBD

V Codesyse je možné využiť tiež jazyk CFC – continuous function chart – diagram nepretržitých funkcií. Tento jazyk je rozšírenie nad rámec normy IEC 61131-3 poskytovaný veľkým množstvom výrobcov programovateľných automatov. Jeho základné charakteristiky sú veľmi podobné s jazykom FBD. Najväčšie odlišnosti sú v type špeciálnych blokov, a v možnostiach spájania blokov, napr. využitím spätnej väzby. Veľkou výhodou je možnosť explicitne riadiť poradie spracovávania programu.

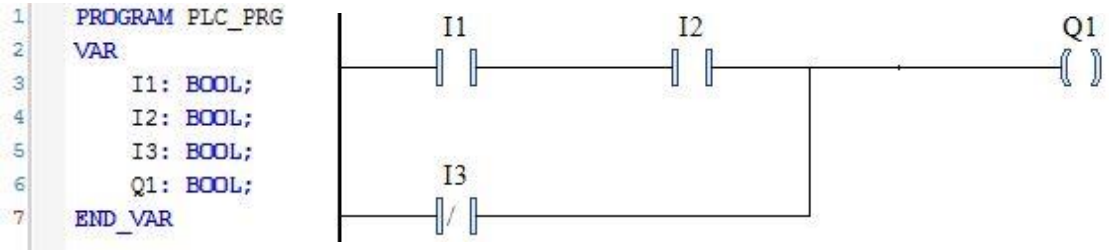
3.5.4 LD – Ladder diagram

LD je druhým grafickým jazykom. Jeho štruktúra pripomína reléové schémy. Program sa tvorí primárne vkladaním spínacích a rozpínacích kontaktov (vstupy), a cievok elektromagnetov (výstupy) medzi dve zvislé zbernice ohraničujúce program zľava (napájanie) a sprava (uzemnenie). Program je možné doplniť o ďalšie funkcie a prvky reprezentované grafickými blokmi. Program sa vytvára obdobným spôsobom ako pri jazyku FBD.

Kontakty sa kreslia ako dve paralelné zvislé čiary (| | resp. negovaný vstup | \ |), ktoré predstavujú booleovskú premennú. Pokiaľ premenná nadobúda hodnotu „pravda“, kontakt sa zopne a signál prejde zľava doprava. V opačnom prípade ostane kontakt rozopnutý a signál sa ďalej nešíri.

Výstupy sa kreslia ako cievky v podobe okrúhlych zátvoriek ( ) resp. (\ ). Výstup nadobúda hodnotu podľa charakteru signálu, ktorý do neho vstupuje z ľavej strany. Špeciálnym typom výstupu sú nastavovacie (S) a nulovacie (R) cievky, ktoré zastávajú úlohu bistabilného preklápacieho obvodu typu RS.

Ukážka jednoduchého programu je zobrazená na obrázku č.16.



Obr. 16 - Deklarácia premenných a ukážka programu v jazyku LD

3.5.5 SFC – Sequential function chart – Sekvenčí funkčný diagram

SFC je grafický jazyk, ktorý umožňuje chronologický popis činností v rámci programu. Jeho štruktúra vychádza z charakteristiky Petriho sietí. Základnou snahou je rozdelenie programu na sériu jednoduchších krokov. Program pozostáva z jednotlivých krokov a prechodov medzi nimi. Každý krok je súbor akcií, ktoré môžu byť naprogramované v ľubovoľnom jazyky zmienenom vyššie. Akcie daného kroku sa vykonajú pokiaľ je krok aktívny. Aktivácia a deaktivácia krokov je podriadená určitým podmienkam, ktoré sú definované v prechodoch.

3.6 Verzie softvéru

Samotný Codesys nie je produktovo orientovaný softvér, na rozdiel od iných špecializovaných nástrojov na programovanie automatov, akými sú napr. Step7 od firmy Siemens [10] alebo Automation Builder od firmy ABB [11]. I keď neexistuje len jediná verzia Codesysu, rôznorodosť je zabezpečená množstvom doplnkov a knižníc, ktoré sú už špeciálne upravené podľa potreby jednotlivých výrobcov hardvéru. Veľkou výhodou je možná kombinácia hardvéru za použitia rovnakého softvéru. Toto sa najčastejšie využíva pri kombinácii riadiaceho systému a bezpečnostného systému alebo riadiaceho systému a systému pre ovládanie pohonu motorov od rôznych dodávateľov.

Staršia verzia softvéru sa označuje v2.3. Táto verzia softvéru je stále používaná vo veľkom vzhľadom na množstvo aplikácii a kompatibilných zariadení vytvorených v minulosti. Podľa dostupných informácii je plánová podpora tohto softvéru do konca roku 2019 [5].

Vývoj novej verzie bol ovplyvnení zmenami v prístupe k daným problémom. Na novú verziu boli kladené väčšie nároky na rýchlosť, možnosti práce s komunikačnými rozhraniami, vizualizáciu, bezpečnosť a mnoho ďalších. Tento vývoj vyústil vo vytvorení novej verzie Codesysu označeného v3.x, aktuálne v3.5.

Najvýraznejším rozdielom medzi oboma verziami je pridanie možnosti objektového programovania do verzie v3.5. Tento krok predchádzal aktuálnemu rozšíreniu normy IEC 61131, ktorá túto funkcionalitu po novom vyžaduje. Medzi ďalšie

významné rozšírenia patrí možnosť programovania viacerých PLC v rámci jedného projektu a viacerých aplikácii v rámci jedného PLC, zlepšenie prekladu pri prepínaní medzi rôznymi jazykmi a možnosť výmeny dát cez XML formát. Vylepšeniami prešlo i komunikačné rozhranie a možnosti vizualizácie. Všetky podstatné rozdiely medzi oboma verziami je možné nájsť v štruktúrovanom dokumente priamo na stránkach Codesysu [13].

3.7 Licencovanie softvéru

Samotný softvér pre koncového užívateľa je voľne dostupný. Podlieha len súhlasu s licenčnou zmluvou pre koncového užívateľa. Doplnky v podobe nadstavbových knižníc a rozšírených možností je potrebné dokúpiť. Táto práca bola vypracovaná na hardvéri od spoločnosti Festo. Upravená verzia a potrebné knižnice sú voľne dostupne na stránkach dodávateľa.

3.8 Podporovaný hardvér

Stránky Codesys-u ponúkajú viac ako 300 typov kompatibilného hardvéru. V mnohých prípadoch ide o celé produktové rodiny, ktoré je možné dostať v niekoľkých desiatkach konfiguráciách podľa potreby a želaní koncových užívateľov. Medzi významných výrobcov hardvéru programovateľného v Codesys-e patrí Bosch Rexroth, Festo, Eaton, Schneider, Lenze, WAGO, ifm a mnoho ďalších [5].

Hardvér možno rozdeliť tiež do skupín podľa jeho charakteru ako je samotné PLC (alebo priemyselný počítač), prvky pre vizualizáciu (HMI – human-machine interface), prvky pre komunikáciu, bezpečnostné prvky a iné. Pri PLC sú základnými charakteristikami výkon procesorovej jednotky CPU, veľkosť pamäte, počet a typ vstupov a výstupov, možnosti napájania a spôsob programovania. Tieto údaje sú špecifické pre jednotlivé PLC a možno ich nájsť v dátových listoch príslušných PLC systémoch.

Výkon alebo rýchlosť procesorovej jednotky nie je vždy údaj, ktorý výrobca udáva. Štandardne sa pohybuje v rámci stoviek megahertzov. Častejšie sa stretávame s údajom o rýchlosti vykonania skenovacích cyklov alebo o počte skenovacích cyklov za definovanú časovú jednotku alebo o rýchlosti vykonaniu jednej inštrukcie. Skenovací cyklus v sebe zahŕňa načítanie vstupnej hodnoty, prevedenie jednoduchej operácie a zápis výsledku do výstupnej premennej. V dnešnej dobe je možné pracovať so systémami, ktorých skenovacia rýchlosť presahuje stotisíc skenovacích cyklov za milisekundu [14].

Každé PLC má niekoľko typov pamätí (programová, dátová, ...). Ich veľkosť sa pohybuje v rámci kB až MB. Väčšina PLC ponúka možnosť rozšírenia kapacity pamäte pomocou pamäťových kariet.

Vstupy PLC slúžia na získavanie dát z okolitých systémov. Môžeme ich rozdeliť medzi digitálne a analógové. Rovnakým spôsobom môžeme rozdeliť aj výstupy. Digitálne vstupy i výstupy predstavujú typickú binárnu logiku (0/1). Analógové vstupy či výstupy pracujú so spojitým signálom a častokrát sa stretávame so špeciálne

upravenými modifikáciami určenými pre špecifické aplikácie (teplotné snímače PT100, riadiace prvky motorov, ...). Štandardné PLC ponúkajú danú kombináciu vstupov a výstupov priamo na základnej platforme a v prípade potreby sú rozšíriteľné do určitej miery pomocou rôznych vstupno-výstupných modulov.

Spôsob napájanie PLC systému volí jeho výrobca. V základe sa stretávame s dvoma variantami, 230AC alebo 24DC.

K PLC je možné pripojiť jednotku HMI, najčastejšie v podobe priemyselného displeja. Displeje slúžia k ovládaniu zo strany obsluhy ako aj ku grafickému znázorneniu aktuálneho stavu zariadenia. Nie je však pravidlom, že každá aplikácia PLC musí byť vybavená displejom.

3.9 Možnosti aplikácie Codesysu

Ako už bolo spomenuté, Codesys slúži hlavne pre programovanie priemyselných aplikácii. Tie je možné rozdeliť na niekoľko samostatných celkov ako priemyselná automatizácia, automatizácia budov, procesná automatizácia a mnoho ďalších.

Codesys bol využitý pri riadení nového typu automatizovaného prekladania konzerv. Pôvodný systém využívajúci systém reťazového pohonu a závor bol nahradený za komplexný magnetický uchopovač. Ten je poháňaný dvojicou servomotorov od firmy Festo. Vďaka synchronizácii oboch motorov pomocou Codesysu je možné prevádzkovať motory paralelne. Celá aplikácia zvýšila produktivitu o 30%. [15]

Spoločnosť VELTRU využila Codesys pre automatizovanú linku na balenie papriky. Hlavnou časťou linky je paralelný kinematický robot spolu s kamerovým systémom. Kamera sníma polohu a orientáciu paprík na dopravníku. Robot papriky odoberá z dopravníka, otáča do požadovanej polohy a ukladá do balenia. Podobným spôsobom vytvára aj balenie trojfarebných paprík. [16]

Nie všetky aplikácie sa skrývajú za stenami fabrík. Na hardvéri od spoločnosti Eaton a softvéri Codesys bol vytvorený Doner robot s označením SCR-Vg 2.1. Tento plne automatický robot poháňaný štvoricou jednosmerných motorov je schopný pomocou série snímačov narezať mäso s milimetrovou presnosťou aj bez prítomnosti obsluhy. [17]

Codesys dokáže pracovať aj v systémoch bez operačného systému na embeded platformách. Tieto sú použité na mnohých aplikáciach s ktorými sa stretávame denne. Napríklad automatické umývacie linky pre automobily, ovládanie žeriavov či priemyselné vážiace systémy. [5]

4 NÁVRH ALGORITMU PÁROVANIA

Pri optimalizácii existuje v dnešnej dobe veľké množstvo zaužívaných procesov, ktoré riešia špecifické problémy unikátnym spôsobom. Optimalizácia môže predstavovať nájdenie globálneho minima alebo maxima účelovej funkcie v definovanom priestore. [18] Tento prístup predstavuje matematické pojatie.

Párovanie súčastí ložiska patrí ku komplexnému procesu. Aby bolo možné vyrábať ložiská splňujúce určité kvalitatívne charakteristiky, je potrebné vyberať vstupné súčasti podľa určitých pravidiel. Tento výber je postupnosťou určitých činností. Pre účely automatizovaného párovanie je potrebné tieto činnosti zefektívniť a vhodne optimalizovať pre využitie v praxi. Vhodnou optimalizáciou sa pritom myslí zrýchlenie vykonávaných procesov, zmenšenie prehľadávaného priestoru či zefektívnenie dielčích činností.

4.1 Požiadavky na automatické párovanie

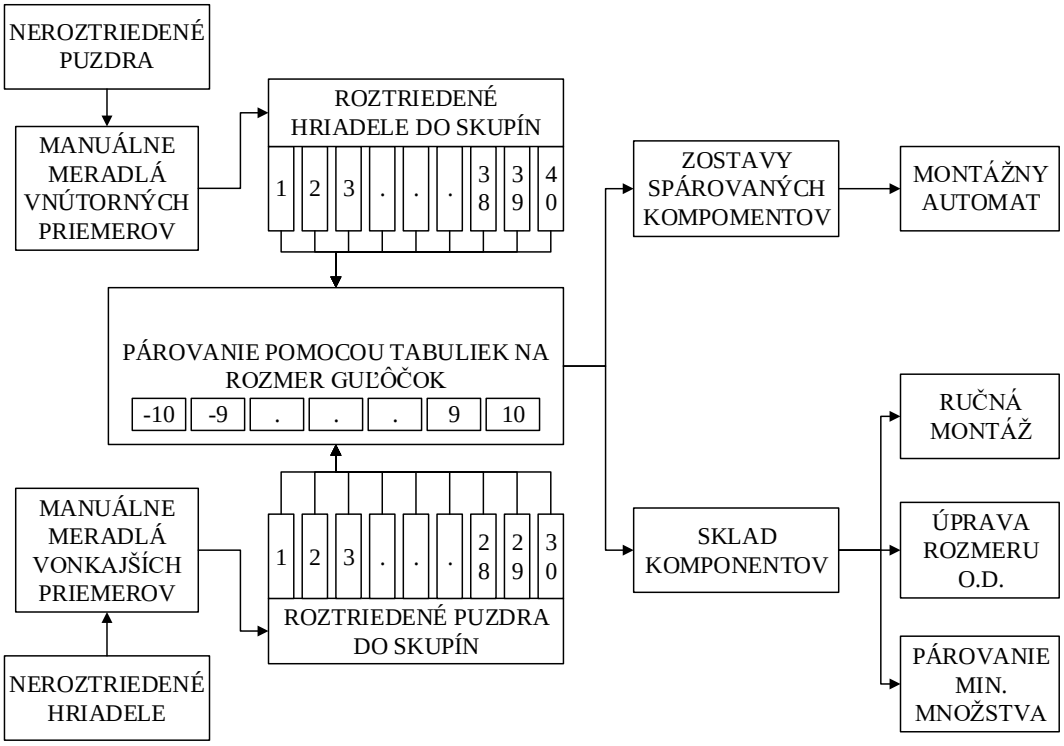
Požiadavky na zariadenie ako aj celý proces sa priamo odzrkadľujú na samotnom algoritme párovania. Na zariadenie sú kladené nasledujúce požiadavky zo strany zákazníka:

- Zariadenie musí byť schopné zmerať priemery rádiusových obežných dráh súčastí ložiska (puzdra i hriadeľa) v tolerancii 1µm a rozdeliť ich do rozmerových rád po 2µm
- Zariadenie musí zabezpečiť vzájomné spárovanie súčastí na predpísanú radiálnu vôľu. Pomôckou pre tento proces sú vzorové párovacie tabuľky.
- Zariadenie bude doplnené pred existujúce zariadenia ako náhrada za ručné rozmeriavanie vykonávané na špeciálnych meradlách.
- Zariadenie musí dodávať spárované súčasti do montážneho automatu v takte cca 7,5s
- Zariadenie musí obsahovať zásobník súčastí (puzdro i hriadeľ) na minimálne 10 minút práce automatu
- Celý proces musí byť automatický

4.2 Proces párovania bez automatizácie

Proces párovania bez automatizácie pozostáva z niekoľkých krokov. Pracovník manuálne meria hriadele na špecializovanom meracom prístroji a podľa veľkosti odchýlok triedi hriadele do predom definovaných skupín. Rovnakým procesom prechádzajú aj puzdra. Na meraní a triedení súčastí sa podieľa niekoľko identických meracích pracovísk. Na základe vnútropodnikového predpisu sa na zvolený typorozmer guľôčky spárujú definované skupiny hriadeľov so skupinami puzdier. Takto pripravené skupiny súčastí sú následne zmontované v montážnom automate. Súčasti, ktoré z rôznych dôvodov neboli vhodné pre párovanie sa umiestnia do skladu. Tieto súčasti sa použijú, keď dosiahnu určité minimálne množstvo vhodné pre montáž v montážnom automate

(rádovo stovky), alebo sa odošlú na úpravu rozmeru obežných dráh (ich prebrúsenie na iný rozmer), alebo sa zmontujú manuálne. Celý proces je zobrazený na obrázku č.17.



Obr. 17 –Proces párovania bez automatizácie

4.3 Proces montáže v montážnom automate

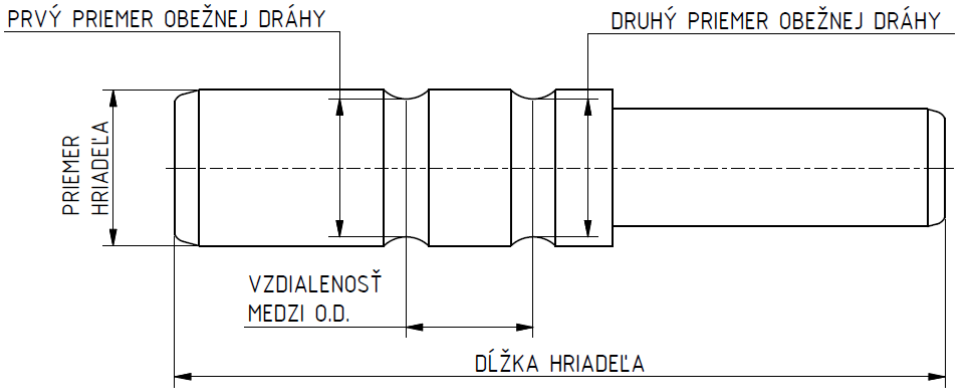
Montážny automat sa skladá z niekoľkých pracovných staníc. Každá stanica vykonáva presne stanovenú operáciu. Montážny proces sa začína vložením puzdra do prvej stanice. Zreťazenie medzi pracovnými stanicami zabezpečuje automatický prekladač. Ten naraz uchopí rozpracované ložiská zo všetkých staníc a presunie ich na stanicu nasledujúcu. Do prvej stanice sa vloží nové puzdro a z poslednej stanice sa odoberie skompletizované ložisko. Ložisko prejde ešte niekoľkými pracoviskami ako je automatický zábeh, meranie vibrácii, laserový popis a automatická paletizácia pomocou priemyselného robota. Posledná spomenutá operácia má za úlohu okrem orientovaného zakladania kusov do plastových paliet aj triedenie kusov vyradených niektorou kontrolnou stanicou automatu. Automat kontroluje prítomnosť (a správnu orientáciu) komponentov, váhu maziva, veľkosť axiálnej a radiálnej vôle.

4.4 Vstupné parametre ložiska

Súčasti ložiska možno rozdeliť do dvoch skupín podľa potreby znalosti odchýlky od menovitého rozmeru. Komponenty ako plastová klietka a plastovo-kovová krytka možno považovať za univerzálne. Odchýlka menovitého rozmeru týchto dielov v rámci povolených tolerancií nemá takmer žiadny vplyv na merateľné výstupné parametre ložiska. Druhou skupinou sú dielce, ktorých odchýlka od menovitého rozmeru priamo zasahuje do výsledných charakteristík ložiska. Tieto dielce sú puzdro, hriadeľ a guľôčky.

4.4.1 Rozmer hriadeľa

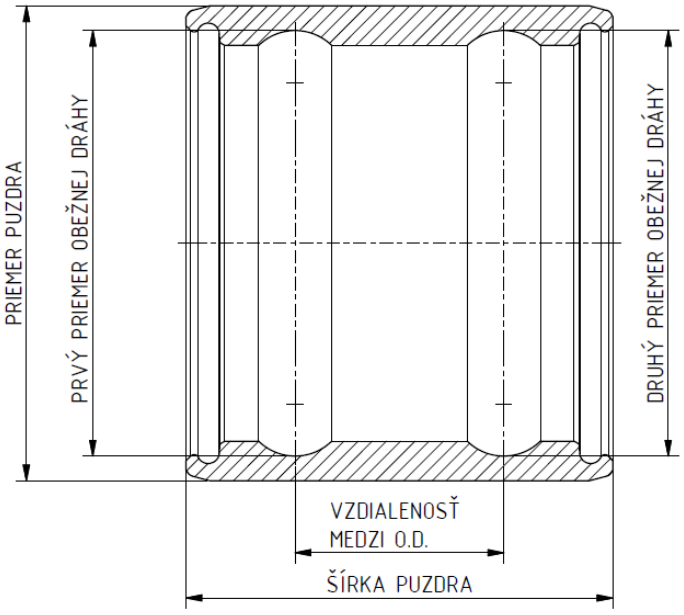
Nominálny priemer hriadeľa v mieste obežných dráh je 16mm. Nominálny priemer obežných dráh sa líši podľa typu montovaného ložiska. Odchýlka priemeru obežnej dráhy od nominálnej hodnoty je povolená v rozsahu od -30 µm do +50 µm. Pri triedení hriadeľov do tried po 2µm predstavuje tento rozsah 40 rôznych rozmerových tried. Hriadele s rozdielom medzi veľkosťami priemerov obežných dráh viac ako definovaná hodnota (štandardne 3µm) sa považujú za nezhodné a vylučujú sa z procesu párovania. Rozmerový náčrt hriadeľa je zobrazený na obrázku č.18.



Obr. 18 – Rozmerový náčrt hriadeľa

4.4.2 Rozmer puzdra

Vonkajší priemer puzdra je 30mm. Vnútorný priemer ako aj priemer obežnej dráhy sa líši podľa typu montovaného ložiska. Odchýlka priemeru obežnej dráhy od nominálnej hodnoty je povolená v rozsahu od -30 µm do +30 µm. Pri triedení puzdier do tried po 2µm predstavuje tento rozsah 30 rôznych rozmerových tried. Pre puzdra platí rovnaká podmienka ohľadne rozdielu odchýlok obežných dráh ako pri hriadeľoch. Rozmerový náčrt puzdra je zobrazený na obrázku č.19.



Obr. 19 - Rozmerový náčrt puzdra

4.4.3 Rozmer guľôčok

Nominálny priemer guľôčky je 6,35mm. Odchýlka od tohto rozmeru sa pohybuje v mikrometroch. Pre účely montáže ložiska sa používajú guľôčky s odchýlkou od -10µm po +10µm s krokom po 1µm. Tento rozsah predstavuje 21 rôznych rozmerových tried. Pri párovaní sa preferujú guľôčky s menšími odchýlkami od nominálnej hodnoty. Zároveň kvôli charakteristikám ložiska i montážnemu postupu je podmienkou použitie toho istého typorozmeru guľôčok pre obe obežné dráhy.

4.5 Výstupné parametre ložiska

Hlavnými kontrolnými parametrami ložiska sú veľkosť radiálnej a axiálnej vôle. Ich hodnoty sú priamo závislé na správnej kombinácii hriadeľa, puzdra a guľôčok. Hodnoty vôlí sú merané v rámci montážneho automatu. Pokiaľ sú namerané hodnoty mimo definovanej tolerancie, robot na konci výrobnej linky tieto kusy vyradí do príslušných chybových zásobníkov. Veľkosť radiálnej a axiálnej vôle pre štandardné valivé ložiská je definovaná v norme ISO 5753. V prípade ložísk pre vodné pumpy sú tieto hodnoty otázkou vnútropodnikových noriem, ktoré sledujú požiadavky zákazníka.

4.5.1 Radiálna vôľa

Radiálna vôľa je hodnota posunutia jedného elementu ložiska voči druhému z jednej krajnej polohy do druhej v smere kolmom na os rotácie. [19]

Meranie radiálnej vôle prebieha automaticky. Hriadeľ ložiska sa pevne upne a na puzdro sa vyvinie presne definovaná sila v jednom smere kolmom na os otáčania a následne v smere opačnom. Výsledná hodnota veľkosti radiálnej vôle je vyjadrená rozdielom medzi oboma krajnými polohami.

Veľkosť radiálnej vôle pre montované ložiská je určená na zostavnom výkrese. Jej hodnota sa definuje ako interval kladných hodnôt pri normovanom zaťažení 50N. Tento interval nikdy nezačína na nule.

4.5.2 Axiálna vôľa

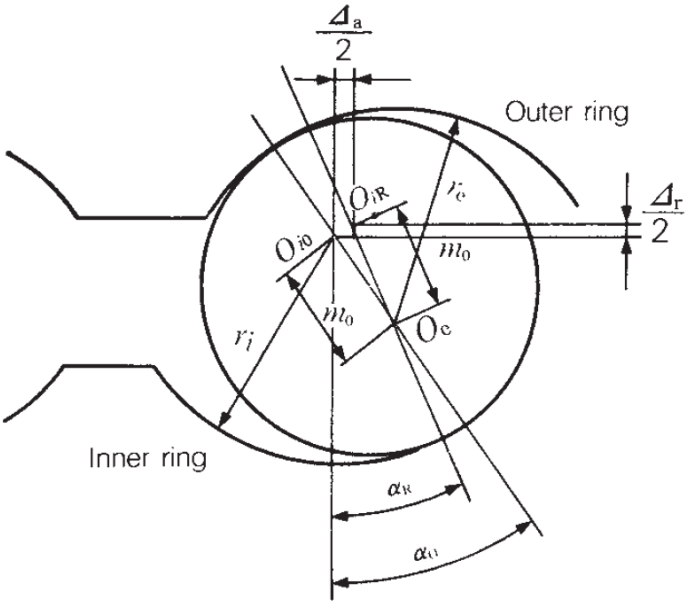
Axiálna vôľa je hodnota posunutia jedného elementu ložiska voči druhému z jednej krajnej polohy do druhej v smere osi rotácie. [19]

Meranie axiálnej vôle prebieha automaticky. Puzdro ložiska sa pevne upne a na hriadeľ sa vyvinie presne definovaná sila v smere osi hriadeľa a následne v smere opačnom. Výsledná hodnota veľkosti axiálnej vôle je vyjadrená rozdielom medzi oboma krajnými polohami.

Podobne ako pri radiálnej vôle, aj axiálna vôľa sa definuje pri normovanom zaťažení 50N. Na rozdiel od radiálnej vôle, pri axiálnej vôli sa predpisuje len jej maximálna hodnota, ktorá sa nesmie presiahnuť.

4.5.3 Závislosť medzi radiálnou a axiálnou vôľou

Geometrické vyjadrenie závislosti medzi radiálnou a axiálnou vôľou je zobrazené na obrázku č.20. Význam jednotlivých parametrov je vypísaný v tabuľke č.1.



Obr. 20 - Závislosť medzi radiálnou a axiálnou vôľou [19]

Δr	Radiálna vôľa
Δa	Axiálna vôľa
α_0	Počiatočný kontaktný uhol pri axiálnom posunutí súčastí
α_R	Počiatočný kontaktný uhol pri radiálnom posunutí súčastí
O_e	Stred zakrivenia obežnej dráhy vonkajšej súčasti
O_{iO}	Stred zakrivenia obežnej dráhy vnútornej súčasti pri axiálnom posunutí
O_{iR}	Stred zakrivenia obežnej dráhy vnútornej súčasti pri radiálnom posunutí
m_0	Vzdialenosť medzi stredmi zakrivenia súčastí.
D_w	Priemer guľôčky
r_i	Polomer zakrivenia vnútornej obežnej dráhy
r_e	Polomer zakrivenia vonkajšej obežnej dráhy

Tabuľka 1 – Význam rozmerových veličín závislosti $\Delta a = f(\Delta r)$

Podľa rozmerového náčrtu (obr.20) je možné zapísať závislosti (1) a (2) [19]:

$$m_0 \sin(\alpha_0) = m_0 \sin(\alpha_R) + \frac{\Delta_a}{2} \tag{1}$$

$$m_0 \cos(\alpha_0) = m_0 \cos(\alpha_R) + \frac{\Delta_r}{2} \tag{2}$$

Pomocou známej závislosti medzi goniometrickými funkciami rovnicou (3)

$$sin^2(\alpha_0) = 1 - cos^2(\alpha_0) \tag{3}$$

je možné prepísať rovnice (1) a (2) do tvaru rovnice (4) [19],

$$\left(m_0 \sin \alpha_R + \frac{\Delta_r}{2}\right)^2 = m_0^2 - \left(m_0 \cos \alpha_R - \frac{\Delta_r}{2}\right)^2 \tag{4}$$

kde vhodnými úpravami dostávame závislosť Δa = f(Δr) v tvare rovnice (5) [19]:

$$\Delta_a = 2\sqrt{m_0^2 - \left(m_0 \cos\alpha_R - \frac{\Delta_r}{2}\right)^2} - 2m_0 \sin\alpha_R \tag{5}$$

Do tejto závislosti vstupuje okrem radiálnej vôle len niekoľko parametrov, ktoré sa môžu pohybovať v rámci určitých tolerancií. Axiálna vôľa montovaných ložísk je navrhnutá tak, aby pri dodržaní požadovanej radiálnej vôle, nepresiahla svoj horný dovolený limit. Z tohto dôvodu postačuje pri párovaní zaistiť požadovanú radiálnu vôľu.

Ďalším dôvodom využitie radiálnej vôle ako hlavného parametra kvality procesu párovania je stabilita rozmerov. Radiálna vôľa je závislá hlavne na priemeroch obežných dráh. Ten je vyrobený opracovaním na výrobnom stroji pomocou nástroja, ktorý sa opotrebováva počas opracovania. Axiálna vôľa je závislá hlavne na vzdialenosti medzi obežnými dráhami. Tento rozmer je lepšie merateľný pri opracovaní ako priemer jednotlivých obežných dráh. Zároveň jeho odchýlky sú rádovo menšie ako pri priemeroch obežných dráh.

Pokiaľ by sa malo zachádzať do úplných detailov, je potrebné zobrať do úvahy aj uhlové naklopenie, zmeny vo vôľach spôsobené rozdielnou teplotou hriadeľa a puzdra, reálne deformácie spôsobené zaťažením, posunutie polohy guľôčok v obežnej dráhe vplyvom deformácie plastovej klietky a mnohé ďalšie.

4.5.4 Párovacia vôľa

Ako je vidno z matematických vyjadrení veľkosti radiálnej vôle v predchádzajúcej kapitole, je výpočet radiálnej vôle pomerne komplikovaný. Taktiež je výpočet vypracovaný len pre dvojradové ložiska s dvoj-bodovým stykom. Pre ložiska so štvôr-bodovým stykom sa celá situácia mierne mení. Montážny automatu je schopný montovať oba typy ložísk vzhľadom na skutočnosť, že po montážnej stránke medzi nimi nie je rozdiel.

Pokiaľ by kontaktný uhol v ložisku bol 0°, bolo by možné zapísať veľkosť radiálnej vôle rovnicou (6).

$$\Delta_r = D_o - D_i - 2D_w \tag{6}$$

Kontaktný uhol v reálnom ložisku má však presne danú teoretickú veľkosť v rozmedzí 15° až 40° v závislosti na type montovaného ložiska. Jeho veľkosť je závislá, okrem priemerov obežných dráh, aj na rozdiele medzi vzdialenosťami obežných dráh puzdra a hriadeľa. Pre účely párovania sa používa parameter označený ako párovacia vôľa. Je vyjadrením určitej linearizácie závislosti radiálnej vôle na reálnych parametroch ložiska vyjadrených pomocou parametrov merateľných vo výrobnom procese. Párovaciu vôľu Δ_{PP} možno vyjadriť rovnicou (7).

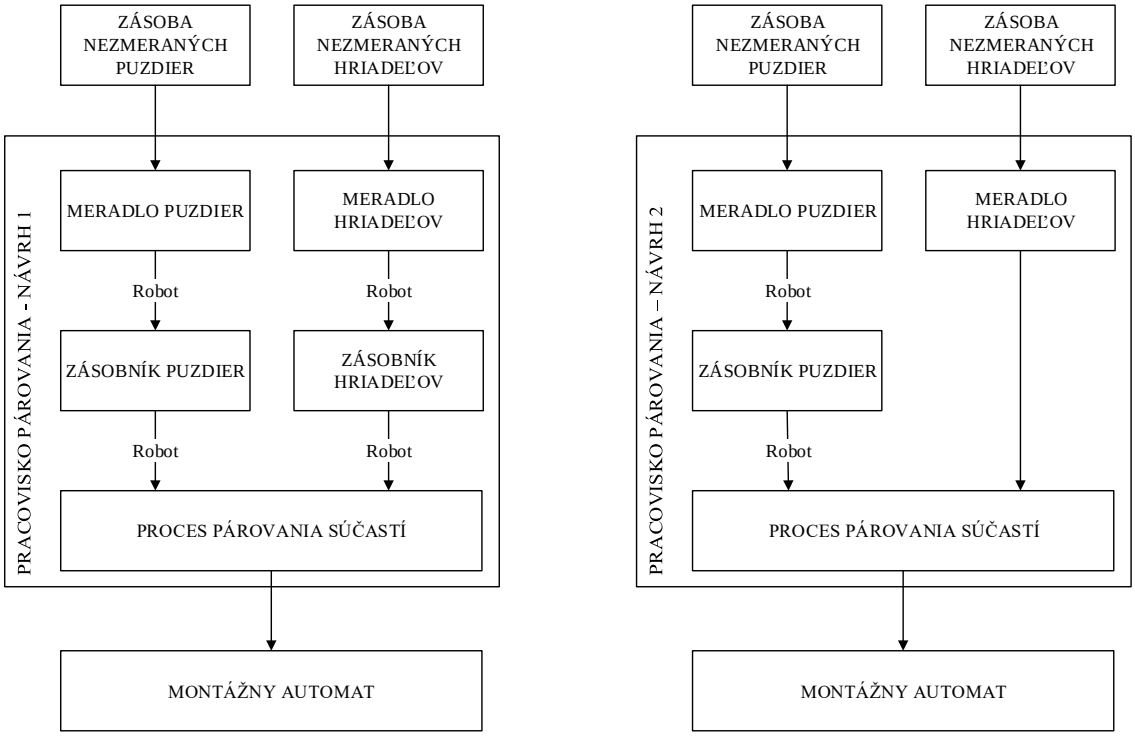
$$\Delta_{PP} = D_o - D_i - 2D_w \tag{7}$$

Dodržaním stanovenej párovacej vôle je zabezpečená hodnota radiálnej vôle v požadovanej tolerancii. Presnosť však klesá so zväčšujúcimi sa odchýlkami obežných dráh od nominálnych hodnôt. Veľkosť párovacej vôle je určená teoreticky aj overením v praxi a považuje sa za know-how výrobcu ložísk.

4.6 Konceptuálny návrh rozloženia pracovísk

Konceptuálny návrh vychádzal z konceptu párovania manuálnym spôsobom (obrázok č.17). Prvý návrh obsahuje samostatné meracie pracoviská pre hriadele i puzdra na ktorých výstupoch sa nachádzajú dostatočne veľké zásobníky. Vhodné páry vybraté z týchto zásobníkov následne vstupujú do montážneho automatu.

Druhý návrh upravoval prvotné riešenie z ekonomických a priestorových dôvodov. Pre jeden komponent bol vynechaný zásobník. Oba návrhy je možno vidieť na obrázku č.21.



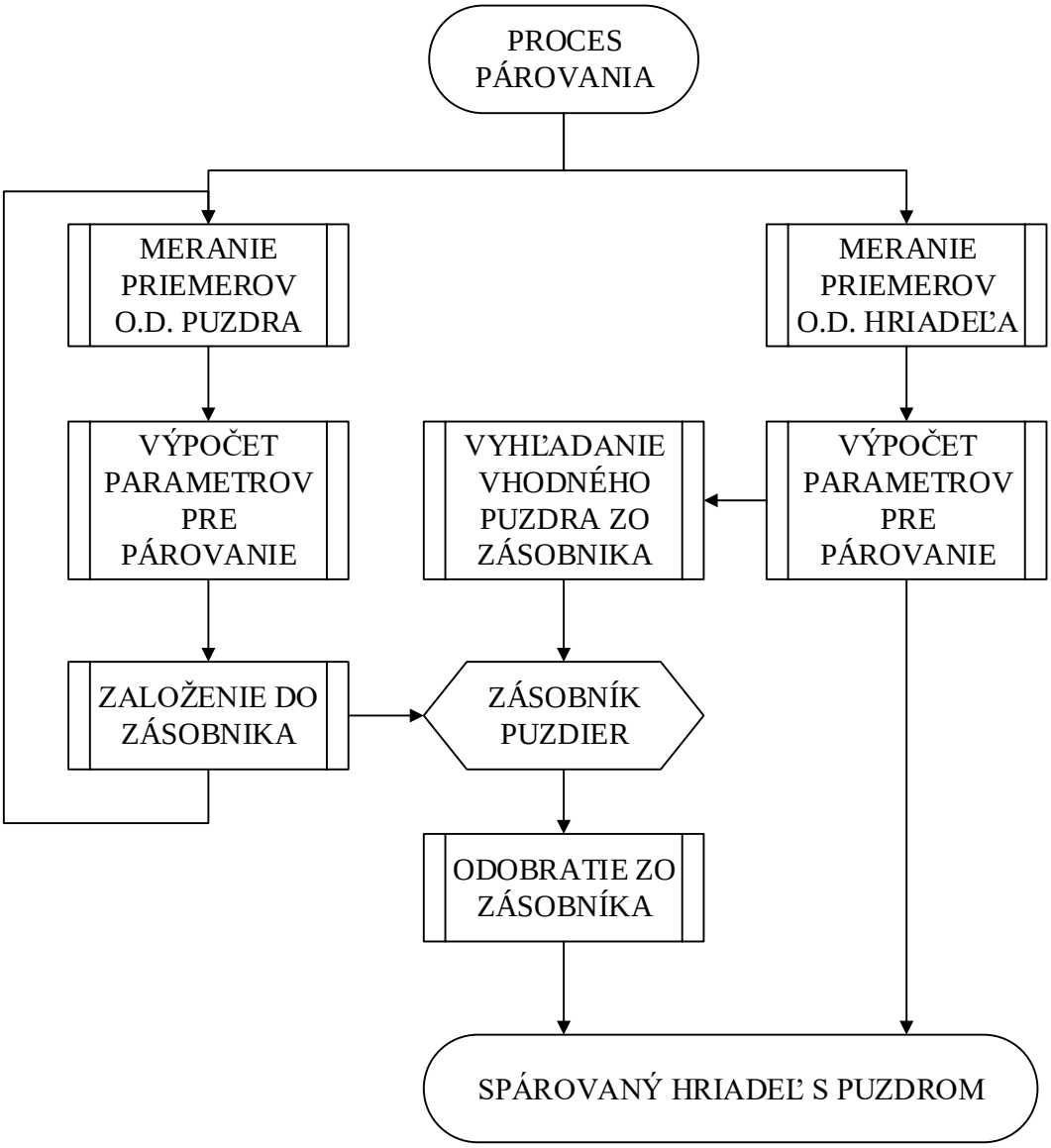
Obr. 21 - Konceptuálny návrh pracovísk

Pri procese párovania, ktorý by využíval dva maticové zásobníky by bolo možné dosiahnuť vyššej efektivity, hlavne pokiaľ sa jedná o súčasti s rozmerom priemeru obežnej dráhy na okraji tolerancie. Celý proces je však časovo obmedzený cyklovým časom čo nedovoľuje zložitejšie výpočty.

Na druhej strane bolo navrhnuté riešenie s jedným zásobníkom. Návrh zaberá menšie zástavbové rozmery a šetrí po ekonomickej stránke nemalé financie za zásobník a dvojicu robotov, ktoré ho obsluhujú. Z pohľadu párovania tiež prispieva k určitému stupňu zjednodušenia. Do algoritmu párovania vstupuje len jedna súčasť, ku ktorej sa hľadá pár pre daný rozmer valivého telieska.

Výsledné riešenie bolo spracované na základe druhého návrhu, kde veľkú úlohu zohrala práve ekonomická stránka riešenia. Pre nájdenie vhodného páru je potrebné vykonať niekoľko krokov. Jednotlivé súčasti je potrebné zmerať, namerané hodnoty spracovať a určiť parametre potrebné pre párovanie. Puzdra je potrebné vkladať do zásobníka. Zakladanie puzdier namiesto hriadeľov bolo zvolené pre menší počet variant puzdier ako aj jednoduchšie konštrukčné riešenie zásobníka. Na základe nameraného

hriadeľa je potrebné v zásobníku nájsť vhodný protikus. Tieto činnosti sú grafický naznačené v obrázku č.22.



Obr. 22 - Procesy potrebné pre párovanie

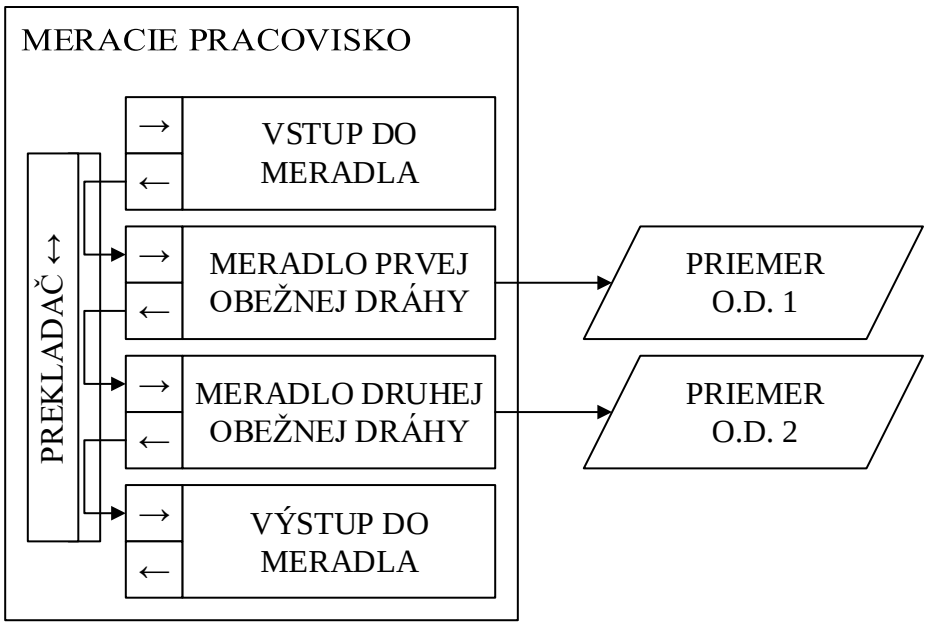
4.7 Meranie rozmerov a spracovanie nameraných hodnôt

Pre správne spárovanie komponentov je potrebné získať rozmerové údaje a vhodne ich spracovať. Pre tento účel boli skonštruované a následne vyrobené meradlá hriadeľov a puzdier.

4.7.1 Meracie pracovisko pre hriadele

Pracovisko pozostáva zo 4 pracovných pozícii. Vstupom do pracoviska je sklzový gravitačný zásobník. Kapacita vstupného zásobníka je 120 ks čím presahuje požadovanú kapacitu 80 kusov pre nepretržitú prácu automatu po dobu 10 min pri cyklovom čase 7,5s. Ďalšími dvoma pozíciami sú identické meradlá vonkajšieho priemeru obežnej dráhy. Poslednou pozíciou je výstup z pracoviska. Na tejto pozícii sa vyraďujú nezhodné hriadele. Zároveň táto pozícia slúži ako štartovací bod pre začiatok vyhľadávania

vhodného páru pre aktuálny hriadeľ. Pokiaľ je nájdený vhodný pár, hriadeľ je presunutý ďalej smerom do montážneho automatu. Pokiaľ sa vhodný pár nenájde, hriadeľ je vyradený. Prekladanie hriadeľov medzi pozíciami zabezpečuje paralelný prekladač. Koncept meradla je zobrazený na obrázku č.23.



Obr. 23 - Koncept meradla

4.7.2 Meracie pracovisko pre puzdra

Pracovisko pozostáva taktiež zo 4 pracovných pozícii. Vstupom do pracoviska je pásový dopravník. Kapacita dopravníka taktiež presahuje minimálnu požadovanú kapacitu 80 kusov. Vstupná pozícia je vybavená kontrolnou kamerou. Puzdra môžu mať na vonkajšom priemere drážku (pre poistný krúžok), ktorej poloha je asymetrická. Automat dovoľuje zmontovanie ložiska s opačne orientovaným puzdrom. Takéto ložisko je vyradené pri vizuálnej kontrole kvality povrchu ako zmätok. Ďalšími dvoma pozíciami sú identické meradlá vnútorného priemeru obežnej dráhy. Poslednou pozíciou je výstup z pracoviska. Na tejto pozícii sa nezhodné puzdra vyraďujú a ostatné sú odoberané robotom, ktorý ich zakladá do maticového zásobníka. Princíp prekladania puzdier medzi pozíciami je identický ako pri hriadeľoch, teda pomocou paralelného prekladača. Koncept samotného meradla je rovnaký ako pri meradle hriadeľov. Pracovisko sa líši z konštrukčnej stránky.

4.7.3 Princíp meradiel

Vzhľadom na krátky cyklový čas je potrebné meranie obežnej dráhy uskutočniť rádovo v sekundách. Meranie každej obežnej dráhy preto prebieha samostatne. Po odčítaní manipulačných časov potrebných na prenos dielca do meradla a jeho následné odobratie z meradla dostaneme čas 3 až 4 sekundy v rámci, ktorého musí prebehnúť celé meranie.

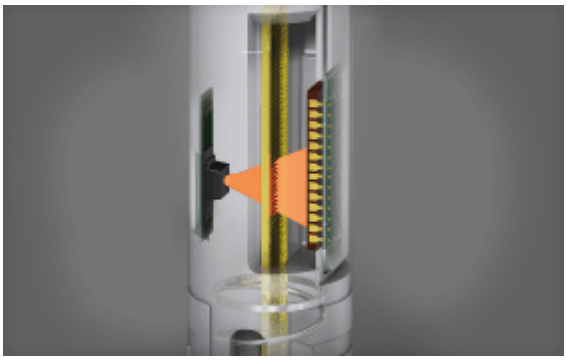
Pre meranie obežnej dráhy sa používa špeciálna konštrukcia, ktorá prevádza meranie priemeru na meranie lineárne. Pneumatický valec pritláča 6 veľmi presných guľôčok uložených v klietke do obežnej dráhy pomocou kužeľa s vrcholovým uhlom 20°.

Veľmi prísna tolerancia vrcholového uhla zabezpečuje, že skreslenie z dôvodu výrobných nepresností nepresahuje 0,5% meranej hodnoty.

Meradlá pracujú porovnávaciou metódou. Na začiatku práce je potrebné každé meradlo skalibrovať pomocou kalibračného kusu s presne definovanými rozmermi. Tieto rozmery sú merané na meracom stredisku v laboratórnych podmienkach. Táto kalibrácia je vyžadovaná opakovane po zmeraní určitého množstva komponentov.

Pre meranie lineárneho posunu sa používa kontaktný snímač typu GT2 od firmy Keyence. Štandardné snímače pre presné merania využívajú systém počítania pulzov (prechod svetla cez dve stupnice so štrbinami alebo zmena magnetických pólov špeciálnej stupnice) alebo pomocou rozdielovej transformátorovej metódy (označované ako LVDT). [20] Nevýhodou týchto metód je zlá teplotná charakteristika a nepresnosť v krajných polohách pri LVDT a neznalosť absolútnej polohy a chybovosť počítania impulzov pri snímačoch so systémom počítania pulzov.

Konštrukcia snímača GT2 je odlišná. V jeho tele sa nachádza zdroj paralelného led svetla formovaného cez špeciálny hranol, stupnica z kremenného skla a výkonný CMOS snímač. Merací dotyk je uložený v lineárnom guľôčkovom ložisku a je pevne spojený so stupnicou, ktorá má v sebe špeciálne rozložené štrbiny. Zdroj svetla a CMOS snímač sú pevne uchytené v tele snímača. Pohyb meracieho dotyku presúva stupnicu, čím zatieňuje rôzne časti na CMOS snímači. CMOS snímač vyhodnocuje osvetlené časti čím dokáže určiť absolútnu polohu meracieho kontaktu. Tento princíp merania sa nazýva Scale Shot System. Rez snímačom je zobrazený na obrázku č.24.



Obr. 24 - Princíp „Scale Shot System“ [21]

Pre meranie bol zvolený snímač s označením GT2-P12K, ktorého presnosť merania je 0,1µm a opakovateľnosť 1µm. Merací rozsah snímača je 12mm. Výstupom snímača je programovateľný analógový výstup zabezpečený komunikačnou jednotkou GT2-71MCP. Snímač a komunikačná jednotka sú zobrazené na obrázkoch č.25 a č.26.

Obr. 25 - Snímač typu GT2-P12K [24]

Obr. 26 - Komunikačná jednotka GT2-71MCP [24]

4.7.4 Spracovanie nameraných hodnôt

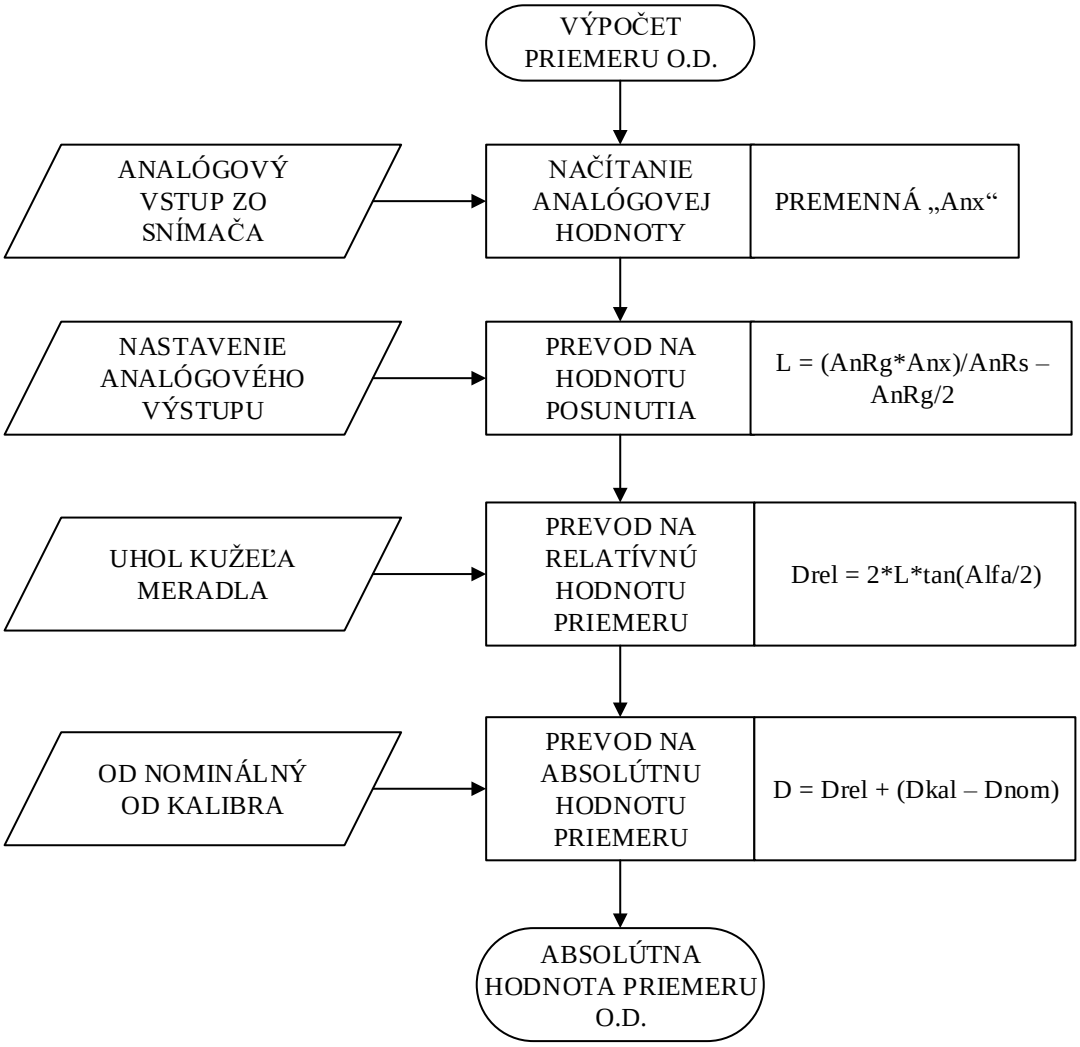
Celé pracovisko párovania obsahuje 4 meradlá, ktoré sú svojou funkciou rovnaké. Výstupom každého meradla je analógová hodnota zo snímača. Túto hodnotu je potrebné vhodne spracovať pre získanie reálneho rozmeru zmeraného priemeru.

Komunikačná jednotka je nastavená tak, aby zobrazovala nulovú hodnotu pri meraní kalibračného kusa. Programovateľný analógový výstup je obmedzený na hranice -1mm až +1mm. Hranice možno upravovať, podmienkou algoritmu je symetrický interval okolo nuly. Za prevod hodnôt je zodpovedný 16b A/D prevodník. Rozlíšenie aj rozsah je potrebné zadať do algoritmu ako parameter.

Druhý prevod meranej hodnoty vyplýva z konštrukcie meradla. Meraným rozmerom je posunutie kužeľa pri zatlačení guľôčok do priemeru obežnej dráhy. Tento prevod možno popísať matematicky pomocou goniometrickej funkcie tangens. Uhol kužeľa je zapísaný do algoritmu ako konštantný parameter.

Zmerané hodnoty je potrebné porovnať s výkresovými hodnotami. Pokiaľ je rozmer mimo požadovanej tolerancie, je daná súčasť vyradená. Zároveň sa kontroluje rozdiel medzi oboma obežnými dráhami. Pokiaľ rozdiel priemerov obežných dráh presahuje povolenú hodnotu, súčasť je taktiež vyradená.

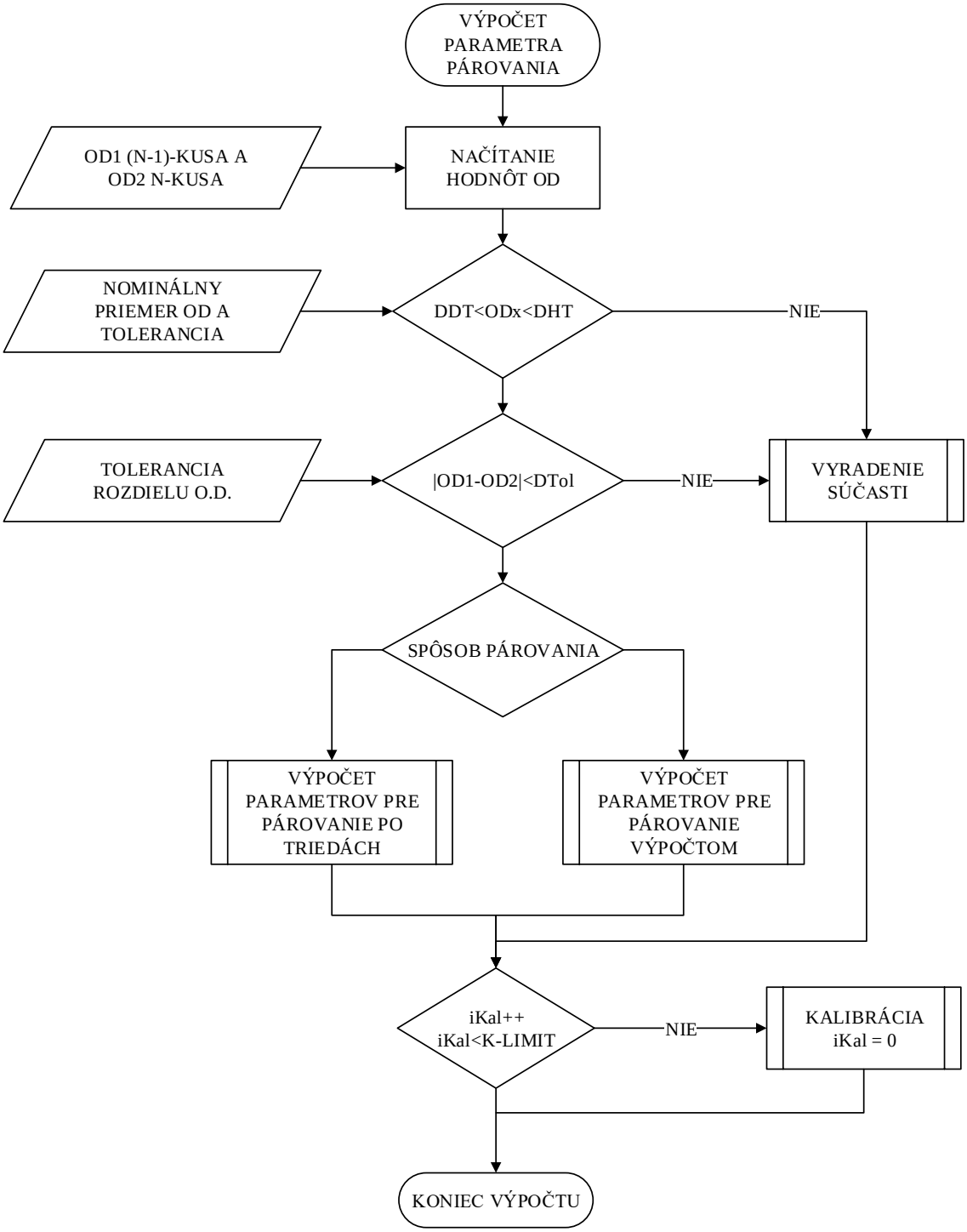
Výsledný algoritmus prepočtu nameranej hodnoty na reálnu hodnotu priemeru obežnej dráhy je zobrazený na obrázku č.27. Obrázok č.28 zobrazuje algoritmus, ktorý triedi súčasti podľa nameraných hodnôt. Tabuľky č.2 a č.3 popisujú význam jednotlivých premenných spolu s ich dátovým typom pre príslušné algoritmy zobrazené nad nimi.



Obr. 27 – Algoritmus pre prepočet analógovej hodnoty

Názov premennej	Typ premennej	Popis premennej
AnX	Real	Analógová hodnota zo snímača GT2
AnRg	Real	Rozsah analógového výstupu nastavený na komunikačnej jednotke
AnRs	Real	Rozlíšenie analógovej karty
L	Real	Hodnota posunutia na snímači GT2
Drel	Real	Relatívna hodnota priemeru
Alfa	Real	Vrcholový uhol kužeľa meradla
Dkal	Real	Hodnota priemeru O.D. kalibra
Dnom	Real	Nominálna hodnota priemeru O.D z výkresu
D	Real	Výsledná hodnota priemeru O.D.

Tabuľka 2 - Význam premenných v algoritme prepočtu analógovej hodnoty



Obr. 28 - Algoritmus predprípravy pre výpočet parametrov párovania

Názov premennej	Typ premennej	Popis premennej
OD1, OD2	Real	Priemer prvej a druhej obežnej dráhy
DT, HT	Real	Dolný a horný hraničný rozmer OD
TOL	Real	Povolená odchýlka medzi O.D.
K-Limit	Integer	Počet meraní medzi kalibráciami
i	Integer	Počítadlo zmeraných kusov

Tabuľka 3 - Význam premenných v algoritme prepočtu analógovej hodnoty

4.7.5 Kalibrácia meradiel

Meradlá pracujú porovnávacou metódou. Výstupná hodnota je reprezentovaná rozdielom medzi meranou obežnou dráhou a obežnou dráhou kalibračného dielca. Pre správnosť merania a dodržanie určitých štandardov je potrebné vykonať kalibráciu po určitom počte meracích cyklov. Počet cyklov určuje vnútropodnikový predpis. Tento parameter sa definuje v receptúrach na displeji. Počet cyklov pre rôzne montované ložiská môže byť odlišný.

4.8 Maticový zásobník

Maticový zásobník je špeciálny typ zásobníka. Dielce v takomto type zásobníka majú presne definovanú polohu, ktorá býva vyjadriteľná v niektorom type súradného systému (karteziánsky, polárny, ...).

Zakladanie a vyberanie dielov zo zásobníka býva riešené viacosími manipulátormi alebo robotmi. Maticové usporiadanie dovoľuje jednoduché definovanie polohy jednotlivých dielov. V zmysle úspory pamäte a jednoduchšiemu prenosu dát o polohe sa častokrát používa prepočet dvojrozmerného poľa do poľa jednorozmerného.

V aktuálnej aplikácii bol použitý jeden maticový zásobník pre puzdra. Jeho obsluhovanie majú na starosti dva roboty Kuka KR6R700sixx. Prvý robot R1 odoberá puzdra z výstupu meradla puzdier. Vyhovujúce puzdra zakladá do definovanej pozície v maticovom zásobníku. Vyradené puzdra odkladá do zásobníka nezhodných puzdier. Druhý robot R2 odoberá puzdra z maticového zásobníka a vkladá ich do vstupného zásobníka montážneho automatu.

Presná poloha jednotlivých pozícii je zadefinovaná v rámci riadiaceho systému robota. Roboty prijímajú čísla programov, ktoré definujú ich nasledovnú činnosť. Založenie do pozície prípadne odobratie z pozície matice je určené bitovou adresou danej pozície.

4.9 Princíp párovania s rozdeľovaním do tried

Párovanie do tried vychádza z princípu, ktorý využíva pôvodný spôsob manuálneho párovania. Hriadele i puzdra sú zaradené do rozmerovej triedy definovanej intervalom po 2µm. Tieto triedy sú následné spárované pomocou párovacích tabuliek na určitý typorozmer valivého telieska.

4.9.1 Párovacie tabuľky

Časť párovacej tabuľky je zobrazená na obrázku č.X. Párovacia tabuľka definuje k určitým typorozmerom guľôčok vhodné páry intervalov. Súčasťou každej tabuľky je aj definovaná párovacia vôľa.

GUĽA 0					GUĽA -1					GUĽA -2			
P		H			P		H			P		H	
28	30	28	30		28	30	30	32		28	30	32	34
26	28	26	28		26	28	28	30		26	28	30	32
24	26	24	26		24	26	26	28		24	26	28	30
22	24	22	24		22	24	24	26		22	24	26	28
20	22	20	22		20	22	22	24		20	22	24	26
18	20	18	20		18	20	20	22		18	20	22	24
16	18	16	18		16	18	18	20		16	18	20	22
14	16	14	16		14	16	16	18		14	16	18	20
12	14	12	14		12	14	14	16		12	14	16	18
10	12	10	12		10	12	12	14		10	12	14	16
8	10	8	10		8	10	10	12		8	10	12	14
6	8	6	8		6	8	8	10		6	8	10	12
4	6	4	6		4	6	6	8		4	6	8	10
2	4	2	4		2	4	4	6		2	4	6	8
0	2	0	2		0	2	2	4		0	2	4	6
0	-2	0	-2		0	-2	0	2		0	-2	2	4
-2	-4	-2	-4		-2	-4	0	-2		-2	-4	0	2
-4	-6	-4	-6		-4	-6	-2	-4		-4	-6	0	-2
…	…	…	…		…	…	…	…		…	…	…	…
…	…	…	…		…	…	…	…		…	…	…	…
…	…	…	…		…	…	…	…		…	…	…	…
-24	-26	-24	-26		-24	-26	-22	-24		-24	-26	-20	-22
-26	-28	-26	-28		-26	-28	-24	-26		-26	-28	-22	-24
-28	-30	-28	-30		-28	-30	-26	-28		-28	-30	-24	-26

Obr. 29 - Časť párovacej tabuľky

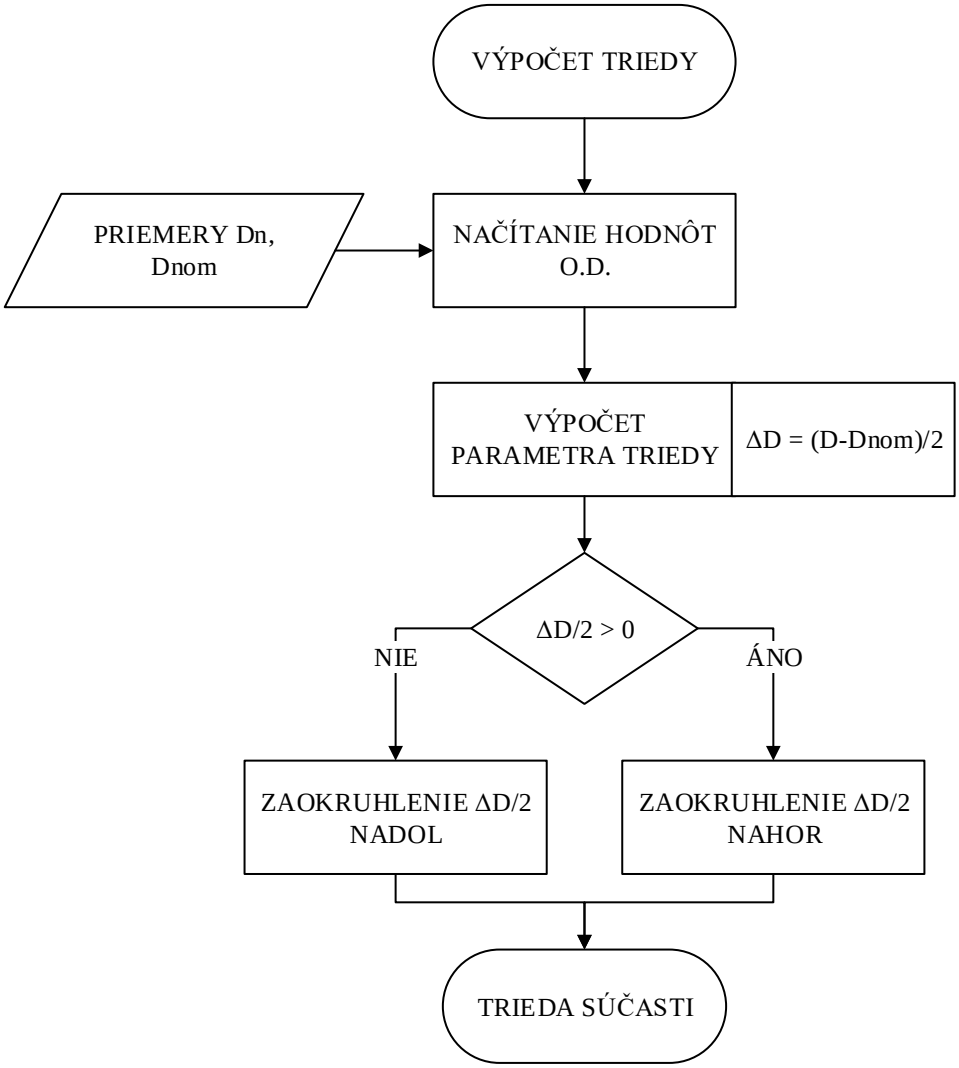
Forma párovacej tabuľky vyhovuje manuálnemu spárovaniu veľkého množstva súčastí ložiska na určitý typorozmer guľôčky. Pre účely automatického párovania bolo potrebné pristupovať k párovaniu iným spôsobom. Tabuľka bola prepísaná do tvaru, kde riadky reprezentujú jednotlivé intervaly puzdier, stĺpce intervaly hriadeľov a na ich priesečníkoch boli zapísané veľkosti guličiek. Intervaly hriadeľov i puzdier boli označené indexami. Interval $\langle 0, 2\}$ bol označený ako nultý.

Interval danej triedy je možno zapísať pomocou výrazu (8),

$$D \in < T_D; T_D + 2) \tag{8}$$

kde D predstavuje priemer obežnej dráhy v µm a T_D príslušnú triedu daného priemeru obežnej dráhy. Kvôli nesymetrickému rozdeleniu tried vzhľadom k nule je potrebné pri hľadaní danej triedy rozlišovať znamienko rozdielu medzi nominálnou

hodnotou priemeru obežnej dráhy a nameranou hodnotou priemeru obežnej dráhy. Krátky algoritmus prepočtu veľkosti priemeru obežnej dráhy na príslušnú veľkosť je zobrazený na obrázku č.30 s popisom premenných v tabuľke 5.



Obr. 30 - Výpočet triedy súčasti

Názov premennej	Typ premennej	Popis premennej
Dn	Real	Zmeraný priemer obežnej dráhy
Dnom	Real	Nominálny priemer obežnej dráhy
Trieda	Integer	Trieda súčasti

Tabuľka 4 - Význam premenných funkcie Výpočet triedy

4.9.2 Princíp výpočtu vhodného páru

Celý proces vychádza z rovnice x. Priemery obežných dráh hriadeľov i puzdier sa nahradia príslušnými triedami. Rovnako sa nahradí aj rozmer guľôčky za jej triedu. Trieda guľôčky je zhodná s veľkosťou odchýlky priemeru guľôčky od nominálneho priemeru 6,35mm vyjadrená v µm. Týmto prepisom vznikne rovnica (9). Význam jednotlivých parametrov popisuje tabuľa 6.

$$\Delta_{PP} = C_{Do} - C_{Di} - C_{Dw} \quad (9)$$

Δ_{PP}	Párovacia vôľa
C_{Do}	Trieda puzdra
C_{Di}	Trieda hriadeľa
C_{Dw}	Trieda guľôčok

Tabuľka 5 - Význam premenných rovnice x

Z charakteru celej aplikácie je potrebné vyhľadať vhodné puzdro k danému hriadeľu. Prepísaním rovnice (9) do tvaru (10) dostávame výraz, ktorý určuje triedu nami hľadaného puzdra.

$$C_{Do} = C_{Di} + C_{Dw} + \Delta_{PP} + \Delta_k \tag{10}$$

Kvôli korekcii výpočtu priamo v priebehu montáže je na pravú stranu rovnice doplnená korekčná premenná Δ_k. Jej význam je popísaný v kapitole (4.12).

4.10 Princíp párovania s nepriamym výpočtom

Nepriamy výpočet využíva rovnicu x. Namiesto priameho určovania veľkosti radiálnej vôle je k výpočtu použitá párovacia vôľa. Priemer obežnej dráhy puzdra možno vyjadriť rovnicou (11).

$$D_o = D_i + 2D_w + \Delta_{PP} + \Delta_k \tag{11}$$

Kvôli korekcii výpočtu je do rovnice pridaná korekčná premenná Δ_k.

Pri párovaní na obe obežné dráhy je potrebné nájsť puzdro v zásobníku ktoré splňuje požiadavky definované rovnicami (11). Význam parametrov je zobrazený v tabuľke 7.

$$\begin{gathered} |D_{OV1} - D_{OZ1}| < Pov\_Tol \\ |D_{OV2} - D_{OZ2}| < Pov\_Tol \\ \min_{MAT}(|D_{OV1} - D_{OZ1}| + |D_{OV1} - D_{OZ1}|) \end{gathered} \tag{12}$$

D_{OV}	Priemer obežnej dráhy ideálneho vypočítaného puzdra
D_{OZ1}	Priemer obežnej dráhy reálneho zmeraného puzdra
Pov_Tol	Povolená tolerancia medzi ideálnym a reálnym priemerom

Tabuľka 6 - Význam parametrov rovnice X

Požiadavky vyjadrujú ohraničenie hľadaných hodnôt obežných dráh. Medzi vyhovujúcimi puzdrami sa vyberie to, ktorého súčet oboch odchýlok medzi vypočítanými a nameranými priemermi je minimálny.

Pre zvýšenie presnosti párovania sa povolená tolerancia definuje pre užšie a širšie hľadanie. Pokiaľ sa nenájde puzdro pri prísnejšej tolerancii pre všetky dostupné guľôčky, hľadá sa opätovne s použitím širšej tolerancie.

Určitý spôsob zjednodušenia celého hľadania ponúka možnosť párovania na priemerovanú hodnotu oboch obežných dráh. Zníženú presnosť spôsobenú priemerovaním je možné čiastočne zvýšiť sprísnením povolených tolerancií.

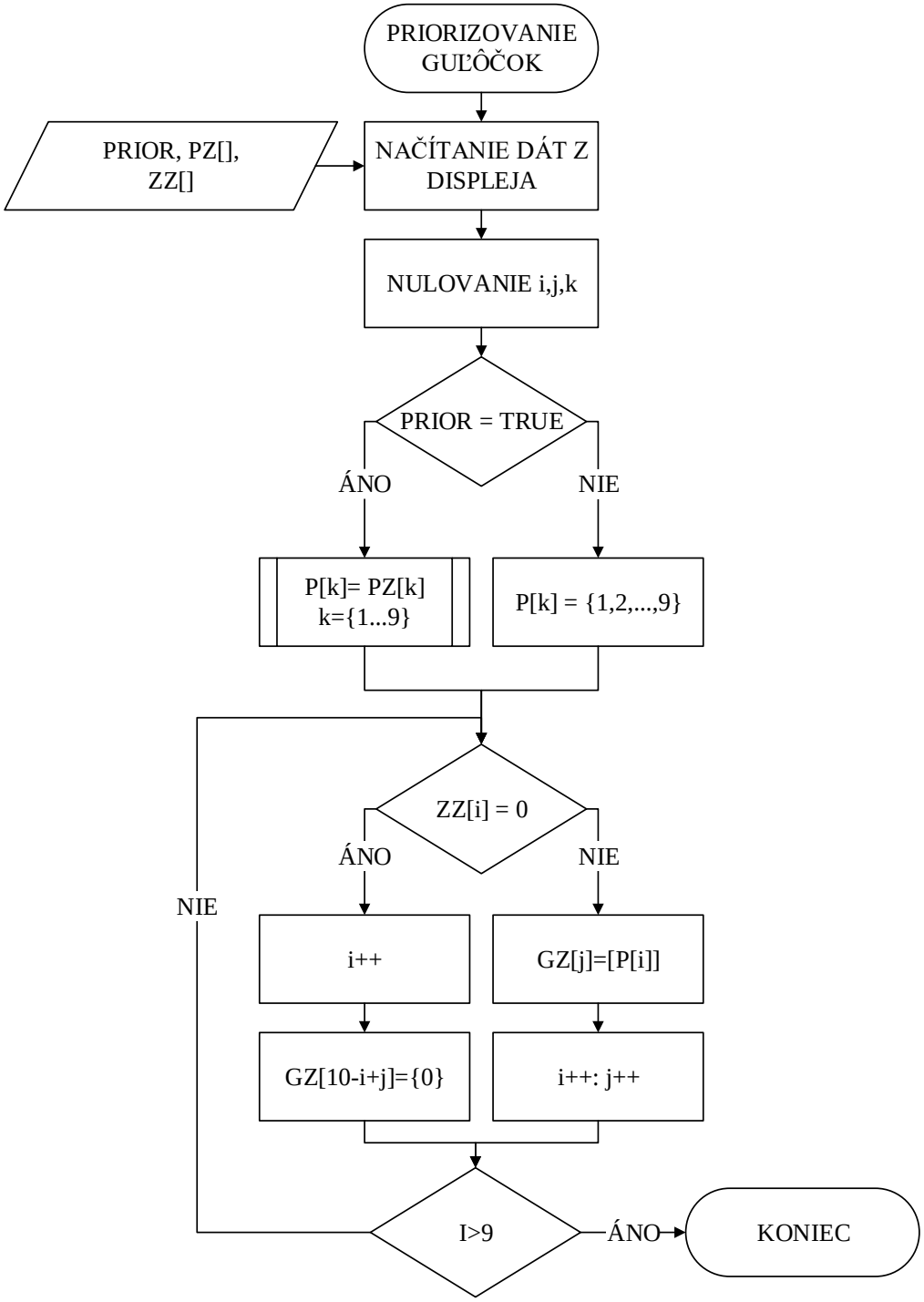
4.11 Priorizovanie zásobníkov guľôčok

Guľôčky sú do puzdier dávkované zo vstupných zásobníkov pre guľôčky. Zásobníky sú konštrukčne identické a ich zostava sa nachádza nad pracoviskom nakladania guľôčok. Počet zásobníkov pre jednotlivé pracoviská je variabilný. Ich počet však nepresahuje 9 kusov.

Zásobník nemá žiadny priamy systém pre určovanie prázdnych zásobníkov ani systém pre rozoznávanie typorozmeru guľôčok. Tieto činnosti vykonáva pracovník obsluhy pomocou displeja. Vypínanie resp. zapínanie zásobníkov sa vykonáva zatlačením priradeného tlačidla na dotykovej obrazovke. Zadávanie hodnôt typorozmeru guľôčky v danom zásobníku sa realizuje vpísaním hodnoty do poľa pod tlačidlo vypnúť/ zapnúť daný zásobník.

Pridaním pracoviska rozmeriavania sa prešlo zo systému využívania jedného zásobníka pre montáž celých zostáv spárovaných súčastí na striedanie jednotlivých zásobníkov. Z tohto dôvodu vznikla požiadavka na možnosť priorizovania jednotlivých zásobníkov. Na displej pribudla teda možnosť zapnúť (resp. vypnúť) priorizovanie zásobníkov a jednoduchý zoznam, ktorý stanovuje prioritu daného zásobníka. Bez zapnutej voľby prioritizácie je nastavená priorita zásobníkov zľava doprava. Po zapnutí voľby prioritizácie sa poradie zásobníkov prezoradí podľa voľby operátora.

Výsledkom tohto procesu je pole zapnutých zásobníkov guľôčok zoradených podľa priority. Algoritmus tohto procesu je vidno na obrázku č.31. Význam premenných použitých v algoritme je popísaný spolu s dátovým typom v tabuľke č.8.



Obr. 31 – Algoritmus priorizovania guľôčok

Názov premennej	Typ premennej	Popis premennej
i, j, k	Integer	Index pre prácu s poľom
PRIOR	Bool	Požiadavka na priorizovanie zásobníkov
PZ[]	Array of integer	Pole priorít zásobníkov z displeja
P[]	Array of integer	Neupravené pole zásobníkov
ZZ[]	Array of bool	Pole s informáciou o stave zásobníka
GZ[]	Array of integer	Pole zoradených zasobnikov

Tabuľka 7 - Význam premenných v algoritme priorizovania guľôčok

4.12 Korekcia výberu spätnou väzbou

Celé zariadenie pracuje s niekoľkými meradlami. Správnosť merania sa zaisťuje cyklickou kalibráciou. Napriek snahe o veľmi presné meranie a spracovanie dát, v systéme sa vyskytujú rôzne faktory, ktoré zapríčiňujú výskyt náhodných i systematických chýb. V prípade výskytu stálej chyby je potrebné výpočet párovania korigovať.

Korekcia sa vykonáva pomocou korekčnej premennej, ktorá priamo ovplyvňuje výpočet veľkosti vhodného priemeru obežnej dráhy puzdra. Korekčná premenná sleduje trend zmeraných radiálnych vôlí. Pokiaľ sa priemerná veľkosť radiálnej vôle posledných n-kusov priblíži k hraničnej hodnote a rozptyl nepresiahne 10% povoleného intervalu radiálnej vôle, program vhodne upraví korekčnú premennú. Pri malých radiálnych vôľach nadobúda kladné hodnoty, pri veľkých zase zápornú. Táto funkcionalita je spravovaná pomocou displeja, ktorý umožňuje efektívne spravovanie štatistických údajov. Korekčná premenná nevstupuje do riadiaceho systému pracoviska, ale priamo upravuje veľkosť párovaciej vôle, ktorá je predávaná ako jeden z parametrov ložiska.

Vzhľadom na vzdialenosť medzi výberom puzdra a meraním radiálnej vôle (viac ako 10 pracovných pozícii) je medzi upravením korekčnej premennej a viditeľnou zmenou určité časové zdržanie. Z tohto dôvodu je medzi zmenami korekčnej premennej definovaný určitý minimálny počet pracovných cyklov.

Korekčná premenná je schopná korigovať len systematické chyby. Tie vznikajú napríklad zmenou okolitej teploty alebo prechodom na iný typ montovaného ložiska.

Automatickú korekciu je možné vypnúť z displeja alebo manuálne zadať hodnotu korekčnej premennej. Zároveň pokiaľ zmena korekčnej premennej nepomáha odstraňovať systematickú chybu, je celý montážny proces zastavený a je vyžiadaná kalibrácia meradiel. Pokiaľ sa nájde vážna chyba, je potrebné vyresetovať i maticový zásobník a vrátiť už zmerané diely na začiatok procesu.

4.13 Triedenie puzdier v zásobníku

Pre účely párovania je potrebné puzdra v zásobníku radiť podľa veľkosti. Podľa spôsobu párovania je možné použiť viaceré možností. Pre všetky možnosti platia určité všeobecné pravidlá :

- Maximálny počet puzdier aktuálnych zariadení je 169. Algoritmus musí byť ľahko modifikovateľný pre iný počet puzdier do budúcnosti.
- Triedenie začína od prvého kusa. Nové kusy sa vkladajú do už zoradeného zoznamu.
- Pracovať s dátami zásobníka môže v jednom okamžiku len jeden program.

V nasledujúcich podkapitolách sa vyskytujú výrazy ako posunutie doľava resp. doprava. Posunutie doľava v poli znamená zníženie indexu pozície o 1, posunutie doprava zase zvýšenie indexu pozície o 1.

4.13.1 Zaraďovanie puzdier pri párovaní s rozdeľovaním do tried

Charakteristickým údajom popisujúcim priemer obežnej dráhy je príslušná trieda. Tento parameter môže nadobúdať jednu z 30 rôznych hodnôt. Algoritmus párovania považuje puzdra rovnakej triedy za rovnocenné. Pre párovanie je potrebné ku každému puzdru prideliť triedu prvej a druhej obežnej dráhy. Poloha puzdra v maticovom zásobníku zodpovedá indexu v poli puzdier. Triedenie vzostupne prebieha nad poľom smerníkov na pole puzdier.

Zoznam tried

Jednou z možností je vytvorenie samostatných polí pre jednotlivé kombinácie tried. Každé pole by obsahovalo smerníky na puzdra z poľa puzdier patriacich do danej rozmerovej kombinácie tried. Počet možných kombinácii je limitovaný podmienkou maximálneho rozdielu medzi veľkosťami priemerov obežných dráh stanovený na 3µm. Toto obmedzenie dovoľuje len kombinácie n-tej triedy prvej obežnej dráhy s triedou druhej obežnej dráhy z intervalu $\langle n-2, n+2\rangle$. Tento prístup vytvára $(5n-6)$ rôznych tried, v našom prípade 144. Vzhľadom na statický spôsob alokovania pamäte, nie je možné meniť veľkosti polí za chodu programu. Z tohto dôvodu by bolo potrebné alokovať pamäť pre všetky kombinácie. Využitie pamäte v našom prípade nepresahuje 1%.

Zjednodušenie metódy prináša úprava charakteristického rozmeru puzdra z dvojice tried na triedu priemernej hodnoty priemerov obežných dráh. Počet možných kategórii sa zmenší na počet n, v našom prípade 30.

Metóda zoznamu tried umožňuje pristupovať k jednotlivým poliam spôsobom FIFO. Tým je možné využívať puzdra z danej kategórie, ktoré sú založené v zásobníku najdlhšie. Tento prístup lepšie rozkladá využitie jednotlivých pozícii, čím zvyšuje životnosť celého zásobníka.

Nevýhodou danej metódy je zlá adaptovateľnosť. Pri zmene povolených tolerancií sa veľkosti a počty zoznamov menia. Potrebné úpravy treba vykonať v programe, ktorý je potrebné opätovne nahrať do riadiaceho systému.

Zhodnotenie metódy je zosumarizované v tabuľke 9.

Výhody	Nevýhody
Rýchle kategorizovanie puzdier	Vysoké nároky na pamäť
Bez potreby triedenia celého zásobníka	Nízke využitie alokovanej pamäte
Informácia o poradí založenia danej triedy	Nízka adaptovateľnosť na zmeny

Tabuľka 8 - Zhodnotenie metódy „Zoznam tried“

Hraničné pole

Ďalšou možnosťou triedenia je využitie poľa s údajmi o hraniciach jednotlivých tried. Špeciálne pole, nazvané hraničné, obsahuje na jednotlivých pozíciách informáciu o polohe prvého prvku danej triedy v druhom poli. Existuje niekoľko možností implantácie tohto prístupu. Väčšina spôsobov počíta s 3 poľami. Základné pole obsahuje informácie o veľkosti puzdra na danej pozícii. Nad týmto poľom je zoradené pole

smerníkov. A nad poľom smerníkov je pole určujúce hraničné miesta prechodov medzi jednotlivými triedami.

Táto metóda predstavuje určitý typ dynamicky sa meniacej veľkosti polí predchádzajúcej metódy. Nevýhodou je nutnosť sledovať obsadenosť pozícii v zásobníku. Toto je možné docieliť sekundárnym poľom booleovských premenných.

Výhody i nevýhody popísanej metódy sú zosumarizované v tabuľke 10

Výhody	Nevýhody
Možnosť počítania prvkov daných tried	Vyššia réžia výpočtov a posunutí
Rýchle založenie puzdra	Len pre priemerovanú verziu výpočtu
	Zložitá implementácia

Tabuľka 9 - Zhodnotenie metódy „Hraničné pole“

Upravené spätné bublinové triedenie

Algoritmus bublinového triedenia je jeden z najznámejších triediacich algoritmov. Algoritmus prechádza pole prvok po prvku a porovnáva susediace dvojice. Pokiaľ nie sú dané dva prvky v požadovanom poradí, algoritmus ich navzájom vymení. Napriek tomu, že daný algoritmus je veľmi jednoduchý, celá metóda je pomalá a neefektívna.[22] Pre možnosť zakladania sa využíva jeho upravená verzia.

Algoritmus pracuje s dvoma poľami. Prvé pole predstavuje maticový zásobník. Index pozície je zhodný s polohou v zásobníku. Údaj na danej pozícii predstavuje triedu založenej súčasti. Druhé pole predstavuje pole smerníkov na pole prvé. Tie sú zoradené vzostupne podľa triedy súčasti, na ktorú ukazujú. Počet založených prvkov označíme premennou „n“. Potom prvých n pozícii poľa smerníkov ukazuje na zoradené puzdra a zvyškové pozície predstavujú voľné miesta v maticovom zásobníku. Algoritmus založí nové puzdro do zásobníka podľa adresy na n+1 pozícii v poli smerníkov. Následne prechádza jednotlivými pozíciami poľa smerníkov smerom doľava. Postupne posúva puzdra s menšou triedou ako má puzdro zakladané smerom doprava. Keď algoritmus nájde vhodnú pozíciu, začne od tohto miesta s vyhľadávaním vhodnej pozície podľa triedy druhej obežnej dráhy. Zhodnotenie metódy je zobrazené v tabuľke 11.

Výhody	Nevýhody
Dostupná poloha založenia puzdra	Zložitejšia implementácia
Automatické posunutie prvkov doprava	
Informácia o voľných pozíciách	

Tabuľka 10 - Zhodnotenie metódy „Upravené spätné bublinové triedenie“

Vkladanie pomocou binárneho vyhľadávania

Základom metódy je delenie intervalu na dve polovice. Jeho funkčnosť je obmedzená na zoradené prvky. Algoritmus pracuje metódou rozdeľ a panuj. Porovnáva stredný prvok zoradeného poľa s hľadaným. Pokiaľ stredný prvok nezodpovedá hľadanému, algoritmus podľa rozdielu medzi hľadaným a stredným prvkom volí jednu

polovicu pôvodného intervalu. Následne opakuje celý proces, pokiaľ nenájde zhodu alebo nevynuluje vyhľadávací interval. [22]

Pre účely vkladania nového prvku do zoradeného poľa sa najprv vyhľadá pozícia kde je daný prvok potrebné založiť. Po nájdení vhodnej pozície pre prvú obežnú dráhu sa od tohto miesta začne s hľadaním vhodnej pozície pre druhú obežnú dráhu. Nový prvok sa založí na vhodnú pozíciu a všetky prvky po pravej strane sa posunú o jednu pozíciu doprava.

Výhody i nevýhody sú spísané v tabuľke 12.

Výhody	Nevýhody
Rýchle zaradenie do poradia	Žiadne doplňujúce informácie
Efektivita rastúca s počtom puzdier	Zložitejšia implementácia

Tabuľka 11 - Zhodnotenie metódy „Vkladanie pomocou binárneho vyhľadávania“

4.13.2 Zaraďovania puzdier pri párovaní s nepriamym výpočtom

Pri zakladaní puzdra do maticového zásobníka je charakteristickým rozmerom veľkosť obežnej dráhy zaokrúhlená na desatinu mikrometra. Pre párovanie je potrebné ku každému puzdru prideliť veľkosť prvej a druhej obežnej dráhy a polohu v zásobníku.

Metódy Zoznam tried a Hraničné pole použiteľné pri párovaní s rozdeľovaním do tried je možné implementovať i do tejto metódy párovania. Avšak počet tried narastie na 600 a viac, čím sa dané metódy stávajú nepoužiteľné.

Metódy Upravené spätné bublinové triedenie a Vkladanie pomocou binárneho vyhľadávania je možné použiť s minimálnymi úpravami.

Pre zakladanie puzdra do zásobníka bola vybratá metóda spätného bublinového triedenia. Hlavným dôvodom bola jej univerzálnosť, teda možnosť využitia pre oba spôsoby párovania bez potreby úprav.

4.14 Výber puzdier z maticového zásobníka

Po zmeraní hriadeľa a spočítaní parametrov ideálneho puzdra pre danú guľôčku je potrebné nájsť v maticovom zásobníku vhodné puzdro. Spôsob vyhľadávania závisí na spôsobe výpočtu ideálneho puzdra a spôsobe zaraďovania nového puzdra do zásobníka.

Netreba zabúdať, že pokiaľ nie je nájdené vhodné puzdro pre prvý rozmer guľôčky, je potrebné hľadať opakovane pre ďalší rozmer guľôčky.

4.14.1 Výber puzdier pri párovaní s rozdeľovaním do tried

Pri použití metódy s rozdeľovaním do tried je potrebné nájsť presnú kombináciu tried obežných dráh. Pokiaľ sa využije zjednodušenie priemerovaním tried oboch obežných dráh, postačuje nájsť požadovanú triedu. Možnosti hľadania požadovaného puzdra vychádzajú zo spôsobu zatriedenia nového puzdra do zásobníka. Všetky metódy končia hľadanie po nájdení prvého vhodného puzdra.

Lineárne vyhľadávanie

Algoritmus lineárneho vyhľadávania je veľmi jednoduchý. Prechádza postupne po jednotlivých prvkoch poľa a testuje ich zhodu s hľadaným prvkom. Výhodou tohto prístupu je jednoduchá implementácia a fakt, že puzdra v zásobníku nemusia byť zoradené. I keď pri metóde s rozdeľovaním do tried postačuje nájsť prvé vhodné puzdro, v najhoršom prípade (žiadna zhoda) je potrebné prejsť celý zásobník 9x (podľa počtu zásobníkov guľôčok). Pokiaľ sú puzdrá zoradené, je možné ukončiť vyhľadávanie skôr.

Vyhľadávanie zo zoznamu tried

Pokiaľ je maticový zásobník roztriedený do zoznam podľa jednotlivých tried, postačuje pri vyhľadávaní vhodného puzdra vybrať prvé puzdro z daného zoznamu. Pokiaľ také puzdro neexistuje ani pre jeden typorozmer dostupnej guľôčky, hriadeľ sa vylúči ako aktuálne nepárovateľný. Tento spôsob je veľmi efektívny. Jeho nevýhody spočívajú skôr v systéme zaraďovania nových puzdier do zásobníka.

Vyhľadávanie pomocou hraničného poľa

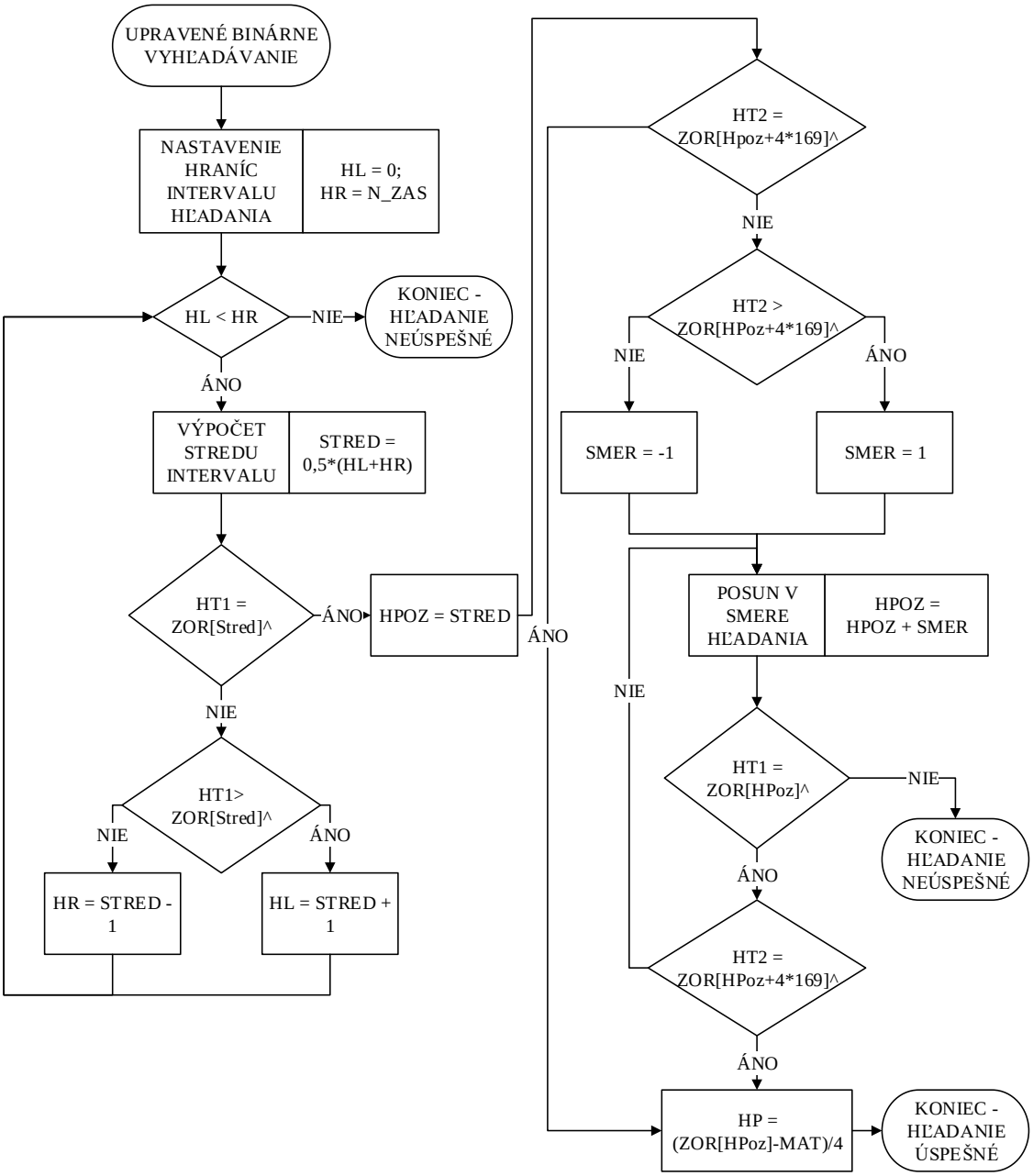
Podľa triedy ideálneho puzdra sa v hraničnom poli zistí poloha prvého puzdra. Pokiaľ je hodnota nulová pre všetky typorozmery guľôčky, vhodné puzdro neexistuje a hriadeľ sa vylúči ako aktuálne nepárovateľný. V opačnom prípade sa zoberie hodnota polohy puzdra, ktoré sa odoberie. Táto metóda je použiteľná len pre párovanie na priemer tried obežných dráh. Pre kombináciu tried vytvára veľké množstvo prvkov v hraničnom poli.

Binárne vyhľadávanie

Binárne vyhľadávanie využíva algoritmus delenia intervalu popísaný v predchádzajúcej kapitole. Veľkou výhodou pri vyhľadávaní ideálneho puzdra je rýchlosť. Vhodné puzdro je nájdené po niekoľkých krokoch a posúvanie pravej strany zoradeného poľa puzdier doľava môže prebiehať paralelne s prekladaním puzdra z maticového zásobníka do montážneho automatu. Algoritmus tohto postupu je zobrazený na obrázku č.32. Význam premenných s ich dátovými typmi je zobrazený v tabuľke 13.

Názov premennej	Typ premennej	Popis premennej
HL, HR	Integer	Hranice intervalu
N_ZAS	Integer	Počet kusov v zásobníku
STRED	Integer	Stred intervalu vyhľadávania
HT1, HT2	Integer	Hľadané triedy OD1 a OD2
MAT[]	Array of Real	Pole s informáciou o triede OD1 a OD2
ZOR[]	Array of Pointer	Zoradené pole smerníkov
HPOZ	Integer	Index pozície v maticovom zásobníku
SMER	Integer	Smer hľadania
HP	Integer	Index pozície vhodného puzdra

Tabuľka 12 - Význam premenných algoritmu podľa obr.32



Obr. 32 – Algoritmus hľadania puzdra pri párovaní po triedach

4.14.2 Výber puzdier pri párovaní s nepriamym výpočtom

Pri vyhľadávaní puzdra pri metóde nepriameho výpočtu sa postupuje odlišne. Je potrebné nájsť puzdro, ktorého odchýlka medzi spočítanou ideálnou obežnou dráhou a nameranou je minimálna a nepresahuje povolenú toleranciu. Pri párovaní na obe obežné dráhy sa hľadá minimálny súčet oboch odchýlok. Pre tento účel je možné použiť niekoľko algoritmov.

Lineárne vyhľadávanie

Algoritmus prechádza maticový zásobník jedno puzdro za druhým. Pre vhodné puzdro spočíta súčet rozdielov medzi obežnými dráhami. Algoritmus je možné vypracovať vo viacerých alternatívach v závislosti na podmienke jeho ukončenia. Vhodná alternatíva sa môže voliť obsluhou. Algoritmus sa môže ukončiť pri nájdení

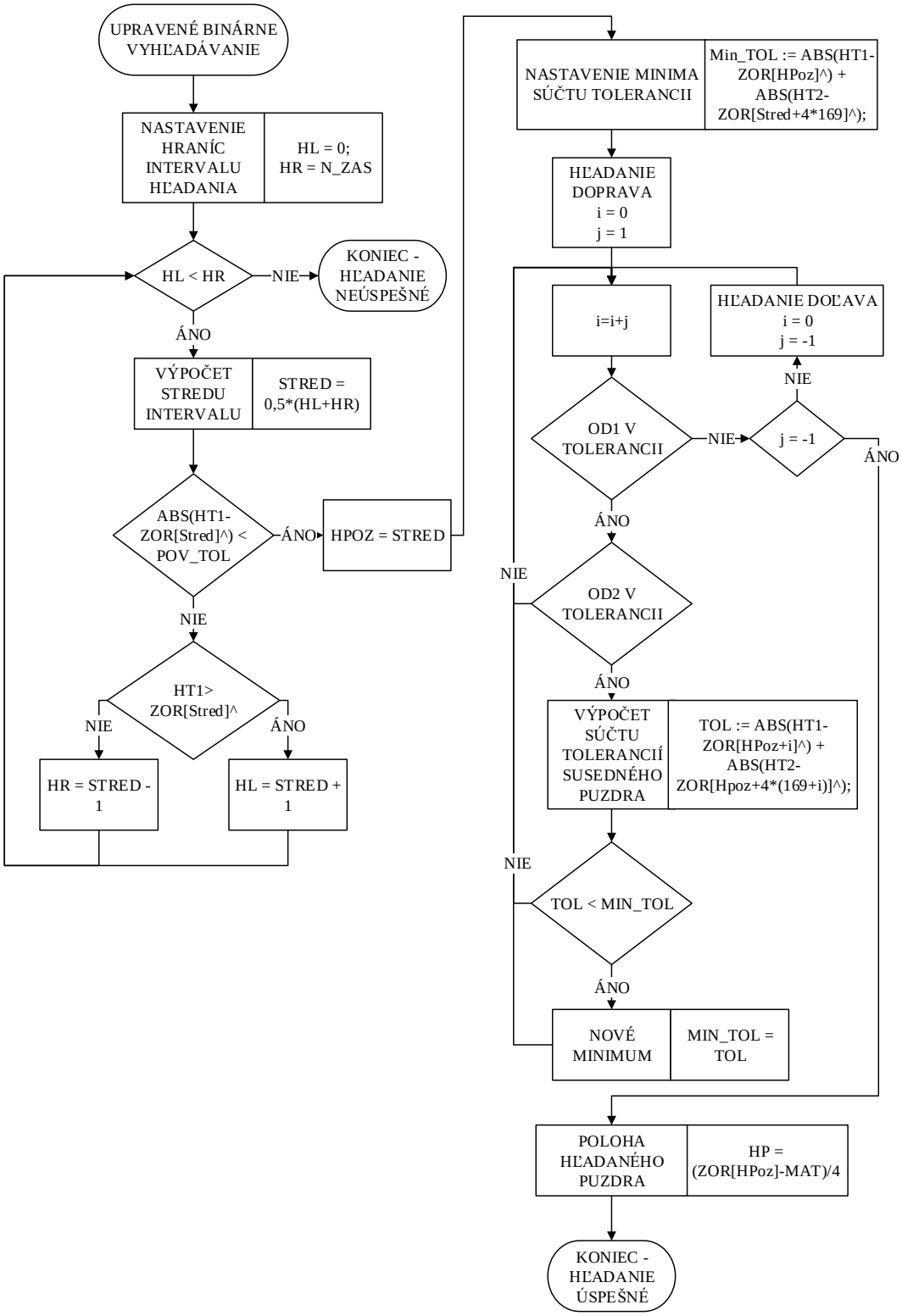
prvého vhodného puzdra alebo po úspešnom prehľadaní maticového zásobníka pre jeden typorozmer guľôčky alebo po prehľadaní maticového zásobníka pre všetky typorozmery guľôčok. Jednotlivé alternatívy majú svoje klady a zápory hlavne z časovej a párovacej efektivity.

Binárne vyhľadávanie

Algoritmus vyhľadá polohu ideálneho puzdra v zásobníku podľa prvej obežnej dráhy. Pre vhodné puzdra v okolí počíta súčet odchýlok puzdier v najbližšom okolí. Výpočet končí v momente, keď sa dostane priemer prvej obežnej mimo tolerancie. Rovnako ako v predchádzajúcom prípade, je možné zvoliť podobné podmienky ukončenia vyhľadávania. Algoritmus vyhľadávania touto metódou je zobrazený na obrázku č.33.

Názov premennej	Typ premennej	Popis premennej
HL, HR	Integer	Hranice intervalu
N_ZAS	Integer	Počet kusov v zásobníku
STRED	Integer	Stred intervalu vyhľadávania
HT1, HT2	Integer	Hľadané triedy OD1 a OD2
MAT[]	Array of Real	Pole s informáciou o triede OD1 a OD2
ZOR[]	Array of Pointer	Zoradené pole smerníkov
HPOZ	Integer	Index pozície v maticovom zásobníku
SMER	Integer	Smer hľadania
HP	Integer	Index pozície vhodného puzdra

Tabuľka 13 - Význam premenných algoritmu podľa obr.33



Obr. 33 - Algoritmus hľadania puzdra pri párovaní po triedach

5 IMPLEMENTÁCIA ALGORITMU V CODESYSE

Celý program pre riadenie párovania pozostáva z viac ako 40 menších podprogramov logických rozdelených podľa potrebných funkcií. Podprogramy pre ovládanie pohybov, komunikáciu medzi zariadeniami či bezpečnostné podprogramy sú z tejto kapitole vynechané. Pre ich napísanie boli použité jazyky FBD a ST. Pre samotné podprogramy párovania bol použitý jazyk ST. Jazyk bol zvolený pre lepšiu prácu so slučkami.

5.1 Deklarácia premenných

Program Codesys umožňuje deklaráciu globálnych premenných do špeciálnych zoznamov označovaných ako GVL (Global Variable List – zoznam globálnych premenných). Pre lepšiu orientáciu je typické vytvorenie viacerých zoznamov. Logicky sú rozdelené na vstupné premenné, výstupné premenné, premenné pre komunikáciu s displejom, premenné získané z receptúr a všeobecné premenné.

Pre účely uchovávania hodnôt pri výpadkoch, reštartoch alebo iných situáciách, ktoré prerušia napájanie riadenia a tým aj zmažú dočasnú pamäť sú dôležité premenné deklarované ako remanentné. Medzi tieto hodnoty patria zmerané hodnoty obežných dráh puzdier uložených v maticovom zásobníku, nastavenia pre meradlá, parametre potrebné pre výpočet priemeru obežných dráh, všeobecné nastavenia zariadenia a štatistické hodnoty.

Jednotlivé podprogramy popísané nižšie v tejto kapitole využívajú častokrát tie isté premenné. Na obrázku č.34 je vidno zoznam týchto premenných aj s komentovaným významom.

```
Prior       : BOOL;                          // Požiadavka na priorizovanie poradia
PZ [ ]      : ARRAY[1..9] OF INT;            // Pole priorít podľa zadania z displeja
ZZ [ ]      : ARRAY[1..9] OF BOOL;           // Pole zapnutých zásobníkov
Povol_Zas   : BOOL;                          // Povolenie prace so zásobníkom
MAT [ ]     : ARRAY [1..2,1..169] OF INT;    // Pole OD1, OD2
ZOR [ ]     : ARRAY [1..169] OF POINTER TO REAL;      // Zoradené pole MAT
N_Zas       : INT;                           // Počítadlo kusov v matici
iKal        : INT;                           // Počítadlo pre kalibráciu
HP          : INT;                           // Index polohy vhodného puzdra
```

Obr. 34 - Zoznam globálnych premenných

5.2 Zadávanie parametrov montovaného ložiska

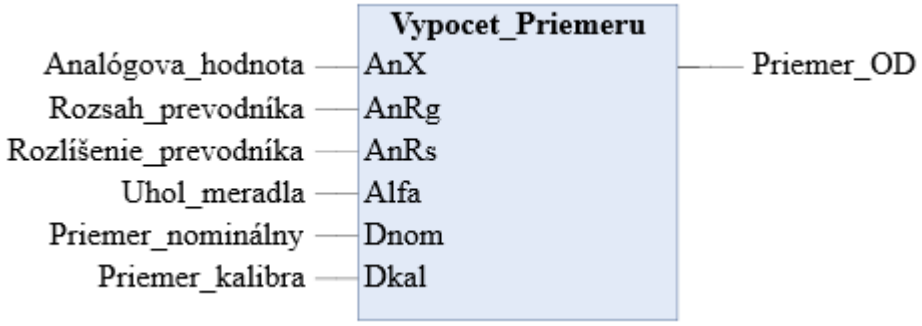
Každé montované ložisko predstavuje viac ako 20 rôznych parametrov potrebných pre správne fungovanie rozmeriavania. Okrem typového označenia ložiska je potrebné zadefinovať výkresové hodnoty, namerané hodnoty z kalibračných dielcov, program pre kamerovú kontrolu orientácie puzdra, maximálnu povolenú odchýlku medzi rozmermi dráh a ďalšie. Tieto hodnoty sa ukladanú do receptúr spravovaných displejom. Parametre zvoleného ložiska sa stlačením tlačidla na obrazovke nahrajú do riadiaceho systému, kde ostávajú v remanentnej pamäti pokiaľ nie je zvolený iný recept, teda typ montovaného ložiska. Na obrázku č.35 je zobrazený zoznam parametrov slúžiacich ako vstupy do procesu párovania. Súbor týchto parametrov je unikátny pre každý typ montovaného ložiska.

```
Typ_L       : STRING;   // Typ ložiska
Puzdro      : STRING;   // Číslo výkresu puzdra
Hriadel     : STRING;   // Číslo výkresu hriadeľa

H_Dnom      : REAL;     // Nominálny priemer OD hriadeľa
H_DTol      : REAL;     // Dolná tolerancia priemeru OD hriadeľa
H_HTol      : REAL;     // Horná tolerancia priemeru OD hriadeľa
P_Dnom      : REAL;     // Nominálny priemer OD puzdra
P_DTol      : REAL;     // Dolná tolerancia priemeru OD puzdra
P_HTol      : REAL;     // Horná tolerancia priemeru OD puzdra
DTol        : REAL;     // Povolená odchýlka medzi dráhami

H_Kal1      : REAL;     // Priemer OD1 kalibra hriadeľa
H_Kal2      : REAL;     // Priemer OD2 kalibra hriadeľa
P_Kal1      : REAL;     // Priemer OD1 kalibra puzdra
P_Kal2      : REAL;     // Priemer OD2 kalibra puzdra

PV1         : REAL;     // Užšia párovacia vôľa
PV2         : REAL;     // Širšia párovacia vôľa
KLimit      : INT;      // Počet kusov medzi kalibráciami
PrgKamera   : INT;      // Číslo programu v kamere
```

Obr. 35 - Premenné získané z receptúry

5.3 Výpočet priemeru obežnej dráhy

Vstupnou hodnotou pre výpočet priemeru obežnej dráhy je analógová hodnota získaná zo snímača GT2. Ďalšími vstupnými hodnotami sú parametre prevodníka a výkresové údaje súčasti ložiska. Grafické zobrazenie vstupov a výstupov je zobrazené na obrázku č.36. Ľavá strana predstavuje potrebné vstupy a pravá strana jediný výstup. Program je riešený pomocou funkcie, ktorá je zobrazená na obrázku č.37. Funkcia je volaná vždy po ukončení merania danej obežnej dráhy.



Obr. 36 - Vstupy a výstupy funkcie pre výpočet priemeru

DEKLARÁCIA

```
FUNCTION Vypocet_Priemeru : REAL
VAR_INPUT
        AnX     : REAL;         // Analógová hodnota zo snímača
        AnRg    : REAL;         // Rozsah analógového výstupu nastavený na kom. jednotke
        AnRs    : REAL;         // Rozlíšenie analógovej kom. jednotky
        Alfa    : REAL;         // Vrcholový uhol meracieho kužeľa [rad]
        Dnom    : REAL;         // Nominálny priemer obežnej dráhy
        Dkal    : REAL;         // Priemer obežnej dráhy kalibračného kusa
END_VAR
VAR
        Drel    : REAL;         // Relatívna hodnota priemeru
        L       : REAL;         // Posunutie meradla
END_VAR
```

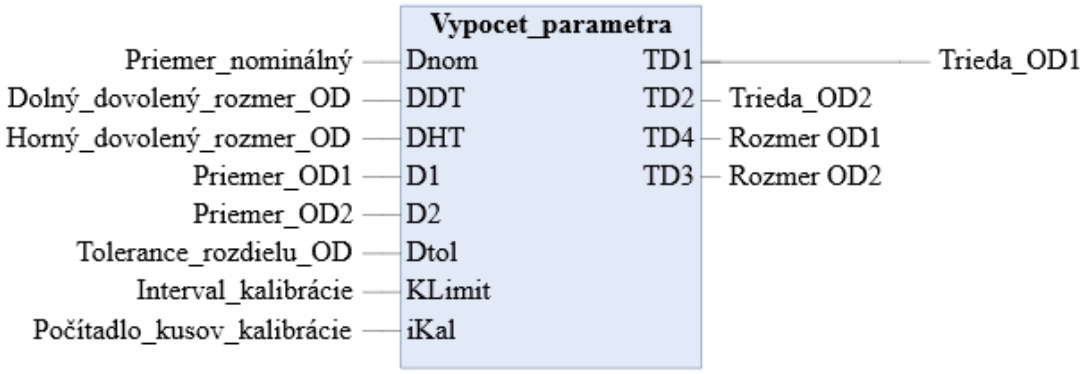
PROGRAMOVÁ ČASŤ

```
IF AnRs > 0 THEN        // Vylúčenie delenia nulou
        L := (AnRg/AnRs)*AnX - (AnRg/2);            // Posunutia meradla
        Drel := 2*L*TAN(Alfa/2);                    // Relatívny priemer
        Vypocet_Priemeru := Drel + (Dkal - Dnom);   // Absolútny priemer
ELSE
        Vypocet_Priemeru := -1;                     // Vracia -1 pri delení nulou
END_IF
```

Obr. 37 - Funkcia pre výpočet priemeru OD

5.4 Výpočet parametrov pre párovanie

Podprogram pre výpočet parametrov pre párovanie vytrieďuje nevhodné súčasti a pre vhodné zapisuje potrebné parametre pre párovanie. Grafické znázornenie potrebných vstupov a výstupov reprezentuje obrázok č.38. Podprogram vykonávajúci výpočet len tolerančne vhodných kusov je zobrazený na obrázku č.39.



Obr. 38 - Vstupy a výstupy programu pre výpočet parametra

DEKLARÁCIA

```
PROGRAM Vypocet_parametra
VAR_INPUT
        D1          : REAL;          // Zmeraný priemer OD1
        D2          : REAL;          // Zmeraný priemer OD2
END_VAR
VAR_OUTPUT
        TD1         : INT;           // Trieda OD1
        TD2         : INT;           // Trieda OD2
        TD4         : REAL;          // Rozmer OD1
        TD3         : REAL;          // Rozmer OD2
END_VAR
```

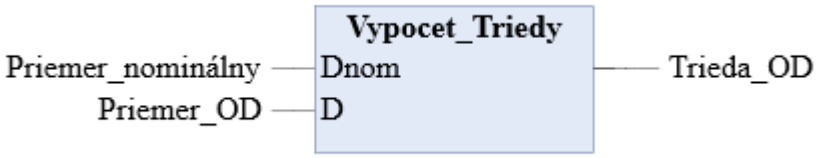
PROGRAMOVÁ ČASŤ

```
// Kontrola OD voči toleranciám
IF (DDT < D1) AND (D1 < DHT) AND (DDT < D2) AND (D2 < DHT)
AND ((ABS(D1-D2))<Dtol) THEN
        // Párovanie s rozdeľovaním do tried
        TD1 := Vypocet_triedy(Dnom, D1);          // Výpočet triedy prvej OD
        TD2 := Vypocet_triedy(Dnom, D2);          // Výpočet triede druhej OD
        // Párovanie s nepriamym výpočtom
        TD3 := D1;                                // OD1
        TD4 := D2;                                // OD2
        // D1 a D2 vstupujú priamo ďalej ako parametre párovania
ELSE                            // Niektorý rozmer mimo tolerancie
        Vyradenie_kusa( );      // Volanie podprogramu pre vyradenie kusa
END_IF

iKal:=iKal+1;                   // Počítadlo kusov pre kalibráciu
IF iKal>KLimit THEN             // Potreba kalibrácie po x-kusoch
        iKal:=0;                // Nulovanie počítadla kusov
        Kalibracia( );          // Volanie podprogramu pre kalibráciu
END_IF
```

Obr. 39 - Podprogram pre výpočet parametrov

5.5 Výpočet triedy obežnej dráhy

Určenie triedy obežnej dráhy je vykonané pomocou funkcie. Vstupy a výstupy funkcie sú graficky zobrazené na obrázku č. 40 a jej vnútro je zobrazené na obrázku č.41.



Obr. 40 - Vstupy a výstupy funkcie pre výpočet triedy

DEKLARÁCIA

```
FUNCTION Vypocet_Triedy : INT
VAR_INPUT
        Dnom    : REAL;         // Nominálny priemer OD
        D       : REAL;         // Zmeraný priemer OD
END_VAR
```

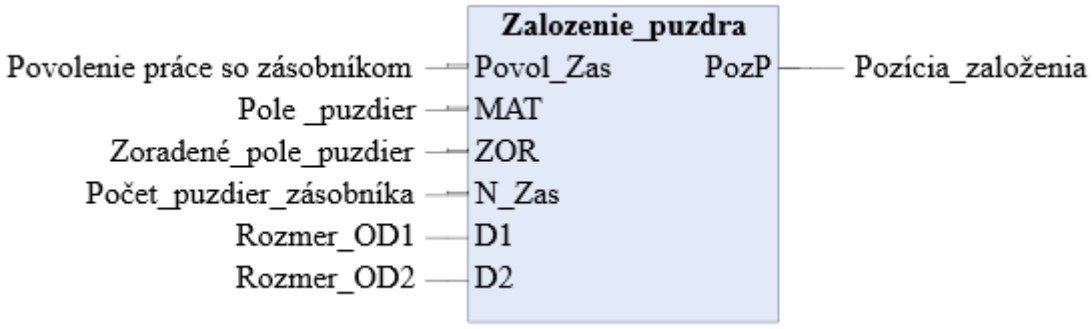
PROGRAMOVÁ ČASŤ

```
// VÝPOČET TRIEDY OBEŽNEJ DRÁHY
IF D >= Dnom THEN       // kladný rozdiel - zaokrúhlenie nadol
                Vypocet_triedy := REAL_TO_INT((D-Dnom)/2);
        ELSE            // záporný rozdiel - zaokrúhlenie nahor
                Vypocet_triedy := REAL_TO_INT((Dnom-D)/2)+1;
END_IF
```

Obr. 41 - Funkcia výpočtu triedy OD

5.6 Zaradenie puzdra do zásobníka

Podprogram pre založenie puzdra založí puzdro na prvú voľnú pozíciu do zásobníka. Následne pre prvý parameter OD prechádza zoradeným polom smerníkov od posledného založeného puzdra a hľadá vhodnú pozíciu. Po každom neúspešnom hľadaní sa posunie o krok doľava. Keď nájde vhodnú pozíciu, pokračuje v hľadaní od tohto miesta pomocou druhého parametra OD. Následne zatriedi adresu puzdra do poradia. Vstupy a výstupy tohto podprogramu sú zobrazené na obrázku č.42 a samotný program je zobrazený na obrázku č.43.



Obr. 42 - Vstupy a výstupy programu pre zaradenie puzdra do zásobníka

DEKLARÁCIA

```
PROGRAM Zalozenie_puzdra
VAR
        i       : INT;                  // Pomocná premenná
        j       : INT;                  // Pomocná premenná
        Poz     : POINTER TO REAL;      // Pomocný smerník
END_VAR
VAR_INPUT
        D1      : REAL;                 // Rozmer OD1
        D2      : REAL;                 // Rozmer OD2
END_VAR
VAR_OUTPUT
        PozP    : INT;                  // Pozícia v zásobníku pre založenie puzdra
END_VAR
```

PROGRAMOVÁ ČASŤ

```
// PROGRAM PRE ZALOZENIE PUZDRA DO MATICE
IF Povol_Zas THEN                       // Kontrola ci sa zásobník pravé nepoužíva
        Povol_Zas := FALSE;             // Zablokovanie prace so zásobníkom pre iné procesy
        Poz := ZOR[N_ZAS+1];            // Skopírovanie adresy prvej voľnej pozície
        PozP := (POZ - MAT) /4;         // Index pozície pre založenie
FOR i:=N_ZAS+1 TO 1 BY -1 DO            // Spätný cyklus od prvej voľnej pozície
        IF ZOR[i-1]^ <= D1 THEN         // Vhodná pozícia pre OD1
                j := i;
                WHILE ZOR[i-1]^ = ZOR[j-1]^ DO          // OD1 bez zmeny
                        IF ZOR[j+168*4]^ <= D2 THEN     // Overenie pozície pre OD2
                                EXIT;
                        ELSE
                                ZOR[j] := ZOR[j-1];     // Posunutie prvku doprava
                                j := j-1;
                        END_IF
                END_WHILE
                        ZOR[i] := Poz;          // Umiestnenie prvku do poradia
                        EXIT;                   // Ukončenie FOR slučky
        ELSE
                ZOR[i] := ZOR[i-1];             // Posunutie prvku doprava
        END_IF
END_FOR
N_ZAS := N_ZAS + 1;                     // Navýšenie počtu puzdier
END_IF
Povol_Zas := TRUE;              // Odblokovanie prace so zásobníkom pre iné procesy
```

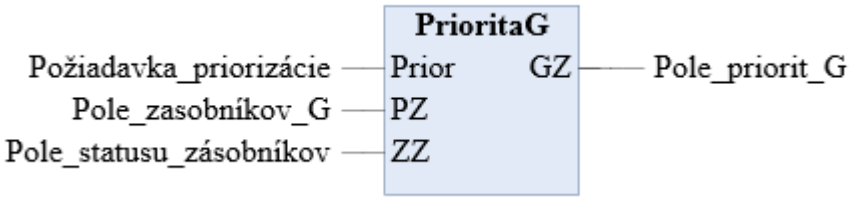
Obr. 43 - Podprogram pre zaradenie puzdra

5.7 Priorizovanie guľôčok

Podprogram priorizovania guľôčok sa volá po spustený zariadenia a v prípade zmeny vykonanej obsluhou na displeji. Zmena priority môže byť vykonaná kedykoľvek. Proces párovania v ďalšom kroku použije nové poradie zásobníkov. Podobne je tomu aj pri zapnutí niektorého zo zásobníkov guľôčok. Problém nastáva v situácii, keď sa niektorý zásobník manuálne vypne, ale montážny automat potrebuje, pre niektoré montované ložisko, guľôčky z daného zásobníka. Táto situácia, ktorá nastáva v prípade poruchy zásobníka alebo jeho vyprázdnenia je riešená kontrolou v riadiacom systéme

samotného montážneho automatu. Riešením je zrušenie montáže daného ložiska a fyzické odobratie súčastí (puzdra i hriadeľa) z procesu montáže.

Výsledkom podprogramu je zoradené pole zapnutých zásobníkov. Grafická reprezentácia vstupov a výstupov je zobrazená na obrázku č.44. Implementácia programu je zobrazená na obrázku č.45.



Obr. 44 - Vstupy a výstupy programu pre výpočet priorít guľôčok

DEKLARÁCIA

```
PROGRAM PrioritaG
VAR
        i       : INT;                  // Pomocne premenne i,j
        j       : INT;
        P       : ARRAY[1..9] OF INT;   // Pole pre prepočet priorít
END_VAR
```

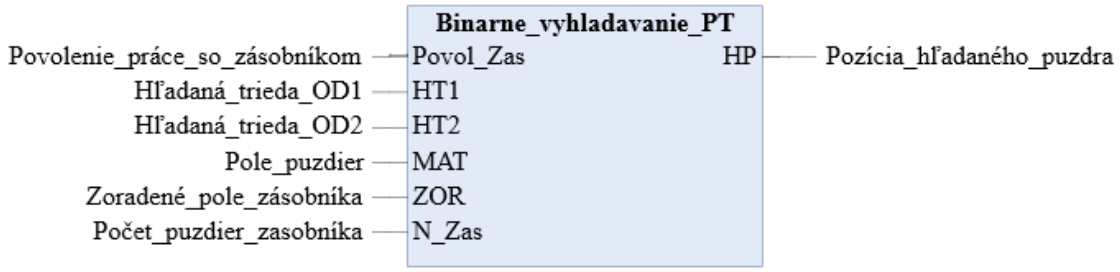
PROGRAMOVÁ ČASŤ

```
// ZAPÍSANIE PORADIA ZÁSOBNÍKOV PODĽA STANOVENEJ PRIORITY
IF Prior = TRUE THEN
                FOR i:=1 TO 9 BY 1 DO // Priorizovane poradie
                        P[i] := PZ[i];
                END_FOR;
        ELSE
                FOR i:=1 TO 9 BY 1 DO // Základne poradie
                        P[i] := i;
                END_FOR;
END_IF
// VYČISTENIE PORADIA ZÁSOBNÍKOV O VYPNUTÉ ZÁSOBNÍKY
j:=0;
FOR i:=1 TO 9 BY 1 DO
        IF ZZ[i] = FALSE THEN           // Vypnutý zásobník
                i:=i+1;
                GZ[10-i+j] := 0;
        ELSE                            // Zapnutý zásobník
                GZ[j]:=P[i];
                i:=i+1;
                j:=j+1;
        END_IF
END_FOR
```

Obr. 45 - Program pre priorizovanie guľôčok

5.8 Upravené binárne vyhľadávanie pri párovaní po triedach

Program pre párovanie po triedach, na základe tried obežných dráh hriadeľa, vyhľadá vhodné puzdro zo zásobníka. Program najprv pomocou metódy binárneho vyhľadávania nájde zhodu pre prvú obežnú dráhu. Následne od tejto pozície začne hľadať zhodu aj pre triedu druhej obežnej dráhy. Pokiaľ je nájdené puzdro vhodných charakteristík, program vracia údaj s indexom polohy v zásobníku. Pokiaľ sa puzdro nenašlo, je zavolaný program pre vyradenie hriadeľa. Vstupy a výstupy podprogramu sú graficky znázornené na obrázku č.46. Samotný program je zobrazený na obrázku č.47.



Obr. 46 -Vstupy a výstupy programu pre vyhľadanie puzdra párovaním po triedach

DEKLARÁCIA

```
PROGRAM Binarne_vyhladavanie_PT      // Binárne vyhľadávanie puzdra pri párovaní triedou
VAR
        HR      : INT;               // Pravá hranica intervalu
        HL      : INT;               // Ľavá hranica intervalu
        Stred   : INT;               // Stred intervalu
        Smer    : INT;               // Smer hľadania druhej OD
        HPoz    : INT;               // Pozícia puzdra podľa OD
END_VAR
VAR_IN_OUT
        Povol_Zas       : BOOL;      // Povolenie prace so zásobníkom
END_VAR
VAR_INPUT
        HT1     : INT;               // Hľadaná trieda OD1
        HT2     : INT;               // Hľadaná trieda OD2
        MAT     : ARRAY [1..3,1..169] OF INT;    // Pole OD1, OD2, Poloha v matici
        N_Zas   : INT;               // Počítadlo kusov v matici
END_VAR
VAR_OUTPUT
        HP      : INT;               // Index polohy vhodného puzdra
END_VAR
```

PROGRAMOVÁ ČASŤ

```
IF Povol_Zas THEN                   // Kontrola či sa zásobník pravé nepoužíva
        Povol_Zas := FALSE;         // Zablokovanie prace so zásobníkom pre iné procesy

        HL      := 1;               // Ľavá hranica vyhľadávania, inicializácia na 0
        HR      := N_Zas;           // Pravá hranica vyhľadávania, inicializácia na počet prvkov
        HPoz    := -1;              // Bez zmeny znamená neúspech pri hľadaní v triedach OD1
        HP      := -1;              // Bez zmeny znamená neúspech pri hľadaní v triedach OD2

        WHILE HL < HR DO                // Binárne vyhľadávanie triedy OD1
                Stred := (HR - HL) / 2;     // Výpočet strednej hodnoty intervalu
                IF HT1 = ZOR[Stred]^ THEN       // Stredný prvok sa rovná hľadanému
                        HPoz := Stred;      // Hľadaná pozícia pre OD1
                        EXIT;               // Ukončenie hľadania
                ELSIF HT1 > ZOR[Stred]^ THEN // Stredný prvok je menší ako hľadaný
                        HL := Stred +1;             // Posun do pravej polovice intervalu
                ELSE                                //Stredný prvok je väčší ako hľadaný
                        HR := Stred -1;             // Posun do ľavej polovice intervalu
                END_IF;
        END_WHILE

        IF HPoz <>  -1 THEN     // Bolo nájdene vhodne puzdro pre OD1
                IF ZOR[HPoz+169*4]^ = HT2 THEN      // OD2 nájdeného puzdra vyhovuje
                        HP := (ZOR[HPoz]-MAT)/4;    // Hľadaná pozícia
                ELSE                                // Hľadanie podľa triedy OD2
                        IF ZOR[HPoz+169*4]^ > HT2 THEN      //Určenie smeru hľadania
                                Smer := -1;         // Hľadanie puzdra v nižších indexoch
                        ELSE
                                Smer := 1;          // Hľadanie puzdra vo vyšších indexoch
                        END_IF
                        WHILE ZOR[HPoz]^ = HT1 DO           // Trieda OD1 bez zmeny
                                HPoz := HPoz + Smer;        // Posuv na ďalší prvok
                                IF ZOR[HPoz+169*4]^ = HT2 THEN
                                        HP := (ZOR[HPoz]-MAT)/4; // Hľadaná pozícia
                                END_IF
                        END_WHILE
                END_IF
        ELSE                        // Nebola nájdená zhoda pre OD1
        Vyradenie_kusa ( );         // Volanie podprogramu pre vyradenie súčasti
        END_IF
        IF HP = -1 THEN                 // Nebola nájdená zhoda pre OD2
                Vyradenie_kusa ( );     // Volanie podprogramu pre vyradenie súčasti
        END_IF
END_IF
Povol_Zas := TRUE;      // Odblokovanie prace so zásobníkom pre iné procesy
```

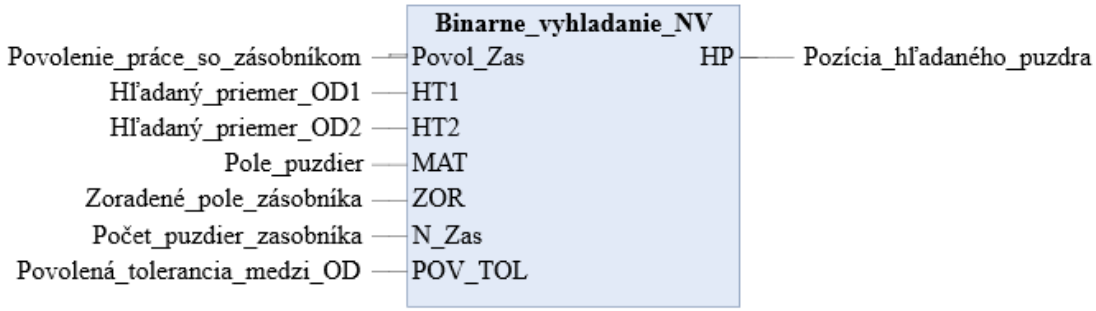
Obr. 47 - Podprogram pre vyhľadávanie puzdra pri párovaní po triedach

5.9 Binárne vyhľadávanie pri párovaní nepriamym výpočtom

Program pre párovanie nepriamym výpočtom hľadá puzdro, ktorého rozdiely medzi obežnými dráhami teoretického a reálneho puzdra sú v rámci definovanej tolerancie a zároveň je ich súčet najmenší.

Program najprv pomocou metódy binárneho vyhľadávania nájde puzdro s prvou obežnou dráhou v rámci tolerancie. Následne prehľadáva vhodné susedné puzdra. Puzdro s najmenšími odchýlkami od hľadaného je programom vrátene ako index polohy puzdra v zásobníku. Pokiaľ sa puzdro nenašlo, je zavolaný program pre vyradenie hriadeľa.

Vstupy a výstupy podprogramu sú graficky znázornené na obrázku č.48. Samotný program je zobrazený na obrázku č.49.



Obr. 48 - Vstupy a výstupy programu pre vyhľadávanie puzdra nepriamym výpočtom

DEKLARÁCIA

```
PROGRAM Binarne_vyhladavanie_PT      // Binárne vyhľadávanie puzdra pri párovaní triedou
VAR
        HR      : INT;               // Pravá hranica intervalu
        HL      : INT;               // Ľavá hranica intervalu
        Stred   : INT;               // Stred intervalu
        HPoz    : INT;               // Pozícia puzdra podľa OD
        Min_TOL     : REAL           // Minimálny súčet tolerancií
        Min_POL     : POINTER TO REAL;   // Poloha hľadaného puzdra
        i       : INT;               // Pomocná premenná
END_VAR
VAR_IN_OUT
        Povol_Zas       : BOOL;      // Povolenie prace so zásobníkom
END_VAR
VAR_INPUT
        HT1     : INT;               // Hľadaná trieda OD1
        HT2     : INT;               // Hľadaná trieda OD2
        MAT     : ARRAY [1..2,1..169] OF INT;    // Pole OD1, OD2
        ZOR     : ARRAY [1..169] OF POINTER TO REAL; // Zoradené pole smerníkov
        N_Zas   : INT;               // Počítadlo kusov v matici
END_VAR
VAR_OUTPUT
        HP      : INT;               // Index polohy vhodného puzdra
END_VAR
```

PROGRAMOVÁ ČASŤ

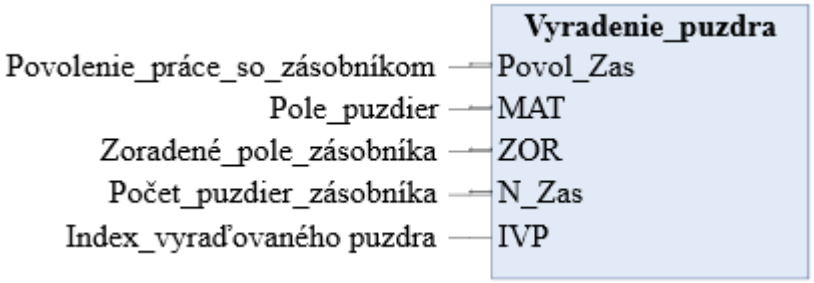
```
IF Povol_Zas THEN                    // Kontrola ci sa zásobník pravé nepoužíva
        Povol_Zas := FALSE;          // Zablokovanie prace so zásobníkom pre iné procesy

        HL := 1;                     // Ľavá hranica vyhľadávania - inicializácia na 0
        HR := N_Zas;                 // Pravá hranica vyhľadávania - inicializácia na počet prvkov
        HPoz := -1;                  // Bez zmeny znamená neúspech pri hľadaní OD1
        HP := -1;                    // Bez zmeny znamená neúspech pri hľadaní OD2

        WHILE HL < HR DO             // Binárne vyhľadávanie nepriamo cez OD1
                Stred := (HR - HL) / 2;     // Výpočet strednej hodnoty intervalu
                IF ABS(HT1-ZOR[Stred]^) < POV_TOL THEN      //
                        HPoz := Stred;      // Hľadaná pozícia pre OD1
                        EXIT;               // Ukončenie hľadania
                ELSIF HT1 > ZOR[Stred]^ THEN // Stredný prvok je menší ako hľadaný
                        HL := Stred +1;     // Posun do pravej polovice intervalu
                ELSE                        //Stredný prvok je väčší ako hľadaný
                        HR := Stred -1;     // Posun do ľavej polovice intervalu
                END_IF;
        END_WHILE

        IF HPoz <>  -1 THEN          // Bolo nájdene vhodné puzdro pre OD1
                Min_TOL  :=  ABS(HT1-ZOR[Stred]^)  +  ABS(HT2-ZOR[Stred+4*169]^);
        // Súčet odchýlok oboch dráh nájdeného puzdra
        Min_POL := ZOR[HPoz];        // Poloha vhodného puzdra
        i:=0;
        WHILE ABS(HT1-ZOR[HPoz+i]^) < POV_TOL    DO
                // Hľadanie smerom doľava pokiaľ prvý priemer zostava v tolerancii
                i:=i+1;
                IF ABS(HT2-ZOR[Hpoz+4*(169+i)]^) < POV_TOL THEN
                // Kontrola odchýlky druhej OD
                TOL := ABS(HT1-ZOR[HPoz+i]^) + ABS(HT2-ZOR[Hpoz+4*(169+i)]^);
                IF TOL<Min_TOL THEN
                        Min_TOL := TOL;        Min_POL := ZOR[HPoz+i];
                END_IF;
                END_IF
        END_WHILE
        i:=0;
        WHILE ABS(HT1-ZOR[HPoz+i]^) < POV_TOL    DO
                // Hľadanie smerom doprava pokiaľ prvý priemer zostava v tolerancii
                i:=i-1;
                IF ABS(HT2-ZOR[Hpoz+4*(169+i)]^) < POV_TOL THEN
                // Kontrola odchýlky druhej OD
                TOL := ABS(HT1-ZOR[HPoz+i]^) + ABS(HT2-ZOR[HPoz+4*(169+i)]^);
                IF TOL<Min_TOL THEN
                        Min_TOL := TOL;        Min_POL := ZOR[HPoz+i];
                END_IF
                END_IF
        END_WHILE
        HP := (Min_POL - Mat)/4;     // Poloha najvhodnejšieho puzdra
                ELSE                 // Nebola nájdená zhoda pre OD1
```

```
            Vyradenie_kusa ( );          // Volanie podprogramu pre vyradenie súčasti
            END_IF
            IF HP = -1 THEN                    // Nebola nájdená zhoda pre OD2
                  Vyradenie_kusa ( );          // Volanie podprogramu pre vyradenie súčasti
            END_IF
END_IF
Povol_Zas := TRUE;       // Odblokovanie prace so zásobníkom pre iné procesy
```

Obr. 49 - Podprogram pre vyhľadávanie puzdra pri párovaní nepriamym výpočtom

5.10 Vyradenie puzdra zo zásobníka

Po vyhľadaní a odobratí puzdra zo zásobníka je potrebné dané puzdro vyradiť z pamäte programu. V poli hodnôt obežných dráh, resp. tried obežných dráh „MAT“ je daná pozícia vynulovaná. V zoradenom poli smerníkov „ZOR“ na pole „MAT“ je vyhľadaný korešpondujúci prvok. Všetky prvky s vyšším indexom sú posunuté na predchádzajúcu pozíciu. Vyraďovaný prvok je následne zaradený na poslednú pozíciu. Vstupy a výstupy podprogramu sú grafické zobrazené na obrázku č.50 a samotný program na obrázku č.51.



Obr. 50 - Vstupy a výstupy podprogramu pre vyradenie puzdra

DEKLARÁCIA

```
PROGRAM Vyradenie_puzdra
VAR
        VP      : POINTER TO REAL;          // Smerník na vyraďované puzdro
        i, j    : INT;                      // Pomocné premenné
END_VAR
VAR_INPUT
        IVP: INT;                           // Index vyradeného puzdra
END_VAR
```

PROGRAMOVÁ ČASŤ

```
//PROGRAM PRE VYRADENIE PUZDRA NA POZICII IVP
IF Povol_Zas THEN               // Kontrola ci sa zásobník pravé nepoužíva
        Povol_Zas := FALSE;     // Zablokovanie prace so zásobníkom pre iné procesy
        VP := ADR (MAT[1,IVP]);             // Načítanie adresy vyradovaného puzdra
        MAT[1,IVP] := 0;                    //Nulovanie hodnôt puzdra
        MAT[2,IVP] := 0;                    //Nulovanie hodnôt puzdra
        N_Zas := N_Zas - 1;                 // Zníženie poctu kusov v počítadle
        FOR i:=1 TO 169 BY 1 DO             //Lineárne prehľadávanie poľa pointerov
                IF VP := ZOR[i] THEN        // Hľadaná zhoda
                        FOR j:=i TO 168 BY 1 DO     //Posuv ostatných prvkov v poli
                                ZOR[j] := ZOR[j+1];
                        END_FOR
                                ZOR[169] := VP; // Posunutie prázdneho miesta na koniec
                        EXIT;                       // Ukončenie prehľadávania
                END_IF
        END_FOR
END_IF
Povol_Zas := TRUE;      // Odblokovanie prace so zásobníkom pre iné procesy
```

Obr. 51 - Program pre vyradenie puzdra

6 ZÁVER

Hlavným cieľom tejto práce bol návrh a implementácia algoritmu v prostredí Codesys pre optimalizáciu procesu párovania súčastí ložiska. Úloha vychádzala z požiadaviek zákazníka na automatizáciu stávajúceho manuálneho procesu.

Úvod práce sa venoval popisu prostredia Codesys. Kapitola popisuje prácu v prostredí, možnosti programovania, programovacie jazyky a na niekoľkých príkladoch ukazuje použiteľnosť aplikácii.

Návrhu algoritmu párovania bola venovaná nasledujúca kapitola. Tá v sebe zlučuje teoretické poznatky s požiadavkami na výsledné zariadenie a skúsenosťami z praxe. Výsledkom tejto kapitoly je séria algoritmov, ktoré zabezpečujú potrebné procesy od prevodu analógovej hodnoty snímača na priemer obežnej dráhy až po správu maticového zásobníka pomocou programov na vkladanie, vyhľadávanie a odoberanie puzdier. Navrhnuté algoritmy obsahujú sumár potrebných vstupov a výstupov s popisom a dátovým typom použitých premenných. Proces párovania je navrhnutý dvoma metódami. Prvá metóda odzrkadľuje manuálny proces párovania pomocou párovacích tabuliek. Druhá metóda pristupuje k párovaniu pomocou nepriameho výpočtu požadovanej veľkosti obežných dráh.

Posledná kapitola je venovaná implementácii navrhnutých podprogramov v prostredí Codesys. Výsledkom tejto kapitoly je séria podprogramov, ktoré boli implementované na dvoch zariadeniach. Programy sú napísane s ohľadom na možnosť implementácie do ďalších podobných zariadení v budúcnosti. Zároveň každý program obsahuje graficky i textovo popísané potrebné vstupy i výstupy.

Vzhľadom na skutočnosť, že obe zariadenia boli uvedené do prevádzky pred odovzdaním tejto práce, je možné zhodnotiť reálne výsledky a prínosy navrhnutých podprogramov.

Ako hlavný ukazovateľ správnosti spárovania súčastí je veľkosť radiálnej vôle. Kusy vyradené z dôvodu veľkosti radiálnej vôle mimo požadovanej tolerancie sa fyzicky triedia na konci montážneho automatu. Ich počet sa zároveň zaznamenáva v riadiacom systéme montážneho automatu. Pri preberacích skúškach zariadenia sa dosiahli reálne výsledky chybovosti z dôvodu radiálnej vôle mimo tolerancie v počte menšom ako 0,5% z celkového počtu vyrobených kusov za jednu osem-hodinovú smenu (normovane 3200ks). Požiadavka zákazníka bola stanovená hranicou 1%.

Každé ložisko obsahuje 12 guľôčok. Pri bezproblémovej výrobe sa spotreba guľôčok na jednej osem-hodinovej smene blíži k 40 tisícom kusov. Pokiaľ z akéhokoľvek dôvodu dôjde k výpadku určitého rozmeru guľôčky (malé skladové zásoby, zistené problémy s kvalitou, ...), nastáva problém. Pokiaľ nemožno čakať na novú dodávku guľôčok je riešením opätovné prepárovanie puzdier s hriadeľmi na iný dostupný rozmer guľôčky. Tento proces je zdĺhavý a neefektívny, a môže nastať i prípad, že ho nemožno vykonať. Párovací algoritmus pracuje s aktuálnymi dostupnými guľôčkami. Okrem

eliminácie problému popísaného vyššie v tomto odstavci je možné využiť tento fakt aj pri snahe minutia malých skladových zásob určitého rozmeru guľôčok.

Táto práca popisuje dva hlavné prístupy k párovaniu komponentov. Jeden sa drží zaužívaného spôsobu párovania podľa párovacích tabuliek. Puzdra i hriadele sú po zmeraní zatriedené do príslušnej rozmerovej triedy rozdelených po 2µm. Algoritmus k triede hriadeľa hľadá puzdro danej triedy z maticového zásobníka. Druhý prístup priamo vypočíta požadovaný rozmer puzdra k nameranému hriadeľu podľa párovacej vôle. Algoritmus následne vyhľadá puzdro v rámci tolerancie oproti vypočítanému rozmeru. Pri testovaní oboch postupov sa druhý prístup prejavil oveľa presnejší ako prístup prvý. Taktiež bol vhodnejší pre riešenie problému s dodávkami guľôčok len v párnych odchýlkach, kde počet vhodných párov pri párovaní podľa párovacej tabuľky klesol na polovicu.

V dnešnej dobe sa okrem bežne používaných rádiusových obežných dráh používajú i nové tvary. Hlavnou úlohou nových tvarov obežných dráh je zväčšenie stykových plôch, čím sa upravujú hlavné charakteristiky ložiska, hlavne únosnosť a životnosť. Všetky meradlá priemerov obežných dráh pracujú so systémom 6 guľôčok. Tým simulujú rovnakú situáciu ako po zmontovaní samotného ložiska. Vďaka tomuto princípu v kombinácii s metódou párovania nie je potrebné upravovať meradlá alebo prepočítavať namerané hodnoty pri zmene tvaru obežnej dráhy.

7 ZOZNAM POUŽITEJ LITERATÚRY

[1] Digitale Wirtschaft und Gesellschaft. *Bundesministerium fuer Bildung und Forschung* [online]. c2017 [cit. 2017-05-25]. Dostupné z: https://www.bmbf.de/de/zukunftsprojekt-industrie-4-0-848.html

[2] *Princípy Industry 4.0* [online]. 2017 [cit. 2017-05-25]. Dostupné z: http://industry4.sk/principy/

[3] Obrázky modelov. In: *Zweireihige Sonderlager für Wasserpumpen der Verbrennungsmotoren* [online]. [cit. 2017-05-25]. Dostupné z: http://www.kinexbearings.de/index.php/produkty?view=category&bearing_caterory=35

[4] Obrázky modelov. In: *Zweireihige Sonderlager für Wasserpumpen der Verbrennungsmotoren* [online]. [cit. 2017-05-25]. Dostupné z: http://www.kinexbearings.de/index.php/produkty?view=category&bearing_caterory=35

[5] *Codesys* [online]. 3S-Smart Software, c2017 [cit. 2017-05-25]. Dostupné z: https://www.codesys.com/the-system.html

[6] Ložiska pro univerzální párování. *SKF* [online]. c2017 [cit. 2017-05-25]. Dostupné z: http://www.skf.com/cz/products/bearings-units-housings/ball-bearings/angular-contact-ball-bearings/single-row-angular-contact-ball-bearings/bearings-for-universal-matching/index.html

[7] IEC 1131 or 61131 : status of the Standard. *PLCopen* [online]. 2008 [cit. 2017-05-25]. Dostupné z: http://www.plcopen.org/pages/tc1_standards/iec_1131_or_61131/

[8] *STN EN 61131-3: Programovateľné regulátory. Časť 3: Programovacie jazyky*. 2013.

[9] STN EN 61131-3. *Úrad pre normalizáciu, metrológiu a skúšobníctvo Slovenskej republiky* [online]. Bratislava, c2011 [cit. 2017-05-25]. Dostupné z: https://www.sutn.sk/eshop/public/standard_detail.aspx?id=118173

[10] CSN EN 61131-3. *Úřad pro technickou normalizaci, metrologii a státní zkušebnictví* [online]. Praha, 2013 [cit. 2017-05-25]. Dostupné z: http://csnonlinefirmy.unmz.cz/html_nahledy/18/93649/93649_nahled.htm

[11] *PLC Programming - STEP 7 (TIA Portal)* [online]. c1996-2017 [cit. 2017-05-25]. Dostupné z: http://w3.siemens.com/mcms/automation-software/en/tia-portal-software/step7-tia-portal/Pages/default.aspx

[12] *Automation Builder* [online]. c2017 [cit. 2017-05-25]. Dostupné z: http://new.abb.com/plc/automationbuilder

[13] Comparison CODESYS V 2.3 vs CODESYS V.3. *Codesys* [online]. 2013 [cit. 2017-05-25]. Dostupné z: https://www.codesys.com/fileadmin/data/Images/System/Comparison_CODESYS_V2_to_V3.pdf

[14] World's fastest PLC based on CPU scanning speed. *Control Engineering* [online]. 2011 [cit. 2017-05-25]. Dostupné z: http://www.controleng.com/search/search-single-display/worlds-fastest-plc-based-on-cpu-scanning-speed/77f43d02495196284e85d9a455fa4f72.html

[15] Simpler System Soups Up Productivity. *Hydraulics&Pneumatics* [online]. 2014 [cit. 2017-05-25]. Dostupné z: http://www.hydraulicspneumatics.com/food-beverage/simpler-system-soups-productivity-0

[16] *VELTRU* [online]. c2015 [cit. 2017-05-25]. Dostupné z: http://www.veltru.com/

[17] Alkadur Robot Systems: "A Device that Never Sleeps". *EATON* [online]. c2014 [cit. 2017-05-25]. Dostupné z: http://www.eaton.no/EatonNO/ProdukterLosninger/Elektriske/Dinvirksomhet/Maskinbyggere/SuccessStories/ProcessingPackaging/AlkadurRobotSystems/index.htm

[18] ZELINKA, I., SNÁŠEL, V., ABRAHAM, A. (eds.): Handbook of Optimization. From Classical to Modern Approach. Berlin, Springer, 2013. ISBN 978-3-642-30-503-0.

[19] Technical Report: 4. Internal Clearance. *NSK* [online]. c2017 [cit. 2017-05-25]. Dostupné z: http://www.nsk.com/services/basicknowledge/technicalreport/04internal.html

[20] How LVDTs Work. *LVDT.co.uk* [online]. c2015 [cit. 2017-05-25]. Dostupné z: http://www.lvdt.co.uk/how-lvdts-work/

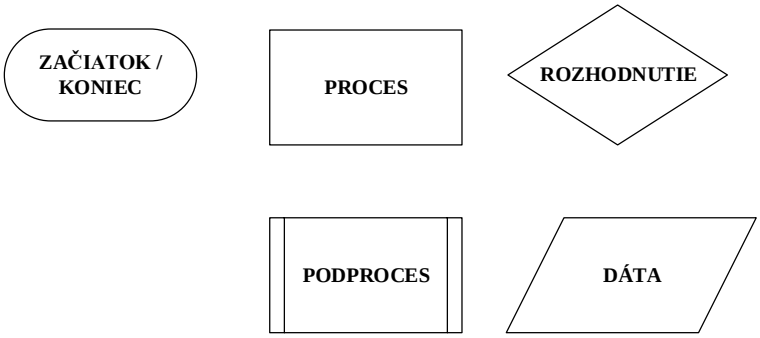
[21] *KEYENCE* [online]. c2017 [cit. 2017-05-25]. Dostupné z: http://www.keyence.com/

[22] CORMEN, Thomas H. *Introduction to algorithms*. 3rd ed. Cambridge, Mass.: MIT Press, c2009. ISBN 9780262033848.

ZOZNAM POUŽITÝCH SYMBOLOV A SKRATIEK

FBD	Jazyk funkčných blokov (Function block diagram)
OD	Obežná dráha
PLC	Programovateľný logický automat (Programmable logic controller)
POU	Programová organizačná jednotka (Program organization unit)
ST	Jazyk štruktúrovaného textu (Structured text)

Význam tvarov použitých v algoritmoch



ZOZNAM OBRÁZKOV

ZOZNAM TABULIEK

ZOZNAM PRÍLOH

CD

Diplomová práca vo formáte pdf

Archív programu pre prostredie Codesys v3.5