ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ

Fakulta elektrotechnická

Model Žonglér pro vzdálenou výuku a řízení CNC stroje

Diplomová práce

Vypracoval: Tomáš Kohout

Vedoucí práce: Ing. Pavel Burget, PhD.

Katedra řídící techniky
Praha 2011

České vysoké učení technické v Praze
Fakulta elektrotechnická

katedra řídicí techniky

ZADÁNÍ DIPLOMOVÉ PRÁCE

Student: **Bc. Tomáš Kohout**

Studijní program: Kybernetika a robotika
Obor: Systémy a řízení

Název tématu: **Model žonglér pro vzdálenou výuku a řízení CNC stroje**

Pokyny pro vypracování:

Cílem této práce je rozšíření modelu žonglér, který je řízen prostředky B+R a programován v návrhovém prostředí Automation Studio, o detekční systém předcházení srážce pohyblivých částí a následně řízení CNC frézky opět v prostředí Automation Studio.

1. Navrhněte a realizujte mechanismus prevence srážky na modelu žonglér.
2. Zapojte Žonglér do systému Lablink.
3. Seznamte se s CNC řízením společnosti B+R. Navrhněte řízení CNC frézky a realizujte jej na reálné frézce.
4. Porovnejte požadavky na řízení obou typů mechanismů.

Seznam odborné literatury:

Dodá vedoucí práce

Vedoucí: Ing. Pavel Burget, Ph.D.

Platnost zadání: do konce letního semestru 2011/2012

L.S.

Prof. Ing. Michael Šebek, DrSc.
vedoucí katedry

prof. Ing. Boris Šimák, CSc.
děkan

V Praze dne 14. 4. 2011

Prohlášení

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

V Praze, dne 12.5.2011 …….

Podpis

Abstrakt

Tato diplomová práce se zabývá připojením modulu Žonglér do systému vzdálené výuky Lablink. V souvislosti s tím je v práci popsán způsob zabezpečení modelu proti možnosti srážky mechanických částí. V teoretické části je blíže popsána průmyslová síť Ethernet Powerlink. Praktická část se zabývá implementací zabezpečovacího algoritmu na použitém hardware.

Druhá část této práce se zabývá programováním CNC vrtačky a frézky jako přídavného zařízení na již používaných pálicích strojích. V první části je popsán CNC systém B&R a v druhé části je popsána programová implementace zařízení.

Na závěr práce obsahuje porovnání programování obou typů zařízení.

Abstract

This diploma thesis deals with connection of model Juggler into system of remote laboratory called Lablink. It contains informations about the method how the prevention of collision of mechanical parts is solved. In theoretical part is described Ethernet Powerlink network as a main tool for solving problems. Practical part deals with implementation of prediction algorithm on hardware which was finally used.

Second part of this work is about programming CNC drill as an additional tool on cutting machines. In first part is described CNC system of B&R and in the second one is described the way how the tool was programmed.

In the end of this document is comparison of programming of this two machines which a worked with.

Poděkování

Na tomto místě bych chtěl poděkovat svému vedoucímu diplomové práce Ing. Pavlu Burgetovi za jeho pomoc při vzniku této práce. Dále patří mé poděkování Pavlu Jarošovi, Lubomíru Prudkovi a Josefu Necidovi, kteří společně se mnou na Žongléru realizovali své práce.

Na závěr chci poděkovat celé své rodině a přátelům, kteří mě v průběhu studia podporovali.

Obsah

1. Úvod ... 1
   1.1. Žonglér ... 1
   1.2. Účel Žongléra ... 2
   1.3. Důvody ke vzniku této práce ... 4
2. Požadavky a cíle zabezpečení ... 5
   2.1. Úpravy žongléra pro vzdálenou výuku ... 5
   2.2. Rizika při programování Žongléra ... 5
   2.3. Cíle zabezpečení ... 6
   2.4. Nové řešení zabezpečení ... 7
3. Průmyslová síť Ethernet Powerlink ... 8
   3.1. Základní vlastnosti sítě Etheret Powerlink v bodech ... 8
   3.2. Referenční model ... 9
   3.3. Fyzická vrstva ... 9
   3.4. Linková vrstva ... 10
      3.4.1. Režim Basic Ethernet ... 10
      3.4.2. Režim Powerlink ... 10
      3.4.3. Typy zařízení ... 10
      3.4.4. Managing Node ... 10
      3.4.5. Controlled Node ... 11
      3.4.6. Služby sítě Powerlink ... 11
      3.4.7. Cyklus sítě Ethernet Powerlink ... 11
      3.4.8. Adresování v síti Powerlink ... 12
      3.4.9. Struktura Rámců Ethernet Powerlink ... 13
      3.4.10. Druhy zpráv v síti Ethernet Powerlink ... 13
   3.5. Síťová + transportní vrstva ... 14
      3.5.1. IP adresování v síti powerlink ... 14
   3.6. Aplikační vrstva ... 15
4. Realizace zabezpečení ... 16
   4.1. Získávání informace o poloze a rychlosti ... 16
      4.1.1. Poll Response rámec Acoposu ... 16
   4.2. Řešení s deskou Shark a modulem NetX ... 17
   4.3. Řešení zabezpečení pomocí hardware B&R ... 19
   4.4. Použitý hardware a jeho konfigurace ... 19
      4.4.1. Dynamické kanály ... 20
      4.4.2. Synchronizace PLC a sítě Powerlink ... 22
      4.4.3. Časování tříd cyklických úloh ... 23
   4.5. Implementace algoritmu – získávání dat ... 25
      4.5.1. Získávání dat z rámců Acoposů – softwarová implementace ... 25
      4.5.2. Přenos dat mezi PP420 a X20 ... 25
   4.6. Teoretické řešení algoritmu ... 26
      4.6.1. Výpočty a vzorce použité pro zabezpečení ... 28
   4.7. Praktický test zabezpečení ... 30
   4.8. Parametry algoritmu ... 33
      4.8.1. Maximální hodnoty zrychlení ... 33
      4.8.2. Krajní polohy, limitní kolizní polohy ... 35
      4.8.3. Jednotky ... 37

4.9. Programové řešení algoritmu ... 38
4.9.1. Prevence srážky ... 38
4.9.2. Kontrola polohy lineárního motoru ... 39
4.9.3. Zastavení stroje ... 39
4.9.4. Ovládání quickstopu přes safety výstupy ... 39
4.10. Časově nekritická část zabezpečení ... 40
4.10.1. Nutné součásti projektu ... 40
4.10.2. Řešení časově nekritické části zabezpečení ... 40
4.10.3. Kontrola konfigurace os ... 41
4.10.4. Knihovna AsIMA ... 42
4.10.5. Kontrola přítomnosti cyklických úloh pro zabezpečení ... 43
5. Zapojení do systému Lablink ... 46
5.1. Systém Lablink ... 46
5.1.1. Rezervační systém ... 46
5.1.2. Vzdálené plochy ... 46
5.1.3. Vizuální přístup k modelu ... 46
5.2. Žonglér v systému Lablink ... 47
5.2.1. Role student ... 47
5.2.2. Role administrátor ... 47
6. Zhodnocení zabezpečení ... 48
6.1. Zhodnocení algoritmu a technického řešení ... 48
6.2. Zhodnocení zabezpečení pro práci studentů ... 48
7. CNC – obecný úvod ... 50
7.1. Možnosti řízení trajektorie v CNC ... 50
7.1.1. Poit to Point řízení ... 50
7.1.2. Lineární řízení ... 50
7.1.3. Řízení trajektorie ... 51
7.2. Programování CNC systémů ... 51
7.2.1. CAM software ... 52
7.3. CNC v systémech B&R ... 53
7.3.1. Softwarový koncept ... 53
7.3.2. CNC Decoder ... 54
7.3.3. Path Generátor ... 55
7.3.4. ARNC0 knihovna ... 55
7.3.5. CNC Systém ... 56
7.3.6. Typy os v CNC systémech ... 57
7.4. CNC program ... 58
7.4.1. M – funkce ... 58
7.4.2. Synchronní M-funkce ... 59
7.4.3. Asynchronní M-funkce ... 59
7.4.4. Komunikce CNC <->PLC ... 59
7.4.5. Vlastnosti M-funkcí ... 60
7.5. Externí parametry ... 60
8. Řízení CNC frézky ... 61
8.1. Pálicí stroje Vanad ... 61
8.1.1. CNC vrtačka a frézka ... 61
8.2. Cíle práce ... 62
8.3. Používaný HW ... 63
8.4. Softwarový koncept ... 63
8.4.1. Obecné zásady pro programování ... 64

8.4.2. Datová struktura cyklické úlohy ... 65
8.4.3. Hierarchie cyklických úloh a komunikace mezi nimi ... 66
8.5. Popis výchozího projektu ... 67
8.5.1. Osy systému ... 67
8.5.2. Osy systému a CNC ... 68
8.5.3. Cyklické úlohy technologií ... 68
8.6. Technologie vrtání a frézování ... 68
8.6.1. Osa Z ... 69
8.6.2. Cyklická úloha technologie vrtání a frézování ... 69
8.6.3. Datová struktura technologie vrtání a frézování ... 71
8.6.4. Externí parametry ... 72
8.6.5. M-funkce ... 72
8.7. Ovládání vřetene vrtačky – Acopos Inverter ... 74
8.7.1. Programové řešení ovládání Acopos Inverter ... 74
8.7.2. Knihovna tk_ACPinv ... 74
9. Porovnání řízení CNC a Žongléra ... 76
9.1. Knihovny pro řízení os ... 76
9.2. Programování systémů ... 76
9.3. Synchronizace ... 76
9.4. Vizualizace a HMI ... 77
9.5. Návrhy na změny v projektech ... 77
10. Závěr ... 78

Přílohy:

A. Ukázky zdrojových kódů ... 81
A.1 Konfigurační objekt pro IMA na X20 ... 81
A.2 Konfigurační objekt pro IMA na PP420 ... 81
B. Zapojení do systému Lablink ... 82
B.1 Pokyny určené pro administrátora ... 82
B.2 Pokyny pro studenty ... 85

Seznam obrázků

Obr. 1.1 Čelní pohled na žonglér ... 3
Obr. 1.2 Vyhazovací ručky Žongléra ... 4
Obr. 2.1 Kolize vyhazovacích misek ... 6
Obr. 2.2 Kolize vyhazovací misky a uchycení magnetické tyče ... 6
Obr. 3.1 Referenční model sítě Ethernet Powerlink ... 9
Obr. 3.2 Cyklus sítě Ethernet Powerlink ... 11
Obr. 4.1 PoolResponse rámec Acoposu ... 16
Obr. 4.2 Deska SHARK s modulem ComX ... 17
Obr. 4.3 Schéma zabezpečení pomocí desky SHARK a modulu ComX ... 18
Obr. 4.4 Přidání PLC X20 do projektu ... 20
Obr. 4.5 Vytváření dynamických kanálů na PLC X20 ... 21
Obr. 4.6 Mapování proměnných z PP420 do X20 ... 22
Obr. 4.7 Nastavení časování CPU a synchronizace se sítí Ethernet Powerlink ... 23
Obr. 4.8 Nastavení časování třídy cyklických úloh ... 24
Obr. 4.9 0sy Žongléra a jejich číslování ... 27
Obr. 4.10 Znázornění významu proměnných použitých v teoretickém rozboru ... 29
Obr. 4.11 Přenos informace sítí Ethernet Powerlink ... 30
Obr. 4.12 Časy přenosu informace sítí Ethernet Powerlink ... 31
Obr. 4.13 Trace osy při zastavení s vyznačenými časy ... 33
Obr. 4.14 Závislost bezpečné vzdálenosti na rychlosti horizontální osy ... 35
Obr. 4.15 Limitní poloha ručky pro prevenci srážky s druhou ručkou ... 36
Obr. 4.16 Limitní poloha pro prevenci srážky s pátou osou ... 36
Obr. 4.17 Rozdíl mezi safety výstupem „direct“ a „via SafeLOGIC“ ... 40
Obr. 4.18 Přístup k PP420 a X20 přes Lablink ... 41
Obr. 4.19 Struktura BR modulu a datový objektu ... 44
Obr. 4.20 Schéma zabezpečení časově nekritické části ... 45
Obr. 7.1 Možnosti řízení trajektorie v CNC systémech ... 51
Obr. 7.2 Vývoj CNC programu ... 52
Obr. 7.3 Softwarový concept CNC systému ... 53
Obr. 7.4 Decoder a Path generator ... 54
Obr. 7.5 Nastavení ARNC0 manageru ... 55
Obr. 7.6 Přidání CNC objektu do Projektu ... 56
Obr. 7.7 Povolení ACP10 osy pro použití v CNC ... 57
Obr. 7.8 CNC program (G-kód) ... 58
Obr. 7.9 Komunikace pomocí M-funkcí ... 59
Obr. 8.1 Schématický nákres vrtacího zařízení ... 62
Obr. 8.2 Systém uživatelů a oprávnění ... 63
Obr. 8.3 Struktura cyklické úlohy ... 64
Obr. 8.4 Datová struktura pro cyklickou úlohu ... 66
Obr. 8.5 Hiearchie cyklických úloh ... 67
Obr. 8.6 Koncept os v CNC aplikaci ... 68
Obr. 8.7 Stavový automat cyklické úlohy vrtání ... 70
Obr. 8.8 Prostorové uspořádání technologií ... 74

Seznam tabulek

Tab. 3.1 Struktura rámce Ethernet Powerlink 13
Tab. 3.2 Typy zpráv v síti Ethernet Powerlink 13
Tab. 4.1 Seznam proměnných pro cyklickou komunikaci 26
Tab. 4.2 Parametry algoritmu a jejich hodnoty 37
Tab. 8.1 Struktura tk_TECH_VRT_FREZ_main_type 71
Tab. 8.2 Struktura tk_TECH_VRT_FREZ_param_type 71
Tab. 8.3 Struktura tk_TECH_VRT_FREZ_cmd_type 72
Tab. 8.4 Struktura tk_TECH_VRT_FREZ_info_type 72
Tab. 8.5 Tabulka externích parametrů 72
Tab. 8.6 Struktura tk_ACPi_main_type 75
Tab. 8.7 Struktura tk_ACPi_param_type 75
Tab. 8.8 Struktura tk_ACPi_cmd_type 75
Tab. 8.9 Struktura tk_ACPi_info_type 75

Kapitola 1

1. Úvod

1.1. Žonglér

Žonglér je automatizované zařízení navržené za účelem žonglování s kulečníkovými koulemi. Celé zařízení bylo zkonstruováno na Katedře řídicí techniky ČVUT v Praze, kde je umístěn v laboratoři.

Stroj se skládá ze dvou otáčivých ramének zakončených miskou, které slouží k chytání a vyhazování koulí. Tato raménka jsou ovládána dvěma synchronními motory, na jejichž hřídel jsou připevněna a tím pádem zprostředkovávají horizontální pohyb žonglujících koulí. Synchronní motory s raménky jsou připevněny k lineárním modulům umístěným vertikálně na zadní straně konstrukce Žongléra. Tím je zajištěn vertikální pohyb při přehazování koulí. Lineární moduly jsou poháněny opět dvojicí synchronních motorů, které jsou navíc vybaveny převodovkou „do pomala“ pro zajištění dostatečného kroutícího momentu respektive zrychlení [2]. Pro vyhazování spadlých koulí zpět do žonglovacího cyklu nebo při začátku žonglování je Žonglér vybaven jedním lineárním motorem umístěným přesně uprostřed mezi vertikálními moduly tak, že je možné vyhozenou kouli chytit oběma ručkami pro žonglování. Sbírání spadlých koulí zajišťuje podavač skládající se z nakloněné dopadové desky s otvorem, zásobníku koulí a dávkovače uvolňujícího kouli na misku lineárního motoru. Dále žonglér obsahuje nezbytné senzory koncových poloh, osvětlující techniku, chlazení lineárního motoru, bezpečnostní prvky a rozhraní s konektory pro připojení k rozvaděči.

Veškeré řízení Žongléru je umístěno v rozvaděči, se kterým je stroj propojen čtyřmi signálovými kabely pro vstupy/výstupy, napájecím kabelem, silovými a signálovými kabely od jednotlivých motorů. Hlavním řídicím systémem je Power Panel 420 s grafickým dotykovým rozhraním umístěným ve dveřích rozvaděče. Dalším řídícím prvkem je Safety PLC zajišťující bezpečnostní funkce celeho stoje [4]. Vedle Safety PLC se nachází modul vstupů a výstupů obsahující jak safety tak i standardní vstupy/výstupy. Pod Safety PLC je umístěno PLC X20 jehož funkce bude popsána v dalších kapitolách této práce. Vedle řídícího systému X20 se nachází průmyslový hub. Níže v rozvaděči jsou umístěny spínací a jistící prvky jak silové tak signálové části obvodů společně se zdrojem napětí 24V. Ve spodní části jsou umístěny servozesilovače Acopos pro jednotlivé motory.

Úplně nejníže se nacházejí konektory pro připojení napájení 3x230V/400V, signálové konektory do Žongléra a zásuvky RJ-45 připojující žonglér do sítě nebo k diagnostice průmyslové sítě. Pro propojení jednotlivých řídicích prvků je použita síť Ethernet Powerlink, která bude podrobněji představena v dalších částech tohoto dokumentu.

V současné době je naprogramováno už několik různých funkcí, které může Žonglér vykonávat. Mezi ně patří dva různé styly žonglování se čtyřmi koulemi – klasické žonglování a vzor fontána, kdy se mezi levou a pravou stranou vyměňují pouze dvě koule a zbylé dvě jsou pouze vyhazovány kolmo do vzduchu. Žonglování se třemi koulemi, kdy jsou koule vyhazovány střídavě levou a pravou stranou, působící více kontinuálním dojmem. Dále přehazování dvou a jedné koule. Poslední funkcí je Katapult, při kterém je koule vyhozena kolmo do výšky skrz připravenou obroučku, nad kterou je opět chycena. Poté následuje puštění koule volným pádem zpět obroučkou a opětovné chycení. Tuto funkci je možné provádět jak s levou tak i s pravou vyhazovací ručkou. Zbývající funkce jsou spíše podpůrného charakteru jako nahození/upuštění koule v klidovém režimu atp. Více informací o žonglování lze nalézt v [3],[2],[5].

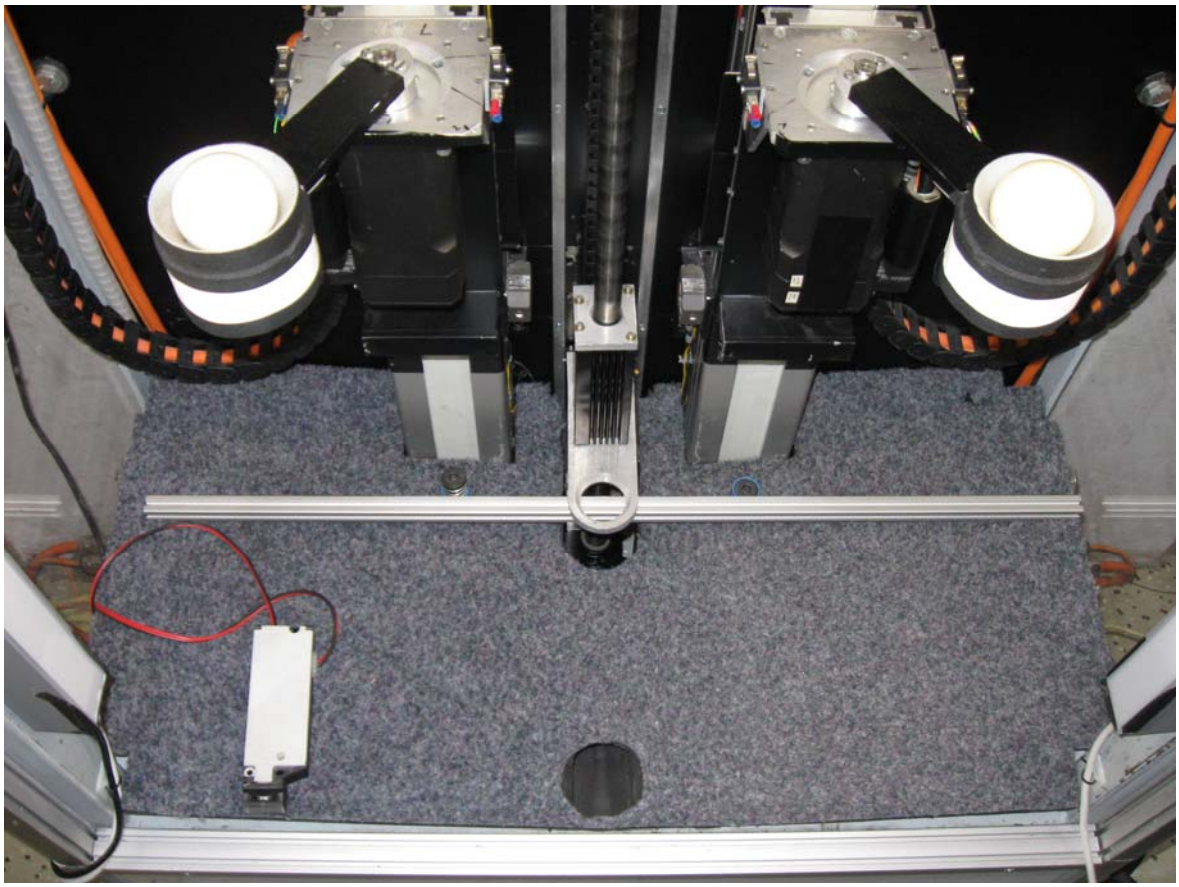
1.2. Účel Žongléra

Účelem stroje je umožnění výuky programování průmyslových systémů na reálné technologii. Specielně Žonglér nabízí i možnost výuky vzájemné synchronizace rychlých servopohonů s vysokými nároky na přesnost. Dále existuje možnost využití průmyslové kamery a softwaru pro zpracování obrazu k poskytnutí dílčí zpětné vazby o žonglovacím procesu [1]. Jedním z úkolů této práce je umožnit studentům vzdálený přístup k modelu pomocí systému Lablink. Na Žonglérovi tedy bude možné pracovat bez nutnosti přítomnosti v laboratoři, kde se žonglér nachází.

Kromě výuky slouží Žonglér také k prezentačním účelům. Prakticky veškeré řízení je realizováno pomocí produktů firmy B&R, která využívá stroj k prezentaci svých produktů a demonstrování jejich možností. Mimo B&R propaguje Žonglér také samotnou Katedru řídicí techniky potažmo celé ČVUT v Praze. Za účelem prezentace byl doposud Žonglér vystavován na tuzemských veletrzích Amper 2010 a Amper 2011. Byl také vystavován a předváděn na zahraničním veletrhu SPS/IPC Drives 2010 v německém Norimberku, kde se nacházel na stánku EPSG a kde mu byla z řad návštěvníků věnována relativně velká pozornost.

Do budoucna by měl Žonglér sloužit zejména pro vzdálenou výuku. Další náměty na zlepšování a vývoj se uvažovaly zejména v řešení lineárního vertikálního pohybu, který je za současného řešení velmi hlučný a pro prezentační účely ne zcela vhodný.

Obr. 1.1 Čelní pohled na žonglér



Obr. 1.2 Vyhazovací ručky Žongléra

1.3. Důvody ke vzniku této práce

Jak již bylo v předchozích odstavcích několikrát zmíněno, hlavním cílem projektu Žonglér bylo zapojit model do systému vzdálené laboratoře. Vzdálená laboratoř v praxi funguje tak, že si student přes webové rozhraní vybere model, se kterým chce pracovat, zvolí si čas, odkdy dokdy chce s modelem pracovat, a provede příslušnou rezervaci. Ve zvolený čas je potom studentovi k dispozici relace vzdálené plochy, na které je nainstalován příslušný software společně s připraveným projektem či dalšími materiály potřebnými k zvládnutí úlohy. Po skončení rezervace je relace vzdálené plochy vyčištěna a připravena pro dalšího studenta. Jednotlivé modely zapojené do systému vzdálené laboratoře jsou vybaveny kamerou, jejíž obraz lze sledovat přes příslušné webové rozhraní v klasickém internetovém prohlížeči. Tímto způsobem je možné vyvíjet úlohy na modelech bez nutnosti fyzické přítomnosti v laboratoři. To umožňuje jednak větší flexibilitu pro studenty, kteří nemusí pracovat pouze se softwarovými simulátory, ale i větší využitelnost a dostupnost modelu po celý den i v nočních hodinách, kdy není přístup do laboratoře možný.

Kapitola 2

2. Požadavky a cíle zabezpečení

2.1. Úpravy žongléra pro vzdálenou výuku

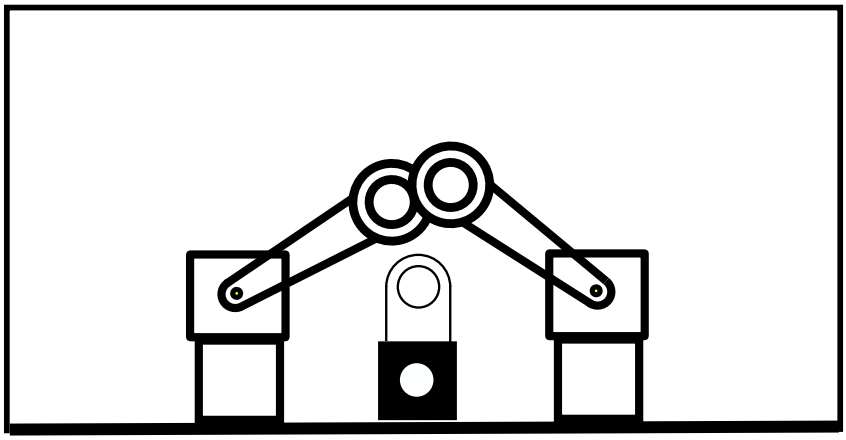
V souvislosti s možností vzdáleného přístupu k modelu vzniká řada nových požadavků, které musí dané zařízení splňovat. Jednak se jedná o instalaci kamery zajišťující on-line sledování s dostatečně krátkou odezvou, dále poskytnutí dostatečného osvětlení pro práci v nočních hodinách a nakonec nejdůležitější požadavek na plnou funkci bez nutnosti vnějšího zásahu cizí osoby. To znamená, že se model nesmí dostat do stavu, ze kterého ho nebude možné dostat pomocí vzdáleného přístupu.

Konkrétně na Žonglérovi bylo osvětlení vyřešeno pomocí lineárních zářivek umístěných po stranách a nahoře v pracovním prostoru. Více informací o volbě osvětlení je k dispozici v mé bakalářské práci Vizuální kontrola chodu stroje [1]. Dále bylo nutné vyřešit bezpečnost stroje z hlediska uživatele a osob, kteří se pohybují poblíž, či by mohli přijít do styku se strojem jiným způsobem. Toto řešení je založeno na funkci bezpečnostního PLC [4]. Zajištění bezproblémového chodu bez nutnosti přítomnosti obsluhy je řešeno pomocí podavače skládajícího se z vhodně vyspádovaného dna s otvorem, kterým se koule dostávají do zásobníku, ze kterého je možné uvolnit jednotlivé koule na vertikální osu, která je vyhazuje zpět do žonglovacího cyklu. Podrobnější popis řešení zásobníku a vertikální osy se nachází v [3].

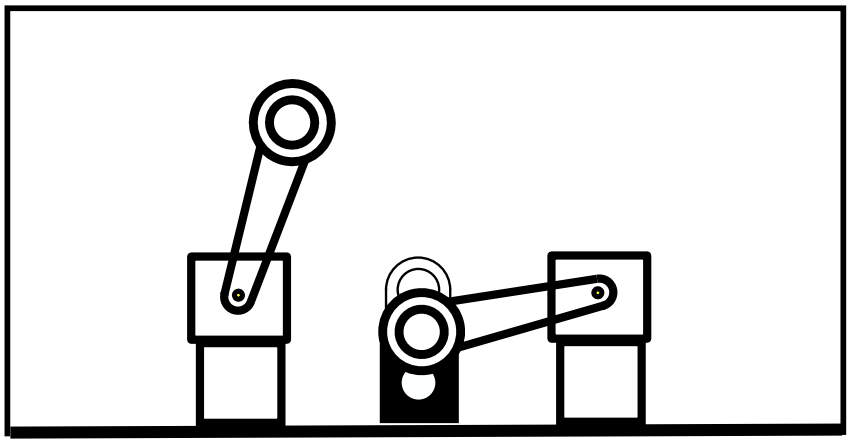
2.2. Rizika při programování Žongléra

Další nutné úpravy se odvíjejí od programování nových aplikací v Žonglérovi. Zde se jako problematické ukázalo uspořádání jednotlivých prvků, které z důvodu funkčnosti stroje nešly umístit jinak. V důsledku toho je možné, že při špatném naprogramování pohybů os systému, může dojít ke srážce vyhazovacích ramének s miskami, následně k jejich zničení a ohrožení osob a věcí nacházejících se v blízkosti stroje. Dále může dojít také ke kolizi mezi vyhazovací miskou a horním upevněním magnetické tyče lineárního motoru (Obr. 2.1). Kromě kolize žonglujících misek mezi sebou hrozí ještě nebezpečí kolize jednotlivých misek s vertikální osou vyhazující koule z podavače (Obr. 2.2). Poslední nutnou úpravou je zamezení propadnutí koule do prostoru pod Žonglérem,

což může nastat při současném uvolnění koule ze zásobníku a nesprávné poloze osy vyhazující koule vzhůru.



Obr. 2.1 Kolize vyhazovacích misek



Obr. 2.2 Kolize vyhazovací misky a uchycení magnetické tyče

2.3. Cíle zabezpečení

Cílem části této práce je vyřešit nebezpečí, která byla zmíněna v předchozím odstavci a zatím nebyla vyřešena v předchozích pracích. To znamená zajištění prevence potencionálních srážek částí stroje.

V mé předchozí práci [1] byla rozebírána možnost rozeznávání poloh vyhazovacích misek pomocí rychloběžné průmyslové kamery. Tento přístup však narážel na mnoho překážek díky nímž byl další vývoj tímto směrem přerušen. Jednalo se jednak o příliš velkou nespolehlivost z důvodu měnících se světelných podmínek společně s možností

náhodného zakrytí prostoru předmětem atd. Druhým klíčovým faktorem vedoucím k zastavení vývoje byla dosažitelná odezva celého systému strojového vidění. Do ní se promítly časy potřebné k pořízení aktuálního snímku, jeho zpracování, interpretace dat, a přenos informace do řídícího systému. Tento celkový čas se s dostupným hardwarem a softwarem pohyboval v řádech desítek milisekund, což je pro zabezpečující funkci nepřijatelná hodnota. V dalších kapitolách bude popsáno nové řešení zabezpečení, které dosahuje o mnoho lepší výsledky než zmiňované předchozí a navíc neobsahuje rizika spojená s osvětlením a další vlivy.

2.4. Nové řešení zabezpečení

Nové řešení problému popisované v této práci je založeno na získávání informace o polohách jednotlivých os systému z cyklické komunikace, která mezi nimi neustále probíhá. Tímto způsobem lze získávat z časového hlediska velmi „aktuální“ data, která navíc nejsou zatížena nejrůznějšími šumy, jak tomu bylo v předchozím řešení.

Pro komunikaci mezi zařízeními je v Žonglérovi použita průmyslová síť Powerlink, jejíž funkce je pro řešení problému zásadní. V následující kapitole ji tedy podrobněji představím.

Kapitola 3

3. Průmyslová síť Ethernet Powerlink

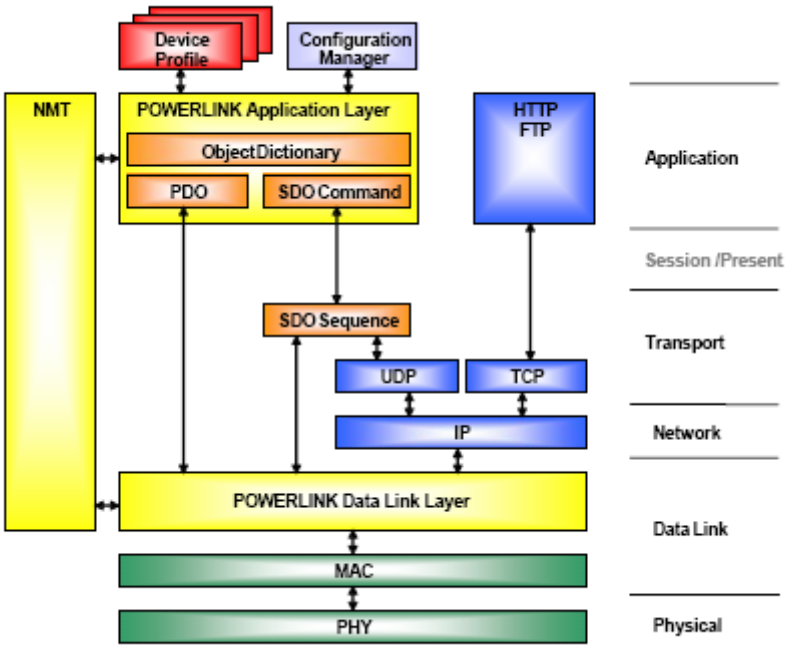
Ethernet Powerlink je deterministický komunikační protokol pro přenos dat v reálném čase založený na standardu Fast Ethernet IEEE 802.3. Protokol funguje na principu cyklické komunikace s pevnou dobou trvání cyklu a velmi malou časovou nejistotou. Krajní hodnoty jsou v současnosti na 100µs pro dobu cyklu a méně než 0.1 µs pro hodnotu časové nejistoty. Díky tomu se hodí k řízení náročných automatizačních celků, synchronizaci zařízení a polohovému řízení, kde je potřeba zpracovávat data od pohonů v reálném čase.

Základním principem komunikace je tzv. „Slot Communication Network Management“. Tento mechanizmus využívá časových slotů, které jsou v každém cyklu přidělovány jednotlivým zařízením na síti. Časové sloty jsou navíc rozděleny zvlášť na izochronní a asynchronní komunikaci. Tím je zajištěno, že dvě nebo více stanic v síti nezačnou vysílat data najednou ve stejný okamžik. Přidělování časových slotů, respektive řízení provozu sítě, má na starosti zařízení pracující v režimu Managing node.

3.1. Základní vlastnosti sítě Etheret Powerlink v bodech

- Přenos časově kritických dat v přesných izochronních cyklech
- Délka cyklu méně než 200µs
- Hodnota jitteru <1 µs
- Vhodné pro aplikace s polohovým řízením
- Výměna dat na principu producent/konzument
- Možnost synchronizace více zařízení v síti s vysokou přesností
- Přenos časově nekritických dat v asynchronní části komunikace
- Funkčnost protokolů z rodiny IP (TCP, UDP) a vyšších (http, FTP..)
- Až 240 zařízení v jednom segmentu sítě (doméně reálného času)
- Kompatibilita s CANopen profily zařízení (DS 301 a DS302)
- Možnost použití standardních ethernetových adaptérů (stejná fyzická vrstva)
- Cross traffic komunikace (přímá komunikace mezi controlled nodes)
- Variabilní topologie sítě

3.2. Referenční model



Obr. 3.1 Referenční model sítě Ethernet Powerlink

Z komunikačního modelu zařízení Ethernet Powerlink je vidět, jak je řešen mechanizmus přenosu izochronních a asynchronních dat. Přístup na síť je řízen pomocí NMT (Network Management), který rozděluje komunikaci na časově závislou – cyklická data z aplikační vrstvy a časově nezávislou – SDO objekty a standardní protokoly.

3.3. Fyzická vrstva

Jak již bylo zmíněno v předchozích odstavcích, tak fyzická vrstva Ethernet Powerlink je totožná s fyzickou vrstvou standardního ethernetu 802.3.

Při návrhu sítě Powerlink je doporučeno používat jako rozbočovače zařízení HUB, u kterých se předpokládá zpoždění signálu pod 500ns a malý jiter pod 70ns. Jako rozbočovače je možné použít také switche, nicméně potom je nutné počítat s řádově větším zpožděním signálu .

Lze využít různé možnosti topologie sítě. Výhodné je použití topologie Line, kde je využíváno obou dvou portů zařízení fungujících jako HUB a tím zjednodušit konstrukci sítě a délku použitých kabelů.

3.4. Linková vrstva

Síť Ethernet Powerlink může fungovat ve dvou operačních režimech. Jednak je to režim POWERLINK a druhý je BASIC ETHERNET. Jak je již z názvů zřejmé, tak v režimu Powerlink se provoz sítě řídí dle pravidel pro real-time komunikaci, kdežto v režimu Basic ethernet se provoz řídí dle standardního přístupu k médiu CSMA/CD (IEEE 802.3).

3.4.1. Režim Basic Ethernet

V tomto režimu se komunikace v síti řídí dle standardních pravidel pro ethernet CSMA/CD (IEEE 802.3) a tudíž je nedeterministická. Data jsou přenášena pomocí standardních protokolů UDP,TCP/IP a dalších protokolů vyšších vrstev.

3.4.2. Režim Powerlink

Režim Powerlink je založen na standardní technologii ethernetu CSMA/CD (IEEE 802.3), tudíž může fungovat na běžném hardware pro ethernet. Determinismus přenášených dat je dosažen díky plánování přenosu zpráv. Ty jsou sdružovány do cyklů, které jsou navíc děleny na izochronní a asynchronní fázi. Jednotlivá zařízení v síti dostávají oprávnění k přenosu svých rámců od Managing Node (MN). Tím je zajištěn bezkolizní přenos dat.

3.4.3. Typy zařízení

V síti Powerlink se vyskytují dva typy zařízení. Jedno, které přiděluje oprávnění ostatním k přenosu dat se nazývá Managing Node (MN). Zbylá zařízení se nazývají Controlled Node (CN)

3.4.4. Managing Node

Powerlink Managing Node (MN) je jediné zařízení, které může přenášet data nezávisle. To znamená, že není závislé na žádném oprávnění od jiného subjektu. Managing node cyklicky vyzývá pomocí unincastových zpráv (frame PReq) jednotlivé Controlled Nody, které poté pomocí multicast zpráv (frame PRes) publikují svá data všem zařízením v síti. Samotný Managing Node může také vysílat svoji multicast všem CN zprávu PRes, která se řadí na konec izochronní fáze.

V jednom segmentu sítě se může nacházet pouze jeden aktivní Managing Node, ve kterém jsou nakonfigurovány všechny ostatní zařízení CN.

3.4.5. Controlled Node

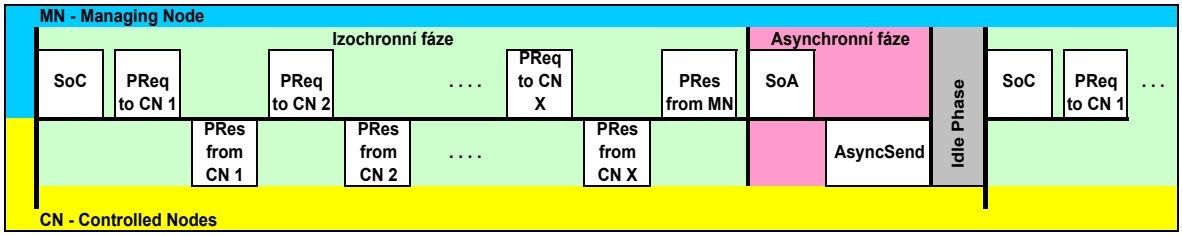
Zařízení typu Controlled Node (CN) je oprávněno vysílat cyklická data pouze tehdy, pokud k tomu je vyzváno MN pomocí rámce PReq. Poté následuje odpověď od CN pomocí PRes, která je typu multicast a mohou ji přijmout všechna zařízení.

Controlled Node může dostávat oprávnění vysílat data každý cyklus nebo každý n-tý cyklus. Tento režim se nazývá Multiplexed Station. Existuje také Asynchronní režim CN, ve kterém dané zařízení nedostává oprávnění vysílat v izochronní části cyklu, ale je dotazováno v asynchronní části, zda nemá nějaká data k přenosu (SoA frame – status request). I v tomto režimu je však schopno přijímat multicast zprávy od ostatních zařízeních v izochronní fázi.

3.4.6. Služby sítě Powerlink

- Izochronní přenos dat – přenos cyklických časově kritických dat
- Asynchronní přenos dat – přenos časově nekritických dat
- Synchronizace zařízení v síti – Synchronizace se provádí pomocí multicast rámce SoC (Start of Cyclic), který se nachází na začátku každé izochronní fáze cyklu. Synchronizace tímto způsobem je velmi přesná.

3.4.7. Cyklus sítě Ethernet Powerlink



Obr. 3.2 Cyklus sítě Ethernet Powerlink

Cyklus sítě se skládá ze tří částí:

- Izochronní fáze
- Asynchronní fáze
- Idle fáze

Na Obr. 3.2 je znázorněn průběh cyklu graficky. Cyklus je zahájen rámcem SoC, který musí být co nejpřesnější bez jitteru. Délka dalších fází cyklu se může již měnit v závislosti na konfiguraci a aktuálním stavu sítě, avšak celková délka cyklu sítě nesmí být překročena.

Izochronní fáze

Izochronní fáze je zahájena ihned po rámci SoC, který je generován na přísně periodickém základě. Poté je zahájeno dotazování aktivních konfigurovaných CN pomocí PReq rámců a jejich komunikace pomocí PRes. Po obsloužení všech zařízení může MN vyslat svůj vlastní PRes frame, pokud tak je nakonfigurován. Poté následuje frame SoA, který zahajuje asynchronní fázi.

Asynchronní fáze

V asynchronní části komunikace může být přístup k síti vyhrazen jednomu zařízení CN nebo MN pro přenos jedné zprávy. Přístup k síti v asynchronní části je přidělován na základě žádosti, která se nachází v PRes rámci daného CN. Přidělování přístupu je poté řízeno plánovačem v MN.

Pro asynchronní fázi existují dva typy zpráv. Powerlink ASnd frame, který používá adresování v síti powerlink a může být poslán jako unicast nebo jako broadcast kterémukoliv jinému zařízení. Druhým typem je standardní ethernet zpráva.

Idle fáze

Tato fáze je zbytkovým časem mezi koncem asynchronní části a začátkem dalšího cyklu (SoC). Během Idle fáze žádné zařízení nepřenáší data.

3.4.8. Adresování v síti Powerlink

Každé zařízení v segmentu sítě Powerlink má unikátní Node ID, které ho jednoznačně identifikuje. Jedná se o jednobajtové číslo, kde jsou hodnoty mezi 1 a 239 vyhrazené pro zařízení typu Controlled Node. Node ID 240 je vždy automaticky přiděleno Managing Node. 0 je zakázaná hodnota. Ostatní hodnoty jsou vyhrazeny pro speciální účely viz. [12].

3.4.9. Struktura Rámců Ethernet Powerlink

Zprávy Ethernet Powerlink jsou zapouzdřeny v rámcích Ethernet II. Délka rámce je omezena na konfigurovanou velikost, aby byla dodržena požadovaná délka cyklu.

Základní powerlinkový rámec obsahuje následující položky:

- Reserved (1 bit)
- Message type (7 bitů) - Typ rámce
- Destination node adress (1 bajt) – Cílová adresa
- Souce node adress (1 bajt) – Adresa zdroje zprávy
- Payload (n bajtů) – Užitečná data

	Bit offset								
Byte offset	7	6	5	4	3	2	1	0	Rámec
0..5	Cílová MAC adresa								Ethernet II
6..11	Zdrojová MAC adresa								
12..12	Typ ethernet zprávy								
14	res	Typ Powerlink zprávy							Ethernet Powerlink
15	Cíl								
16	Zdroj								
17..n	Data								
n+1..n+4	CRC32 - kontrolní součet								Ethernet II

Tab. 3.1 Struktura rámce Ethernet Powerlink

3.4.10. Druhy zpráv v síti Ethernet Powerlink

Typ zprávy	ID/Zkratka	Typ přenosu
Start of Cycle	SoC	Multicast
PollRequest	PReq	Unicast
PollResponse	PRes	Multicast
Atart of asynchronous	SoA	Multicast
Asynchronous Send	ASnd	Multicast

Tab. 3.2 Typy zpráv v síti Ethernet Powerlink

Start of Cycle (SoC)

Rámec Start of Cycle se nachází vždy na začátku každého powerlink cyklu. Jeho poloha v čase je přísně deterministická, tudíž lze právě od něj provádět synchronizaci zařízení. Nepřenáší žádná procesní data.

Poll Request (PReq)

Poll Request přiřazuje oprávnění jednotlivým stanicím vysílat data. Je vždy určen jednomu konkrétnímu zařízení (Unicast). Může obsahovat procesní data od MN pro daný CN.

Poll Response (PRes)

Rámec Poll Response je odpovědí CN na oprávnění vysílat Poll Request. Obsahuje procesní data od daného CN pro všechny stanice v síti (broadcast)

Start of Asynchronous (SoA)

Tento rámec vysílaný Managing Nodem ukončuje izochronní část cyklu a začíná asynchronní fázi. Je v něm obsažena informace o typu akce, která bude v asynchronní části probíhat. Neobsahuje žádná procesní data.

Asynchronous Send (ASnd)

Asynchronous Send slouží k přenosu časově nekritických dat v asynchronní části komunikace. Stejně jako SoA obsahuje informaci o jaký typ akce se jedná. Podrobnosti viz. [12]

Rámce jiných protokolů

Všechny ostatní rámce mohou být přenášeny v asynchronní fázi cyklu na základě žádosti daného CN a povolení od MN.

3.5. Síťová + transportní vrstva

Fungování síťové a transportní vrstvy se týká především asynchronní části komunikace. Pro tyto vrstvy jsou preferovány protokoly IPv4 (RFC 791) a transportní protokol TCP (RFC 793) a UDP (RFC 768)

3.5.1. IP adresování v síti powerlink

Každé zařízení v síti Powerlink schopné pracovat s protokolem IP obdrží adresu, masku sítě a výchozí bránu. Tyto parametry se řídí následujícími pravidly:

IPv4 Adresa

Jako adresa sítě je doporučeno použít adresu třídy C 192.168.100.0. Poté je adresa zařízení tvořena způsobem 192.168.100.Powerlink Node ID

Maska sítě

Maska sítě by měla odpovídat síti třídy C 255.255.255.0

Výchozí brána

Výchozí brána by měla být nastavena na 192.168.100.254. Může být však změněn na libovolnou jinou hodnotu.

3.6. Aplikační vrstva

Jak lze vyčíst z Obr. 3.1, tak aplikační vrstva Ethernet Powerlink je založena na struktuře, která se nazývá Object Dictionary. Object Dictionary je částečně pevně definovaná struktura obsahující záznamy/objekty, ke kterým lze přistupovat například ze sítě, nebo aplikace a zařízení tímto způsobem ovládat či konfigurovat. Existují různé profily zařízení udávající strukturu části Object Dictionary, pomocí které lze dané zařízení ovládat nebo konfigurovat. Tyto profily mohou být různé například pro servozesilovače, ventily, senzory aj. Podrobnější popis aplikační vrstvy je nad rámec tohoto dokumentu a lze jej nalézt v dokumentaci [12].

Kapitola 4

4. Realizace zabezpečení

4.1. Získávání informace o poloze a rychlosti

Jak již bylo v představeno v kapitole 2.4, tak získávání poloh a rychlostí jednotlivých os systému je řešeno analyzováním síťové komunikace Ethernet Powerlink. Každá osa systému je řízena servozesilovačem Acopos, který je vybaven rozhraním pro Ethernet Powerlink a je jím připojen do sítě.

Tímto způsobem komunikují samotné Acoposy jednak mezi sebou a zároveň s hlavním řídícím PLC Power Panelem 420. Vzájemná komunikace je realizována pomocí Poll Response rámců, ve kterých jsou obsažena potřebná data. Informace o poloze a rychlosti se v rámci nachází na pevném místě, které se s časem nemění, tudíž je možné potřebná data jednoduše každý cyklus z rámců získávat.

4.1.1. Poll Response rámec Acoposu

Na Obr. 4.1 je zobrazen příklad Pool response rámce od Acoposu. Šedou barvou jsou zvýrazněna užitečná přenášená data (payload) a v nich červeně ohraničené informace o poloze a rychlosti. Důležitou informací je offset dat v části Payload, který je pro polohu 16 bajtů a pro rychlost 20 bajtů.

```
0000  01 11 1e 00 00 02 00 60  65 00 49 02 88 ab 04 ff   .......` e.I.....
0010  02 fd 01 00 00 00 4c 00  89 01 bc 3d 84 42 00 00   ......L. ...=.B..
0020  00 00 00 af 11 00 10 00  00 00 00 00 00 00 00 00   ........ ........
0030  00 00 00 00 00 00 00 00  c7 00 7d 01 fe ff 15 43   ........ ..}....C
0040  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
0050  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
0060  00 00 00 00                                        ....
```

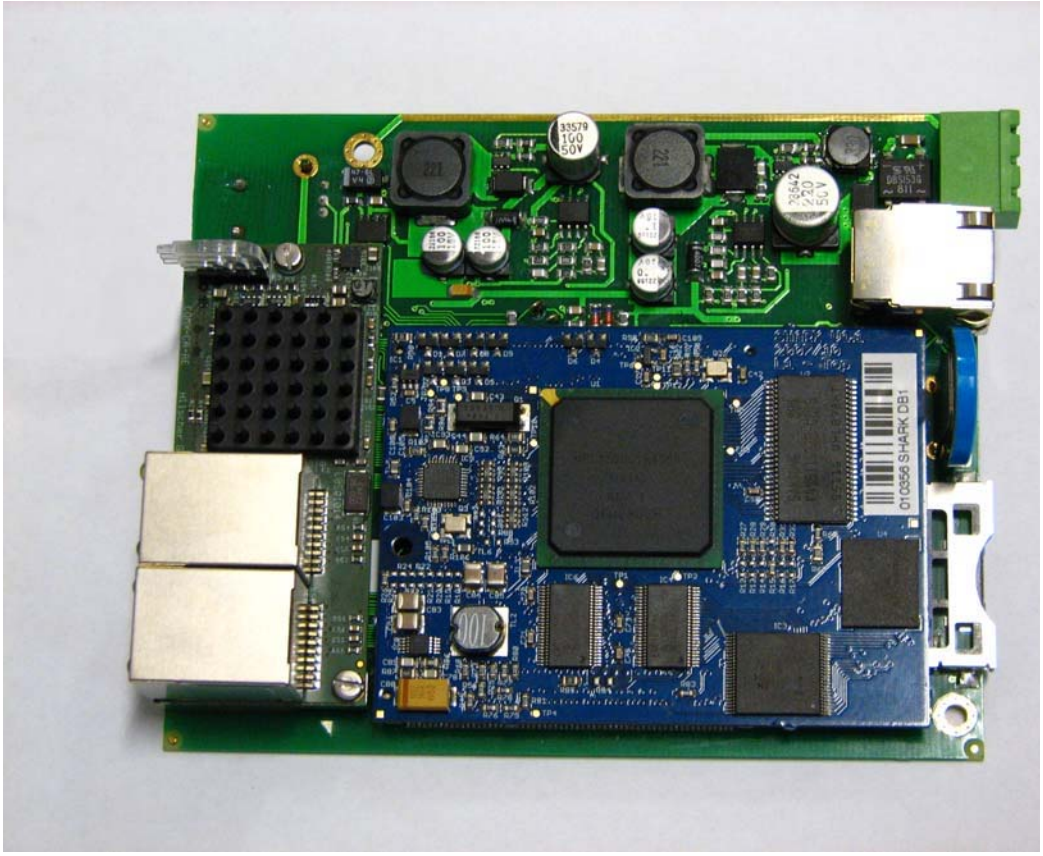
Obr. 4.1 PoolResponse rámec Acoposu

Se známou polohou potřebných dat v rámci, který je v síti přístupný všem ostatním zařízením je možné tyto data nějakým způsobem získávat a analyzovat. V následujících odstavcích bude popsán způsob řešení pomocí desky Sharks procesorem MPC 5200.

4.2. Řešení s deskou Shark a modulem NetX

Pro zabezpečení stroje nebylo uvažováno pouze využití standardního HW, který je k dispozici od B&R automotion, ale chtěli jsme vyzkoušet možnosti použití i jiných řešení. Druhým způsobem zabezpečení je použití desky SHARK (Obr. 4.2) od společnosti Microclima s procesorem MPC 5200 architektury Power PC. Deska obsahuje také modul ComX s implementovaným stackem Ethernet Powerlink, pomocí kterého je možné komunikovat po průmyslových sběrnicích používajících jako fyzické médium ethernet, tudíž i Ethernet Powerlink. Druh použité průmyslové sítě je závislý na aktuálním nahraném firmware s příslušným stackem.

Cílem bylo tyto dva přístupy mezi sebou porovnat a získat tak maximální možný výkon ve smyslu zkrácení reakční doby na nebezpečnou situaci. Dalším důvodem byla redundance těchto dvou systémů a tím pádem zajištění větší míry bezpečnosti.



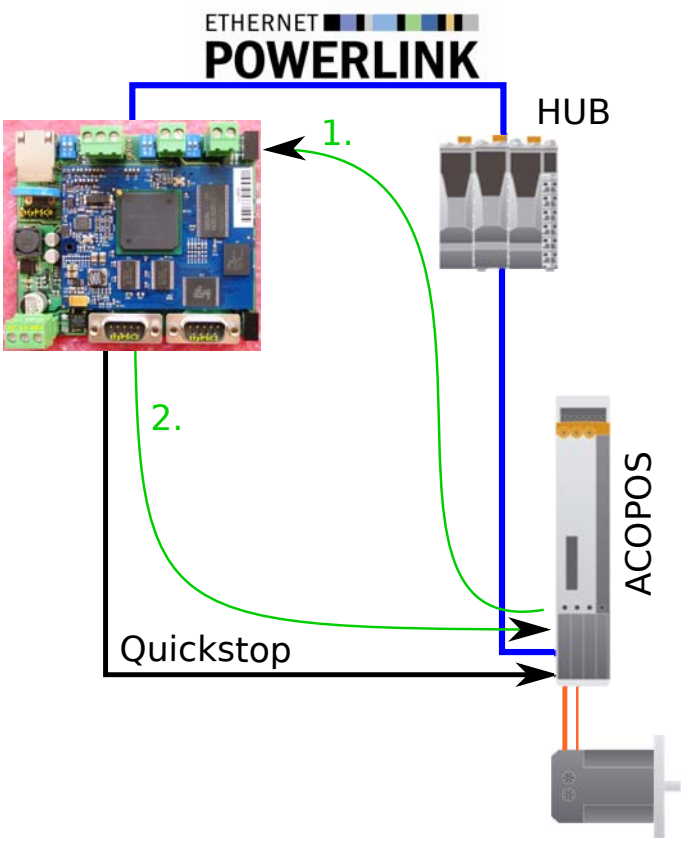
Obr. 4.2 Deska SHARK s modulem ComX

Jako první krok bylo nutné desku implementovat vhodným způsobem do systému. V hardwarové konfiguraci Power Panelu byla deska přidána jako Generic Powerlink Station, což je typ stanice používaný pro zařízení jiných výrobců než B&R. Přidáním

do konfigurace je deska zařazena do cyklické komunikace a je mezi ní a Power Panelem možná výměna dat. Za tímto účelem byla na straně PP420 využita knihovna Powerlnk popsaná v části 4.5.1.

Pro funkci zabezpečení je však zásadní komunikace mezi modulem ComX a jednotlivými Acoposy. Protože veškerá cyklická data od Acoposů jsou broadcastové zprávy, jsou k dispozici přímo modulu ComX v každém cyklu sběrnice bez nutnosti přenášení dat přes Power Panel. Tento přístup získávání aktuálních poloh a rychlostí je jednou ze zásadních výhod oproti výslednému řešení s použitím standardního hardware B&R, které bude popsáno v dalších částech této práce. Nicméně, aby mohl být tento způsob získávání dat realizován, je nutné, aby stack v modulu ComX podporoval takzvanou cross komunikaci, což je výměna dat jednotlivými Controlled Nody. S aktuální verzí stacku, která je volně zdarma k dispozici, je možná pouze komunikace mezi ComX jako Controlled Nodem a PP420 jako Managing Nodem. Díky tomuto faktu jsem se zaměřil na vývoj zabezpečovacího systému s využitím standardního hardware od B&R.

I přes tuto překážku je však užitečné toto řešení uvažovat v případě, že by v budoucnu byl k dispozici stack umožňující CN – CN komunikaci.



Obr. 4.3 Schéma zabezpečení pomocí desky SHARK a modulu ComX

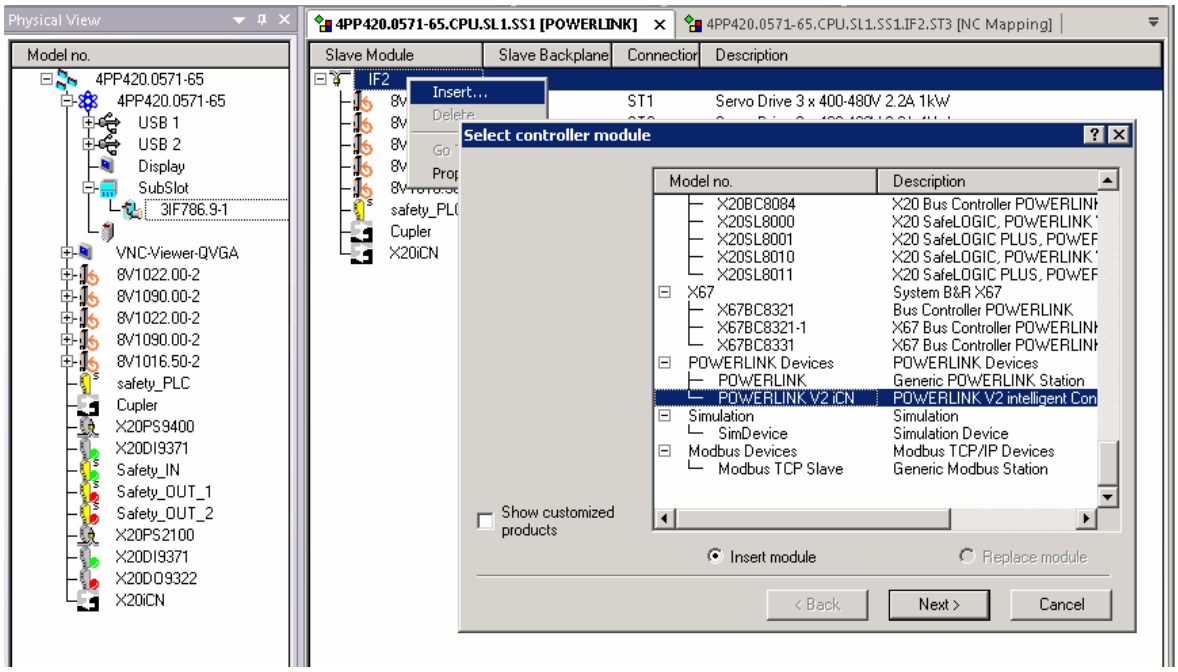
4.3. Řešení zabezpečení pomocí hardware B&R

Řešení využívající hardware B&R je založeno na přidání samostatného PLC do původního systému, ve kterém bude probíhat kontrola poloh a rychlostí os systému. Bohužel s dostupnými knihovnami není možné implementovat cross komunikaci mezi kontrolním PLC a Acoposy, tudíž bylo nutné zvolit řešení, kdy informace z Acoposů jsou čteny Power Panelem jakožto Managing Nodem a následně přes něj předávány do kontrolního PLC. Z kontrolního PLC bude informace o případném nouzovém zastavení systému přenášena do safety PLC a safety výstupů ovládající Quickstop vstupy Acoposů.

4.4. Použitý hardware a jeho konfigurace

Jako PLC pro zabezpečovací funkci je použito PLC X20 CP1485. Obsahuje procesor Celeron 400 a komunikační rozhraní USB, RS232, Ethernet a Ethernet Powerlink. Dále je možné přidat ještě jeden modul do volného slotu.

Do hardwarové konfigurace je X20 přidáno jako Inteligent Controlled node. Toto nastavení se používá pro PLC, kdy je použito v síti jako CN. V projektu je nejprve nutné vytvořit novou konfiguraci v Configuration view. Pro PLC X20 je nazvána X20iCN. Poté je možné přidat takovéto zařízení do sítě powerlink. V záložce Powerlink Power Panelu se zadá Insert a následně vybere z nabízených možností Powerlink V2 iCN (Obr. 4.4). V dalších krocích se vybere konfigurace, která byla předtím vytvořena. Tím je PLC přidáno do konfigurace a je možné vytvářet komunikační kanály mezi těmito zařízeními.



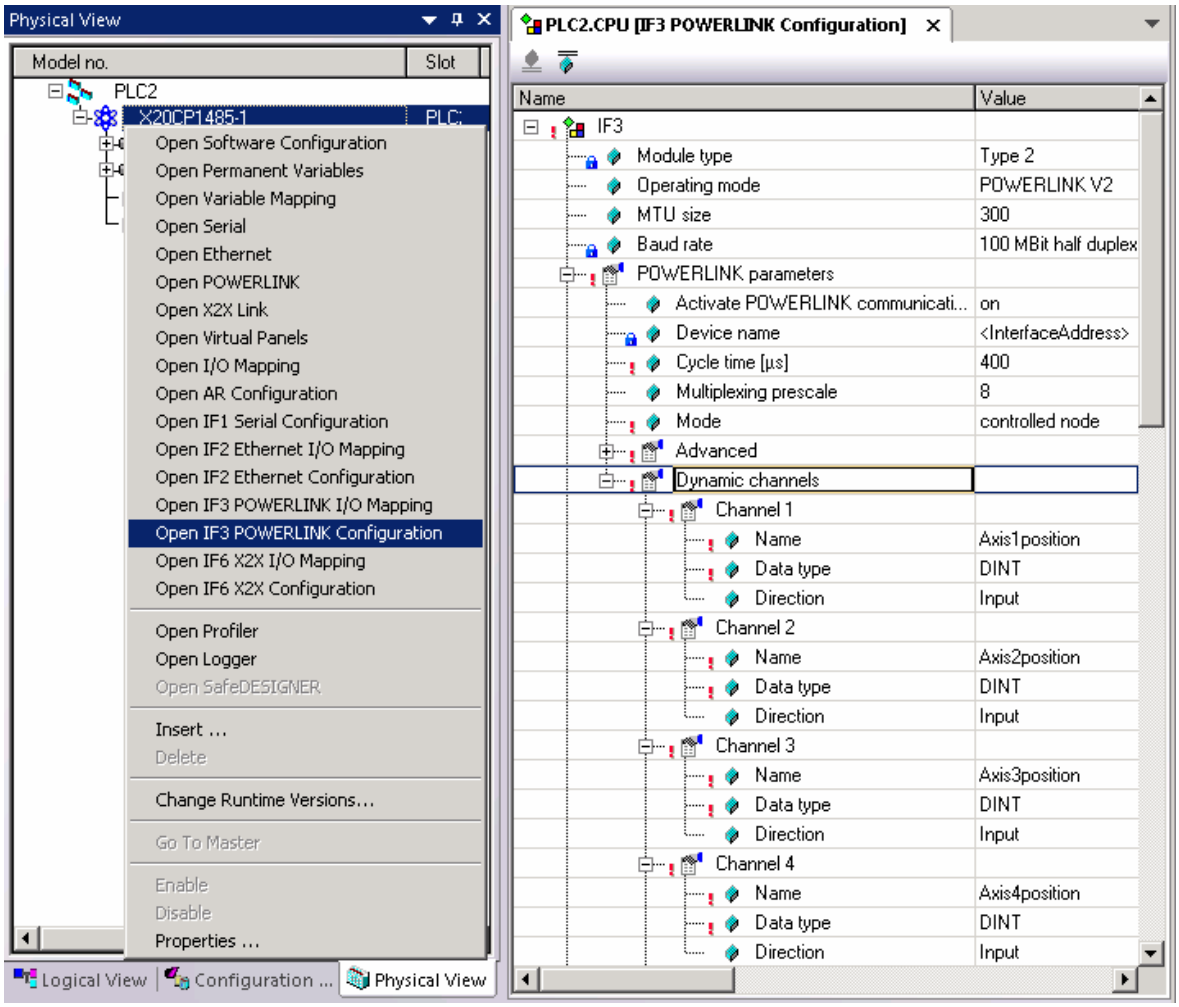
Obr. 4.4 Přidání PLC X20 do projektu

4.4.1. Dynamické kanály

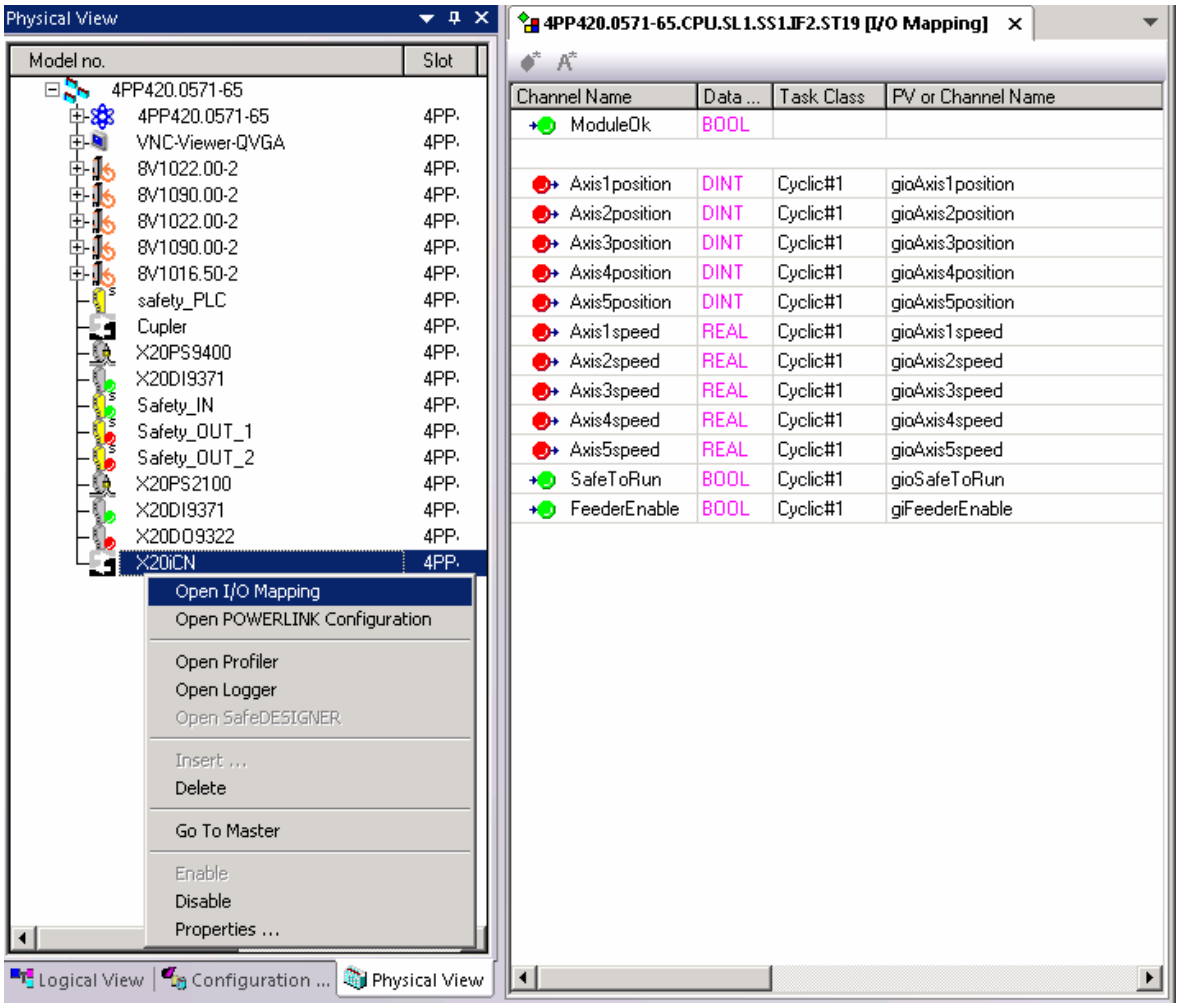
Dynamické komunikační kanály slouží k cyklické výměně procesních dat mezi zařízeními. Pro účely zabezpečení je potřeba takováto data vyměňovat mezi PP420 a X20. Zde je důležité, aby bylo PLC X20 v projektu přidáno jako Inteligent Controlled node, jak bylo ukázáno v předchozím odstavci.

Vytváření dynamických kanálů se provádí nejprve v konfiguraci X20 v nastavení Powerlink configuration (Obr. 4.5). Zde je možné modifikovat parametry komunikace přes ethernet powerlink. Pro komunikaci mezi PLC je důležitá položka Dynamic channels. Jeden kanál odpovídá jedné přenášené proměnné mezi zařízeními. U kanálu se definuje jméno, datový typ přenášené proměnné a směr přenosu z hlediska X20.

Po vytvoření kanálů je možné do nich mapovat proměnné. Mapování v konfiguraci X20 se nachází v nastavení IF3 Powerlink I/O mapping, kde jsou po vytvoření kanálů všechny zobrazeny. V konfiguraci PP420 se kanály objeví v I/O mapping pro X20 (Obr. 4.6), nikoliv v I/O mapping pro IF2.



Obr. 4.5 Vytváření dynamických kanálů na PLC X20

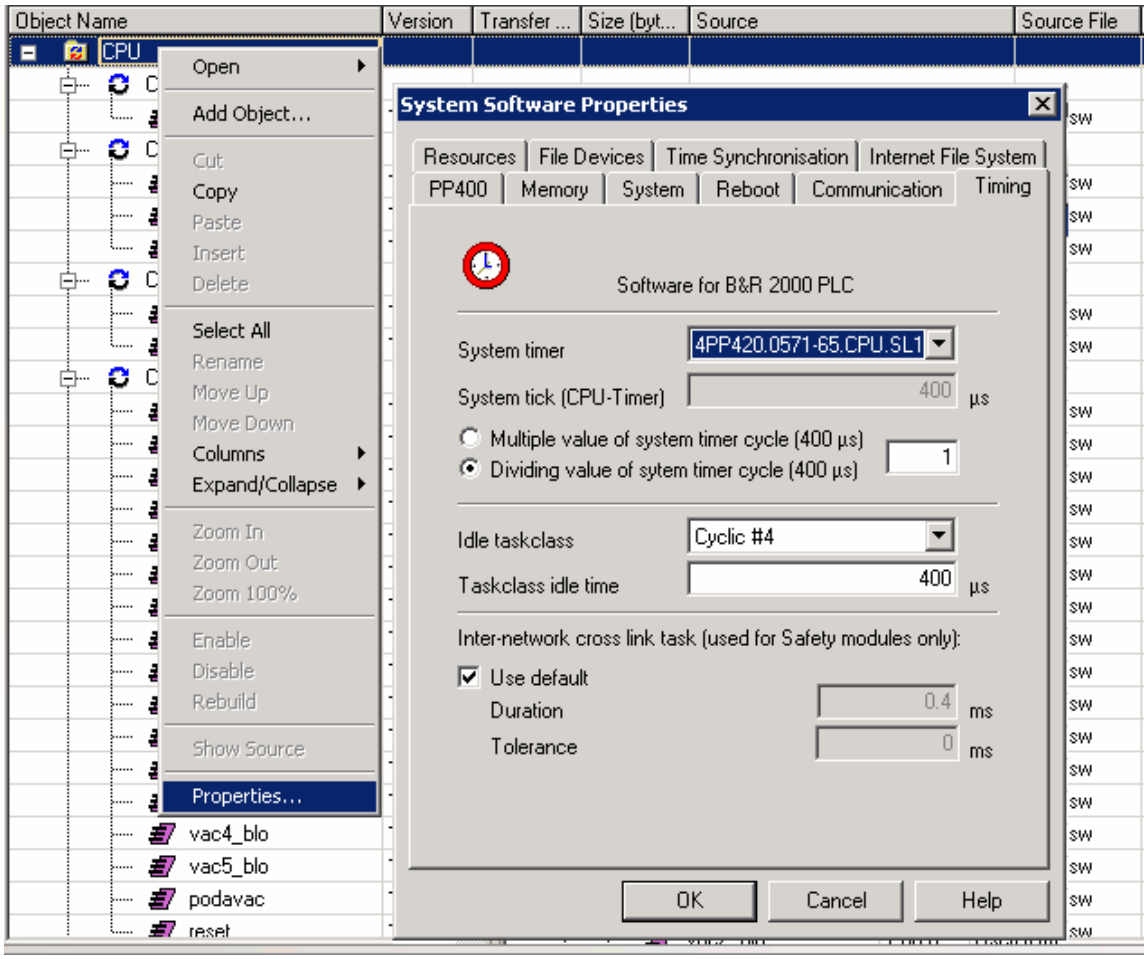


Obr. 4.6 Mapování proměnných z PP420 do X20

4.4.2. Synchronizace PLC a sítě Powerlink

Pro správnou funkci zabezpečení, je potřeba aby byla ze sítě zpracována všechna data bez vynechání některého cyklu sítě. To lze zajistit právě synchronizací cyklu PLC s cyklem sítě Powerlink. Díky synchronizaci dosáhneme také konzistentní časové odezvy při komunikaci mezi jednotlivými zařízeními.

Natavení se provádí ve vlastnostech daného CPU. CPU -> Properties -> Timing -> System timer (Obr. 4.7). Jako zdroj systémového času je nutné nastavit powerlink interface a v možnostech násobení/dělení systémového času zadat 1. Toto nastavení bylo potřeba udělat jak pro CPU X20 tak i pro Power panel 420. Postup byl vždy stejný. Systém timer se tudíž odvíjí od cyklu sítě Powerlink, která má v projektu nastavený cyklus na hodnotu 400 µs (nastaveni Powerlink configuration).



Obr. 4.7 Nastavení časování CPU a synchronizace se sítí Ethernet Powerlink

4.4.3. Časování tříd cyklických úloh

Mimo systémového časování, které bylo popsáno výše bylo potřeba také nastavit časování jednotlivých cyklických tříd úloh, ve kterých se budou spouštět časově kritické algoritmy pro zabezpečení nebo se v nich bude pracovat s daty pro zabezpečovací účely.

Na každém PLC byla pro tyto účely použita cyklická třída Cyclic #1. Ve vlastnostech dané třídy (Obr. 4.8) jsem nastavil periodu spouštění na 400 µs, což je při časování systému nejnižší možná hodnota. Díky této hodnotě je zajištěno, že budou data z každého cyklu sítě stihnuta být zpracována.

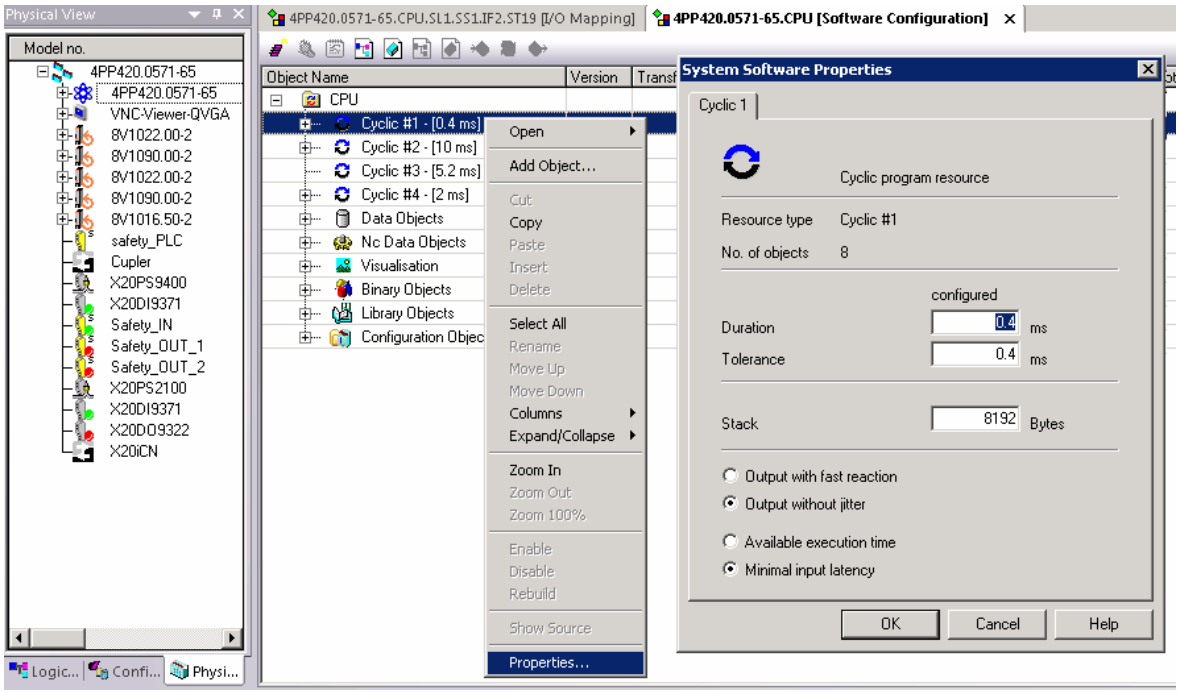
Další potřebné nastavení je tolerance periody cyklické třídy, která udává její maximální možné překročení. Pro časově kritické třídy úloh by bylo logické použít nejmenší možnou hodnotu, což ale nebylo možné kvůli problémům při nahrávání nových částí projektu do PLC a následného restartování PP420. Pokud byl proveden restart PP420 a tím pádem přerušena na nějaký čas komunikace přes Ethernet Powerlink, se kterou je cyklická úloha v X20 synchronizována, nebyla tolerance dodržena. To mělo za následek fatální chybu

„cycle time violation“ po níž bylo PLC restartováno do režimu Service. Tím pádem přestala fungovat nejen cyklická třída pro zabezpečení, ale i komunikace pomocí IMA (viz. 4.10.4) a tudíž nebylo možné restartovat X20 zpět do režimu RUN bez pomoci projektu pro X20 v Automation Studiu a připojení přes Ethernet, které bude pro studenty nedostupné.

Z uvedených důvodů byla tolerance nastavena na dobu 5,2ms, která se v praxi ukázala jako dostatečná aby nedocházelo k restartování PLC do režimu Service. Zde je vhodné podotknout, že tolerance byla překračována právě pouze v případě restartu PP420 a při standardním běhu programu byla po celou dobu testování dodržena, tudíž není pravděpodobné, že by toto nastavení ohrozilo bezpečnost zabezpečovacího algoritmu.

V nastavení pro třídu Cyclic #1 jsem navíc zvolil volbu „Output with fast reaction“, která zajišťuje nejrychlejší možné zapsání proměnných spojených s výstupem. V mém případě se to týká proměnné pro zastavení stroje – `goSafeToRun`.

Dále jsem zvolil volbu „Minimal input latency“. Díky této volbě je vykonávání cyklu spuštěno až v okamžiku, kdy jsou k dispozici vstupní data ze sítě Powerlink. Tím je možné dosáhnout zpracování dat ještě v tomtéž cyklu, ve kterém byla přijata.



Obr. 4.8 Nastavení časování třídy cyklických úloh

4.5. Implementace algoritmu – získávání dat

4.5.1. Získávání dat z rámců Acoposů – softwarová implementace

Získávání informací o poloze a rychlosti je softwarově implementováno v souboru powerlink.c, který je umístěn ve třídě Cyclic #1 v Power panelu. Pro vyčítání dat z rámců je použita knihovna powerlnk obsahující funkce pro rozhraní Ethernet Powerlink. Konkrétně jsem použil funkci `plCECreate()` obsahující 4 parametry – datapoint, adress, taskclass a pIdent.

Datapoint

Parametr datapoint obsahuje string definující místo, odkud se mají data kopírovat/zapisovat. V mém případě byla použita hodnota "SL1.SS1.IF2.%ID1.0.16", kde SL1.SS1 definuje procesor, IF2 interface, ID definuje směr (Input) a datový typ (Double), 1 udává číslo powerlink node, ze kterého mají být data čtena, 0 značí definici místa pomocí Byte offsetu a 16 je bajtový ofset. V tomto případě se tudíž jedná o datapoint pro pozici osy 1.

Adress

Parametr adress je adresa proměnné v PLC, kam/ze které mají být data čtena/zapisována.

Taskclass

Taskclass je třída úloh, ve které má být cyklické kopírování hodnoty prováděno a pIdent je adresa vytvořeného datapointu.

Pomocí funkcí plCECreate() jsou vytvořeny cyklické kopírovací instrukce, které získávají automaticky každý cyklus ze sítě aktuální data a kopírují je do příslušné proměnné. Takto jsou data k dispozici v PP420 a mohou být přenesena do X20.

4.5.2. Přenos dat mezi PP420 a X20

Pro přenos dat mezi PP420 a PLC X20 jsou použity vytvořené kanály mezi těmito zařízeními dle popisu v odstavci 4.4.1. Konkrétně se přenášejí údaje o rychlosti, poloze os a proměnná povolující pohyb os a uvolňování koule z podavače.

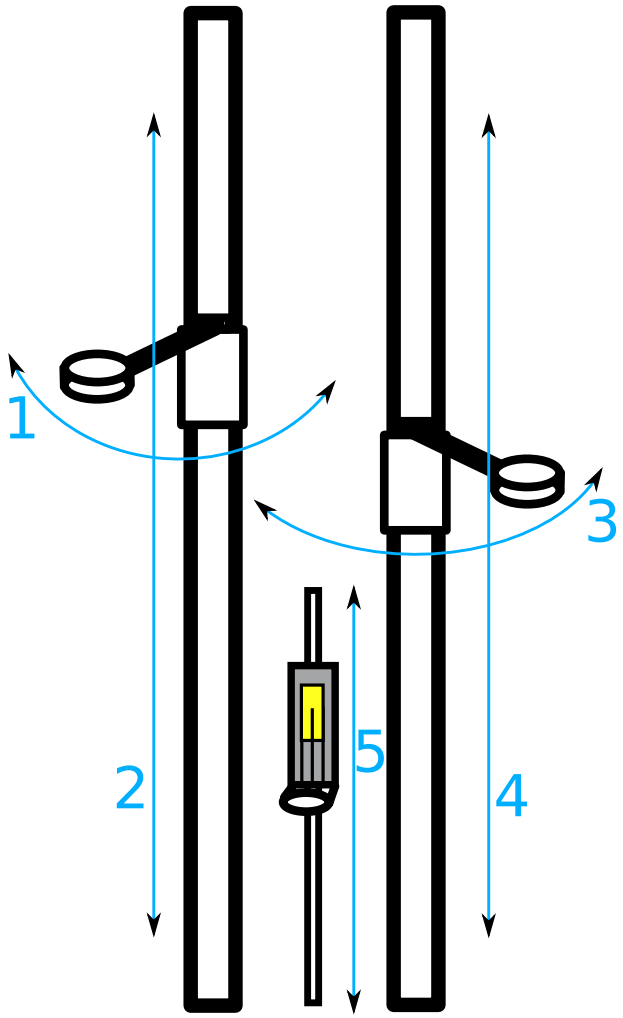
Název kanálu	Proměnná v PP420	Proměnná v X20	Směr přenosu
Axis1position	gioAxis1position	giAxis1position	PP420 → X20
Axis2position	gioAxis2position	giAxis2position	PP420 → X20
Axis3position	gioAxis3position	giAxis3position	PP420 → X20
Axis4position	gioAxis4position	giAxis4position	PP420 → X20
Axis5position	gioAxis5position	giAxis5position	PP420 → X20
Axis1speed	gioAxis1speed	giAxis1speed	PP420 → X20
Axis2speed	gioAxis2speed	giAxis2speed	PP420 → X20
Axis3speed	gioAxis3speed	giAxis3speed	PP420 → X20
Axis4speed	gioAxis4speed	giAxis4speed	PP420 → X20
Axis5speed	gioAxis5speed	giAxis5speed	PP420 → X20
SafeToRun	gioSafeToRun	goSafeToRun	X20 → PP420
FeederEnable	giFeederEnable	goFeederEnable	X20 → PP421

Tab. 4.1 Seznam proměnných pro cyklickou komunikaci

V předchozích částech byl popsán způsob získávání a přenos dat pro analýzu a zabezpečení os žongléra. Nyní je možné tyto hodnoty zpracovávat zabezpečujícím algoritmem, který bude popsán v dalších odstavcích.

4.6. Teoretické řešení algoritmu

Algoritmus hlídá polohy všech pěti os systému. Jejich uspořádání a číslování je znázorněno na Obr. 4.9.



Obr. 4.9 0sy Žongléra a jejich číslování

Z hlediska bezpečnosti je nutné předcházet překročení koncových poloh jednotlivých os. U vertikálních os systému (2,4,5) jsou krajní polohy opatřeny koncovými spínači zajišťující bezpečnostní funkci. Algoritmus v tomto případě funguje jako pojistka, při selhání a překročení těchto spínačů. U horizontálních os systému (1,3) je koncovým spínačem hlídána pouze vnější krajní poloha. Algoritmus tudíž zajišťuje plnou funkci zamezující překročení limitních poloh. Na vnější straně zabraňuje nárazu vyhazovací misky do lineárního modulu. Na vnitřní taktéž a navíc na úseku, kde se nachází lineární motor kontroluje případnou kolizi s magnetickou tyčí a jejím upevněním, u kterého je nebezpečí zlomení vyhazovací ručky zvlášť vysoké.

Hlavní funkcí algoritmu je však zamezení vzájemné srážky vyhazovacích ruček, ke kterému může dojít pokud jsou obě ručky poblíž vnitřní krajní polohy a vzájemně se míjejí.

4.6.1. Výpočty a vzorce použité pro zabezpečení

Predikční horizont algoritmu

Predikčním horizontem algoritmu se rozumí čas, který uplyne od okamžiku vzniku nebezpečné situace do okamžiku, kdy se aktivuje akce na její odstranění. V praxi to znamená zahrnout čas potřebný pro přenos dat po síti, čas potřebný na běh algoritmu a dobu přenosu informace o zastavení na příslušný výstup. Celkový čas se potom vypočítá:

$$T_p = T_{IN} + T_{CPU} + T_{OUT} \tag{1}$$

Konkrétní hodnoty časů společně s postupem jak jsem je získal je uvedeno v následující části práce 4.7.

Výpočet dosažitelných poloh na horizontu predikce

Jako první algoritmus počítá teoreticky dosažitelné polohy os. Z matematického hlediska se jedná o výpočet rovnoměrně zrychleného pohybu na daném časovém intervalu – horizontu predikce, který je dán dobou mezi vznikem nebezpečného stavu systému a okamžikem, kdy je započata akce na jeho odstranění (přenos a vyhodnocení informace).

Pro každou osu je predikována poloha s uvažováním maximálního možného zrychlení v pozitivním a v negativním směru.

$$s = v_0 T_p + \frac{1}{2} a_{acc} T_p^2 \tag{2}$$

kde jsou proměnné:

s … predikovaná poloha osy

v_0 … aktuální rychlost osy

T_p … horizont predikce

a_{acc} … maximální hodnota zrychlení osy (pozitivní a negativní směr)

Kontrola bezpečného stavu

Výsledné teoreticky dosažitelné polohy z předchozího výpočtu jsou poté porovnávány s hodnotami, které nesmí být překročeny (krajní polohy, vzdálenosti jednotlivých os …). V matematickém vyjádření pak platí:

$$
\begin{gathered}
s_{lim} > s + vt - \frac{1}{2} a_{dec} t^2 \\
t = \frac{v}{a_{dec}} \\
\hline
|s_{lim} - s| > \frac{v^2}{2a} \\
v < \sqrt{2a\,|s_{lim} - s|}
\end{gathered}
\tag{3}
$$

kde jsou proměnné:

s_{lim} … poloha, která nesmí být dosažena

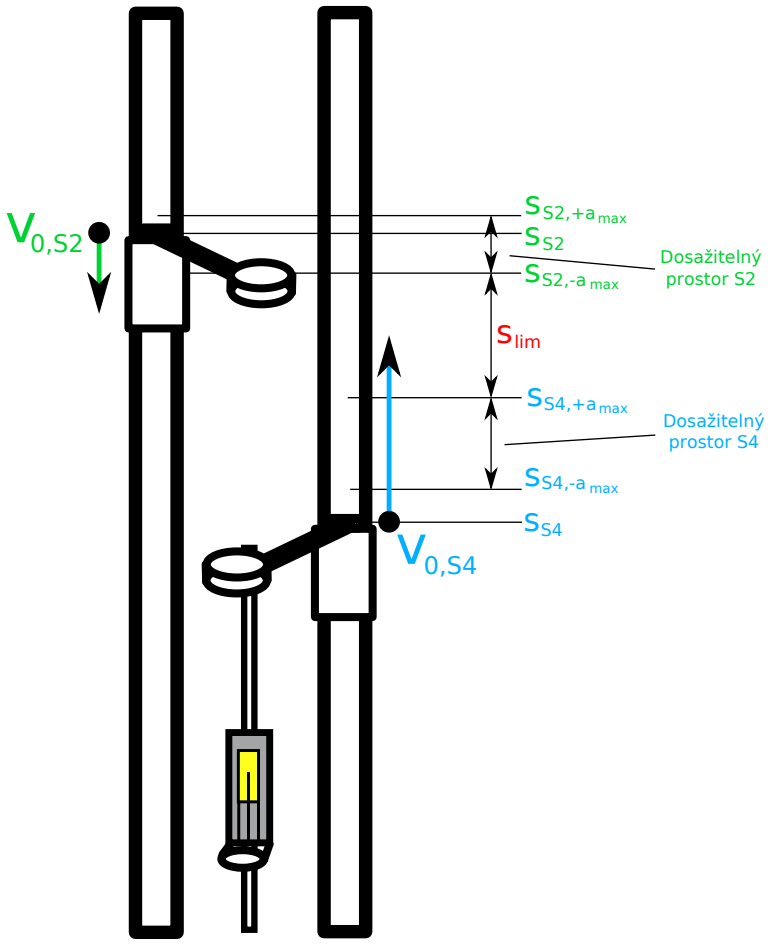
s … predikovaná poloha osy

v … predikovaná rychlost osy

a_{dec} … zrychlení osy při brzdění (zpomalení)

t … čas brzdění na nulovou rychlost

Pro názornost jsou proměnné znázorněny v následujícím obrázku:



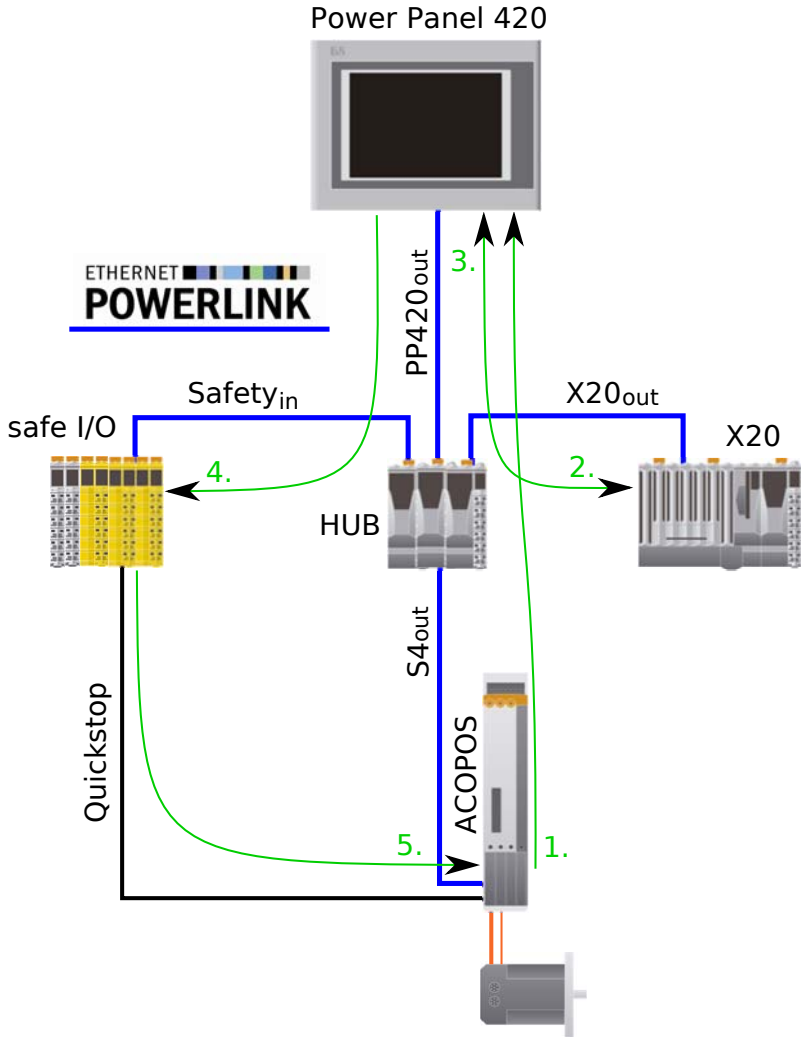
Obr. 4.10 Znázornění významu proměnných použitých v teoretickém rozboru

Na obrázku Obr. 4.10 je znázorněna situace, kde díky pozici vyhazovacích ramének existuje jejich potencionální srážka. Osy 2 a 4 se pohybují směrem proti sobě rychlostí $v_{0,S2}$ a $v_{0,S4}$. Díky svým maximálním zrychlením a_{acc} jsou schopny na horizontu predikce T_p dosáhnout poloh $S_{S2,+a_{\max}}$, $S_{S2,-a_{\max}}$, $S_{S4,+a_{\max}}$, $S_{S4,-a_{\max}}$.

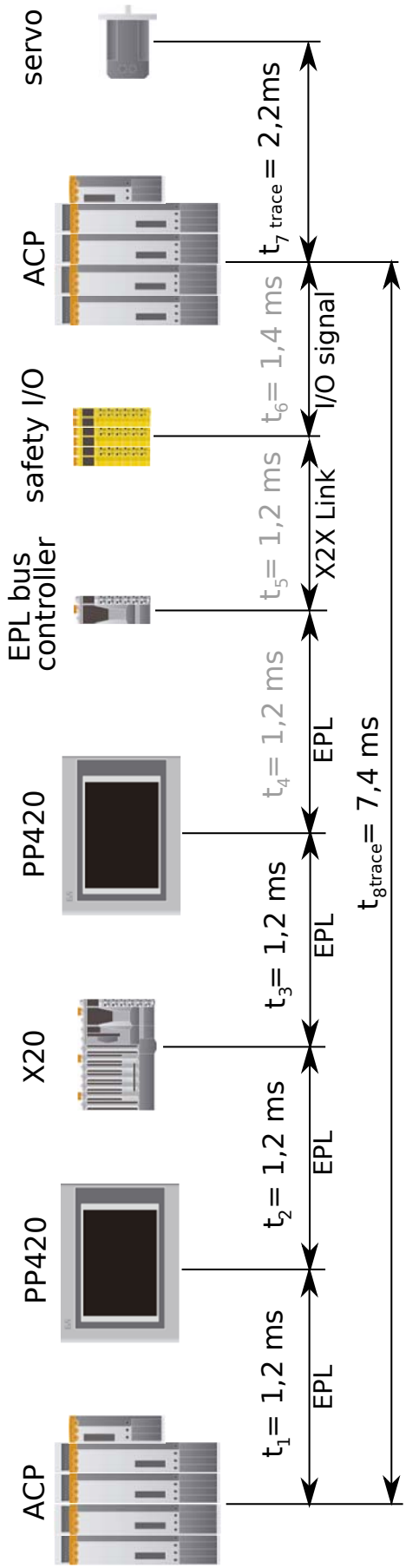
Tyto dosažitelné polohy jsou poté použity ve vztahu (3). V tomto konkrétním případě je bráno v potaz, že se protisměrné predikované rychlosti sčítají.

4.7. Praktický test zabezpečení

Na Obr. 4.11 a Obr. 4.12 je znázorněna datová cesta společně s vyznačenými změřenými časy. Tato situace byla vytvořena podmínkou, která při překonání polohy větší než 10 000 unitů stroj zastavila. Během testu jsem zaznamenával potřebné hodnoty pomocí programu Wireshark na analýzu komunikace sítě.



Obr. 4.11 Přenos informace sítí Ethernet Powerlink



Obr. 4.12 Časy přenosu informace sítí Ethernet Powerlink

Na Obr. 4.12 je znázorněna cesta informace sítí společně s časovými zpožděními, které jednotlivé prvky v síti způsobují. Časové zpoždění je myšleno vždy od doby kdy se data objeví na výstupu prvku nalevo do doby, kdy se objeví na výstupu prvku napravo. Zpoždění s dolním indexem trace byly získány pomocí nástroje Trace a praktického testu, ostatní byly zjištěny analýzou komunikace pomocí programu Wireshark. Hodnoty uvedené šedou barvou jsou pouze odhadované na základě teoretického předpokladu o cyklu zařízení a cyklu komunikačního média (Powerlink, X2X link) a byly dopočítány na základě hodnoty t_{8trace}. Z výsledných časů je patrné, že se teoretické hodnoty až na drobné odchylky shodují s praktickým měřením.

Z výsledků měření jsem zjistil následující hodnoty:

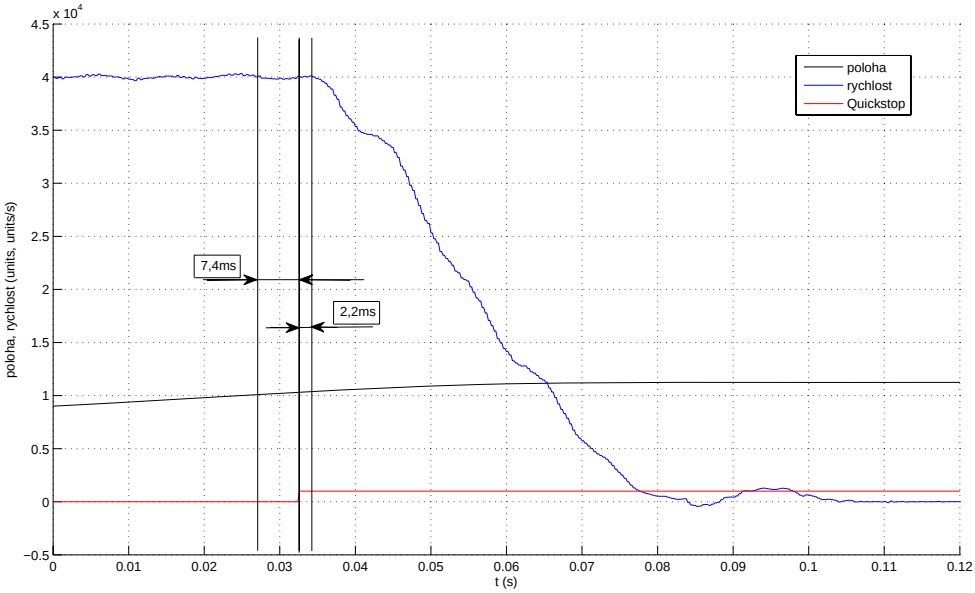
$T_{IN} = 1,2\,ms$

$T_{CPU} = 1,2\,ms$

$T_{OUT} = 7,2\,ms$

$T_p = 9,6\,ms$

Díky nástroji Trace, který je k dispozici v Automation Studiu je možné zaznamenávat parametry os systému v závislosti na čase se zvolenou časovou přesností. Na Obr. 4.13 je zobrazena situace odpovídající předchozímu měření a v ní jsou vyznačeny časy použité pro odvození výsledků uvedených v předchozím odstavci.



Obr. 4.13 Trace osy při zastavení s vyznačenými časy

4.8. Parametry algoritmu

Kromě časových parametrů obsahuje algoritmus další údaje, které bylo potřeba správně identifikovat. Jedná se zejména o hodnoty zrychlení, se kterými se může počítat, koncové limitní hodnoty os systému a stanovené rezervy.

4.8.1. Maximální hodnoty zrychlení

Při stanovování maximálních hodnot zrychlení se ukázaly jako problematické výkony motorů vykonávající otáčivý pohyb vyhazovacích ruček. Pro funkci žonglování jsou jejích výkony značně předimenzovány, tudíž bylo nutné počítat s velmi vysokým zrychlením. V práci [2] bylo maximální zrychlení stanoveno teoretickým výpočtem na 2000 rad/s. V praxi se ukázalo, že osa dosahuje ještě většího zrychlení, nicméně pro ilustraci problému použiji tuto hodnotu.

Aby nedošlo k zastavení stroje, tak musí být od aktuální polohy osy volný prostor na vzdálenost danou maximálním možným zrychlením po dobu času predikce T_p a dráhou potřebnou k zpomalení pohybu z teoreticky dosažitelné polohy. Matematicky vyjádřeno:

$$s_{safe} = v_0 T_p + \frac{1}{2} a_a T_p^2 + v_p t_b - \frac{1}{2} a_b t_b^2$$

Kde:

s_{safe} … potřebná bezpečná vzdálenost

v_0 … počáteční rychlost osy

T_p … horizont predikce

a_a … maximální možné zrychlení osy

v_p … predikovaná rychlost při maximálním zrychlení

a_b … zrychlení osy při brzdění

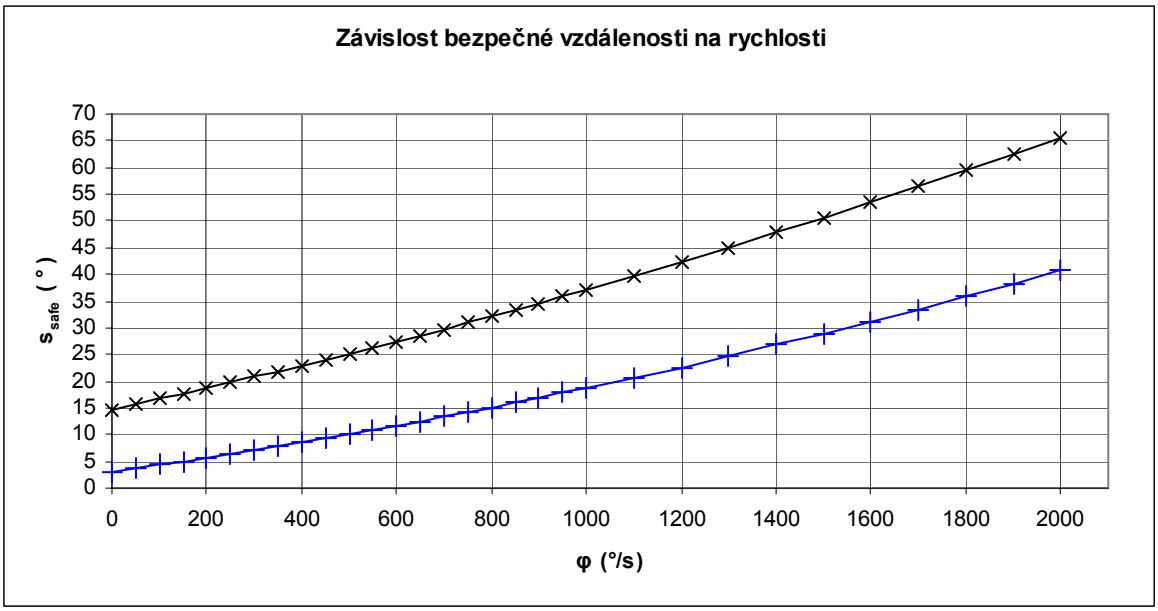
t_b … doba potřebná pro zpomalení na nulovou rychlost

Při použití stejného maximálního možného zrychlení i zpomalení dle teoretického výpočtu 2000 rad/s^2, což představuje 114 649,7°/s^2 vychází, že při nulové počáteční rychlosti musí být volný prostor 14,7°. Potřebný volný prostor roste se zvyšující se rychlostí dle grafu na obrázku Obr. 4.14 černou barvou. Na grafu je také vidět jak roste tato vzdálenost v rozmezí používaných rychlostí pro žonglování.

Tato vzdálenost se ukázala pro již naprogramované žonglování v mnoha pohybech jako moc vysoká a docházelo k zastavování stroje. Abych potřebnou volnou vzdálenost snížil, tak jsem zajistil rozdílené hodnoty pro akceleraci osy a pro brzdění. Konkrétně se jednalo o parametry ve struktuře ACP10AXIS_typ a1_pos, a1_neg, a2_pos, a2_neg a parametr a_stop. Tyto parametry jsem nastavil na hodnotu 50 000°/s^2 (436,1 rad/s^2), která udává maximální možné zrychlení dané osy. Aby zpomalování osy při brzdění bylo rychlejší a nevyužívalo stejné limitní parametry jako normální pohyby, tak jsem v parametru decel_ramp zvolil hodnotu ncTRQ_LIMIT, díky kterému osa zpomaluje nejvyšším možným kroutícím momentem, který je schopna vyvinout.

Zpomalení způsobené tímto momentem jsem opět určil pomocí nástroje Trace. Jako výslednou hodnotu pro zpomalení v horizontálním směru jsem zjistil 160 000°/s2. Obdobným způsobem jsem změřil i zpomalení na vertikálních osách, které činí 130 m/s2.

Pro účely žonglování s kulečníkovými koulemi je postačující hodnota 50 000°/s^2 u horizontálních os a 50 m/s^2 u vertikálních. Tuto hodnotu jsem dosadil do parametrů uvedených výše. Po aplikování těchto nových hodnot se minimální potřebná volná vzdálenost při nulové rychlosti sníží na 3,1° a její vývoj s přibývající rychlostí je na obrázku Obr. 4.14 modrou barvou.

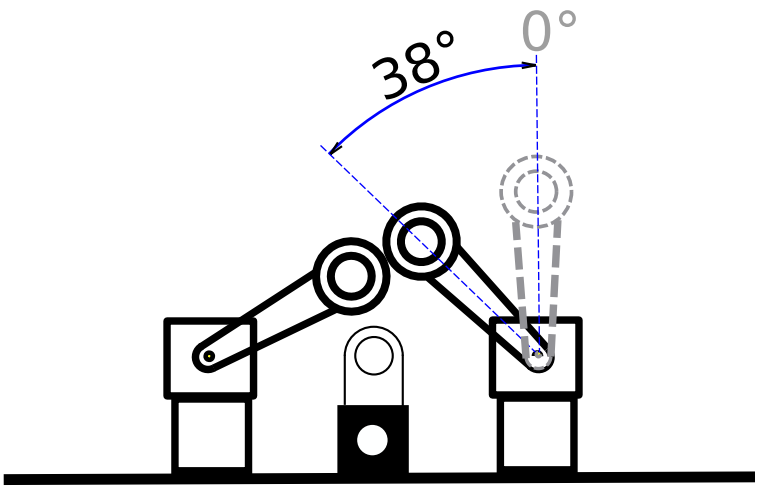


Obr. 4.14 Závislost bezpečné vzdálenosti na rychlosti horizontální osy

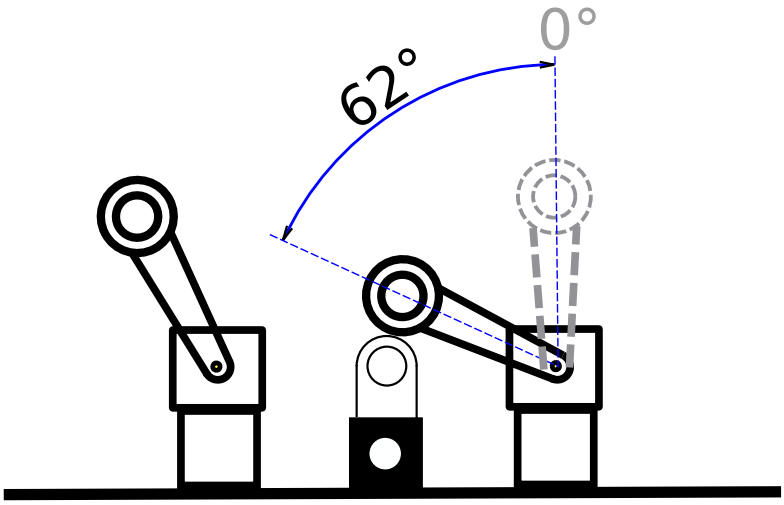
4.8.2. Krajní polohy, limitní kolizní polohy

Vyhazovací ručky vykonávající otáčivý pohyb mají při referování nastavenou nulovou polohu tak, aby se nacházely kolmo k zadní stěně Žongléra (Obr. 4.15). V polohách ±90° se tudíž nacházejí rovnoběžně se zadní stěnou a poblíž jedné této polohy (-90°) se nacházejí i koncové spínače sloužící i jako referenční bod.(Přesná poloha reference se u jednotlivých os liší díky drobným odchylkám v konstrukci vyhazovacích misek a uchycení koncového spínače). Krajní polohy, které jsou hlídány algoritmem jsou nastaveny na ±100°. Díky tomu je možné s ručkama pohybovat v poloze 90° například při funkci Katapult.

Dalšími důležitými body jsou polohy, za jejichž hranicí hrozí srážka misek při míjení vertikálním pohybem. Tato situace je znázorněna na Obr. 4.15. Mezní polohu jsem změřil na 38°. Stejné nebezpečí existuje i při míjení vyhazovací misky a lineárního motoru (Obr. 4.16). Pro tento případ je kritická hodnota 62°. V souvislosti s lineárním motorem ještě navíc hrozí kolize s upevněním jeho magnetické tyče. V tomto případě je bezpečný prostor až do hranice 92°.



Obr. 4.15 Limitní poloha ručky pro prevenci srážky s druhou ručkou



Obr. 4.16 Limitní poloha pro prevenci srážky s pátou osou

Pro vertikální osy jsem stanovil krajní polohy na -10cm a 2m, přičemž nula se nachází na spodních referenčních spínačích. Od této nuly se také odvíjejí koncové polohy pro lineární osu uprostřed, která má pak krajní polohy na hodnotách -76cm a 34cm.

Dalším parametrem, který se v algoritmu objevuje je zvětšení objektu vyhazovací misky ve vertikálním směru aby byl respektován jejich rozměr a navíc byla stanovena bezpečnostní rezerva. Zvětšení jsem zvolil 5cm v obou směrech.

Celkový souhrn parametrů algoritmu a jejich hodnot je uveden v následující tabulce:

Parametr		Proměnná	Hodnota (units, units/s)
	Horizont predikce	REACTION_DELAY	0,0096 s
	Mezní poloha pro kolizi osy 1 a 3	AXIS_1_3_COLLISION_LIMIT	380
	Mezní poloha pro kolizi os 1,3 a 5	AXIS_24_5_COLLISION_LIMIT	620
	Mezní poloha pro kolizi osy 1,3 s uchycením osy 5	AXIS_13_ROD_COLLISION_LIMIT	920
osa 1,3	Zrychlení osy v pozitivním směru	AXIS_X_ACCELERATION_UP	500000
	Zrychlení osy v negativním směru	AXIS_X_ACCELERATION_DOWN	500000
	Koncový limit pozitivní	AXIS_X_LIMIT_UP	-1000
	Koncový limit negativní	AXIS_X_LIMIT_DOWN	1000
	Brzdné zrychlení v pozitivním směru	AXIS_X_BRAKE_UP	1600000
	Brzdné zrychlení v negativním směru	AXIS_X_BRAKE_DOWN	1600000
	Zvětšení osy v pozitivním směru	AXIS_X_OVERSIZE_UP	0
	Zvětšení osy v negativním směru	AXIS_X_OVERSIZE_DOWN	0
osa 2,4	Zrychlení osy v pozitivním směru	AXIS_X_ACCELERATION_UP	1000000
	Zrychlení osy v negativním směru	AXIS_X_ACCELERATION_DOWN	1000000
	Koncový limit pozitivní	AXIS_X_LIMIT_UP	20000
	Koncový limit negativní	AXIS_X_LIMIT_DOWN	-1000
	Brzdné zrychlení v pozitivním směru	AXIS_X_BRAKE_UP	1300000
	Brzdné zrychlení v negativním směru	AXIS_X_BRAKE_DOWN	1300000
	Zvětšení osy v pozitivním směru	AXIS_X_OVERSIZE_UP	500
	Zvětšení osy v negativním směru	AXIS_X_OVERSIZE_DOWN	500
osa 5	Zrychlení osy v pozitivním směru	AXIS_5_ACCELERATION_UP	2000000
	Zrychlení osy v negativním směru	AXIS_5_ACCELERATION_DOWN	2000000
	Koncový limit pozitivní	AXIS_5_LIMIT_UP	3400
	Koncový limit negativní	AXIS_5_LIMIT_DOWN	-7600
	Brzdné zrychlení v pozitivním směru	AXIS_5_BRAKE_UP	2000000
	Brzdné zrychlení v negativním směru	AXIS_5_BRAKE_DOWN	2000000
	Zvětšení osy v pozitivním směru	AXIS_5_OVERSIZE_UP	500
	Zvětšení osy v negativním směru	AXIS_5_OVERSIZE_DOWN	500

Tab. 4.2 Parametry algoritmu a jejich hodnoty

4.8.3. Jednotky

V praxi algoritmus nepracuje s jednotkami, které jsem uváděl v předchozích odstavcích, ale s jednotkami, které jsou dány nastavením jednotlivých os. Veškeré výpočty jsou tedy prováděny v unitech. Pro horizontální otáčivý pohyb je použit vztah 1° = 10 unitů. U vertikálních os odpovídá 1mm 10 unitům. U lineárního motoru je dokonce rozlišení 100 unitů na milimetr, tudíž se musí poloha pro algoritmus předem přepočítat poměrem 1:10.

4.9. Programové řešení algoritmu

4.9.1. Prevence srážky

Pro naprogramování algoritmu jsem využil maker jazyka C, ve kterém je algoritmus napsán. Konkrétně se jedná o soubor X20safety.c. Na začátku kódu jsou nadefinovány konstanty obsahující limitní hodnoty a parametry os, se kterými algoritmus pracuje.

Dále jsou v kódu pomocí maker vytvořeny funkce pro výpočet odhadu stavu na konci časového horizontu predikce a také funkce jejichž výsledkem je logická hodnota true nebo false v závislosti na splnění nebo nesplnění podmínky.

Použitá makra:

`PREDICT_AXIS(id)`

Počítá stav na konci časového horizontu predikce. Pro výpočet je použit vztah pro standardní rovnoměrně zrychlený/zpomalený pohyb.

`DECELERATABLE(distance, speed, aceleration)`

Vrací hodnotu true, pokud lze při dané rychlosti a zrychlení zbrzdit pohyb na nulovou hodnotu na dané vzdálenosti. To je dáno vztahem (3).

`WATCH_AXIS_LIMITS(id)`

Toto makro kontroluje pozici osy vůči jejím krajním limitním hodnotám. Bere v úvahu predikovanou pozici a rychlost směrem ke krajnímu limitu a tyto hodnoty vyhodnocuje opět pomocí makra DECELERATABLE. Výsledkem je logická hodnota true pokud nehrozí, že by se osa dostala za koncový limit.

`WATCH_AXIS_REGION(id, limit_max, limit_min)`

Pomocí tohoto makra lze definovat oblast, ve které se NESMÍ daná osa nacházet. Vrací true, pokud nehrozí, že by se osa do zakázané oblasti mohla dostat.

`WATCH_AXIS_COLISION(id1,id2)`

Toto makro hlídá kolizi mezi horizontálními nebo vertikálními osami. Výsledná logická hodnota true je vrácena v případě, že kolize nehrozí.

Pomocí těchto maker lze vytvářet logické výrazy hlídající nebo omezující pohyb os. Výstupem algoritmu je tedy booleovská proměnná `goSafeToRun`, která svou hodnotou FALSE zastaví veškeré pohyby stroje.

4.9.2. Kontrola polohy lineárního motoru

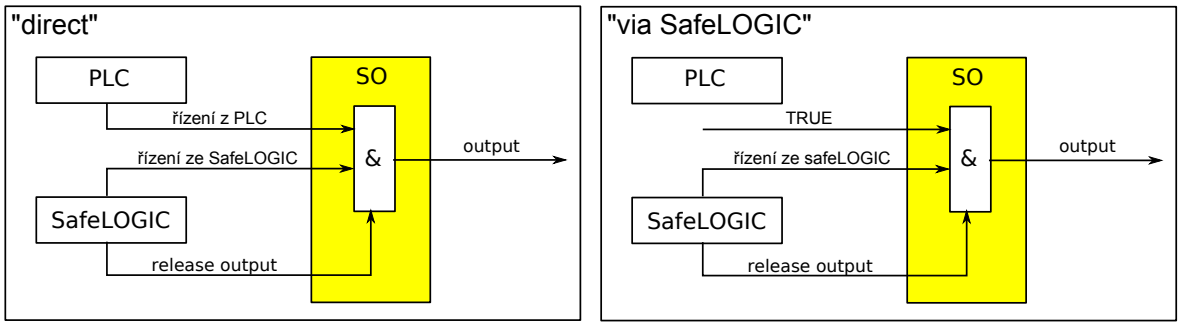
Algoritmus také kontroluje polohu lineárního motoru pro nadhazování spadlých koulí. Pokud je lineární motor svou polohou nad otvorem, kterým do jeho misky přicházejí koule z podavače, propadne koule do prostoru pod žonglérem. To by znamenalo problém při vzdáleném přístupu k modelu. Proto je pozice kontrolována a pokud je osa v přípustné pozici je proměnná `goFeeder_en` v logické 1. Tato proměnná je poté použita v PP420, kde povoluje vypuštění koule ze zásobníku.

4.9.3. Zastavení stroje

Výsledkem běhu algoritmu jsou proměnné `goSafeEnable` povolující pohyb os a `goFeederEnable` povolující vypuštění koule ze zásobníku. Tyto proměnné jsou opět mapovány do kanálu SafeToRun a FeederEnable a cyklickou komunikací přes Ethernet Powerlink přenášeny do PP420, kde jsou z kanálů mapovány na proměnné `gioSafeToRun` a `giFeederEnable`. V cyklické úloze powerlink.c proměnná `giFeederEnable` maskuje logickým součinem proměnnou `feeder_pawl_2`, která ovládá vypouštění koule ze zásobníku. `gioSafeToRun` se kopíruje do proměnné `goSafeEnable` ovládající přímo quickstop acoposů.

4.9.4. Ovládání quickstopu přes safety výstupy

Jako quickstop je u Acoposů použit digitální vstup trigger 2, na nějž je připojen výstup ze safety výstupního modulu. V projektu se konkrétně jedná o modul Safety_OUT_2 a jeho digitální výstup DigitalOutput03. Ten je běžně ovládán programem v safety PLC. Aby mohl být výstup ovládán z aplikace běžící v PP420 je nutné provést potřebné nastavení. U modulu Safety_OUT_2 v I/O Configuration je potřeba nastavit output signal path u výstupu 3 do režimu direct. Tím se výstup DigitalOutput03 zobrazí v I/O mapping výstupního modulu a je možné do něj namapovat proměnnou goSafeEnable, která je výstupem z bezpečnostního algoritmu a zastavuje všechny osy. Ovládání výstupu ze safety PLC je zachováno tím způsobem, že výstup je logickým součinem signálu z aplikace a ze safety PLC.



Obr. 4.17 Rozdíl mezi safety výstupem „direct“ a „via SafeLOGIC“

4.10. Časově nekritická část zabezpečení

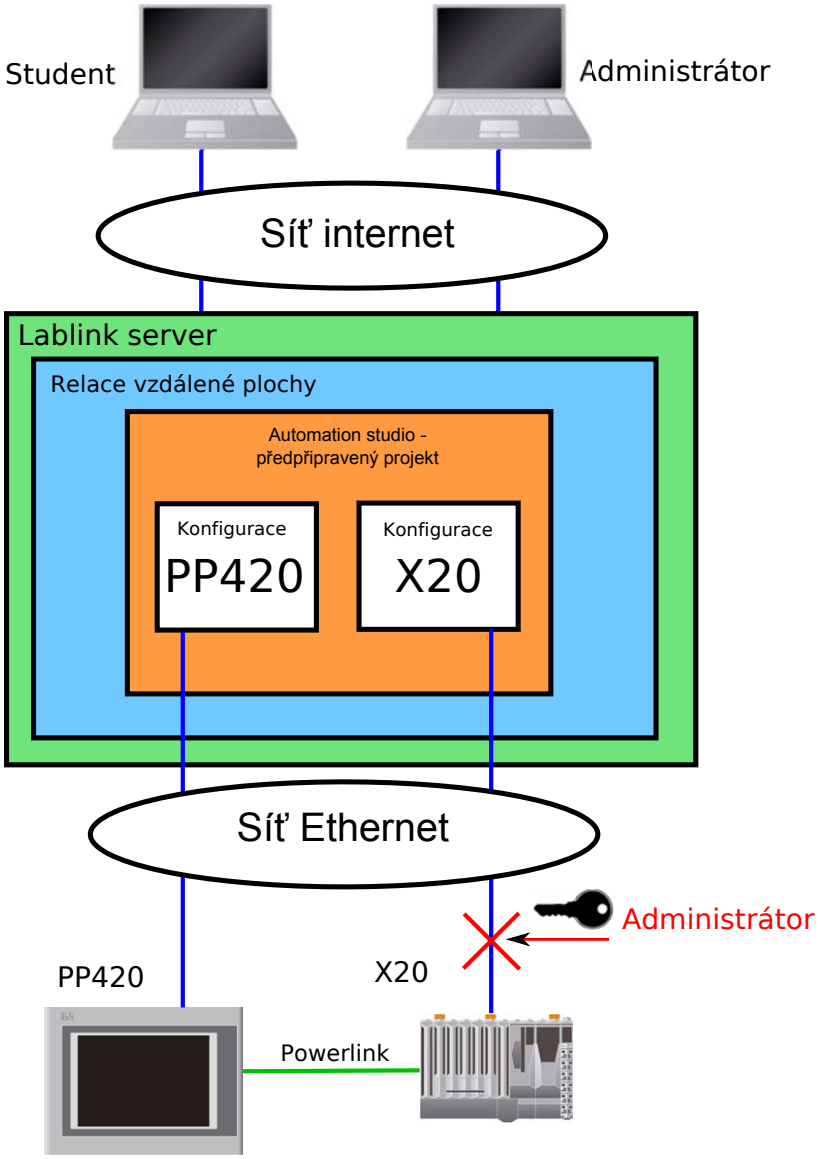
Část zabezpečení, která není kriticky časově náročná, řeší bezpečnost Žongléra z hlediska správné hardwarové konfigurace a kontroluje přítomnost předpřipravených cyklických úloh souvisejících se zabezpečením.

4.10.1. Nutné součásti projektu

Z důvodu možnosti nastavení real-time komunikace (dynamické kanály) mezi PP420 a X20 je nutné, aby se v jednom projektu nacházely konfigurace pro obě zařízení. V případě projektu se kterým budou pracovat studenti to znamená, že budou moci konfiguraci a cyklické úlohy obou zařízení libovolně měnit, což je pro funkci zabezpečení nežádoucí. Z tohoto důvodu jsem byl nucen navrhnout řešení, které by tyto možnosti omezení funkčnosti zabezpečení omezovalo.

4.10.2. Řešení časově nekritické části zabezpečení

Podstata řešení je založena na možnosti přístupu k PLC X20. Toto PLC obsahuje úlohy, které kontrolují správnou konfiguraci a cyklické úlohy na PP420. Přístup k X20 má pouze pověřená osoba, která zná IP adresu a fyzicky vlastní klíč od rozvaděče, kde může PLC fyzicky připojit do ethernetové sítě. Při práci studentů je PLC ze sítě fyzicky odpojeno. V projektu v konfiguraci X20 mohou studenti tudíž dělat libovolné změny, nicméně je nejsou schopny do PLC nahrát.



Obr. 4.18 Přístup k PP420 a X20 přes Lablink

4.10.3. Kontrola konfigurace os

Kromě hlídání pozic a rychlostí jednotlivých os je také potřeba hlídat nastavení parametrů jednotlivých os, aby údaje o poloze či rychlosti odpovídaly realitě. Tato funkce kontroluje, aby studenti při vzdálené výuce nemohli nastavení os překonfigurovat a tím pádem zkreslit data pro zabezpečení.

Veškeré nastavení se nachází v objektu ncAXIS reprezentovaný datovým typem ACP10AXIS_typ obsahující další podobjekty rozdělující parametry logicky do skupin. Každá osa je reprezentována jedním objektem ncAXIS, ve kterém je možné osu konfigurovat. Z hlediska bezpečnosti je nutné hlídat vybrané parametry, aby nebyly změněny.

Limit values – ACP10AXLPA_typ

Tato struktura obsahuje limitní hodnoty pro pozici, rychlost a zrychlení. Hlídány jsou nastavené parametry pro zrychlení a_stop, a1_pos, a1_neg, a2_pos, a2_neg. Ty byly pro každou osu nastaveny tak, aby bylo možné realizovat žonglování až se čtyřmi koulemi a zároveň aby zabezpečovací algoritmus nebral v úvahu tak velký dosažitelný prostor na horizontu predikce.

Homing parameters - ACP10HOMPA_typ

V této struktuře se nachází nastavení pro proceduru inicializace („homování“). Kontrolován je především ofset, který je nastaven při dosažení referenčního spínače. Dále se kontrolují parametry definující způsob, kterým bude referenční spínač hledán a status udávající zdali byla inicializační procedura úspěšně provedena. Pomocí této informace lze poznat zdali je možné data o pozici brát za relevantní odpovídající skutečnosti.

Encoder parameters - ACP10ENCPA_typ

Zde jsou nastaveny parametry pro enkodér – pozitivní směr čítání a škálování.

Digital inputs - ACP10DILEV_typ

Tato struktura udává na jakou fyzickou hodnotu vstupu acoposu bude daný vstup aktivní.

Dále jsou kontrolovány parametry **node_nr** a **serial_nr,** aby nemohlo dojít k záměně jednotlivých os a parametr mode ze struktury move, který udává jaký druh pohybu je právě prováděn. Tato hodnota je využívána pro povolení pohybu při inicializaci, kdy algoritmus nefunguje, protože nemá relevantní hodnoty o pozici.

4.10.4. Knihovna AsIMA

Část zabezpečení, která kontroluje parametry os a cyklické úlohy využívá pro komunikaci mezi X20 a PP420 taktéž síť Powerlink, ale komunikace se odehrává v asynchronní části cyklu. Konkrétně je použit protokol INA 2000.

Knihovna AsIMA obsahuje funkce pomocí kterých může být využíván INA2000 Manager, který funguje jako klient a umožňuje přenos proměnných, modulů a další

funkcionality mezi klientem a serverem. Výhodné je, že není nutné cokoliv programovat na straně serveru, kterým může být jakýkoliv Automation Runtime.

Komunikace se zahajuje voláním funkce IMAInit, které je předávána adresa s názvem konfiguračního objektu, který obsahuje veškeré nastavení komunikace.

Konfigurační objekt

Tento datový objekt je uložen na straně klienta a obsahuje informace potřebné pro navázání komunikace a komunikaci samotnou. Skládá se z několika sekcí, které mohou být řazeny libovolně za sebou. Každá sekce je uvozena názvem uzavřeným v hranatých závorkách. Syntaxe je tvořena klíčovými slovy, které začínají vždy lomítkem (/). Na začátku každé sekce musí být uveden typ o jaký druh sekce se jedná. Následují klíčová slova specifická pro objekty v dané sekci. Podrobný popis syntaxe je popsán v [13]

Pro funkce zabezpečení je potřeba pouze navázat spojení a číst objekty ncAXIS. Konfigurační objekt je v příloze A.1. Zde uvedu krátký příklad:

```
"[ROUTING_PATH]"
"/SECTION=INTERFACE"
"/CN=IF3.1"
"[PV_LIST]"
"/SECTION=PV"
"/SV=READ /SERVPV=gRAxis01  /CLNTPV=X20IMAcom:axis_1_param"
```

Klíčové slovo CN udává rozhraní použité pro komunikaci. IF3 je adresa rozhraní a 1 číslo INA stanice která figuruje v komunikaci jako server. SV = READ specifikuje službu (pouze čtení) a CLNTPV se SERVPV obsahují názvy úlohy a proměnných na straně klienta a serveru odkud kam se má hodnota kopírovat.

4.10.5. Kontrola přítomnosti cyklických úloh pro zabezpečení

Kromě kontroly parametrů os systému jsou navíc kontrolovány cyklické úlohy potřebné pro zabezpečení nacházející se na PP420. Navíc je potřeba také kontrolovat mapování proměnných učastnících se v systému pro zabezpečení.

Jak cyklické úlohy, tak i nastavení mapování proměnných se z PC do PLC přenáší ve formě binárních souborů takzvaných BR modulů. V této formě jsou na příslušném PLC také uložena v paměti, kterou lze zvolit Automation Studiu. Jelikož je nežádoucí, aby

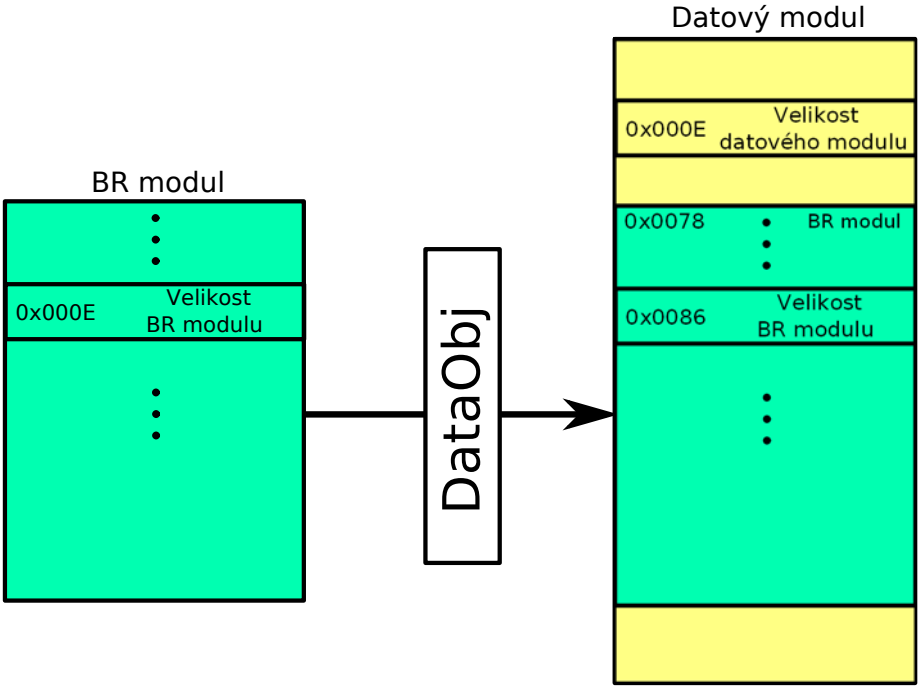
studenti tyto soubory jakkoliv měnili, tak je cílem pro zabezpečení tyto BR moduly přenášet do PLC X20 a tam provést jejich kontrolu.

K přenosu BR modulů jsem stejně jako pro struktury os použil knihovnu AsIMA a protokol INA2000. V konfiguračním objektu je pro tento účel vytvořena nová sekce BRMODUL (viz. A.1), ve které je přenos potřebných souborů nakonfigurován. SV značí směr přenosu, LD paměť, kam má být modul uložen, MO název modulu a CMDPV proměnnou, která spouští přenos souboru.

Problémy při přenosu BR modulů

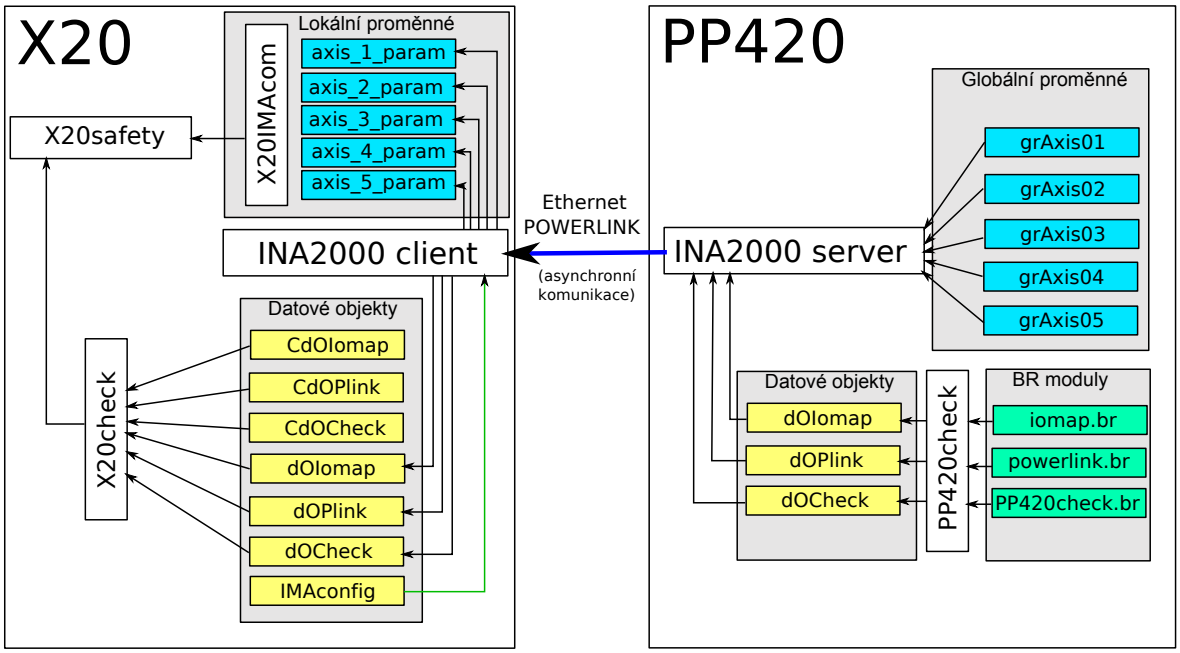
Při přenosu BR modulů přímo z PP420 do X20 se však vyskytly problémy s přímým přístupem k modulům. U přenosu modulu pro mapování byl naopak problém na straně X20, kdy přenášený modul nahradil původní modul používaný PLC X20.

Z těchto důvodů jsem přidal na PP420 cyklickou úlohu, která z vybraných BR modulů vytváří jiný typ souboru – datové objekty, se kterými již při přenosu ani při ukládání na druhé straně není problém. Za tímto účelem jsem použil knihovnu DatObj, která obsahuje funkční bloky pro práci s datovými objekty a knihovnu sys_lib, pomocí které jsem pracoval s BR moduly.



Obr. 4.19 Struktura BR modulu a datového objektu

Na straně X20 jsou přenesené moduly v datových objektech zkontrolovány, zdali mají stejnou délku jako moduly referenční. Výsledek kontroly je poté vyhodnocován společně s výsledky kontroly parametrů a cyklické kontroly pozic a rychlostí. Schematicky je kontrola modulů znázorněna na Obr. 4.20.



Obr. 4.20 Schéma zabezpečení časově nekritické části

Kapitola 5

5. Zapojení do systému Lablink

Systém pro vzdálenou výuku Lablink se skládá z webového rezervačního systému, serveru s relacemi vzdálených ploch a samotných modelů, se kterými lze přes systém Lablink pracovat.

5.1. Systém Lablink

5.1.1. Rezervační systém

Pomocí rezervačního systému si student může rezervovat model, se kterým chce pracovat, na daný čas a délku doby práce. Do systému se přihlašuje pomocí uživatelského jména a hesla. Součástí rezervačního systému jsou i informace o modelech, které student potřebuje k práci s nimi. Dále jsou k dispozici *.rdp soubory pro přístup k relacím vzdálených ploch.

5.1.2. Vzdálené plochy

Druhou částí systému Lablink je server s relacemi vzdálených ploch, ke kterým se lze připojit pomocí RDP protokolu. Tyto plochy jsou vždy připraveny pro daného uživatele na základě předchozí rezervace přes webové rozhraní. Nikdo jiný se k ploše v daný čas nemůže přihlásit. Po skončení práce na ploše a uplynutí doby rezervace je relace připravena pro dalšího uživatele. Všechna osobní data včetně rozpracovaných projektů jsou odstraněna.

5.1.3. Vizuální přístup k modelu

Pro vizuální kontakt s modelem jsou zařízení vybavena kamerou k níž má student přístup přes běžný internetový prohlížeč.

5.2. Žonglér v systému Lablink

5.2.1. Role student

Pro práci studentů na Žongléru se bude na vzdálené ploše nacházet předpřipravený projekt s hardwarovou konfigurací a zabezpečením popisovaným v této práci. Z tohoto projektu budou studenti při své práci vycházet a doplňovat do něj své cyklické úlohy, vačkové profily atd.

V úlohách a nastavení vztahující se k zabezpečení nesmí studenti cokoliv měnit.

5.2.2. Role administrátor

Administrátor bude mít k dispozici výchozí projekt obsahující navíc cyklické úlohy k zabezpečení nacházející se na PLC X20. Dále má fyzický přístup do rozvaděče Žongléra, aby mohl zabezpečovací PLC fyzicky připojit k síti a nahrávat do něj projekt. Tuto možnost studenti nemají.

Podrobný popis pro zapojení Žongléra do vzdálené výuky je se nachází v příloze B. této práce.

Kapitola 6

6. Zhodnocení zabezpečení

Závěrečné zhodnocení bych rozdělil na dvě části. V první se budu věnovat zabezpečení ve smyslu hlídajícího algoritmu zastavující stroj při nebezpečné situaci. V druhé části zhodnotím zabezpečení z hlediska zapojení do systému vzdálené laboratoře a práce studentů s vývojovým prostředím a strojem jako takovým.

6.1. Zhodnocení algoritmu a technického řešení

Při testování funkčnosti algoritmu se potvrdily teoretické předpoklady, ze kterých jsem vycházel v kapitole 4.6.1. Při testování uvažovaných kolizních situací bylo střetu vždy zabráněno s dostatečnou rezervou.

Na konceptu zabezpečení však lze nalézt vylepšení, která by mohla snížit nezbytně nutné bezpečné vzdálenosti respektive reakční čas. S použitím hardwaru B&R by se jednalo o umožnění takzvané cross komunikace mezi Acoposy a PLC X20, kterou nebylo možné s dostupnými knihovnami realizovat. Touto „zkratkou“ pro data by bylo možné ušetřit reakční čas.

Další možné zrychlení reakce by mohlo přinést řešení, kdy by X20 obsahovala I/O modul a z něj by vedl výstup přímo na vstup quickstop Acoposů.

Jiným celkovým řešením by bylo použití HW s možností provozovat ethernetové rozhraní v promiskuitním režimu a následně jednotlivé packety analyzovat. Dále by musel HW používat některý z real-time operačích systémů. Výstup ze zařízení by mohl být připojen přímo na quickstop nebo na safety vstup.

Pro potřeby vzdálené výuky se však použité řešení ukázalo jako dostatečné a ve všech testech prokazovalo správnou funkci.

6.2. Zhodnocení zabezpečení pro práci studentů

Pro zabezpečení vůči změně parametrů os a úloh souvisejících se zabezpečením se mi podařilo najít řešení formou přenosu BR modulů a jejich kontroly v PLC X20.

Je nutné však podotknout, že se jedná o ochranu spíše proti nechtěnému zásahu do zmíněných částí projektu, než o stoprocentní zabezpečení proti úmyslnému vyřazení

kontrolní funkce. Slabinou této části zabezpečení je možnost pevného nastavení hodnoty proměnné z Automation Studia při běhu programu pomocí online monitoru a funkce „Force Variable“. Touto možností lze vymaskovat výstup zabezpečovacího algoritmu do safety výstupu zastavujícího všechny osy systému.

Kapitola 7

7. CNC – obecný úvod

Společně s vývojem elektroniky a jejich aplikací do průmyslových oborů se začaly během 80. let objevovat i první počítačem řízené stroje. Zpočátku se jednalo o takzvané NC (Numeric contrlol) stroje, které umožňovaly stejně jako stroje s ruční obsluhou pouze lineární pohyby nástroje rovnoběžné s osami stroje. S rostoucím výkonem výpočetní techniky se začaly objevovat první CNC (Computer Numeric Control) stroje, které dokázaly pohybovat dvěma osami najednou a vzájemně je interpolovat. Další vývoj následoval v podobě možnosti interpolace více os a řízení pohybu v prostoru pomocí prostorových transformací, čímž se od CNC řízení dospělo k robotickým úlohám.

V klasickém CNC řízení se používá prostorových systémů. Základem je klasické 2D řízení, kde mohou být interpolovány dvě osy v jedné z kartézských rovin. Rozšířením je 2 1/2D řízení, kde lze rovinu libovolně naklonit. Ve 3D systému mohou být vzájemně interpolovány všechny tři osy X,Y,Z současně.

7.1. Možnosti řízení trajektorie v CNC

7.1.1. Poit to Point řízení

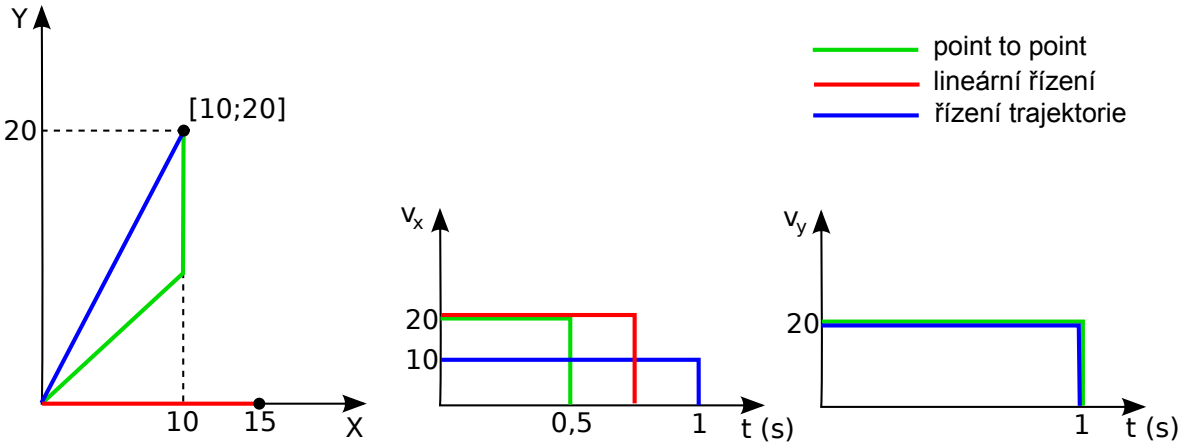
Pří způsobu řízení Point to Point se definuje pouze cílová poloha os systému. Obvykle se k ní osy přibližují maximální rychlostí nezávisle na sobě, tudíž trajektorie vykonaná nástrojem není nijak definovaná a může se měnit s různým nastavením systému. Předpokládá se tedy, že nástroj bude použit až na koncové pozici (vrtání, svařování, pick and place).

7.1.2. Lineární řízení

U lineárního řízení lze určit pozičně-rychlostní profil trajektorie pouze pro jednu osu. Pohyb do cílové polohy je vykonáván postupně za sebou pro každou osu. Tento způsob řízení se používá v procesech vyžadujících konstantní rychlost jako přímočaré řezání nebo vrtání.

7.1.3. Řízení trajektorie

Při řízení trajektorie jsou interpolovány minimálně dvě osy najednou. Osy jsou řízeny tak, aby co nejvíce kopírovaly jak poziční tak rychlostní profil. Výsledná trajektorie je tudíž přesně definovatelná pro potřeby dané technologie, která je na CNC systému použita.



Obr. 7.1 Možnosti řízení trajektorie v CNC systémech

7.2. Programování CNC systémů

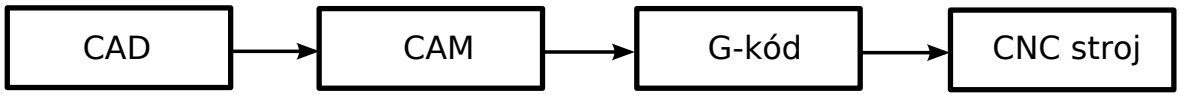
Pohyby CNC strojů jsou v naprosté většině programovány pomocí G-kódu, což je programovací jazyk určený k programování pohybů CNC strojů. G-kód není striktně definován a lze ho najít v mnoha implementacích. Nejčastěji se používá evropský standard ISO 6983, nicméně existují i jiné často používané standardy různých zemí např. DIN 66035, PN-73M-55256, PN-93/M-55251 atd. Pro každý stroj existuje soubor G–funkcí, které mohou mít různý význam v závislosti na výrobci. Příklad G-kódu může být následující:

```
N001 M63
N005 G00 X100 Y100 F300
N010 EXW60=3600 EXF61=100
N020 M60
N030 G01 X110 Y110
N040 M60
N030 G01 X120 Y120 A35
N040 EXW60=1800 EXF61=60 EXF62=50
N050 M60
```

G-kódy mají obvykle tvar GXX což představuje něco jako instrukci programu, která říká stroji co má vykonat. Na začátku instrukce však mohou být i jiná písmena.

7.2.1. CAM software

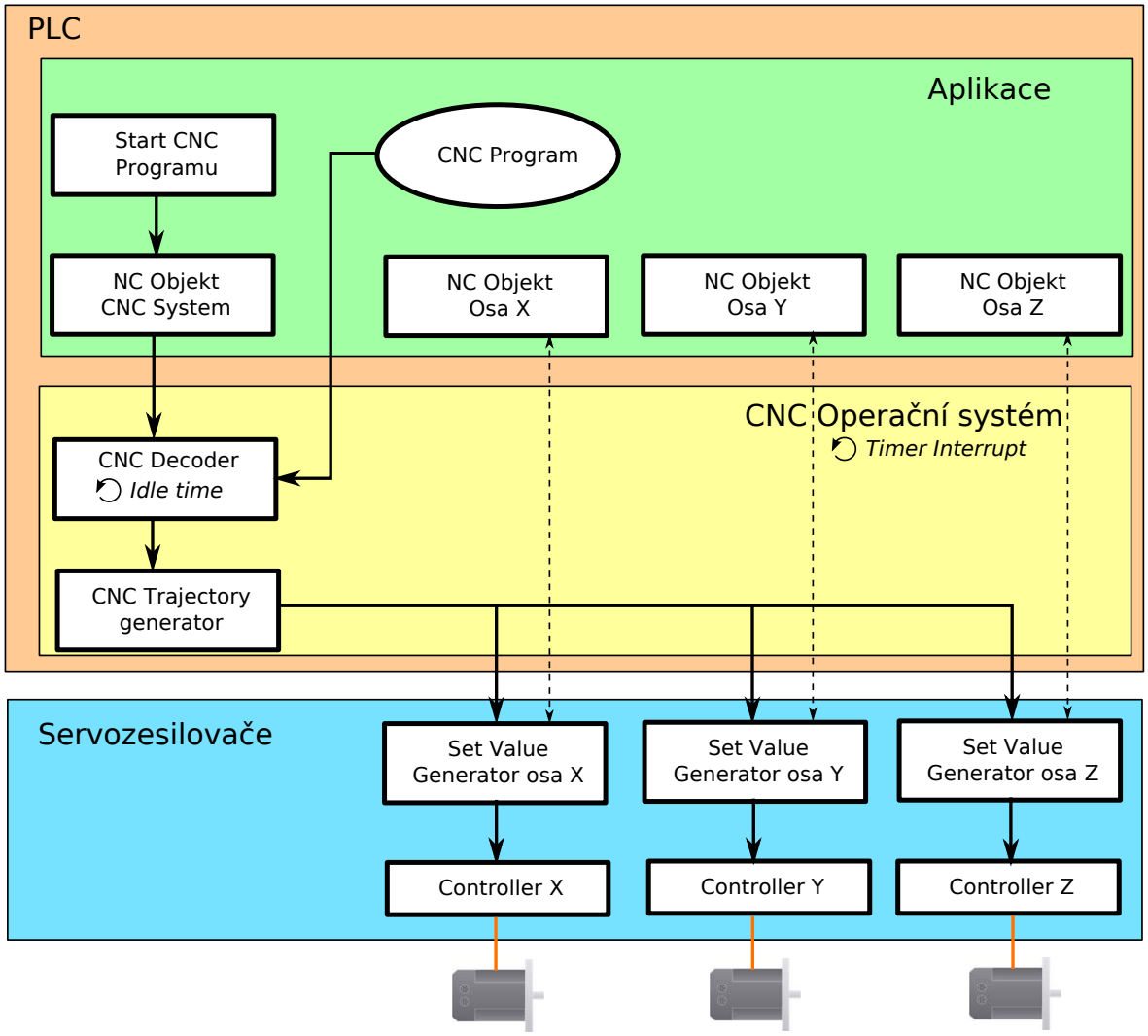
Tvorba G-kódu většinou není záležitost pro pracovníka, který by stroj přímo programoval. Nejčastěji se používá některý z CAM softwarů (Computer Aided Manufacturing) jako Samartcam, Edgecam, Surfcam a další… Tyto aplikace používají k tvorbě G-kódu překladače – postprocesory, které lze konfigurovat v závislosti na použitém cílovém CNC stroji aby pro něj generovaný kód souhlasil.



Obr. 7.2 Vývoj CNC programu

7.3. CNC v systémech B&R

7.3.1. Softwarový koncept



Obr. 7.3 Softwarový concept CNC systému

Na obrázku Obr. 7.3 je znázorněn softwarový koncept jak je řešeno řízení CNC systému na platformě B&R. Aby mohl CNC systém fungovat jsou zapotřebí alespoň 2 osy, které jsou softwarově reprezentovány NC objekty.

NC objekt má podobu struktury, ve které jsou logicky rozděleny parametry a nastavení vztahující se k dané ose.

Stejně jako osy tak samotný CNC systém je reprezentován NC strukturou, která obsahuje parametry a nastavení celého CNC systému. Z tohoto konceptu vyplývá, že na jednom PLC mohou být současně řízeny dva nezávislé CNC systémy, jejichž činnost lze díky tomu jednoduše synchronizovat.

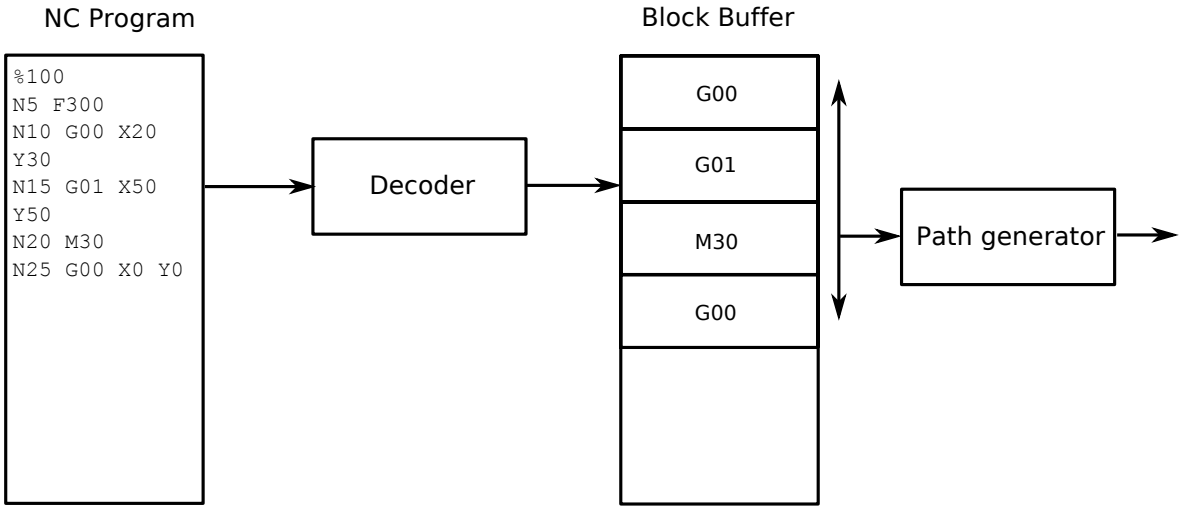
Jádrem CNC řízení je CNC operační systém, který pracuje právě s NC objekty prostřednictvím knihovny ARNC0. Viz 7.3.4.

7.3.2. CNC Decoder

CNC Decoder je vlastně interpreter, který překládá NC program v podobě G-kódu do příkazů pro další část CNC operačního systému – pro Path generátor.

Velmi důležitou vlastností CNC Decoderu je, že jeho činnost je vykonávána neustále během Idle time procesoru, kdy není potřeba obsluhovat některou z cyklických úloh v daném cyklu. NC program je tedy co nejrychleji interpretován a ukládán do Block Bufferu ,zatímco je trajektorie nástroje teprve na začátku sekvence pohybů. Poté co je již Block buffer naplněn, lze se v něm pohybovat oběma směry . Velikost Block Bufferu je nastavitelná v NC struktuře pro CNC.

Z tohoto důvodu jsou hodnoty externích parametrů použity ve stavu, v jakém byly při interpretaci dekodérem. Jsou tudíž Decoder Synchronní. Pokud by bylo potřeba některý z parametrů vyhodnotit současně s vykonáním příkazu G-kódu, tak je k dispozici funkce G170 (Decoder synchronization), která interpretaci zastaví a spustí až ve chvíli, kdy je Decoder „dostižen“ Path generátorem. Existují samozřejmě také skupiny parametrů, které jsou Path synchronní.



Obr. 7.4 Decoder a Path generator

7.3.3. Path Generátor

Path generátor je část systému, která rozkládá požadovanou trajektorii na pohyby jednotlivých os systému. Generuje pro ně tedy žádanou pozici a rychlost, která je cyklicky posílána po síti Powerlink jednotlivým servozesilovačům.

7.3.4. ARNC0 knihovna

Datové typy (NC objekt pro osy a CNC) a funkce potřebné k vytvoření CNC aplikace jsou obsaženy v knihovně ARNC0 (Automation Runtime Numeric Control 0). Součástí ARNC0 je i ARNC0 Manager zajišťující následující funkce:

- Práce s NC objekty os
- Práce s NC objekty CNC
- Zajišťuje komunikaci mezi NC objekty os a CNC
- Zajišťuje komunikaci mezi PLC a pohony
- Obsahuje vlastní nastavení ARNC0 Manageru

Arnc0cfg.ncc [NC Manager Configuration]

Name	Value
Configuration data for ARNC0	
Sampling rate of NC operating system [us]	1600
Number of data records per ACOPOS for Network Command Trace	3000
Size of data buffer for cyclic ARNC0 trace	16#0005_0000
Enable motion packet log	Yes
Task class for NC Manager task	Cyclic #1
Scaling for CPU load from background tasks	1
Use global PV as NC object	Yes
Virtual interface[0]	
Interface name	SDC:SDC_IF1
PV name	
Mode	0
Virtual interface[1]	
Interface name	ST1.IF1
PV name	VIRTUAL
Mode	0
POWERLINK interface[0]	
Interface name	IF6

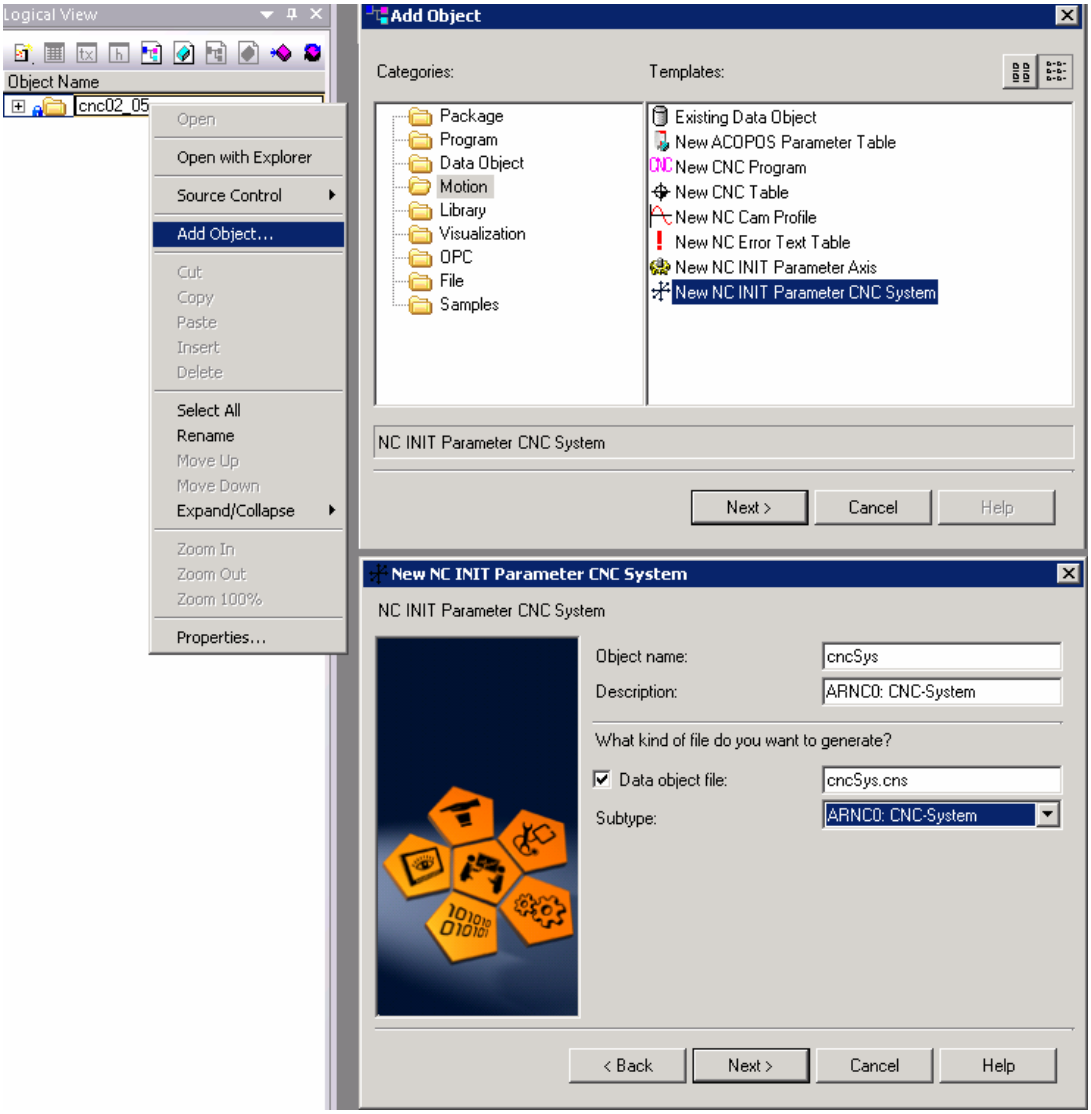
Obr. 7.5 Nastavení ARNC0 manageru

Na Obr. 7.5 je nastavení ARNC0 Manageru. Nejdůležitější položky jsou Sampling rate a Task class, které určují cyklus NC manageru.

7.3.5. CNC Systém

CNC Systém je v softwarovém projektu reprezentován NC strukturou respektive strukturou ARNC0CNC_typ z knihovny ARNC0. Přidání CNC systému do projektu je znázorněno na Obr. 7.6. Samotný CNC Systém zajišťuje:

- Komunikace s Decoderem
- Komunikace s Path generátorem
- Práce s CNC parametry
- Možnost provozovat několik nezávislých CNC systémů



Obr. 7.6 Přidání CNC objektu do Projektu

Osy systému a kompatibilita knihoven ACP10 a ARNC0

V současnosti lze použít k řízení os dva typy knihoven. Jednak již zmíněnou ARNC0, ale také ACP10, která navíc umožňuje využití standardizovaných knihoven PLCopen. ACP10 však neobsahuje CNC Systém. Pokud tedy chceme zároveň používat knihovnu ACP10, PLCopen a CNC Systém tak musíme v nastavení os povolit jejich použití v CNC Systému (Obr. 7.7).

AxesNcMap.ncm [NC Mapping]*

NC Object N...	Module Address	Nc Obje...	Channel	NC INIT ...	ACOPOS P...	Advanced	Additional Data
M1Ax1	SDC_IF1.ST2	ncAXIS	1	sdcax_X	sdcax_par		CNC_Enabled="1"
M1Ax2	SDC_IF1.ST3	ncAXIS	1	sdcax_Y	sdcax_par		CNC_Enabled="1"

Obr. 7.7 Povolení ACP10 osy pro použití v CNC

Poté je možné osy používat jednak pro CNC řízení tak pro ovládání pomocí PLCopen funkčních bloků, přičemž použití funkčního bloku má za následek ukončení případného běžícího CNC programu.

7.3.6. Typy os v CNC systémech

ncCNC

Jedná se o klasický typ CNC osy, která definuje pracovní polohu nástroje v kartézských souřadnicích obvykle značených X,Y,Z.

ncLINEAR

Lineární osa, která má spíše doplňující charakter. Způsobuje zastavení na přechodu mezi částmi trajektorie. Obvykle se využívá na funkce přidržení a pohyby nástrojů či jednotek. ncLINEAR může mít rozšíření ncLINEAR+ncNO_STOP nezpůsobující zastavení na přechodu.

ncTANGENT

Velmi důležitým typem osy je tangenciální osa. Obvykle se jedná o rotační osu kolmou k pracovní rovině. Natočení tangenciální osy závisí na aktuálním směru vykonávané trajektorie (nastavitelné). S tím souvisí řada vyvstávajících problémů zejména na přechodech trajektorie. Chování osy v těchto místech je závislé na nastavených parametrech.

ncROTARY

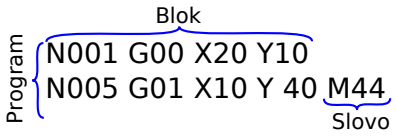
Jedná se o rotační osu, jejíž jednotky jsou ve stupních (monitor CNC objektu v rozsahu 0 – 360, monitor osy reálná pozice). V defaultním nastavení osa způsobí zastavení na přechodu trajektorie. Lze přidat nastavení ncNO_STOP a ncSHORT_PATH upravující absolutní polohování, kdy je například místo pohybu o 390° vykonán pohyb o 30°.

Unit Factor

Pro všechny osy je důležité nastavení unitfactor v datovém objektu CNC systému. Tento údaj udává přepočet mezi jednotkami používanými v CNC systému a jednotkami os systému (unity). Při rozlišení 100 unitů na milimetr je při používání metrických jednotek (mm) v CNC hodnota nastavena na 100.

7.4. CNC program

CNC program je textový soubor v ASCII formátu, který obsahuje popis trajektorie a informace o technologických funkcích. Jeden řádek programu se nazývá blokem, který může být složen z více slov. Slova programu jsou adresy registrů (G, N, X …) s jejich hodnotami (200, 01, 0.443 …). Bloky jsou vykonávány jeden po druhém nezávisle na jejich číslování (N001, N010 …), přičemž slova se vykonávají dle definovaného pořadí (např. M slova před G slovy).



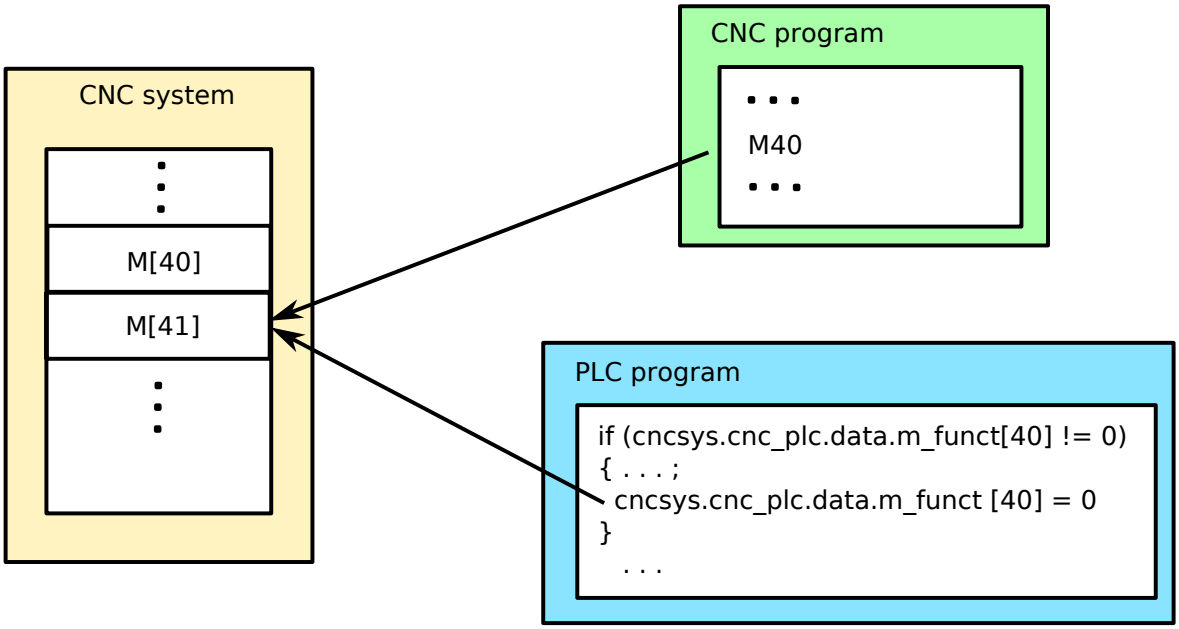
Obr. 7.8 CNC program (G-kód)

V G-kódu je možné provádět standardní matematické operace (+, - , *, / , cos, acos, mod, …) a také používat řídicí struktury s podmínkami (IF, SWITCH, FOR …).

7.4.1. M – funkce

M-funkce představují velmi důležitý nástroj pro spolupráci CNC programu s vlastní aplikací implementovanou v PLC. Díky nim lze jednoduše synchronizovat běh CNC programu například se složitým technologickým postupem, jiným CNC kanálem nebo externím zařízením (upínací čelisti, čerpadlo chlazení, spindle …).

M-funkce jsou implementovány v paměti jako bitové pole o velikosti 1024 bitů, kde offset bitu značí číslo příslušné funkce. Voláním M-funkce z G-kódu je příslušný bit nastaven do „1“.



Obr. 7.9 Komunikace pomocí M-funkcí

7.4.2. Synchronní M-funkce

Při volání synchronní funkce je vykonávání CNC programu pozastaveno do té doby, než je příslušný bit resetován zpět do „0“ aplikací v PLC.

7.4.3. Asynchronní M-funkce

Voláním asynchronní M-funkce není vykonávání programu nijak pozastaveno. Funkci je možné znovu volat až po resetováním bitu z aplikace.

7.4.4. Komunikce CNC <->PLC

Nastavování bitů M-funkcí v bitovém poli je nutné volat pomocí funkce `nccnccom(cncsys)`. Jakmile je funkce zavolána z CNC programu tak je příslušný bit nastaven až po volání této funkce. Je jí tudíž vhodné volat každý cyklus PLC.

7.4.5. Vlastnosti M-funkcí

M-funkce jsou zpracovávány na začátku NC bloku a jsou aktivovány zároveň. Pokud jsou v po sobě jdoucích blocích, tak jsou každá zpracována v jednom cyklu CNC.

Po dekódování je M-funkce v samostatném bloku Block Bufferu. Při zpětném pohybu je tudíž nejdříve vykonán pohyb a až poté vykonána funkce.

7.5. Externí parametry

Použití externích parametrů umožňuje stejně jako M-funkce spolupráci CNC programu a aplikace v PLC. Oproti funkcím jsou externí parametry určeny pro výměnu dat.

Externí parametry jsou implementovány v paměti jako struktura (ARNC0EXPAR_typ) obsahující 4 pole různých datových typů o velikosti 100. V CNC systému jsou k dispozici následující typy:

- B – Byte parametr (datový typ INT8)
- W – Word parametr (datový typ INT16)
- L – Long parametr (datový typ INT32)
- F – Float parametr (datový typ FLOAT)

V CNC programu se požívá syntaxe EX<data_type><index>=<value>

V PLC je přístup k parametru pomocí struktury :

<název_struktuty>.EX<data_type>[<index>]=<value>

Kapitola 8

8. Řízení CNC frézky

8.1. Pálicí stroje Vanad

Praktická část druhého oddílu této diplomové práce se bude zabývat návrhem řízení CNC frézky, která bude instalována na současných pálicích strojích Vanad jako přídavné zařízení.

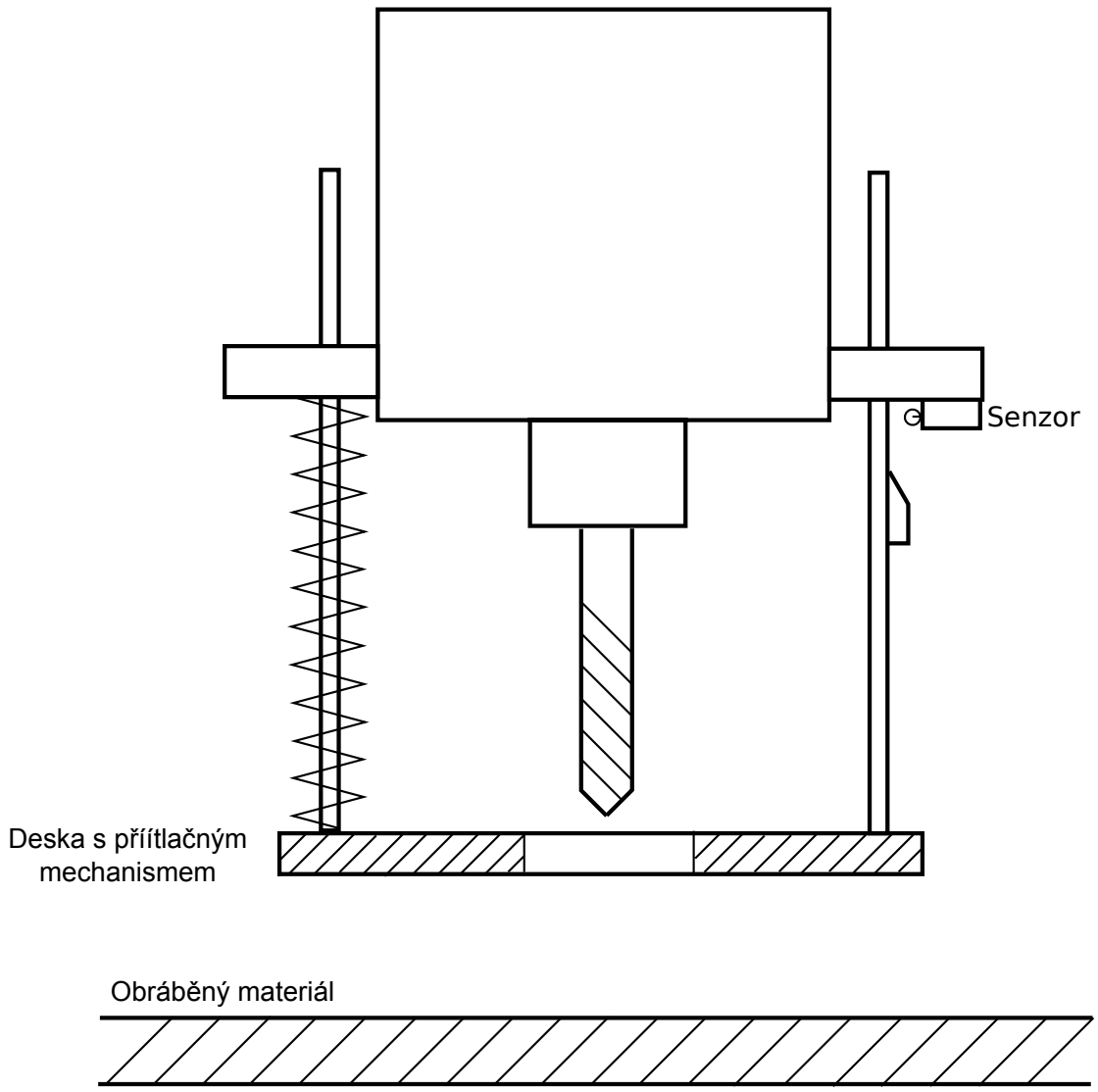
Pálicí stroje Vanad se vyrábějí ve třech základních velikostech – Arena, Proxima a Kompakt, přičemž mohou být osazeny plazmovou dělící jednotkou nebo kyslíkovou. Tloušťka děleného materiálu může být až 200mm. Volitelně lze pak přidat technologii značení, navrtávání, rýsování či jednotku pro regulaci výšky hořáku.

Stroje jsou zkonstruovány z portálu pohybujícího se po kolejnicích ve směru osy X. Portál je osazen pojezdem, na který lze umístit jeden nebo více supportů. Pohyb po portálu reprezentuje pohyb v ose Y. Jednotlivé supporty jsou polohovatelné vertikálně, což zajišťuje pohyb v ose Z. Supporty jsou osazeny nosnými profily, na které lze jednoduše instalovat libovolnou technologii. Pohyb os zprostředkují synchronní nebo krokové motory různých výkonů v závislosti na rozměru stroje.

8.1.1. CNC vrtačka a frézka

Technologie vrtání a frézování bude realizována jako přídavné zařízení (do budoucna i samostatné) k pálicímu stroji a bude mít vlastní support. Pro pohyb v ose y bude použit support pálicí s nímž bude pevně spřažen.

Technologie bude určena k vrtání menších otvorů do průměru 13mm do plochých materiálů maximální tloušťky 200mm. Pro technologii frézování a gravírování je předpokládáno použití maximálního průměru nástroje 10mm pro maximální hloubku 1,5mm do oceli a do 8mm do měkkých materiálů (dřevo, plast ..).
Schématicky je zařízení znázorněno na Obr. 8.1.



Obr. 8.1 Schématický nákres vrtacího zařízení

8.2. Cíle práce

Úkolem této práce je popsat programové řešení pro ovládání technologie vrtačky a frézky. Popsán bude jednak obecný koncept použitý pro programování modulárních technologií, za kterou lze portálový stroj se zařízeními považovat, a také konkrétní programové řešení. Na závěr provedu porovnání požadavků na řízení pro oba typy stroje, které jsem v této práci popisoval

8.3. Používaný HW

K řízení pálicích strojů Vanad jsou používány průmyslové PC APC 620 a APC620 Embeded, která obsahují zabudovaná komunikační rozhraní pro Powerlink, Etherntet, CAN, RS232 a USB. K polohovému řízení jsou použity servozesilovače Acopos v závislosti na použitých typech pohonů. Jako pohony jsou používány synchronní, stejnosměrné a krokové motory. Pro pohon vrtačky je potom použit asynchronní motor s frekvenčním měničem Acopos Inverter.

Vizualizace je zprostředkována dotykovým LCD panelem s klávesnicí přizpůsobenou pro CNC aplikace. Další možností je mobilní operátorský panel MP50 s 6,5“ displejem.

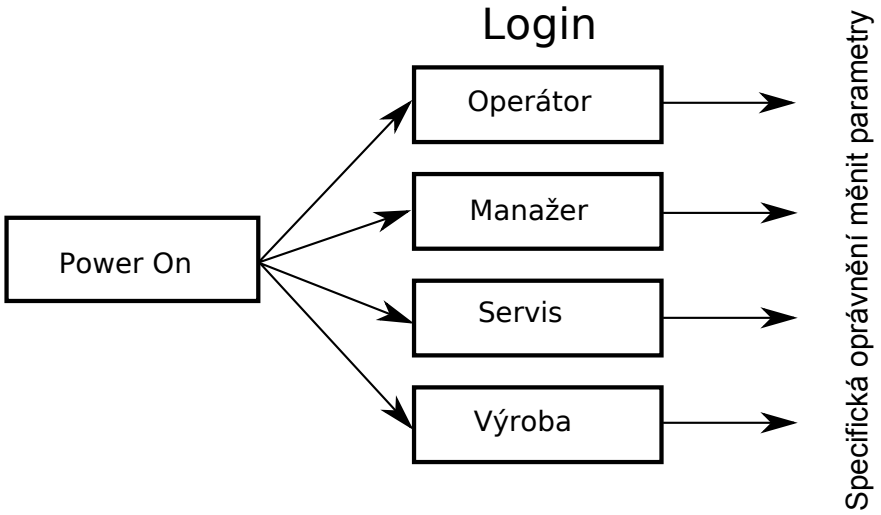
Vstupy a výstupy jsou zajištěny vstupně-výstupními moduly řady X20.

8.4. Softwarový koncept

Programové řešení řízení pálicích strojů Vanad je realizováno jako jeden softwarový projekt v prostředí Automation Studio, který obsahuje dvě hardwarové konfigurace pro jednotlivé typy používaných APC.

Obě konfigurace pak obsahují veškeré hardwarové moduly, které se v různých variantách stroje používají. Softwarová konfigurace je na obou typech APC shodná.

Tímto přístupem je dosaženo, že se do všech pálicích strojů nahrává stejný projekt bez ohledu na konkrétní velikostní provedení nebo vybavenost přídavnými zařízeními. Konfigurace pro daný typ pálicího stroje se provádí až po nahrání projektu do zařízení a přihlášení uživatele s právy měnit výrobní nastavení stroje.



Obr. 8.2 Systém uživatelů a oprávnění

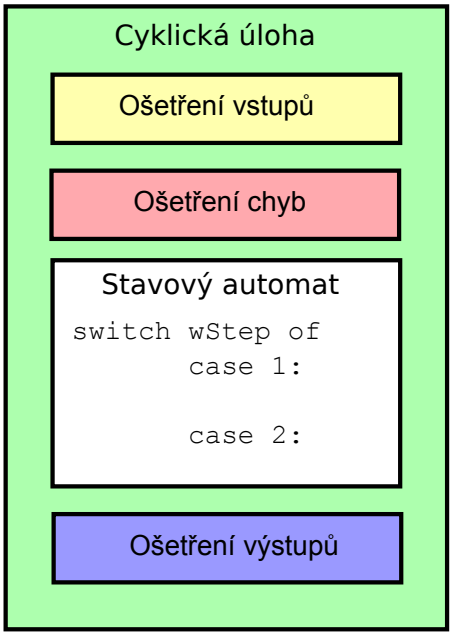
Programově je projekt rozdělen do samostatných cyklických úloh, z nichž většinou každá obstarává řízení některé části stroje nebo samostatnou technologii.

Cyklické úlohy bych rozdělil do tří základních skupin. První skupinou jsou úlohy řídící pohyb jednotlivých os systému. Druhou jsou úlohy řídící jednotlivé technologie, kterými může být stoj osazen a poslední skupinou jsou úlohy pro parametrizaci, komunikaci, HMI atd.

V následujících odstavcích popíšu blíže koncept podle kterého jsou programovány úlohy řídící jednotlivé části systému. Tato předloha může sloužit jako doporučení pro programování podobných systémů a zařízení za účelem úspory psaného zdrojového kódu, ale hlavně z důvodu zachování konzistence a přehlednosti projektu.

8.4.1. Obecné zásady pro programování

Zdrojový kód by měl být utříděn pokud možno do logických celků. Takovým celkem může být v případě systémů B&R cyklická úloha, knihovna funkcí, nebo funkčních bloků. Ve většině případů je potom vhodné použít strukturu dané cyklické úlohy takovou, jaká je uvedena na Obr. 8.3.



Obr. 8.3 Struktura cyklické úlohy

Na začátku cyklické úlohy je vhodné ošetřit vstupy, které danou část technologie ovlivňují. Následuje kontrola, zdali se nevyskytla nějaká chyba. V případě, že ano, tak je

vhodné uložit potřebné parametry a „poslat“ stavový automat popsaný v dalším odstavci do stavu řešící chyby procesu.

Hlavní částí cyklické úlohy je stavový automat. Obvykle obsahuje inicializační stav, kde vyčkává na požadavek, při jehož příchodu následuje některý z dílčích stavů řešící daný úkol. Po skončení této procedury se opět vrací do inicializačního stavu. Programově je implementován pomocí řídicí struktury CASE resp. SWITCH.

Na konci cyklické úlohy se nachází ošetření výstupů daného procesu nejen ve smyslu výstupních proměnných, ale i proměnných ovlivňující jiné části technologie (jiné cyklické úlohy).

8.4.2. Datová struktura cyklické úlohy

Procesní data pro danou část technologie je vhodné uchovávat pohromadě v jednom strukturovaném datovém typu. V něm jsou proměnné rozděleny podle jejich určení do skupin:

Cmd

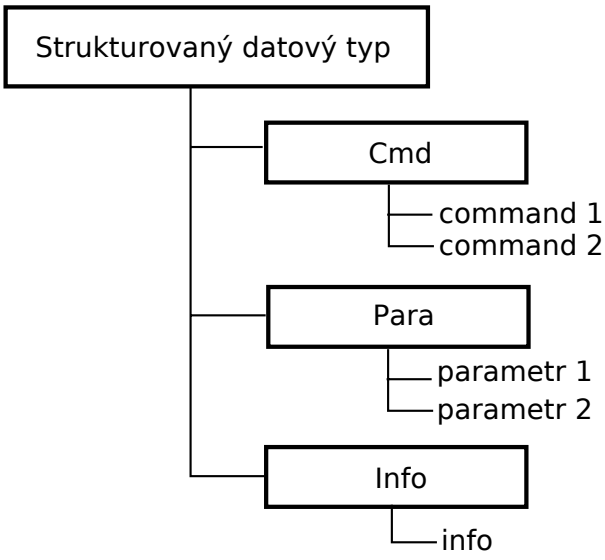
Obsahuje většinou booleovské proměnné, pomocí kterých se daná část technologie řídí. Je to tudíž vstup do daného subsystému.

Para

Část struktury Para obsahuje parametry pro danou technologii. Proměnné jsou většinou také vstupem do systému.

Info

Poslední část struktury Info obsahuje informace o stavu technologie. Lze ji považovat za výstup dané části technologie.



Obr. 8.4 Datová struktura pro cyklickou úlohu

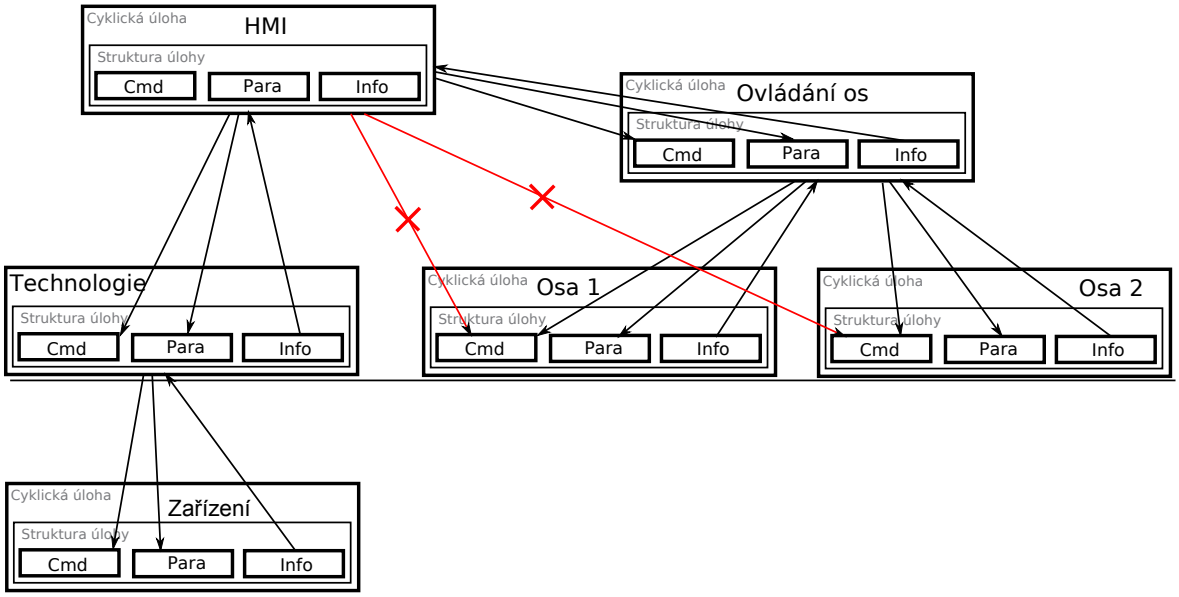
Datová struktura cyklické úlohy je potom umístěna v globálních proměnných a lze k ní přistupovat i z ostatních cyklických úloh. Tento přístup by se opět měl řídit určitými pravidly.

8.4.3. Hierarchie cyklických úloh a komunikace mezi nimi

Jak bylo zmíněno v předchozím odstavci, tak datová struktura by měla sloužit ke komunikaci mezi jednotlivými cyklickými úlohami. Každá cyklická úloha by měla mít tedy svoji vlastní strukturu globálního rozsahu, aby k ní bylo možné přistupovat z jiných úloh a tím způsobem úlohu ovládat.

Obecně je vhodné utřídit do hierarchie jak znázorněná v příkladě na Obr. 8.5.

Pravidlem je pak, že nadřazené cyklické úlohy smějí zapisovat hodnoty do struktur Cmd a Para úloh podřízených, přičemž u struktury Cmd se jedná pouze o nastavení příslušné proměnné do logické „1“. O resetování proměnné se stará sama cyklická úloha. Struktura Info pak slouží ke komunikaci směrem z cyklické úlohy ven a z vnějšku by se do ní nemělo odnikud zapisovat.



Obr. 8.5 Hiearchie cyklických úloh

Na příkladu na Obr. 8.5. je také znázorněn červenou barvou méně koncepční přístup, který má za následek velikou složitost nadřazené úlohy (v tomto případě HMI), která má navíc z logického pohledu jiný účel, než operace, které by se v ní takto vyskytovaly.

8.5. Popis výchozího projektu

8.5.1. Osy systému

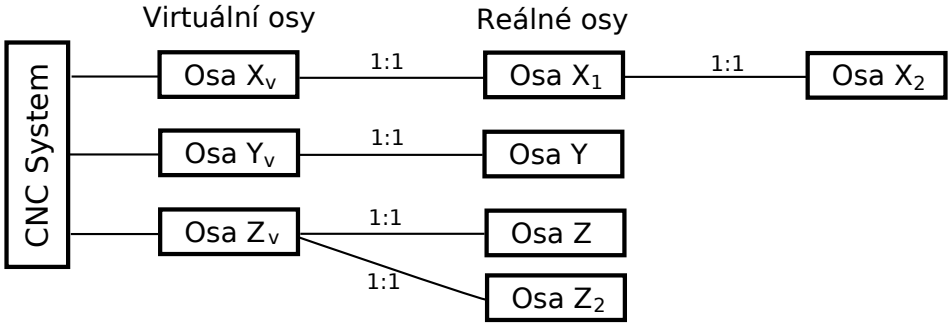
Jelikož systém obsahuje mnoho os, bylo by psaní ovládaní každé osy zvlášť velmi náročné a případná potřebná změna by znamenala zásah do všech cyklických úloh os systému.

Z tohoto důvodu je pro řízení os vytvořena knihovna (pkAxis_2) obsahující funkční blok, pomocí kterého lze jednotlivé osy ovládat. Dále je v knihovně definován strukturovaný datový typ pk_AX_l_main_type, jehož adresa je jediným vstupem funkčního bloku. Tento datový typ obsahuje mimo jiné adresu datového objektu osy (ARNC0AXIS), což je klíčový objekt pro její řízení. Uvnitř funkčního bloku je osa řízena pomocí NC akcí a je použita knihovna ARNC0.

Každá osa má svoji vlastní cyklickou úlohu, ve které je funkční blok volán, jsou ošetřeny vstupy, parametry a výstupy. Funkční blok je ovládán strukturou pk_AX_g_main type, která má globální rozsah.

8.5.2. Osy systému a CNC

Systém os v projektu je navržen tak, aby byl flexibilní vůči různým konfiguracím strojů a také aby umožňoval ovládání jednak CNC systémem a jednak standardně aplikací. Z toho důvodu jsou vytvořeny virtuální osy, které jsou nalinkovány do CNC systému. Reálné osy jsou s nimi potom svázány lineární vačkou 1:1. Koncept os je naznačen na Obr. 8.6.



Obr. 8.6 Koncept os v CNC aplikaci

Výhodou tohoto uspořádání je možnost rozvazbení dvojice os při pauze programu, změnění polohy nástroje a pokračování v pálicím plánu.

8.5.3. Cyklické úlohy technologií

Technologie řezání plazmou, kyslíkem a technologie značení jsou implementovány podle způsobu popsaném v 8.4.Protože na jednom stroji mohou být instalovány více než jedna řezací plazmová hlavice, je program pro řízení plazmy realizován pomocí funkčního bloku, jako tomu je i u os systému.

8.6. Technologie vrtání a frézování

Technologie by měla umožňovat vrtání děr jednak pomocí přímého naprogramování v G-kódu (vrtacím a pálicím plánu) a také automatické vyvrtání díry po zadání parametrů buď ve vrtacím plánu nebo ve vizualizaci. Dále by měla být technologie schopna zjišťovat polohu materiálu vůči nástroji pomocí odměřovacího zařízení.

8.6.1. Osa Z

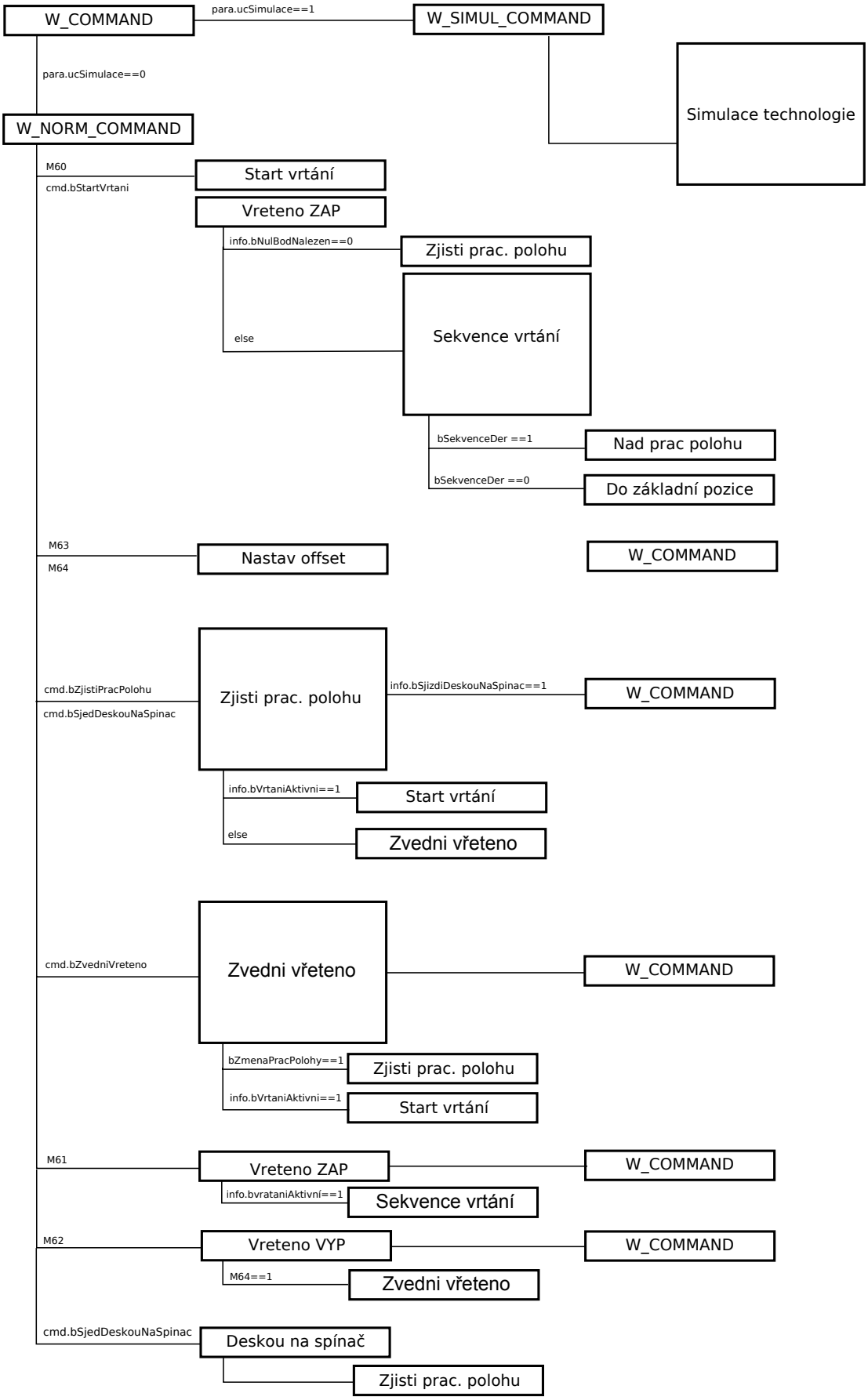
Přidání nové technologie vrtání, která využívá vertikálního pohybu supportu si vyžádalo přidání nové fyzické osy do projektu. Nová osa pro vrtání frézování se jmenuje zvf a byla vytvořena podle konceptu popsaném v 8.5.2.

Aby mohla být osa ovládána z CNC pálicího/vrtacího plánu, je navazbena na virtuální osu Z. Ostatní nastavení vychází z nastavení pro osu z technologie autogenní tříhořákové hlavy na jejímž supportu je vrtačka instalována.

8.6.2. Cyklická úloha technologie vrtání a frézování

Cyklická úloha pro technologii vrtání byla vytvořena opět podle konceptu popsaném v 8.5.3. Jedná se tedy o stavový automat, který lze řídit přes příkazy z části struktury Cmd nebo přes M-funkce CNC systému.

Na následujícím obrázku je znázorněn hlavní stavový automat s nejvýznamnějšími operacemi, které se v úloze vyskytují. Přechody mezi jednotlivými stavy a sekvencemi stavů jsou prováděny na základě hodnot z řídící struktury úlohy nebo interních proměnných.



Obr. 8.7 Stavový automat cyklické úlohy vrtání

8.6.3. Datová struktura technologie vrtání a frézování

Struktura pro technologii vrtání a frézování se nazývá tk_TECH_VRT_FREZ_main_type. Jednotlivé členy tohoto typu jsou uvedeny v Tab. 8.1 až Tab. 8.4 společně s jejich významem a případnými jednotkami.

tk_TECH_VRT_FREZ_main_type	
cmd	tk_VRT_FREZ_cmd_type
para	tk_VRT_FREZ_param_type
info	tk_VRT_FREZ_info_type
wStep	UINT

Tab. 8.1 Struktura tk_TECH_VRT_FREZ_main_type

tk_VRT_FREZ_param_type			
Proměnná	Typ	Jednotky	Význam
ucSimulace	USINT	0 = OFF, 1 = ON	Přepíná technologii do simulace, osy X,Y pracují, ostatní stojí
fTolerZmenyPracPolohy	REAL	mm	Maximální možná odchylka pracovní polohy při průběžném měření (při překročení je poloha znovu změřena)
ucModeVrtFrez	USINT	0 = vrtačka, 1=frézka	
fNajezdNastrojNadPP	REAL	mm	Výška, do které se má přiblížit nástroj před začátkem vrtání
fNajezdDeskaNadPP	REAL	mm	Výška, do které se má přiblížit přítlačná deska max. rychlostí nad materiál
fHledaniPracPozice_v	REAL	mm/min	Rychlost hledání pracovní polohy
fPriblizeniNastrojPP_v	REAL	mm/min	Rychlost sjíždění z výchozí pozice nad materál
fDelkaNastroje	REAL	mm	Délka nástroje
fVyskaHrotuNastroje	REAL	mm	Výška hrotu nástroje nad materiálem (při sepnutém ref. spínači)
fHloubkaVrtani	REAL	mm	Hloubka vrtání
iOtackyVretene	INT	ot/min	Otáčky vřetene
fRychlostVrtani	REAL	mm/min	Rychlost vrtání
fVychoziPozice	REAL	mm	Výchozí pozice pod horním koncovým spínačem
fVyjizdeniZDiryVen_v	REAL	mm/min	Rychlost vyjíždění z otvoru po vyvrtání
fRefZasunuti	REAL	mm	Vzdálenost uražená od prvního kontaktu odměřovacího mechanismu až o sepnutí referenčního spínače

Tab. 8.2 Struktura tk_TECH_VRT_FREZ_param_type

tk_VRT_FREZ_cmd_type		
bZjistPracPolohu	BOOL	Zjistí pracovní polohu (výšku materiálu)
bStartVrtani	BOOL	Spustí sekvenci vyvrtání díry dle aktuálních parametrů
bStopTechnol	BOOL	Zastaví technologii
bZvedniVreteno	BOOL	Zvedne vřeteno do výchozí polohy
bSjedDeskouNaSpinac	BOOL	Sjede deskou na referenční spínač, aby byl právě sepnutý

Tab. 8.3 Struktura tk_TECH_VRT_FREZ_cmd_type

tk_VRT_FREZ_info_type		
bHledaniPPaktivni	BOOL	Probíhá právě hledání pracovní polohy
bVrtaniAktivni	BOOL	Probíhá právě sekvence vyvrtání otvoru
bTechnologieAktivni	BOOL	Technologie je aktivní
ucStopTechnol	USINT	Technologie právě zastavuje (zastavení vřetene, návrat do základní polohy)
bNulBodNalezen	BOOL	Byla již nalezena pracovní poloha (po zapnutí stroje = 0)
bSjizdiDeskouNaSpinac	BOOL	Probíhá právě sjíždění deskou na ref. spínač

Tab. 8.4 Struktura tk_TECH_VRT_FREZ_info_type

8.6.4. Externí parametry

Aby se dalo nakonfigurovat automatické vrtání děr z vrtacího plánu, bylo nutné využít možnosti komunikace mezi CNC a PLC. Takovou možnost nabízejí externí parametry. Seznam použitých externích parametrů, jejich datový typ a význam je uveden v následující tabulce:

Externí parametry		
Ext. Parametr	Význam	Jednotky
EXW[60]	Otáčky vřetene	ot/min
EXF[61]	Rychlost vrtání	mm/min
EXF[62]	Hloubka vrtání	mm

Tab. 8.5 Tabulka externích parametrů

8.6.5. M-funkce

Technologii vrtání lze ovládat z vizualizace manuálně a z CNC vrtacího plánu. To je zajištěno pomocí synchronních M-funkcí a externích parametrů. Použité M-funkce jsou popsány v následujících odstavcích:

M60 – Vyvrtání otvoru

Tato M-funkce má stejný efekt jako příkaz cmd.bStartVrtani. Spouští sekvenci vyvrtání jednoho otvoru s právě nastavenými parametry (z vizualizace nebo z vrtacího plánu externími parametry).

Sekvence se skládá z dílčích kroků. Nejdříve se aktivuje rotace vřetene na nastavenou rychlost, přítlačná deska se přiblíží maximální rychlostí nad materiál a poté se nižší rychlostí (fpriblizeniNastrojPP_v) přiblíží nástroj nad obráběný materiál a aktivuje se upínání materiálu (prototyp stroje upínání nemá). Následuje vyvrtání otvoru nastavenou rychlostí do nastavené hloubky, vyjetí z vrtaného otvoru a uvolnění uchycení materiálu. Poslední krok závisí na nastavení, zdali je aktivována funkce vrtání sekvence otvorů. Pokud tomu tak je, tak support vyjede do pozice, kdy je přítlačná deska nad materiálem. V opačném případě se vrátí do výchozí polohy.

M61 – zapnutí rotace vřetene

Zapíná rotaci vřetene nastavenou rychlostí.

M62 – vypnutí rotace vřetene

Zastavení rotace vřetene.

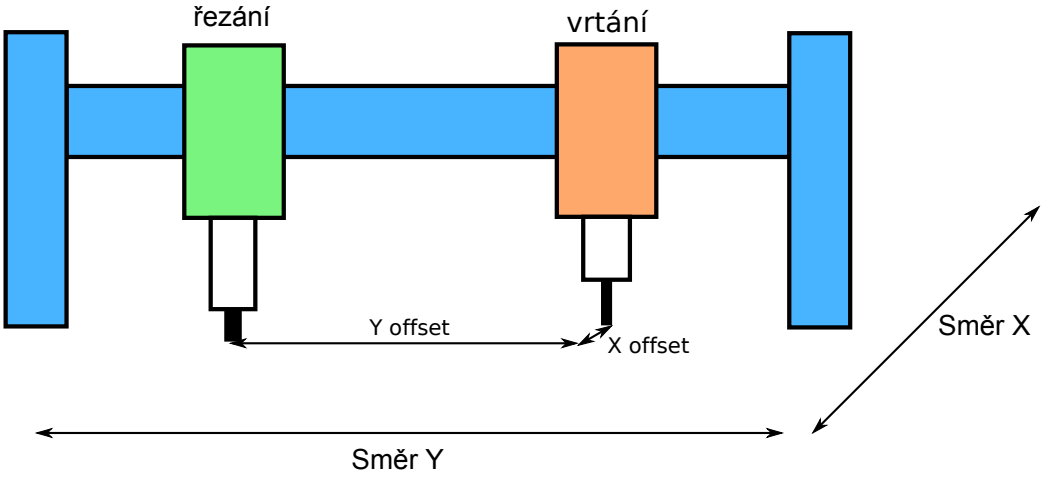
M63 – aktivace technologie vrtání/frézování

Aktivace technologie slouží především k posunu kartézských os systému, který je způsoben mechanickým uspořádáním stroje viz. Obr. 8.8. Při změně používané technologie jsou reálné osy X a Y odvazbeny od virtuálních os, které jsou ovládány CNC systémem. Poté jsou reálné posunuty o offset mezi technologiemi a zpět zavazbeny na virtuální. Poté je možné pokračovat v CNC programu bez zbytečného připočítávání offsetu v G-kódu.

Aktivace technologie zabraňuje také použití jejích funkcí při běhu jiné technologie.

M64 – aktivace technologie řezání

Jedná se o stejnou funkci jako předchozí M63, nýbrž pro aktivaci řezacích technologií.



Obr. 8.8 Prostorové uspořádání technologií

8.7. Ovládání vřetene vrtačky – Acopos Inverter

Na rozdíl od ostatních os systému je pro vřeteno vrtačky použit asynchronní motor. K řízení asynchronních motorů se v systémech B&R používají frekvenční měniče Acopos Inverter. Pro vývojový prototyp je použit asynchronní motor Siemens o výkonu 1,5kW a frekvenční měnič řady P84 o stejném výkonu.

8.7.1. Programové řešení ovládání Acopos Inverter

Programově je řízení asynchronního motoru řešeno tak, aby odpovídalo konceptu os systému popsaném v této práci. Na rozdíl od servomotorů se pro řízení asynchronního motoru přes frekvenční měnič nepoužívá žádná struktura osy, ale ovládá se pomocí namapovaných proměnných, které lze nastavit po přidání měniče do projektu.

8.7.2. Knihovna tk_ACPinv

Strukturovaný datový typ pro ovládání invertoru – tk_ACPi_interface

Pro ovládání frekvenčního měniče jsem vytvořil nový datový typ tk_ACPi_interface, jehož členy se shodují s kanály pro mapování ovládacích proměnných frekvenčního měniče. Členy typu tk_ACPi_interface jsou namapovány do těchto kanálů, tudíž je měnič ovládán pomocí jedné struktury a všechny proměnné jsou pohromadě.

Funkční blok tk_ACPi

Funkce ovládání Acoposu Invertor jsou implementovány ve funkčním bloku tk_ACPi. Tento přístup je opět motivován možností řídit více frekvenčních měničů pomocí jednoho funkčního bloku a při případné změně zdrojového kódu stačí upravit pouze jeden soubor s programem pro daný blok.

Struktura pro řízení funkčního bloku tk_ACPi_main_type

Řízení cyklické úlohy pro Acopos Inverter je řešeno opět strukturou podobnou jaké jsou použity u řízení ostatních os systému.

tk_ACPi_main_type	
cmd	tk_ACPi_cmd_type
para	tk_ACPi_param_type
info	tk_ACPi_info_type
wStep	UINT

Tab. 8.6 Struktura tk_ACPi_main_type

tk_ACPi_param_type			
Proměnná	Typ	Jednotky	Význam
iSpeed	USINT	ot/min	Otáčky motoru
bReverseRotation	REAL		Změna směru motoru
bSimulation	USINT		Simulace zařízení

Tab. 8.7 Struktura tk_ACPi_param_type

tk_ACPi_cmd_type		
bStart	BOOL	Start motoru dle nastavených otáček
bStop	BOOL	Zastavení motoru

Tab. 8.8 Struktura tk_ACPi_cmd_type

tk_ACPi_info_type		
ucCmdToReady	BOOL	Zařízení připraveno

Tab. 8.9 Struktura tk_ACPi_info_type

Kapitola 9

9. Porovnání řízení CNC a Žongléra

9.1. Knihovny pro řízení os

Nároky na řízení obou strojů vycházejí hlavně z účelu, ke kterému mají sloužit. Jelikož jedním z cílů Žongléra je sloužit jako nástroj pro výuku řízení servopohonů, tak je žádoucí, aby bylo umožněno využít co nejvíce technik programování. Z tohoto důvodu je pro řízení os Žongléra použita knihovna ACP10, která umožňuje použití standardních bloků PLCopen vycházejících z normy IEC 61131-3 pro řízení průmyslových systémů.

Pro řízení os u pálicích (vrtacích) strojů Vanad je použita knihovna ARNC0, která obsahuje CNC systém, jenž je pro celý projekt zásadní. Kromě chybějící podpory pro řízení pomocí PLCopen nabízí knihovna stejné možnosti řízení jako ACP10 u Žongléra. Zde jen poznamenám, že osy řízené knihovnou ACP10 je možné použít v CNC systému knihovny ARNC0. Obě knihovny mohou být použity tedy na jednom systému zároveň.

9.2. Programování systémů

V případě Žongléra se objevil nezvyklý požadavek, aby studenti nemohli měnit určité části projektu vztahující se k systému zabezpečení proti kolizím. Žádnou takovou možnost vývojové prostředí Automation Studio nenabízí. Tento problém je tudíž řešen pomocí druhého PLC, ke kterému studenti nemají přístup a které kontroluje přítomnost potřebných cyklických úloh na hlavním PLC Power Panelu. Více v části 4.10.

U CNC systému se objevuje požadavek, aby se pomocí jednoho softwarového projektu mohlo řídit více lišících se zařízení. Tento problém je v tomto případě řešen možností simulací jednotlivých os či technologií.

9.3. Synchronizace

Společným znakem obou zařízení je synchronizace více os navzájem. V CNC aplikacích se jedná o projetí přesné trajektorie definovanou rychlostí. U Žongléra je kladen důraz především na synchronizaci z časového hlediska, aby se osy nacházely ve správný čas na správném místě s pevně definovanou rychlostí.

Prostředkem pro synchronizaci u CNC systémů je samotný CNC operační systém, respektive Path generátor (Obr. 7.3), který vytváří požadované trajektorie jednotlivých os. Dalším nástrojem pro synchronizaci jsou například M-funkce popsané v 7.4.1.

V případě Žongléra hraje zásadní roli pro synchronizaci vačkový automat. Jeho základem je virtuální „Master“ osa, která představuje časovou osu. K ní jsou zavazbeny jednotlivé reálné osy systému, jejichž pohyb je definován pomocí vačkových profilů. Díky tomu jsou osy precizně časově i pozičně synchronizovány.

9.4. Vizualizace a HMI

Obě zařízení obsahují i vizualizaci a HMI. U Žongléra jsou požadavky na tyto části relativně nízké. Jelikož se jedná o statické zařízení umístěné stále na jednom místě, postačuje k ovládání dotykový displej na Power Panelu.

U pálicích strojů jsou požadavky značně vyšší. Jednak samotná vizualizace musí být dostatečně velká, aby bylo možné zobrazit rozumným způsobem pálicí plán. K tomu slouží Zobrazovací panel s klávesnicí. U rozměrově větších strojů je požadavek, aby se mohla obsluha pohybovat kolem zařízení a stále jej ovládat. K tomu slouží mobilní panel MP50 s grafickým displejem.

9.5. Návrhy na změny v projektech

Na základě získaných zkušeností během tvorby této práce jsem vyvodil návrhy pro zlepšení u prezentačního projektu Žongléra. Doporučení se týká stylu a zásad programování, které bych upravil tak, aby odpovídaly způsobu popsaném v 8.4. Projekt by se tak stal lépe čitelný pro případné nové studenty, kteří by s ním pracovali.

Kapitola 10

10. Závěr

V této diplomové práci jsem se snažil splnit všechny cíle uvedené v zadání práce.

V první části práce jsem se zabýval zabezpečením Žongléra, aby mohl být nasazen v systému vzdálené výuky Lablink. Tento cíl jsem vyřešil s použitím hardwarového řešení se standardními komponentami firmy B&R, jejíž produkty je model řízen. V práci jsem také uvedl další možná řešení problému s použitím jiného hardware, která by zlepšovala některé parametry zabezpečení. Nicméně konečné řešení, které bylo nakonec realizováno, splňuje všechny požadavky, tudíž je možné ho použít k zabezpečení práce na Žongléru.

Druhá část práce popisuje část vývoje aplikace řízení CNC frézky pomocí systémů B&R. Cílem bylo navrhnout a realizovat řízení prototypu tohoto zařízení. V první polovině druhé části jsem rozebral koncept, možnosti a úskalí řízení CNC aplikací obecně. Bohužel celkový rozsah celého CNC řízení B&R výrazně překračuje možnosti této práce. Proto ho lze považovat jen za nezbytný minimální přehled pro vytvoření aplikace jako je realizované vrtací zařízení. V druhé polovině jsem popsal některé zásady pro programování PLC systémů, které vedou ke zpřehlednění programu a které jsem společně s poznatky z první části aplikoval při programování aplikace technologie vrtání.

Konkrétní detaily programu a projektu nelze v této práci uvést z důvodu ochrany zadavatele projektu a jeho know-how.

V této části práce však byla technologie otestována pouze bez části řídící frekvenční měnič, který fungoval jen v simulaci. Tento nedostatek byl způsoben chybou při prvotním zprovoznění měniče a následnými problémy s jeho dostupností.

Na závěr práce jsem popsal přístup k řízení obou typů mechanismů, se kterými jsem během této práce pracoval.

Během realizace této práce jsem získal mnoho zkušeností z oblasti řízení průmyslových systémů, což pro mě hodnotím jako veliký přínos z důvodu mého budoucího pracovního zaměření, které se s danými tématy shoduje. Kromě zkušeností s realizací řízení se naskytlo také mnoho příležitostí získat cenné zkušenosti s prezentací Žongléra na veletrzích nebo souvisejících prací ve studentských soutěžích.

Seznam použité literatury

[1] KOHOUT TOMÁŠ, *Bakalářská práce: Vizuální kontrola chodu stroje*, Katedra řídicí techniky ČVUT, Praha 2009

[2] JAROŠ PAVEL, *Bakalářská práce: Dynamika rychlých servopohonů,* Katedra řídicí techniky ČVUT, Praha 2009

[3] PRUDEK LUBOMÍR, *Bakalářská práce: Řízení rychlých servopohonů*, Katedra řídicí techniky ČVUT, Praha 2010

[4] NECID JOSEF, *Diplomová práce: Bezpečnost strojů a opatření ke snížení rizik*, Katedra řídicí techniky ČVUT, Praha 2011

[5] JAROŠ PAVEL, *Diplomová práce: Rozšíření modelu žonglér a řízení CNC stroje*, Katedra řídicí techniky ČVUT, Praha 2011

[6] GRUDININ ILJA, *Diplomová práce: Řízení rychlých servopohonů*, Katedra řídící techniky ČVUT, Praha 2008

[7] BOČEK KAREL, *Diplomová práce: Vývoj zařízení PROFINET IO Device*, Katedra řídící techniky ČVUT, Praha 2009

[8] NĚMEC TOMÁŠ, *Diplomová práce: Positional visual feedback control*, Katedra řídící techniky ČVUT, Praha 2007

[9] PŠENIČKA JAN, *Diplomová práce: Synchronizace servopohonů*, Katedra řídící techniky ČVUT, Praha 2007

[10] MEZERA PAVEL, PRUDEK LUBOMÍR, KOHOUT TOMÁŠ, JAROŠ PAVEL, BURGET PAVEL, *Visual-Feedback Juggler With Servo Driv*es, Katedra řídicí techniky ČVUT, Praha 2009

[11] MEZERA PAVEL, PRUDEK LUBOMÍR, KOHOUT TOMÁŠ, NECID JOSEF, JAROŠ PAVEL, BURGET PAVEL, *Trajectory prediction in motion control with fast servo drives*, Katedra řídicí techniky ČVUT, Praha 2009

[12] EPSG, *Ethernet POWERLINK Communication profile specification EPSG DS 301 V1.1.0*, Ethernet POWERLINK Standardisation Group 2008

[13] *Automation Studio Online help*, Bernecker & Rainer GMBH

Přílohy

A. Ukázky zdrojových kódů

A.1 Konfigurační objekt pro IMA na X20

```
"[ROUTING_PATH]"
"/SECTION=INTERFACE"
"/CN=IF3.1"

"[PV_LIST]"
"/SECTION=PV"
"/SV=READ /SERVPV=gRAxis01  /CLNTPV=X20IMAcom:axis_1_param"
"/SV=READ /SERVPV=gRAxis02  /CLNTPV=X20IMAcom:axis_2_param"
"/SV=READ /SERVPV=gRAxis03  /CLNTPV=X20IMAcom:axis_3_param"
"/SV=READ /SERVPV=gRAxis04  /CLNTPV=X20IMAcom:axis_4_param"
"/SV=READ /SERVPV=gRAxis05  /CLNTPV=X20IMAcom:axis_5_param"
"/SV=READ /SERVPV=PP420check:dat_obj_created
/CLNTPV=X20check:dat_obj_created"
"/SV=READ /SERVPV=PP420check:errPP420  /CLNTPV=error_PP420"

"[BRMODUL]"
"/SECTION=BRMODUL"
"/SV=UPLOAD /LD=DRAM /MO=dOCheck
/CMDPV=X20check:download_dOCheck"
"/SV=UPLOAD /LD=DRAM /MO=dOPlink
/CMDPV=X20check:download_dOPlink"
"/SV=UPLOAD /LD=DRAM /MO=dOIomap
/CMDPV=X20check:download_dOIomap"
```

A.2 Konfigurační objekt pro IMA na PP420

```
"[ROUTING_PATH]"
"/SECTION=INTERFACE"
"/CN=SL1.SS1.IF2.2"

"[PV_LIST]"
"/SECTION=PV"
"/SV=READ /SERVPV=X20paramsOK  /CLNTPV=X20paramsOK"
"/SV=READ /SERVPV=X20homeOK  /CLNTPV=X20homeOK"
"/SV=READ /SERVPV=X20check:dat_obj_created
/CLNTPV=PP420check:dat_obj_created"
```

B. Zapojení do systému Lablink

V následujících bodech uvedu postup a návod pro řešení problémů při zapojování Žongléra do systému vzdálené výuky Lablink. Uvedené pokyny rozdělím na část určenou pro administrátora a na část určenou pro studenty pracující s modelem.

B.1 Pokyny určené pro administrátora

Rozdělení projektů

Softwarové projekty jsou zvlášť pro administrátora a pro studenty. Administrátorský projekt obsahuje veškeré cyklické úlohy a proměnné potřebné pro zabezpečení:

- Powerlink – Řeší přenos poloh a rychlostí a Acoposů do X20
- IMAcfgX20 – Konfigurační objekt pro IMA komunikaci pro X20
- X20IMAcom – Časově nekritická část zabezpečení – kontrola parametrů
- X20check - Časově nekritická část zabezpečení – kontrola cyklických úloh
- X20safety – Časově kritická část zabezpečení
- IMAcfg420 - Konfigurační objekt pro IMA komunikaci pro PP420
- PP420IMAcom – Časově nekritická komunikace mezi PP420 a X20
- PP420check - Časově nekritické zabezpečení – kontrola cyklických úloh

Ze studentského projektu jsou odstraněny úlohy, které se nachází na PLC X20. Projekt tedy obsahuje jen ty, které pracují na PP420:

- Powerlink – Řeší přenos poloh a rychlostí a Acoposů do X20
- IMAcfg420 - Konfigurační objekt pro IMA komunikaci pro PP420
- PP420IMAcom – Časově nekritická komunikace mezi PP420 a X20
- PP420check - Časově nekritické zabezpečení – kontrola cyklických úloh

Nahrání projektů do PLC

Před začátkem práce studentů je nutné nejprve nahrát administrátorský projekt do obou PLC. V Configuration view je nutné poklepat pravým tlačítkem myši na danou konfiguraci a zvolit Set as Active Configuration. Poté je možné konfiguraci zkompilovat a nahrát do PLC. Takto je nutno postupovat pro obě PLC.

Kontrola BR modulů

Pokud by bylo nutné někdy udělat zásah do cyklických úloh nebo konfiguračních souborů na PP420, které jsou kontrolovány, je potřeba nově vytvořené datové objekty (cyklické úlohy) přenést do PLC X20 aby mohly být kontrolovány.

Aby bylo možné toto provést, je nutné tyto objekty na PP420 vytvořit se stejným názvem jaký poté používají v PLC X20 (nestačí je přejmenovat v prostředí Windows).

- CdOCheck.br
- CdOIomap.br
- CdOPlink.br

Po nahrání projektu se změnami do PP420 se přejde do online monitoru (symbol lupy na panelu nástrojů). Po otevření cyklické úlohy PP420check se do variable watch přidá proměnná `MakeCopyObjects` (pravé tlačítko myši a volba Insert variable)a zadá hodnota TRUE (logická „1“). Tím jsou v paměti vytvořené potřebné datové moduly a jsou tudíž viditelné v Software configuration (otevře se poklepáním na symbol procesoru v Physical view) v online monitoru.

Nyní je nutné tyto moduly přenést do projektu. Jejich označením, poklepáním pravým tlačítkem a zadání volby Online->Load from Target jsou moduly nahrány do konfigurace do projektu. Nyní je lze nalézt ve složce projektu v adresáři Binaries a podadresáři PP420.

Dalším krokem je zkopírování modulů do složky Logical, kde se přepíšou původní používané moduly.

Nyní je možné konfiguraci pro PLC X20 zkompilovat a nahrát do PLC. Nakonec je potřeba v konfiguraci PP420 v Software configuration vymazat stažené moduly z PLC, aby se do něj zbytečně znovu nenahrávaly.

Touto sekvencí operací je změna v cyklických úlohách vyřešena a kontrolní algoritmus nebude vyhodnocovat chybu.

Změna parametrů os

V případě změny parametrů os v konfiguraci PP420 je nutné tyto změny provést i v kontrolní úloze na X20.

Změna se provede v lokálních proměnných cyklické úlohy X20IMAcom. Zde jsou pro každou osu skupiny parametrů, které jsou kontrolovány a mají přiřazené pevné hodnoty již z projektu, které je potřeba změnit, aby souhlasily s konfigurací PP420. Po provedení změn je nutné projekt znovu zkompilovat a nahrát do PLC.

Změna parametrů zabezpečení

Změna parametrů zabezpečení se provádí v cyklické úloze X20safety, kde jsou všechny parametry na začátku úlohy definovány pomocí maker #define.

Tvorba nových omezení pohybu os

V případě potřeby vytvoření nového omezení pohybu nebo změny stávajících omezení je potřeba změnit, nebo přidat nové členy do logického výrazu umístěného v úloze X20safety.c.

Logický výraz:

```
goSafeToRun    =      WATCH_AXIS_LIMITS(1)          \
                      &&     WATCH_AXIS_LIMITS(2)          \
                      &&     WATCH_AXIS_LIMITS(3)          \
                      &&     WATCH_AXIS_LIMITS(4)          \
                      &&     WATCH_AXIS_LIMITS(5)          \
                      &&     (WATCH_AXIS_COLISION(2, 4) || (WATCH_AXIS_REGION(1,
AXIS_1_LIMIT_UP, AXIS_1_3_COLLISION_LIMIT) ||  WATCH_AXIS_REGION(3, AXIS_3_LIMIT_UP,
AXIS_1_3_COLLISION_LIMIT)))   \
                      &&     (WATCH_AXIS_COLISION(2, 5) || WATCH_AXIS_REGION(1,
AXIS_1_LIMIT_UP, AXIS_24_5_COLLISION_LIMIT))\
                      &&     (WATCH_AXIS_COLISION(4, 5) || WATCH_AXIS_REGION(3,
AXIS_3_LIMIT_UP, AXIS_24_5_COLLISION_LIMIT))\
                      &&  (WATCH_AXIS_REGION(2, 6000, 5000) || giAxis1position <
AXIS_13_ROD_COLLISION_LIMIT)                                    \
                      &&  (WATCH_AXIS_REGION(4, 6000, 5000) || giAxis3position <
AXIS_13_ROD_COLLISION_LIMIT);
```

Dostupná makra pro tvorbu omezení jsou popsána v části 4.9 této práce.

Řešení problémů

Při problému s výstupem celého zabezpečení je při hledání důvodu vhodné postupovat následujícím způsobem:

Otevřít cyklickou úlohu X20safety, přejít do Online monitoru a do Variable watch přidat následující proměnné:

`goSafeToRun`

V případě hodnoty FALSE je nutné hledat příčinu, proč je výstup algoritmu v logické 0.

`X20homeOK`

Pokud je hodnota této proměnné FALSE, je příčina v tom, že nejsou osy zinicializovány.(“homing”).

`X20paramsOK`

V případě hodnoty FALSE je problém v parametrech os, které nesedí. Další řešení problému je nutné hledat v úloze X20IMAcom stejným způsobem pomocí Online monitoru, kde je nutné sledovat booleovské proměnné končící „OK“.

`allModulesOK`

V případě hodnoty FALSE je problém v parametrech os, které nesedí. Další řešení problému je nutné hledat v úloze X20check.

B.2 Pokyny pro studenty

Projekt pro studenty

Pro studenty je určen samostatný projekt, ve kterém jsou připraveny nakonfigurované osy systému, zabezpečení stroje a základ vizualizace.

Projekt neobsahuje cyklické úlohy a proměnné používané na kontrolním PLC X20. Tento projekt bude umístěn na vzdálené ploše a bude pro studenty výchozí.

Co v projektu studenti nesmí měnit

Studenti nesmí v projektu měnit části vztahující se k zabezpečení. Jedná se o tyto části:

- Cyklická úloha Powerlink
- Cyklická úloha PP420check
- Cyklická úloha PP420IMAcom
- Datový objekt IMAcfg420
- Init parametry os systému (složka Axis config v Logica view)
- Jakkoliv používat globální proměnné vztahujíí se k zabezpečení (již přítomny v připraveném projektu)

Maximální možná zrychlení

Studenti mají povoleno ve svých projektech používat maximální zrychlení 100m/s^2 ve vertikálním směru a 50 000 °/ s^2 horizontálním směru. Rychlosti jsou omezeny na 10m/s vertikálně a 5000°/s horizontálně.

Prodlení při startování Acoposů

Při špatném nastavení parametrů je ve vizualizaci aktivní alarm „Špatné nastavení os“. Po startu systému je tento alarm aktivní, protože jsou vypnuty serozesilovače Acopos. Po stisku tlačítka Reset ve vizualizaci a inicializaci Acoposů („bootování“) tento alarm zanikne. V jiném případě je nastavení os opravdu špatné.

Chyba „Quick stop input active…“

Pokud se objeví na některé z os tato chyba, je stroj zastaven výstupem z bezpečnostního algoritmu.