FAKULTA MECHATRONIKY,
INFORMATIKY A MEZIOBOROVÝCH
STUDIÍ TUL

Diplomová práce

Software pro řízení experimentální linky na výrobu nanovlákenných produktů

Studijní program: N0714A270010 Mechatronika
Autor práce: **Bc. Pavel Vaščuk**
Vedoucí práce: Ing. Martin Diblík, Ph.D.
Ústav mechatroniky a technické informatiky

Liberec 2024

FAKULTA MECHATRONIKY,
INFORMATIKY A MEZIOBOROVÝCH
STUDIÍ TUL

Zadání diplomové práce

Software pro řízení experimentální linky na výrobu nanovlákenných produktů

Jméno a příjmení:	**Bc. Pavel Vaščuk**
Osobní číslo:	M22000060
Studijní program:	N0714A270010 Mechatronika
Zadávající katedra:	Ústav mechatroniky a technické informatiky
Akademický rok:	2023/2024

Zásady pro vypracování:

1. Seznamte se s aktuálním mechanickým, elektrickým a softwarovým řešením experimentální linky KOPRIS1 na výrobu nanovlákenných produktů.
2. Na základě zjištěných skutečností navrhněte úpravy softwarového vybavení linky tak, aby tyto reflektovaly aktuální požadavky uživatelů.
3. Pro ovládání linky navrhněte a realizujte nové HMI rozhraní s využitím technologie mappView pro novější grafický dotykový panel.

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

Seznam odborné literatury:

[1] JOHN, Karl-Heinz a Michael TIEGELKAMP. IEC 61131-3: programming industrial automation systems: concepts and programming languages, requirements for programming systems, decision-making aids. Second edition. Berlin : New York: Springer, 2010. ISBN 978-3-642-12014-5.

[2] JAROSLAV, Beran; MARTIN, Bílek; MARTIN, Diblík; ONDŘEJ, Baťka; JOSEF, Skřivánek et al. *Line for production of flat composite nanofibrous materials using AC electrospinning. Research report of sub-project TN01000015 NCK for year 2020*. 2021-01-14T18:05:55Z

[3] VALTERA, Jan; KALOUS, Tomáš; POKORNÝ, Pavel; BAŤKA, Ondřej; BÍLEK, Martin et al. *Fabrication of dual-functional composite yarns with a nanofibrous envelope using high throughput AC needleless and collectorless electrospinning*. NATURE PUBLISHING GROUP, MACMILLAN BUILDING, 4 CRINAN ST, LONDON N1 9XW, ENGLAND, 2019-09-30T07:39:08Z. Dostupné z: https://doi.org/10.1038/s41598-019-38557-z.

Vedoucí práce:	Ing. Martin Diblík, Ph.D. Ústav mechatroniky a technické informatiky
Datum zadání práce:	12. října 2023
Předpokládaný termín odevzdání:	14. května 2024

L.S.

prof. Ing. Zdeněk Plíva, Ph.D.
děkan

doc. Ing. Josef Černohorský, Ph.D.
vedoucí ústavu

V Liberci dne 12. října 2023

Prohlášení

Prohlašuji, že svou diplomovou práci jsem vypracoval samostatně jako původní dílo s použitím uvedené literatury a na základě konzultací s vedoucím mé diplomové práce a konzultantem.

Jsem si vědom toho, že na mou diplomovou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

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

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

Současně čestně prohlašuji, že text elektronické podoby práce vložený do IS/STAG se shoduje s textem tištěné podoby práce.

Beru na vědomí, že má diplomová práce bude zveřejněna Technickou univerzitou v Liberci v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů.

Jsem si vědom následků, které podle zákona o vysokých školách mohou vyplývat z porušení tohoto prohlášení.

18. března 2024 Bc. Pavel Vaščuk

SOFTWARE PRO ŘÍZENÍ EXPERIMENTÁLNÍ LINKY NA VÝROBU NANOVLÁKENNÝCH PRODUKTŮ

ABSTRAKT

Tato práce se v teoretické části zabývá programovatelnými logickými automaty, jejich programovacími jazyky a teorií designu datových struktur. Popisuje řešení softwarové a hardwarové bezpečnosti v kontextu programovatelných logických automatů. V obecné rovině se zaměřuje na základní principy tvorby a výroby nanovlákenných struktur. Popisuje konstrukci a zabezpečení experimentální linky KOPRIS1.
V praktické části práce analyzuje za pomoci vývojového prostředí B&R Automation Studio stávající softwarový projekt pro experimentální linku KOPRIS1. Zabývá designem a tvorbou struktury podporující hardwarovou rekonfiguraci stroje z úrovně vykreslování VC4 a MappView.

Klíčová slova: B&R, MappView, Zvlákňování střídavým proudem, Design datových a programových struktur stroje

ABSTRACT

The theoretical part of this work deals with programmable logic controllers, their programming languages and the theory of designing data structures. It describes solutions for software and hardware security in the context of programmable logic controllers. In general, it focuses on the basic principles of creating and producing nanofiber structures. Additionally, it describes the construction and security of the experimental line KOPRIS1.
In the practical part of the work, it analyzes the existing software project of the experimental line KOPRIS1 using the B&R Automation Studio development environment. Furthermore, it addresses the design and creation of a structure supporting hardware reconfiguration of the machine from the level of VC4 rendering and MappView.

Keywords: B&R, MappView, Electrospinning using alternating current, Design of data and program structures

PODĚKOVÁNÍ

Úvodem bych chtěl vyjádřit své poděkování všem, kteří umožnili vznik této diplomové práce.
Zejména chci poděkovat Ing. Martinovi Diblíkovi Ph.D., vedoucímu diplomové práce, za odborné vedení, dohled, i za možnost účastnit se na dalším rozvoji tak zajímavé technologie, jakými jsou zařízení pro tvorbu a zpracování nanomateriálů.
Dále bych chtěl poděkovat Ing. Tomášovi Myslivcovi za konzultace při řešení problematiky designu programů a nového vizuálního rozhraní MappView, kolegyni Anežce Barákové za odborné rady v oblasti výroby textilií a nanomateriálů a reprezentantům české odnože společnosti B&R za poskytnutí přístupu ke studijním materiálům.

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM ZDROJOVÝCH KÓDŮ

SEZNAM ZKRATEK

FB	„Function block“ Funkční blok
PLC	„Programable logical controller“ Programovatelný logický automat
DB	„Data block“ Blok dat programu
UDP	„User Datagram Protocol“ Nespojový Ethernetový protokol
TCP	„Transmission Control Protocol“ Spojový Ethernetový protokol
HS	„Hard Stop“ Nouzové zastavení
CAN	„Controller Area Network“ Komunikační protokol
OMC	„Open Motion Control“ Standard komunikace s pohony
PID	„Proporcional Integral Derivative“ Regulace za pomoci proporcionální, integrační a derivační složky
MPC	„Model Predictive Control“ Regulace za pomoci modelu systému
VNC	„Virtual Network Computing“ Vzdálené grafické uživatelské rozhraní
CPU	„Central processing unit“ Procesor
MTBF	„Mean Time Between Failures“ Střední hodnota mezi poruchami
I2C	„Inter-Integrated Circuit“ Komunikační protokol
MODBUS	Komunikační protokol

UART	„Universal Asynchronous Receiver / Transmitter“ Komunikační protokol
LAN	„Local Area Network“ Lokální Ethernetová síť
MAC	„Medium Access Control address“ Identifikátor síťového zařízezí
ASIC	„Application-Specific Integrated Circuit“ Integrovaný obvod
PL	„Performance Level“ Úroveň bezpečnosti součástí řídicích systémů
SIL	„Safety Integrity Level“ Úroveň bezpečnostních požadavků
IO	„Inputs and Outputs“ Vstupy a výstupy

ÚVOD

Zvyšování produkce nanovláken a snižování obtížnosti jejich výroby je klíčovým krokem v průmyslu i vědě. Stávající zařízení a technologie doposud umožňovaly produkci nanovlákenných materiálů pouze v omezené míře. Intenzivním vývojem a výzkumem v oblasti nanotechnologií dochází k postupnému vzniku procesů umožňujících výrobu nanomateriálů i v průmyslovém měřítku. Technická univerzita v Liberci, jako průkopník výzkumu nanomateriálů, nemohla zůstat pozadu, a tak vznikla myšlenka na zadání softwarové inovace experimentální linky pro tvorbu nanovláken KOPRIS1. Cílem projektu bylo zajistit jednoduchou modifikovatelnost linky, vytvoření funkčních bloků, které by byly aplikovatelné i na jiné stroje. Zajistit hardwarovou rekonfigurovatelnost linky, zpracovat novou vizuální formu za pomoci technologie MappView a připravit komunikační rozhraní s externími zařízeními.

Mým osobním cílem bylo dosáhnout u projektu stavu, který se blíží standardu průmyslového stroje s tím, že stávající funkčnost zařízení KOPRIS1 nebude nijak ovlivněna.

TEORETICKÁ ČÁST

2.1 PLC

2.1.1 Přehled

PLC neboli programovatelný logický automat je v současné době nedílnou součástí každého většího technologického celku. S využitím v průmyslu, lékařství, výzkumu atd. PLC nám poskytuje vysokou bezpečnost, modulárnost a adaptabilitu při vývoji stroje, a to díky svému programovatelnému chování. PLC umožňuje přijímat, zpracovávat a reagovat na podněty, které jsou většinou zprostředkovávány formou elektrických signálů. Tyto signály jsou zaznamenávány za pomoci digitálních, analogových, či jiných vstupů a speciálních rozhraní.

V současné době jsou hlavními představiteli výrobců PLC [7]:

- Siemens
- Rockwell Automation
- Mitsubishi Electric
- Schneider Electric
- ABB (B&R Industrial Automation je součástí divize společnosti ABB)
- HoneyWell

2.1.2 Vlastnosti PLC

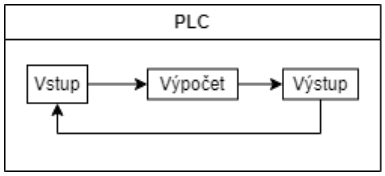
Nastavitelnost vstupů a výstupů

PLC je typické stylem čtení vstupů, případně nastavování výstupů, kdy se tyto vstupy a výstupy kontrolují, případně mění, všechny v jeden nebo více časových okamžiků. To nám zajišťuje koherentnost dat.

PLC oproti standardním mikropočítačům disponuje větším množstvím režimů manipulace se vstupy a výstupy. První forma režimu načítaní vstupů a výstupů zpracuje všechny vstupy a výstupy najednou a potom provede výpočty.

Druhá forma provede načtení vstupů, výpočty a potom nastavení výstupů.

U některých PLC lze tyto změny nastavit i na střed výpočetního cyklu.



Obrázek 2.1: Ukázka vzorkovacího cyklu PLC

Standardní výbava PLC

Výbavu PLC rozdělujeme na 2 úseky, a to na hardwarovou a softwarovou.

Hardwarová výbava se primárně zaobírá interakcemi s prostředím. Příkladem dané interakce jsou hardwarové spouště vyvolávající danou sekvenci operací. Používají se zde speciální komunikační sběrnice pro komunikaci s měniči, případně řídícími jednotkami pro krokové motory. Dále pak specializované hardwarové bloky pro odečítání síťového napětí, senzorů tepla, sériových sběrnic, CANu atd.

PLC disponují řadou knihoven a protokolů, například OMC (Open Motion Control), PID regulátory s autotuningem. V pokročilejších verzích MPC regulátory umožňující řídit a adaptovat se na komplexní dynamické systémy. PLC podporují v omezené míře komunikaci i přes internet, a to převážně za pomoci TCP a UDP protokolů. Často používanou nadstavbou těchto protokolů je používání OpcUA serveru. Součástí softwarového vybavení PLC je vizualizace uživatelského prostředí, které se v rámci současných trendů přesouvá z virtuální vizualizace VNC do prostředí webových stránek.

Provoz PLC

PLC zpracovává programy sekvenčně dle úlohové třídy. Úlohová třída definuje prioritu vykonávání programu, jeho toleranci na zpoždění a následného časování propisů dat na vstupy a výstupy z PLC.

Provoz PLC se dělí do 5 základních stavů [8]:

- Start (BOOT) - Inicializace proměnných, zavádění kódu
- Diagnostika (DIAGNOSTICS) - Základní kontroly periferií a stavu PLC
- Běh (RUN) - Standardní provoz
- Servisní režim (SERV) - Poruchový režim PLC. Může být vyvolán problémem s časováním, nebo poruchou na úrovní programu
- Zastavené (STOP) - Zastavení veškeré činnosti. V tomto stavu se nastaví všechny výstupy do bezpečného stavu

Přechody ze stavu STOP do stavu RUN jsou:

- Prvotní start (COLD Start)
- Restart (WARM Start)

2.1.3 Časování PLC

Časování úloh v PLC je velmi důležitým úsekem celého programování. Ovlivňuje totiž vytížení procesoru (CPU), zpoždění reakce na úlohy. V extrémních případech dokáže ovlivnit regulace systémů a způsobit selhání celého stroje.

2.1.4 Pohony PLC

PLC je schopné ovládat většinu pohonů. V případě nutnosti řídit pohyb pohonů přesněji, případně se vyrovnávat s další dynamikou nebo synchronizací více pohonů, vyvstává potřeba do pohybu zapojit vyšší úroveň komunikace a zpravidla i další pomocná zařízení a kontrolní jednotky. V minulosti bylo takovéto řízení pohonů velice obtížné a drahé. Přenosy dat mezi řídícími jednotkami motorů a výpočetními jednotkami musely být přenášeny optickými vlákny z důvodu zpoždění ostatních komponent a regulace. V současné době pro tyto účely vznikl standard Open Motion Control [9], který nám standardizuje základní funkční bloky pro komunikaci s pohony a pohonnými jednotkami. Při komunikaci s řídícími jednotkami pohonů je kontrolováno zpoždění polohy motorů, ztráta kontroly nad pohonem, nedosažení komunikace v řádném čase, chyby vstupů a výstupů atd.

Základní pohyby pohonů

Základní sekvence každého pohonu obsahuje přechod z deaktivovaného režimu do stavu bez pohybu. K tomu slouží funkce MC_Power. Dále je nutné pohon referencovat na počáteční bod, aby byl povolen jeho pohyb. K tomu se využívá funkce MC_Home.

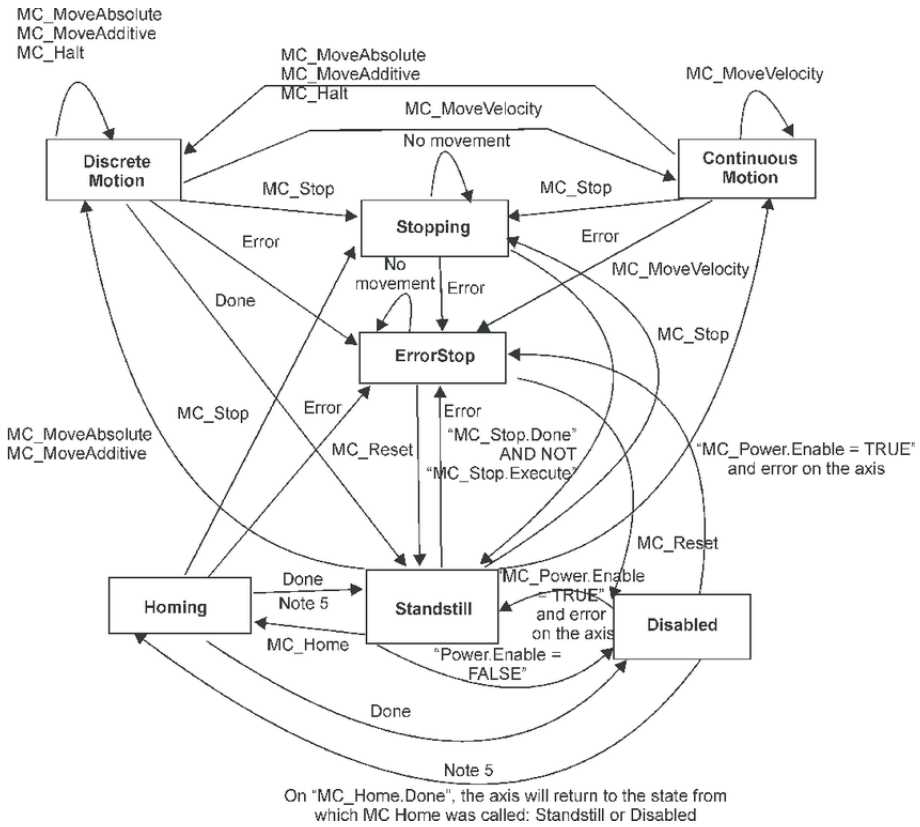
Následně je možné provádět řadu pohybů např:

- Diskrétní - Po bodech
- Spojitý - Řešený předpisem funkce
- Modulovaný za pomoci předdefinované charakteristiky

Pro zastavení pohonu existují 2 mechanismy, a to funkce MC_Halt, která rozváže všechny vazby mezi osami a ukončí pohyb, nebo funkce MC_Stop, která aktivuje nouzové zastavení os. MC_Stop jako jediná funkce zaručuje zastavení os za jakýchkoliv podmínek.

Pokročilé pohyby pohonů

Při koordinaci většího počtu os je potřeba dosáhnout synchronizace povelů ve stejný moment. Pro tyto účely implementuje Open Motion Control funkci MC_GearIn, která umožňuje vytvářet převodové vazby mezi osami za pomoci převodového čísla. Zpravidla se vytvoří vazba mezi virtuální hlavní osou a fyzickou sekundární osou. V některých případech je potřeba koordinovat pohyby os jinak než za pomoci přímého převodu. Například vyvolání série specifických diskrétních pohybů, které je potřeba vzájemně synchronizovat. Jejich současnou synchronizaci umožňuje vyrovnávací paměť každé osy, a to za pomoci jednoho centrálního programu.



Obrázek 2.2: Diagram přechodů Open Motion Control [1]

PLC a prostředí Simulink PLC Coder

Pro tvorbu a regulaci komplikovaných dynamických systémů, jako jsou třeba vačkové mechanismy ve spolupráci s pohony nebo třeba chemické reaktory, lze integrovat do PLC kód ze Simulink PLC Coder. Simulink PLC Coder [10] umožňuje automaticky generovat PLC kód za pomoci překladu simulačního prostředí do Structured Textu a Ladder diagramů. S podporou většiny v současné době používaných vývojových prostředí je velmi užitečným a cenným nástrojem pro každého vývojáře. MathWorks, společnost vyvíjející Simulink, také nabízí v rámci svého balíčku tvorbu testovacího prostředí

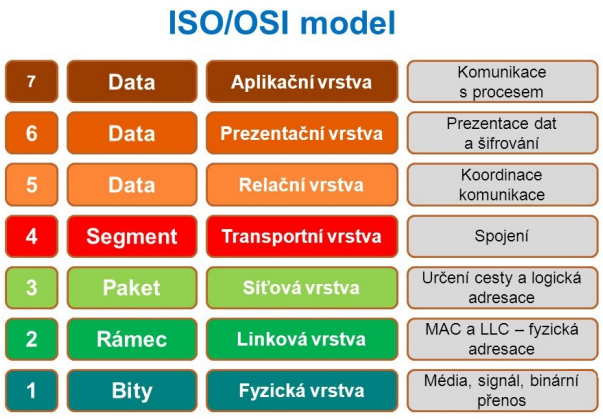
pro PLC a dokonce nástroje pro certifikaci kódu podle norem IEC 61508 a ISO 26262.

Životnost PLC

Celkovou životnost PLC definujeme podle třídy požadované bezpečnosti stroje a bezpečnosti samotného PLC. Každý modul PLC má svoji vlastní střední hodnotu selhání (MTBF), kdy prvky jako monitory a CPU mají významně nižší životnost, nežli vstupní a výstupní karty. Za předpokladu, že je dodržena norma IEC 61508-3 [11] z hlediska kvality napsaného kódu, můžeme předpokládat, že lze dodržet bezpečnost SIL2 za pomoci klasického PLC po dobu až 5 let. Doporučený interval výměny pro vykreslovací panel HMI je 5 až 7 let a 12 let pro ostatní hardwarové prvky. [12] S předpokládanou životností průmyslového stroje 20 až 30 let je takto možné odhadnout výměnu PLC a s tím spojeného hardwaru na 3x až 6x za celkovou dobu provozu stroje.

Komunikace přes Ethernet

PLC podporují řadu komunikačních rozhraní. Např.: CAN, MODBUS, I2C, UART, Ethernet. Ethernet je segment modelu komunikačního standardu ISO/OSI a řadí se do vrstvy fyzické a linkové. Společně s TCP/IP nám zajišťuje přenos komunikace přes fyzické rozhraní. Těmito rozhraními jsou moduly se specializovaným hardwarem s fyzickými porty LAN. Ethernet je pro použití v PLC nevyhovující z důvodu časové neurčitosti. To bylo impulzem pro vznik průmyslového Ethernetu, který doplňuje komunikaci klasického Ethernetu o zprávy protokolu přesného času (PTP) a dále umožňuje seřadit komunikaci dle priorit.



Obrázek 2.3: OSI Model[2]

Komunikace prováděná prostřednictvím Ethernetu je ovlivněna vzdáleností mezi zdrojovým a koncovým zařízením, vnějším rušením, vzájemnou interakcí více zařízení a nastavením sítě. V důsledku toho vznikají časové prodlevy

a výpadky při přenosu dat. Z tohoto důvodu je nutné rozhodnout, jestli bude upřednostňováno dodání dat nebo jejich aktuálnost.

Komunikace přes POWERLINK

Powerlink je speciální komunikační protokol se zvláštní hardwarovou implementací. Datový tok Powerlinku je podobný průmyslovému Ethernetu. Umožňuje navíc synchronní i asynchronní komunikaci v rámci prostředí PLC [13, 14]. Obsahuje komunikační protokol OPEN Safety [15] a splňuje bezpečnostní úroveň SIL3.

Protokol TCP a jeho použití v PLC

TCP protokol je charakteristický svým zaručeným, ale potenciálně pomalým dodáním dat. Celkové zpoždění (RTT - Round trip time) 1 zprávy TCP protokolu může přesáhnout pro velké sítě 100 ms [16]. Z tohoto důvodu je TCP nevýhodné pro časově kritické komunikace.

Protokol UDP a jeho použití v PLC

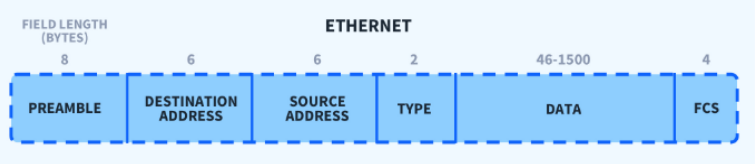
Protokol UDP se vyznačuje vysokou rychlostí. V případě poškození zprávy posílané UDP protokolem je ale daná zpráva „zahozena“. Což znamená, že daná data nikdy nedorazí. UDP protokol je primárně používán pro krátké přenosy malých datových toků. Z tohoto důvodu se využívá pro časově kritické komunikace. Celkové RTT 1 zprávy UDP protokolu se pohybuje v řádu do 20 ms [16].

Časování komunikace PLC

V rámci Ethernetové komunikace je nutné dodržet časování PLC, protože v případě, že dojde k překročení odesílacího nebo přijímacího času, tak může dojít k aktivaci hlídače času (Watch dog). Z tohoto důvodu mají UDP a TCP funkční bloky limitace na velikost přenášených dat.

Formátování Ethernetové komunikace

Ethernetová komunikace se skládá z preambule, MAC adresy zdroje, MAC adresy příjemce, identifikátoru typu, dat a kontrolního součtu. O celkové velikosti přenášených dat potom rozhoduje poskytovatel komunikace, switch nebo router, které mohou umožnit přenos až cca 9 kB na jeden packet. Překlady a přenosy dat jsou prováděny specializovaným hardwarem, z pravidla ASICem.



Obrázek 2.4: Formát Ethernetové komunikace[3]

Serializace dat

Při práci s Ethernetem je nutné provádět výměnu dat. Výměna dat je komplikovanou činností, protože je přímo závislá na vzájemné kompatibilitě komunikace obou stran. A to jak ve způsobu připojení, tak i ve formátu, v jakém jsou data přenášena. Nejrozšířenější formy serializace dat jsou:

- JSON - Jednoduchý, rychlý, univerzální, náročnější na velikost zprávy
- XML - Použití ve firemních aplikacích, podporuje všechny formáty jako JSON + další (např.: Obrázky)
- YAML - Formát preferovaný v konfiguračních souborech pro svoji „čitelnost člověkem“
- CSV - Formát dělící data středníkem. Umožňuje tvořit tabulky a v omezené míře databáze dat

Většina výrobců PLC podporuje jistou formu serializace dat. Společnost Siemens podporuje formát XML a JSON, společnost B&R Automation Industry jen formát JSON. Reprezentace těchto serializovaných dat v paměti PLC se liší implementací. Siemens používá statickou reprezentaci, zatím co B&R Automation Industry dynamickou reprezentaci.

2.2 PROGRAMOVÁ A DATOVÁ STRUKTURA PLC

2.2.1 Programovací jazyky

PLC podporují podle normy IEC61131-3 pět forem programování. Jazyky pro toto programování se dělí do dvou podskupin [17, 18]:

Textové:

- IL (Instruction List)

  \+ Paměťově efektivní, nízko úrovňový jazyk

  \- V současné době už méně používaný

- ST (Structured text)

+ Vysokoúrovňový jazyk, podporující rychlou tvorbu kódu, podobný jazyku C

- Náchylný na chyby

Grafické:

- FBD (Function Block Diagram)

  + Využívá funkčních bloků jako stavebních bloků, zlepšuje modularitu kódu

  - Vyžaduje plánování dopředu

- SFC (Sequentual Function Chart)

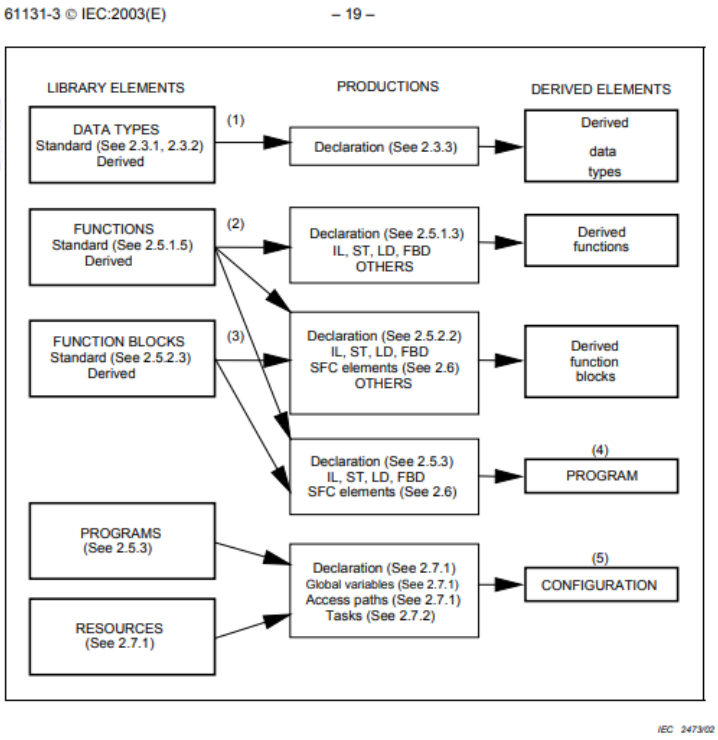
  + Diagram vhodný pro tvorbu stavových automatů

  - Nepoužitelný samostatně

- LD (Ladder diagram)

  + Vhodný pro jednoduché logiky

  - Zhoršuje modularitu a strukturování kódu



Obrázek 2.5: Diagram interakcí funkcí, programů a dat PLC [4]

2.2.2 Programové organizační jednotky

Norma IEC61131-3 [4] zavádí pojem „Programová organizační jednotka“ (POU) definující tvar programu, funkčního bloku a funkce. Každá POU obsahuje část deklarace proměnných a část postupu popsaného kódem za pomoci klíčových slov.

Deklarace proměnných:

- VAR je paměť. Může být předem definována na jednom místě. Např.: VAR %MW35 - Paměť (MEMORY) proměnné o šířce slova (WORD) na pozici 35
- VAR_ACCESS je speciální definice proměnné umožňující manipulovat s proměnnými jiných programů. Tato proměnná musí obsahovat identifikátory přístupu READ_ONLY nebo READ_WRITE. V případě kolize programů je na výrobci PLC, aby daný problém vyřešil
- VAR_INPUT kopíruje vstupní proměnné do vlastní paměťové oblasti a tím předchází kolizím v programu. Funkce nebo funkční blok nemají možnost tuto proměnnou jakkoliv měnit
- VAR_OUTPUT definuje výstupní proměnnou z vlastní paměti POU a je poskytována ostatním POU
- VAR_IN_OUT jedná se o přenos parametru referencí. To znamená, že POU netvoří vlastní paměťovou oblast, ale odkazuje se na paměťovou oblast již vytvořenou. Přístup k dané proměnné je rychlejší
- VAR_TEMP je speciální forma VAR, která je mazána mezi jednotlivými zavoláními instancí funkčních bloků
- VAR_GLOBAL jsou proměnné, které jsou přístupné ze všech POU

Deklarační úsek: Funkce (FUNCTION), musí poskytnout na konci svého procesu výstup, a to formou VAR_IN_OUT nebo VAR_OUTPUT a nesmí obsahovat žádné interní stavy. Norma IEC61131-3 definuje řadu standardních konverzních, logických a matematických funkcí.

Funkční bloky (FUNCTION_BLOCK) dovolují v segmentu VAR vazby s dalšími funkčními bloky, což jim umožňuje prohlubovat komplexitu kódu bez zbytečného zhoršování čitelnosti. Funkční bloky jsou v praxi používány jako forma „oddělovače“ přístupu ke kódu, podobně jako je ve vyšších jazycích používána forma tříd. Výhodou funkčních bloků je výborná testovatelnost. Vzhledem k tomu, že VAR je předem definovaná jedna oblast paměti, tak všechny funkční bloky stejného druhu sdílejí danou paměťovou oblast. (V ostatních jazycích se pro toto chování objevuje termín „static“). V důsledku toho VAR lze používat buď jako sdílenou paměť, prostor pro konstanty anebo odkazy na již zmíněné funkce a funkční bloky.

V případě potřeby vícečetného použití funkčního bloku je nutné vytvořit blok jako nestavový (transientní). A současně s tím definovat blok dat (parametrickou instantizaci) [19], která umožňuje mezi blokem v kontextu procesu programu přepínat.

2.2.3 Datové struktury běžného stroje

Datové struktury PLC se skládají ze dvou segmentů:

- Souboru datových typů (objekt)
- Instance daného objektu (proměnné)

Proměnné se dělí, co se týče rozsahu působnosti na globální (Global), s vybraným rozsahem působnosti (Scoped) a lokální (Local). Soubory datových typů se označují předponou, případně koncovkou „t".[4]

- Global je viditelná ve všech místech programu. (Předpona g, G)
- Scoped je viditelná ve sdílených oblastech, zpravidla složky.
- Local je viditelná jen pro samotný jeden program. (Předpona l,L)

Při tvorbě proměnných je doporučováno používat způsob formátování „camelCase" případně formátování pro zvýšení přehlednosti. Ukázka syntaxe proměnných:

- gStTest.bData - Globální struktura Test, s vnořenou Booleanovskou hodnotou data
- lFbSectiCisla - Lokální funkční blok sečti čísla
- lEnumDny - Lokální počitatelná proměnná „Enumerátor" dnů
- gKONSTANTA - globální konstanta.

Proměnné se mohou lišit z hlediska uložení na:

- Dočasné (Volatile)
- Zapamatovatelné po restartu (Remanent)
- Zapamatovatelné po přehrání (Persistent)

V rámci ideologie tvorby proměnných se objevují 2 formy pro design v Globálním segmentu.

1. Tvorba proměnných pro každý úsek odpovídající jednomu fyzickému prvku stroje, automatu atd. Tento postup je výhodný zejména u strojů, u kterých dochází ke značným změnám a obsahují menší množství fyzických prvků. Nevýhodou této metodiky je omezení komplexnosti celého stroje. Programy nerespektují design a homogenitu celé

architektury. Programy jsou nepřehledné a obtížně přenositelné na jiné stroje.

2. Tvorba proměnné, která obsahuje celou datovou strukturu stroje. Tato metodika vyžaduje práci s funkčními bloky za účelem udržení přehlednosti projektu. Datové typy v logické návaznosti kopírují architekturu stroje, což umožňuje intuitivnější práci. Tento postup je náročnější na změny, ale vynucuje modulární design, který umožňuje lepší přenositelnost, rozšiřitelnost a formu. Ve volatilní formě nám navíc proměnná stroje zajišťuje vždy definovaný start.

Stroj řízený PLC zpravidla obsahuje následující oblasti [20]:

- gMS - Globální stavový automat, který kontroluje a řídí globální stavy stroje a tím všechny ostatní podřízené automaty
- IO - Datová struktura definující všechny vstupy a výstupy
- gPRG - Prostor databloků jednotlivých programů
- gPM - Prostor managementu hesel
- gMC - Prostor manipulace s pohyblivými částmi stroje
- gFH - Prostor manipulace se soubory
- gCMD - Prostor globálního řízení

Všechny tyto objekty ukládají do centrální proměnné, kterou označujeme gPLC.

2.2.4 Doporučené styly programování

V rámci programování je nutné zajistit následující:

- Přehlednost - Hloubka vnoření proměnných by neměla přesáhnout vrstvu 4. [21] Kód by měl být dělený do funkčních bloků.
- Pokrytí - Každý kód by měl být kontrolovatelný a testovatelný v plném rozsahu. [11] Pokud testovatelnost daného kódu není možná, tak musí být uveden důvod proč.
- Bezpečnost - Jakákoliv složka stroje, která může ohrozit uživatele nebo sama sebe, musí mít přímou, nebo 100% pokrytou cestu pro zajištění bezpečnosti. Bezpečnostní činnost by měla nastat buď okamžitě v reakci na externí podnět, nebo nejdříve, jak je to možné.
- Homogenita - Stroj by měl mít homogenní základní strukturu, tzn. funkcionality a proměnné kritické k provozu a interakcím programů by se měly opakovat.

- Čitelnost - Z kódu by mělo být jednoznačně jasné, o jakou proměnnou se jedná.

2.2.5 Problematika bezpečnosti softwarového procesu

Paměťová oblast zařízení podporuje dva režimy přístupu k proměnným, a to čtení nebo čtení a zápis. Zápisové operace pro danou proměnnou může v daný moment provádět jen jeden program. Čtecí operace může provádět větší množství programů současně. Toto pravidlo se uplatňuje u všech systémů a programů. V případě jeho nedodržení dochází ke kritickým chybám. Z tohoto důvodu je vhodné před provedením změn proměnné vytvořit její kopii, na kterou v dané paměťové oblasti má přístup jen jeden daný proces. Pro sdílené přenosy dat je specificky určená globální oblast datové struktury PLC. Lokální oblast je určena pro již zmíněné kopie proměnných.

V průběhu vývoje může nastat situace, kdy je třeba přistoupit do proměnné na lokální úrovni. To znamená vytvoření přenosu z lokální vrstvy zpět do globální. Tento postup je korektní, ale zdlouhavý.

Další variantou je zažádat si přes paměťovou adresu o přístup do cizího POU a proměnou si vzít. To ale může vyvolat blokaci zdrojového programu a potenciálně způsobit nedodržení vyžadovaného času.

Třetí variantou je udržování „lokální“ kopie objektu nebo proměnné dat na globální úrovni a přistupovat do dané oblasti přes reference. Tak dojde k blokaci jen omezené oblasti paměti.

2.3 HARDWAROVÁ A SOFTWAROVÁ BEZPEČNOST

Hardwarová bezpečnost a softwarová bezpečnost je neopomenutelným prvkem každého stroje. A to jak z hlediska bezpečnosti člověka, tak i z bezpečnosti stroje samotného. Norma IEC 61508 zavádí pojem SIL [22], neboli „úroveň bezpečnostních požadavků“ a definuje úrovně bezpečnosti, které stroj může dosahovat. Norma EN ISO 13849-1 definuje pojem Performance Level (PL), který se zaměřuje na pravděpodobnost selhání úseku stroje.

Bezpečnost rozdělujeme do 3 úrovní.

- Primární - Snaha danému problému předejít, a to buď ve formě kontrolních senzorů, spínačů nebo softwarových zásahů
- Sekundární - Má za úkol zastavit a zamezit šíření dané poruchy

- Terciální - Zablokování daného stroje, pokud byla daná chyba dostatečně vážná

Bezpečnostní činnost by měla urychleně ukončit pohyb všech hmot. Odpojit dodávku elektřiny v případě, že došlo k zahoření prvku nebo selhání spínacího prvku v uzavřeném stavu.

2.3.1 Hardwarové bezpečnostní prvky a jejich zapojení

Základním prvkem bezpečnosti každého průmyslového stroje je tlačítko nouzového zastavení, které se řadí do sekundárního segmentu bezpečnosti. Tato zařízení jsou zpravidla dvoukanálová, s velmi vysokou úrovní SIL. Pro dodržení bezpečnosti je doporučováno zapojení maximálně 2 bezpečnostních tlačítek v sérii, a to ještě ve speciální konfiguraci. Běžně jsou zapojována paralelně. Nevhodné zapojení nouzových tlačítek značně snižuje úroveň PL bezpečnostního okruhu [23]. Signály z nouzových tlačítek jsou přenášeny do bezpečnostních obvodů, jako jsou bezpečnostní stykače a bezpečnostní PLC. Tyto signály jsou navíc většinou modulované, takže lze detekovat i přemostění vodiče.

2.3.2 Softwarová bezpečnost

Softwarová bezpečnost se řadí do primárního segmentu bezpečnosti. Jejím úkolem je zamezit nebo předejít vzniku kritických stavů. Kritickým stavům se předchází blokací funkcionalit, úpravou proměnných a tvorbou audiovizuálních upozornění. V případě, že dojde k selhání, je úkolem softwaru v co možná nejkratším časovém úseku reagovat a omezit pravděpodobnost vzniku škody nebo ublížení na zdraví. Mezi možné formy zásahu patří silný vizuální vjem na panelu, alarm případně sirény, fyzické zábrany omezující potenciální škody nebo zranění.

2.3.3 Testování softwarové bezpečnosti pro PLC

Pro testování programové bezpečnosti a pokrytí všech stavů jsou voleny 3 způsoby kontroly:

1. Testování samotným programátorem, který může testovat program za komplikovanějších podmínek, ale současně s rizikem chyby a možnosti, že daná chyba nebude zachycena.
2. Tvorba testů za pomoci automatů. Příkladem takového programu je Test Suite od společnosti Siemens nebo Unit Test od společnosti B&R Automation Industry. Tyto automaty umožňují testovat pomocí programátorem předem definovaných sekvencí.
3. Plně syntetické testy. Tyto testy hledají všechny variace, co mohou nastat a vyhodnocují testy podle předem nastavených parametrů.

V současné době jsou tyto testy předmětem vývoje. [24, 25]

2.4 B&R Industrial Automation prostředí a PLC

2.4.1 Automation Studio

Automation Studio je vývojové prostředí pro PLC od společnost B&R Industrial Automation. Umožňuje vývoj softwaru v jazycích dle normy IEC61131-3. K tomu přidává možnost vyvíjet v jazyce C++, Automation Basic a C.

Automation Basic je jazyk vyvíjený B&R Industrial Automation jako alternativa k Structured textu.

Automation Studio poskytuje integraci MappTechnology, systém debuggovacích nástrojů, unit testů a záznamů.

2.4.2 MappTechnology

Mapp technologie je soubor funkcionalit umožňující vyvíjet tzv.: Mapp komponenty. Tyto komponenty mají za úkol zjednodušit vývoj softwaru stroje. Mezi tyto technologie se řadí MappServices, MappView, MappMotion, MappCocpit, MappControl a MappSafety.

2.4.3 MappService

MappService je technologie společnosti B&R Industrial Automation umožňující integraci funkcionalit alarmů, systému receptů, systémových nastavení, komunikace s databází, záznamem informací a formy vykreslování za pomoci webů. Významnou výhodou této služby je provázanost daných funkcionalit a jejich jednoduchý propis do MappView.

AlarmX

AlarmX je MappService funkce zaměřena na sběr a reakce na poruchové stavy. Její výhodou, oproti klasické tvorbě chybových seznamů, je možnost konfigurace typu chyby (trvalá / krátkodobá). Umožňuje nastavit sekvenci ochranných činností a záznamu poruch. AlarmX má dále integraci do MappView ve formě modulů.

2.4.4 MappView

MappView je služba umožňující vývojáři vytvářet interaktivní webové prostředí. Zprostředkovává sadu základních stavebních bloků pro sestavení

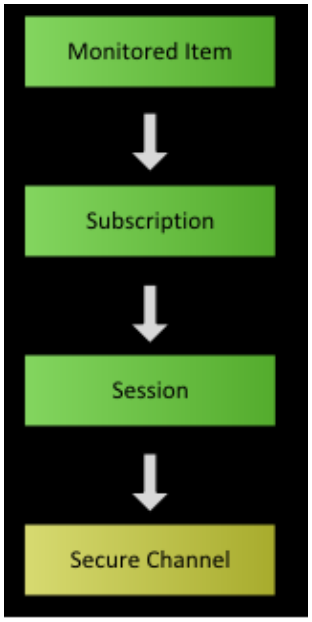
designově „pěkné“ stránky. Přičemž je schopná podporovat interakce více uživatelů najedou. Umožňuje vytvářet vlastní scripty a modifikace. V současné době je cílem B&R Industrial Automation touto službou nahradit zastaralý styl vykreslování pomocí VC4. MapView spolupracuje s OpcUA serverem, který je používaný jako zdroj proměnných.

OpcUA

OpcUA je standard definující průmyslovou implementaci komunikace [26] s průmyslovým zařízením přes internet a definuje implementace databáze. Vyznačuje se vysokou bezpečností, rozšiřitelností, možností používat většinu komunikačních protokolů. Reaguje na události a zároveň je vyvolává, což má významný vliv na výkonnost PLC. OpcUA můžeme rozdělit do dvou základních segmentů, a to segmentu serveru a klienta. Server poskytuje klientovi proměnné přes takzvané „nody“. Tyto nody definují datový typ, data a název proměnné.

Na základně žádosti klienta server poskytne přístup k daným proměnným. V průběhu žádosti dochází k výměně autorizačních klíčů, testování spojení a poté k vytvoření bezpečného kanálu pro komunikaci.

Součástí OpcUA implementace od B&R Industrial Automation je služba AccessAndSecurity, která zajišťuje přístupy uživatelů ke stroji, řídí nastavení firewallu a nastavuje práva pro zápisy do proměnných.



Obrázek 2.6: Diagram přechodů odběru dat OpcUA [5]

2.4.5 VC4

VC4 je „stará“ forma vykreslování pro PLC společnosti B&R Industrial Automation. Oproti MappView je architektonicky odlišná používáním systému vykreslování po vrstvách. Je významně jednodušší, a to jak z hlediska funkcionalit, tak i vývoje. Modifikace VC4 lze dosáhnout v krátkém časovém úseku, což je využíváno zejména v prototypovém vývoji.

MappMotion

MappMotion je službou řízení pohonů od společnosti B&R Automation Industry. Umožňuje implementovat rozsáhlejší kontrolní struktury pohonů s integrací simulací a bezpečnostních prvků. Lze implementovat osy robotů, CNC a dalších komplexních struktur. Všechna data z pohonů lze integrovat do vizuálního prostředí MappView pomocí widgetů, případně je kontrolovat za pomoci diagnostického rozhraní [27].

MappCockpit

Služba MappCockpit zajišťuje rozhraní pro kontrolu, konfiguraci a sběr dat stroje. Úzce spolupracuje s MappView a tvoří tak uživatelsky jednoduše použitelné prostředí.

MappControl

Služba MappControl umožňuje kontrolu většiny dynamických systémů. Disponuje funkcemi od modifikovaných PID regulátorů až po Kalmanovy filtry a MCP regulátory. Dále obsahuje předpřipravené balíčky pro řešení kontroly teploty, procesů tvorby plastů a řízení hydrauliky. MappControl může spolupracovat s MapleSim nebo IndustrialPhysics pro tvorbu virtuálních dvojčat [28].

MappSafety

Služba MappSafety je součástí řady Safety společnosti B&R Automation [29]. Prvky bezpečnosti (SafetyNODE) zajišťují bezpečnost řídících jednotek, vstupů a výstupů, pohonů a dalších bezpečnostních jednotek. Safety prvky jsou vybaveny hardwarovými spínači s možností nezávisle na PLC provádět nouzové akce. S řadou Safety je v B&R Automation Studiu integrovaný tzv.: Safe DESIGNER [30], který implementuje speciální vývojové prostředí pro tvorbu bezpečnostních logik. Bezpečnostní proměnné lze vyměňovat s dalšími bezpečnostními prvky za pomoci Safety Gateway, která komunikuje přes POWERLINK.

2.4.6 Interní komunikace PLC

PLC od B&R Industrial Automation používají jako základní komunikační rozhraní X2X Link [31]. Toto rozhraní je velmi odolné proti okolnímu rušení, má datovou propustnost 8 kB a použitelnou délku rozvodů až 100 m. X2X se používá při komunikaci mezi jednotlivými moduly. Při použití vysílačů a přijímačů X2X lze rozšířit celou soustavu PLC až na 255 modulů. Toto rozhraní lze použít pro komunikaci s měniči a dalšími periferiemi od společnosti B&R Industrial Automation. V PLC od B&R Industrial Automation se dále používá POWERLINK, který se primárně objevuje v páteřních oblastech PLC. Například ve spojení CPU a BUSu. Mezi další základní periferie PLC od B&R Industrial Automation patří USB, které umožňuje ukládání souborů na externí disky nebo se užívá jako vstup pro licenční klíč.

2.5 ELEKTROSTATICKÉ ZVLÁKŇOVÁNÍ

2.5.1 Úvod

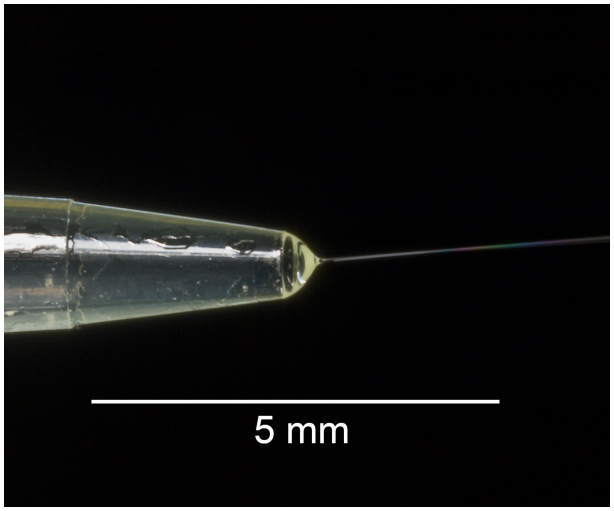
Vlákno definujeme jako útvar, jehož délka významně přesahuje jeho průměr. Vlákna z hlediska původu dělíme na chemická a přírodní. Vlákna chemického původu jsou uměle vyrobené polymery, například polyester, polyamid a polycotton. Přírodní vlákna se dělí podle původu na živočišná a rostlinná. Mězi živočišná patří vlna a hedvábí. Mezi rostlinná patří například bavlna, len, kokos a bambus. Vlákna se dále dělí na dvě základní skupiny, a to monofilamenty, které se skládají pouze z jednoho materiálu a na multifilamenty, které se skládají z vícero materiálů.

Příze je dloužený útvar skládající se z vláken. Příze rozdělujeme na lineární, která je obvykle opatřena zákrutem a navíjena na cívky nebo plošnou, která se objevuje v netkanných textíliích.

2.5.2 Nanovlákna

Nanovlákna se vyznačují velkým měrným povrchem a malou délkou. Příze z nanovláken drží pohromadě i bez použití zákrutu a dosahuje vysoké pevnosti a tažnosti. Nanovlákenné materiály jsou využívány v oblasti medicíny, elektroniky, v chemickém průmyslu a kosmetice [32]. Jsou z nich vyráběny filtry, nosiče tekutin, buněk, textilní pleťové masky, krycí materiály na popáleniny. Základem výroby nanovláken je destabilizace roztoku polymeru a oddělení rozpouštědla. Existuje řada způsobů tvorby nanovláken. Například odstředivé zvlákňování, fázová separace nebo tlakové zvlákňování. Mezi způsoby tvorby nanovláken patří také elektrostatické zvlákňování, u kterého se využívá k destabilizaci povrchu roztoku polymeru vysoké napětí [33]. Vysoké napětí vyvolává rozdíl mezi povrchovým napětím kapaliny, přesněji řečeno hydrostatickým tlakem

a tlakem vyvolaným elektrickou silou. Tak vzniká kapalinná vlna. Vlny se zužují a prodlužují do tvaru trysky. Zužování dané trysky umocňuje působení elektrického pole a zapříčiňuje vznik extrémně úzkého kapalinného kužele (Taylorova kužele). Při dostatečném zúžení dojde vlivem externích sil a tepla k oddělení polymeru od rozpouštědla. Tento děj poprvé popsal britský matematik Geoffrey Ingram Taylor v roce 1969.



Obrázek 2.7: Ukázka Taylorova kužele [6]

2.5.3 Historie technologie

První myšlenky elektrostatického zvlákňování se začaly objevovat na počátcích 20. století. Počáteční pokusy se uskutečňovaly za pomoci dutých jehel. Stroje před dvaceti lety využívaly pro výrobu nanovláken velkých skupin jehel. Výroba touto technologií byla velmi náročná, a to hlavně z hlediska údržby. Později se objevil nápad využívající rotujícího disku s ostrou hranou, na kterém se tvořilo větší množství kapalinných trysek najednou. Oproti výrobě prostřednictvím jehel se produkce nanovláken zvýšila a zjednodušila se údržba strojů.

Technologie NANOSPIDER™ 2005 vyvinutá TUL se zase ubírala směrem tvorby Taylorových kuželů na rotujícím válci. Technická univerzita v roce 2011 patentovala technologii přeplavovacího zvlákňování, kde se 2 polymery přelévaly přes vodivou hranu.

Současný nejnovější výzkum v oblasti výroby nanovláken se zaměřuje na postupy tvorby lineární příze z čistých nanovláken. Je snaha o transformaci plošné příze nanovláken do příze lineární za pomoci zákrutu.

Fyzikální základ

Roztoky polymerů jsou nemagnetické, proto se na ně převážně uplatňuje jen Coulombovská složka Lorentzovy síly.

Lorentzova síla:

$$F = q(E + v \times B)$$

Coulombova síla:

$$F = E \cdot q$$

F - Lorentzova síla $[N]$
E - Intenzita elektrického pole $[V \cdot m^{-1}]$
q - Elektrický náboj $[C]$
v - Pohyblivost náboje $[m \cdot s^{-1}]$
B - Intenzita magnetického pole $[T]$

Díky tomu, že nanovlákna jsou extrémně lehká je dokáže unášet elektrický vítr.

$$v_d = k \cdot \frac{U}{d}$$

v_d - Elektrický vítr $[m \cdot s^{-1}]$
k - Pohyblivost iontů $[m^2 \cdot V^{-1} \cdot s^{-1}]$
U - Elektrické napětí $[V]$
d - Vzdálenost elektrod $[m]$

Zvlákňování střídavým elektrickým polem

Střídavé elektrické pole je oproti stejnosměrnému asi 10x efektivnější [34]. Není nutné dosahovat tak vysokých napětí a není potřeba aktivní druhá elektroda. Ke zvlákňování dochází v kladné půlvlně a je snaha maximalizovat efektivní napětí. Sběr vzniklých nanovláken je potom zajišťován tzv. kolektory.

Kolektory

Kolektory nanovláken můžeme rozdělit do dvou úseků. Na statické, kde se aplikuje jen vliv elektrického pole a pohyblivé, kde se aplikuje jak elektrické pole, tak rotační pohyb, za pomoci kterého dochází ke směrování vláken.

Základní dělení kolektorů:

- Statický plošný
- Statický prostorový
- Pohyblivý diskový kolektor s rameny
- Pohyblivý válec
- Pohyblivý strunný kolektor (kde se střídaly směrově orientované a neorientované vrstvy)

2.6 KOPRIS1

KOPRIS1 je experimentální linka určená pro výrobu nanovláken. Vyznačuje se vysokou a často využívanou modularitou, lze za ní vytvářet plošnou i lineární přízi. KOPRIS1 dokáže vyrábět 2 druhy lineární příze. Z multifilamentu, za pomoci navinu nanovláken na nosné jádro nebo monofilamentu, kdy lze tvořit a zakrucovat nanovlákna samostatně. Dále je zde možno vytvářet plošnou netkanou přízi za pomoci pohyblivého bubnového kolektoru.

Obrázek 2.8: Ukázka stroje KOPRIS1

2.6.1 Mechanické prvky linky KOPRIS1

- Náviny - KOPRIS1 disponuje 2 způsoby návinu. Jeden v prostoru zvlákňovací komory v podobě bubnu a jeden vně komory v prostoru za sušícími trubicemi.
- Galety - Na stroji jsou umístěny galety pro účely dodržení správného předpětí příze.
- Pumpy - Stroj je osazen 5 pumpami. Tři jsou šnekové s magnetickou vazbou a dvě peristaltické.
- Zákruty - Zákruty mají dosažitelnou rychlost otáčení až 20000 otáček za minutu. Zákruty jsou výrobkem TUL [34].
- Vysoušecí potrubí - KOPRIS1 obsahuje 2 vysoušecí potrubí. Vysoušecí potrubí má celkový tepelný výkon 1000 W s vnitřním průměrem 30

mm a funkční délkou cca 2 metry. Potrubí bylo zakázkově vyrobené společností ELKOP Technik s.r.o [34].

- Dmychadlo - Ve spolupráci s vysoušecím potrubím je zapojené dmychadlo Mistral 6 od společnosti Leister Technologies AG [34].
- Zdroj vysokého napětí - KOPRIS1 má k dispozici 2 formy vstupů vysokého napětí. Interní napětí je poskytováno zdrojem KGUG 36 36000/230V [34], tento zdroj je měřící transformátor zapojený ve zvyšovacím režimu s regulací za pomoci autotransformátoru. Maximální dosažitelné efektivní napětí tohoto zdroje je 32 kV. Externím zdrojem vysokého napětí může být libovolný generátor.

2.6.2 Základní řešení bezpečnosti linky KOPRIS1

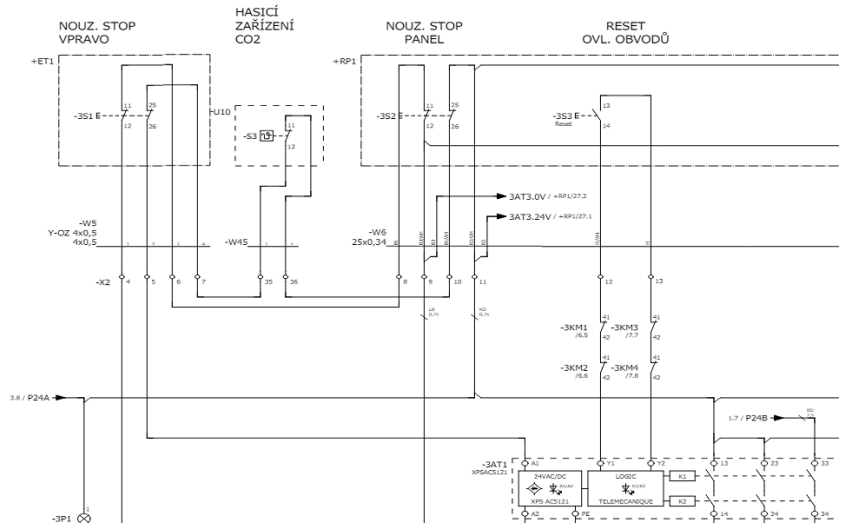
Z povahy experimentálního charakteru linky KOPRIS1 vyplývá, že často dochází k jejím úpravám a technickým modifikacím. Proto musí existovat nástroje pro obcházení některých bezpečnostních obvodů a zároveň musí být zajištěna základní bezpečnost zařízení.

Dveře

Systém dveří má jednokanálové zapojení snímače polohy se zámkovým mechanismem. Pro účely údržby linky je možno systém zabezpečení dveří obejít a spustit tak stroj v servisním režimu.

Nouzové tlačítko zastavení

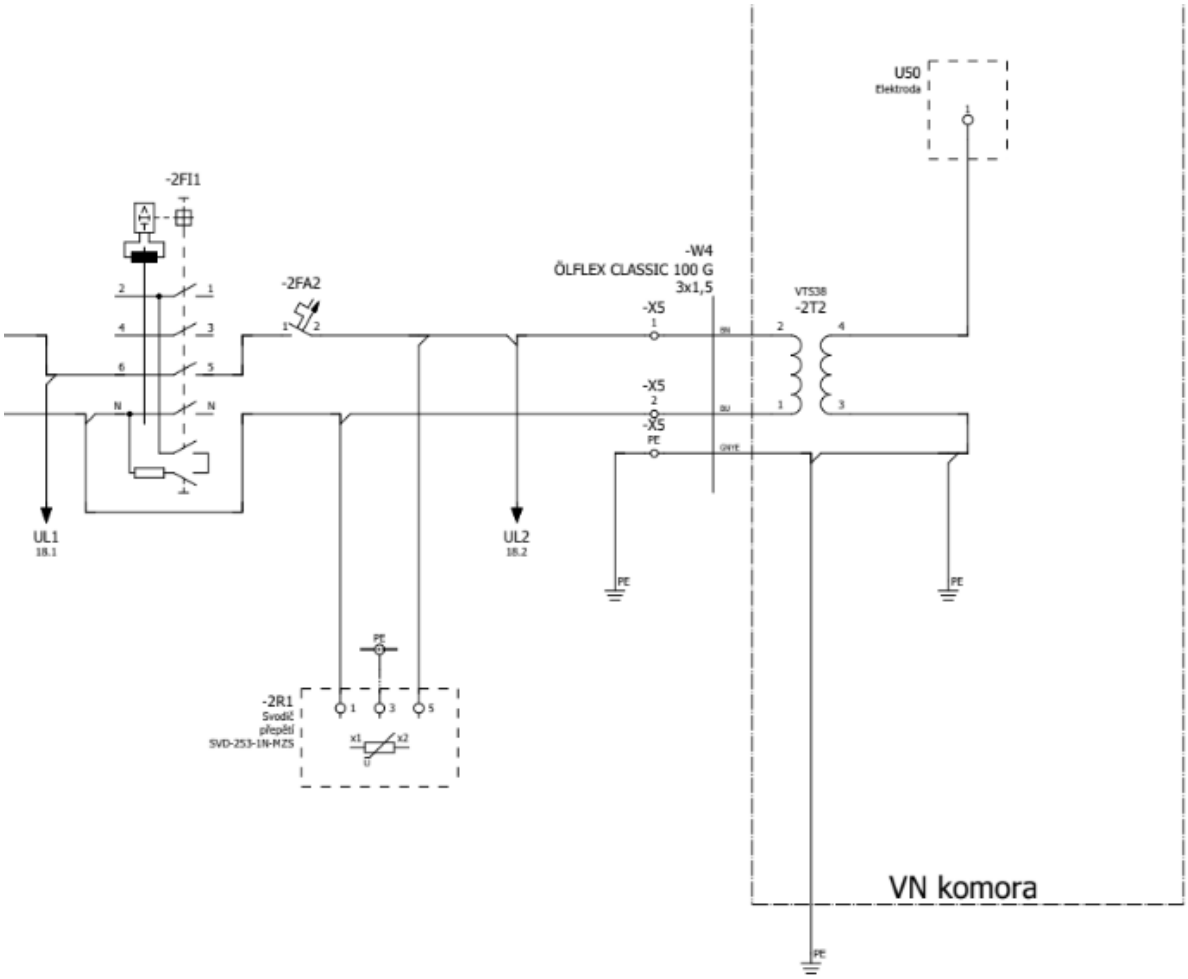
KOPRIS1 je vybaven dvěma 2 dvoukanálovými tlačítky nouzového zastavení zapojenými v sérii. Nouzový stav se přenáší do bezpečnostního PLC XPSAC5121, které je nezávislé vůči řídícímu PLC.



Obrázek 2.9: Výňatek z dokumentace nouzového okruhu

Řešení bezpečnosti prostoru s vysokým napětím

Vysoké napětí je kontrolované dvěma mechanismy: Za pomoci modulu X20AP3121 kontrolujícího síťové napětí a napětí primárního okruhu transformátoru. Sekundární okruh se kontroluje převodovým poměrem transformátoru. Dále je v okruhu s vysokým napětím proudový chránič v atypickém zapojení jednoho vodiče, kde se kontroluje velikost procházejícího proudu. Na úrovni komory jsou umístěny svodiče vysokého napětí a pro případ vznícení roztoku je na stroji umístěn hasící systém s CO2 a zásobníky s práškovou náplní.



Obrázek 2.10: Výňatek z dokumentace obvodu VN

PRAKTICKÁ ČÁST

3.1 POŽADAVKY NA NOVÝ SOFTWAROVÝ PROJEKT PRO LINKU KOPRIS1

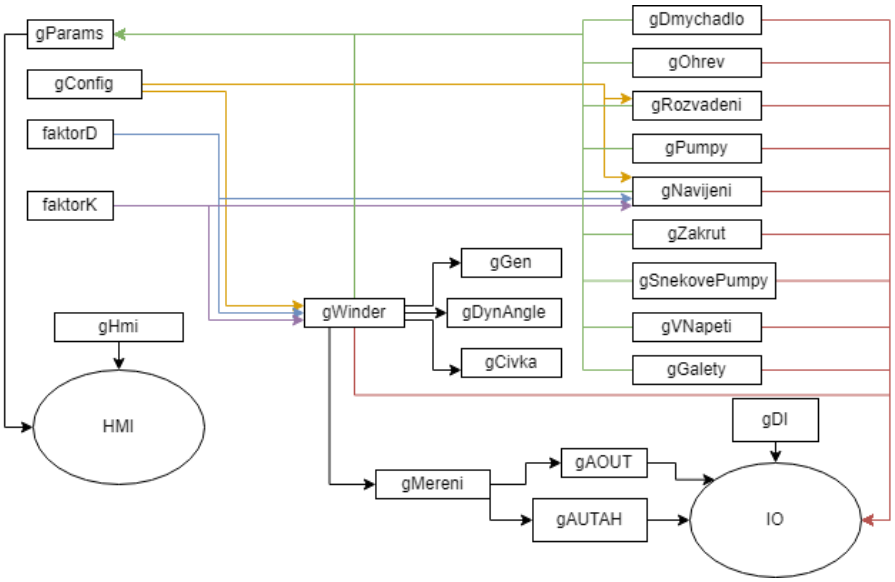
1. Analýza využití vstupů a výstupů stroje
2. Možnost přepínat hardwarovou konfigurací stroje z pozice uživatele
3. Upravení projektu do formy, která je použitelná na dalších strojích
4. Vytvoření serveru pro komunikaci s dalšími zařízeními
5. Vytvoření možnosti tvorby receptů za pomoci MappService Recipe
6. Modifikace programu hlavního návinu (Mainwinder) pro navíjení v referenci na jinou osu

Z této skupiny byly vybrány úlohy 1 až 4 jako doplňkové požadavky od uživatelů experimentální linky KOPRIS1 a staly se tak doplňujícími cíli této diplomové práce.

V následujícím textu se budou objevovat **tučně** zvýrazněná slova. Tyto slova jsou termíny s vazbou na softwarový projekt KOPRIS1.

3.2 ANALÝZA PŮVODNÍHO SOFTWAROVÉHO PROJEKTU

3.2.1 Původní architektura objektů stroje



Obrázek 3.11: Zjednodušený diagram proměnných starého projektu

Z výše zobrazené architektury původního softwaru je zřejmé, že veškeré **HMI** interakce procházejí permanentní proměnnou **gParams**. S proměnnou **gParams** byly provázané veřejné proměnné odpovídající interním stavům a konfiguraci každého programu, který manipuloval s hardwarem.

Veřejné proměnné programů byly propisovány přímo do vstupů a výstupů stroje. V důsledku toho bylo **IO** decentralizované. Toto řešení bylo pro původní požadavky na softwarové vybavení stroje vyhovující, viz. ideologie tvorby proměnných.

Pro účely rekonfigurace linky bylo nutné zavést řadu dalších programů. Původní struktura pro tyto nové programy byla nevyhovující.

3.2.2 Nedostatky původní struktury projektu

Systém datových typů nebyl rozdělen na separátní oblasti odpovídající programům. Architektura stroje tak byla značně nepřehledná.

Programová data proměnné **gParams** obsahující většinu parametrů stroje, byla inicializována z trvalé paměti. Mohlo tak dojít ke startu stroje v nedefinovaném stavu.

Datová struktura obsahovala skupinu proměnných bez inicializace. Například: **faktorD** a **faktorK**. Tyto proměnné musely být zpětně získány z trvalé paměti linky KOPRIS1.

IO proměnné byly nejednoznačné, často matoucí a decentralizované.

Absence knihoven - C307 [21]

Nepoužívání vlastních funkčních bloků - C307 [21]

„Magická“ čísla - C314 [21]

3.2.3 Inicializační sekvence stroje

Inicializační sekvence linky byla zajištěna programem **MAIN**. Program **MAIN** reagoval na nouzové tlačítko formou přímé manipulace potvrzení napájení periferií. Dále v rozporu s principem odpovědnosti za 1 úsek [35] manipuloval s parametry automatu **gAuto**. Parametry automatu **gAuto** řídily sekvenční spuštění os návinu, rozvádění, spouštění vysokého napětí a zákrutů.

3.2.4 Záznam poruch

Sběr reportu chyb byl vyřešen programem **ERROR**, který je zaznamenával do pole Booleanů. Oznámení byla statická a často neodpovídala reálnému stavu poruchy. U poruch se nerozeznávala jejich závažnost, což bezdůvodně vyvolávalo nouzový stav celého stroje. Dále program **ERROR** obsahoval kontrolu přetržení vlákna při navíjení. Čímž byl opět porušen princip odpovědnosti za 1 úsek [35].

3.2.5 Vizualizace za pomoci technologie VC4

Vizualizace **VC4** obsahovala značné nedostatky v oblasti limitace parametrů a vstupů. Uživatel byl nucen znát interní strukturu programů. Bez této znalosti mohlo dojít ke kolizi automatů a k překročení limitních hodnot. **VC4** se vícenásobně odkazoval na stejné parametry. Zejména mezi parametry programů **Navíjení** a hlavního návinu (**MainWinder**), což mohlo potenciálně ohrozit provoz stroje.

3.2.6 Nebezpečné chyby

Simulací PLC byla zachycena chyba zapříčiňující osám druhu **MD_cam_typ** totální a neresetovatelnou blokaci při použití funkce **MC_STOP**. Tento problém byl částečně vyřešen aplikací **MC_HALT** i v případě, kdy mělo dojít k nucenému zastavení. Původní implementace automatu **MainWinder** obsahovala mechanismus pro autonomní reset poruch os bez upozornění uživatele.

3.2.7 Závěr analýzy

Na základě provedené analýzy lze dospět k závěru, že původní softwarové vybavení linky KOPRIS1 odpovídalo experimentálnímu provozu.

Rozborem původního softwarového vybavení byly identifikovány tyto nedostatky: nepřehlednost, obsah duplicitního kódu, potencionálně nebezpečné části kódu, které mohly způsobit kolizi automatů nebo selhání linky KOPRIS1.

Z hlediska záchytu poruch byla linka KOPRIS1 dostatečně pokryta, dané chyby ale neodpovídaly skutečným stavům. Koncové stavové automaty byly až na případ **MainWinderu** ve vyhovujícím stavu. Použité automaty byly typu Moore. Z hlediska dlouhodobé udržitelnosti není původní softwarový projekt vyhovující, a to z důvodu nedostatku softwarové modularity, nepoužívání funkcí a funkčních bloků. Při analýze využitelnosti **IO** vstupů bylo shledáno, že lze na stroji využít 1 analogový vstup, 8 digitálních stupů typu sink a 9 výstupů typu source. Prvky manipulující s pohony byly ignorovány. Celkové procento využití **IO** stroje bylo 63%.

3.2.8 Příprava na změny

V rámci přípravy na změny v projektu byl v první fázi vypracován návrh úprav architektury. Úkolem těchto úprav architektury bylo významným způsobem zlepšit modulárnost stroje.

Ve druhé fázi byl vytvořen testovací projekt pro otestování postupu tvorby nového vizuálního prostředí za pomoci Mapp technologie **MappView**. Bylo zjištěno, že je nutné centralizovat parametry vizualizace do uzavřené oblasti. Optimálně jedné proměnné nebo typu, za účelem minimalizace náročnosti budoucích změn a možnosti aplikovat rekurzivní používání práv přístupu do proměnných.

Ve třetí fázi došlo k vytvoření konceptu manažera funkčních bloků (**FBManager**). Zde bylo ověřeno použití parametrické instantizace funkčních bloků. To položilo základy metodiky pro tvorbu většiny dalších funkčních bloků.

3.3 UPGRADE CPU

Za účelem vývoje nové vizuální formy za pomoci technologie**MappView** bylo nutné změnit typ displeje z 4PP065.0571 do 4PPC70.0702-20W. Tato změna způsobila odstranění podpory většiny doposud používaných fyzických tlačítek. Následkem změny již zmíněných displejů došlo k narušení vazeb **ACP manageru** se sběrnicí Powerlink. Toto narušení vazeb se projevilo nefunkčností pohonů. Tento problém byl v průběhu testování opraven.

3.4 DESIGN NOVÝCH DATOVÝCH STRUKTUR A PROGRAMŮ

Původní architektura decentralizovaných proměnných pro jednotlivé hardwarové prvky odpovídala jednodušší logice stroje.

Z důvodu požadavku změny hardwaru na softwarové úrovni při zajištění bezpečnosti, bylo nutné pozměnit architekturu tak, aby umožňovala aktivaci a deaktivaci programů, aktivaci a deaktivaci jejich bezpečnostních smyček a případně blokaci pohybu.

V reakci na předchozí požadavek bylo nutné vytvořit novou formu hlavního stavového automatu, zajišťujícího bezpečnost stroje i v podmínkách deaktivace všech pomocných bezpečnostních smyček.

Z důvodu přehlednosti bylo potřeba vytvořit novou strukturu zajišťující sběr stavů všech programů a zprostředkovávající daná data hlavnímu stavovému automatu. Dále vznikla potřeba redesignu vykreslování, které doposud neumožňovalo každý prvek hardwaru vizuálně označit za blokovaný.

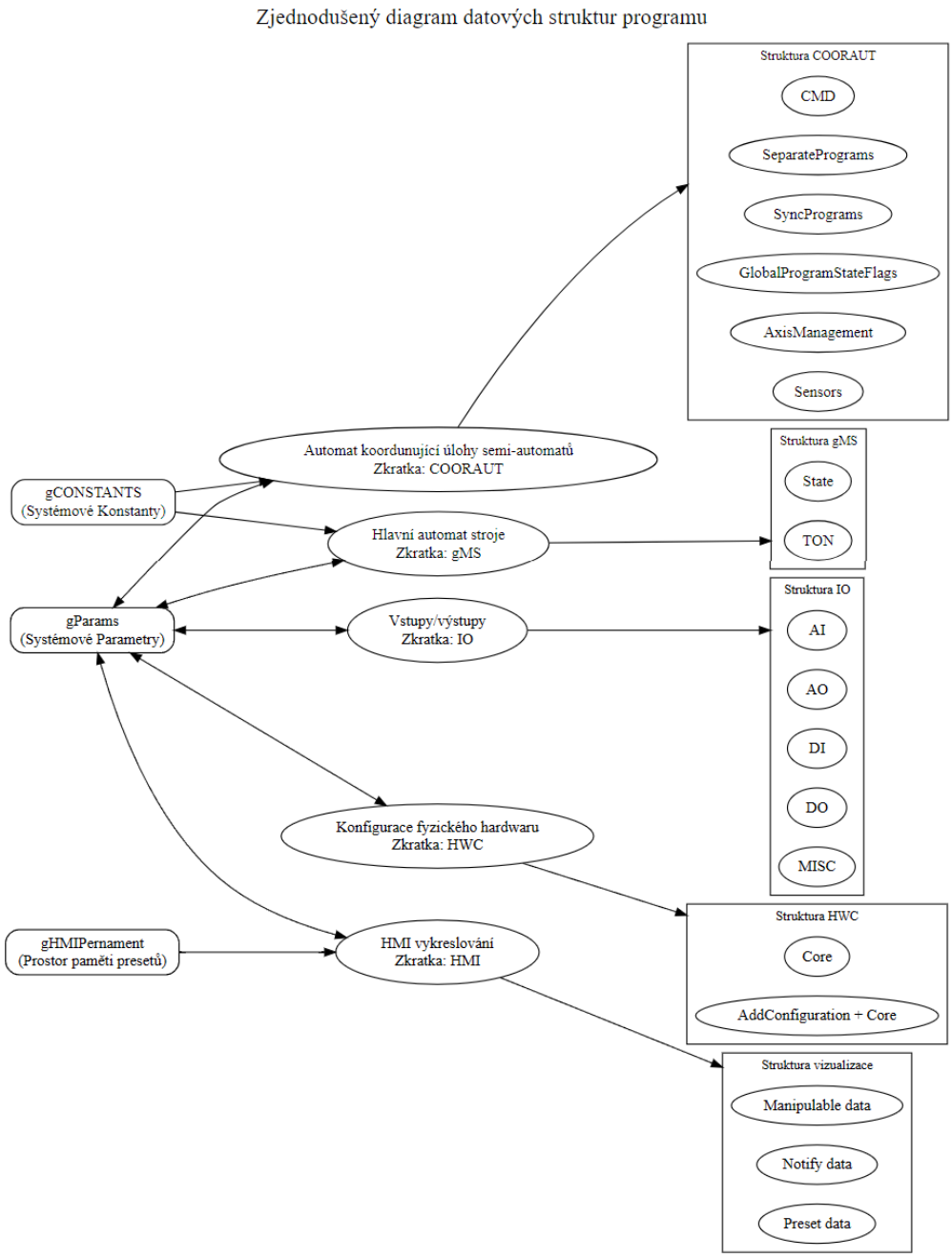
Centrální struktura linky měla být navržena tak, aby byla jednoduše rozšířitelná. Byl kladen důraz na přenositelnost datové a programové struktury na jiné stroje.

3.4.1 Vrcholové proměnné nové struktury

Globální proměnná **gParams** byla zbavena persistentní vlastnosti, tím bylo dosaženo stejného počátečního stavu startu stroje. Proměnná **gParams** navíc nově reprezentuje „instanci linky“, tzn., že v případě potřeby je možné za pomoci 1 PLC řídit více linek.

Dále byla navržena nová struktura reprezentující globální konstanty stroje **gCONSTANTS**, s primárním využitím pro hlavní stavový automat a indexy vykreslovacího prostředí.

gHMIParameters vznikla za účelem nahrazení předchozí paměti stroje a stala se jistou formou profilů pro daný stroj. Výsledkem tvorby této struktury je možnost přepínat mezi jednotlivými profily / presety stroje a tím ulehčit přípravu stroje. Potencionální budoucí využití této proměnné by mohlo spočívat v tvorbě komplexních sekvenčních automatů a tvorby receptů.



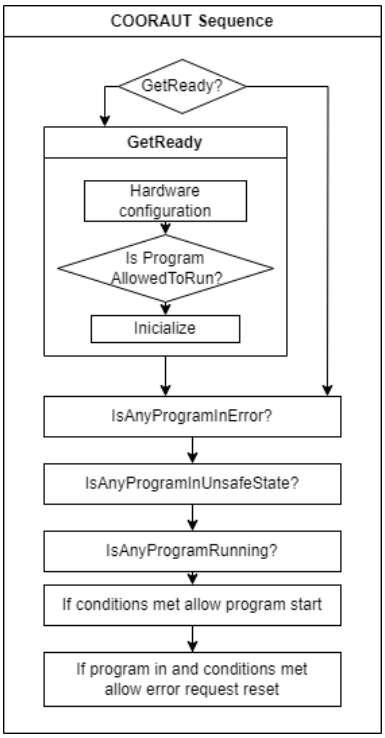
Obrázek 3.12: Diagram nové datové struktury stroje

Koordinační automat - COORAUT

Program **COORAUT** se stal sekcí inicializace, konfigurace, řízení a sběru dat programů.

Účelem tohoto programu a struktury bylo vytvořit specializovaný prostor, který manipuluje s jednotlivými programy. Registruje jejich stavy a zasílá je globálnímu automatu pro další analýzu. Z globálního stavového automatu

přijímá rozhodnutí o připravenosti celého stroje a dané rozhodnutí zprostředkovává ostatním programům. V případě poruchy dále rozhoduje o možnosti návratu z chybového stavu programu. Tento program vytvořil pomyslnou „bariéru“ mezi programy a uživatelem / automaty. Tato „bariéra“ nově znemožňuje startovat nepřipravené programy nebo nedostupný hardware. Vedlejším efektem této „bariéry“ byla jistá forma adaptace automatů na základně dostupného hardwaru.



Obrázek 3.13: Diagram sekvence COORAUTU

Hlavní automat - GMS

Program **MAIN** byl přejmenován na **gMS**. Funkcionalita sběru stavu programů byla nahrazena proměnnými z **COORAUTU**.

Přímá manipulace s bezpečnostními proměnnými byla odstraněna a nahrazena funkcí blokace stavu „RUN“ ve všech programech.

GMS s použitím **gCONSTANTS** umožnil každému automatu sledovat globální stav stroje. Stavy globálního automatu byly rozšířeny o vynucení zavření dveří při hardwarové konfiguraci. Účelem vzniku tohoto stavu bylo zajištění bezpečnosti při nefungujících pomocných bezpečnostních smyčkách programů.

3.4.2 Vstupy a výstupy - IO

IO bylo přepracováno a centralizováno do jedné struktury. Výsledkem této činnosti bylo zjednodušení změny výstupních vazeb stroje. Datová struktura byla rozdělena do 5 základních struktur:

- **AI** - Analogové vstupy
- **AO** - Analogové výstupy
- **DI** - Digitální vstupy
- **DO** - Digitální výstupy
- **MISC** - Ostatní IO, např.: pulzní šířková modulace

3.4.3 Hardwarová konfigurace - HWC

Datová struktura **HWC** vznikla za účelem jednoznačného definování hardwarové konfigurace. Každá možná konfigurace je zde reprezentována Booleanovskou proměnnou, která zajišťuje jednoznačnou konfiguraci hardwaru a režimů stroje. **HWC** se skládá ze dvou segmentů, a to jádra (**Core**), které obsahuje identifikátory a nastavení hardwaru a **AddConfiguration** zaštiťující případné další stavy a režimy.

3.4.4 Vykreslování - HMI

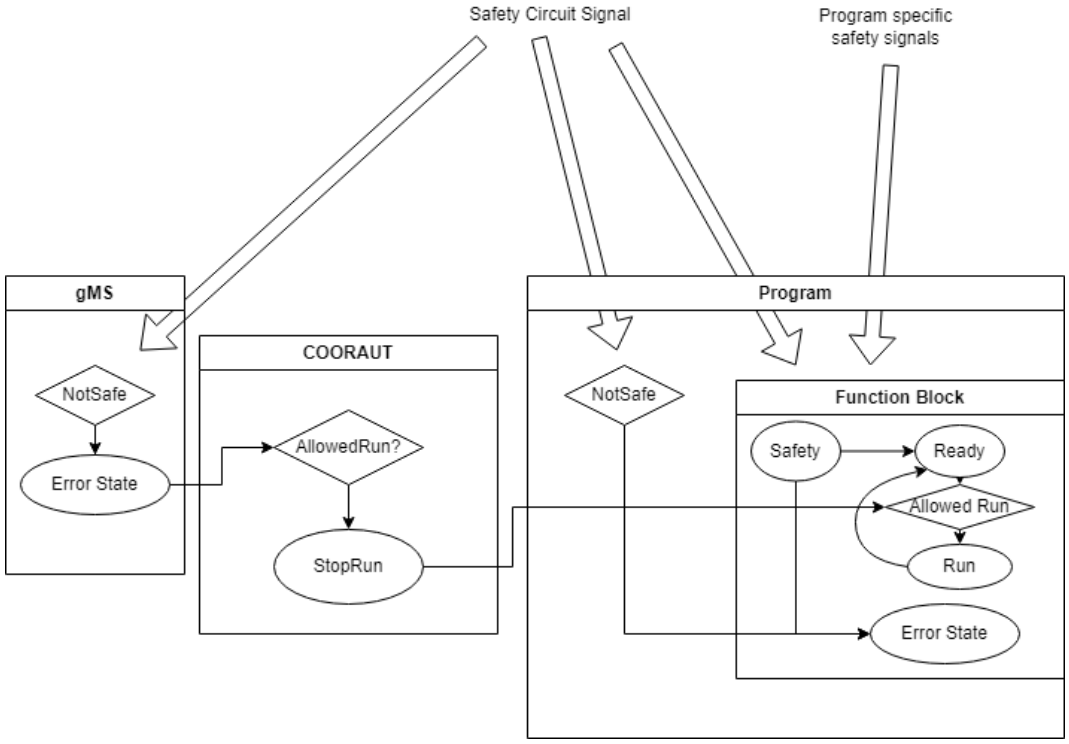
Struktura **HMI** byla vytvořena za účelem centralizace vstupních parametrů z vykreslování. Pro strukturu **HMI** bylo nutné vyhledat všechny interakce programů s vykreslovacím prostředím **VC4**, což bylo časově značně náročné. Celý design názvů struktury **HMI** vycházel z původní idexace stránek **VC4**, která byla ale rozšířena o popis unikátní vlastnosti dané stránky.

Jednotlivé stránky byly umístěny do prostoru „manipulovatelných proměnných“ (**ManipulableData**). Dále byla vytvořena substruktura oznámení pro uživatele (**Notify data**) a seznam profilů (**Preset data**). Při transformaci dat byly odstraněny nedostatky jako například: nepoužívané proměnné, vícenásobné vazby, chyby v přechodech a překladech. Pouze pro účely tvorby nové vizualizace bylo vytvořeno nebo přeformátováno okolo tisíce proměnných a vazeb.

3.4.5 Řešení lokální a globální bezpečnosti

Bezpečnost stroje je nově rozdělena do 3 oblastí. Segment hlavního automatu, segment koordinačního automatu a segment programu. V případě zaznamenání poruchy na úrovni hlavního nebo koordinačního automatu nyní dojde k nucenému zastavení programů. V případě detekce poruchy

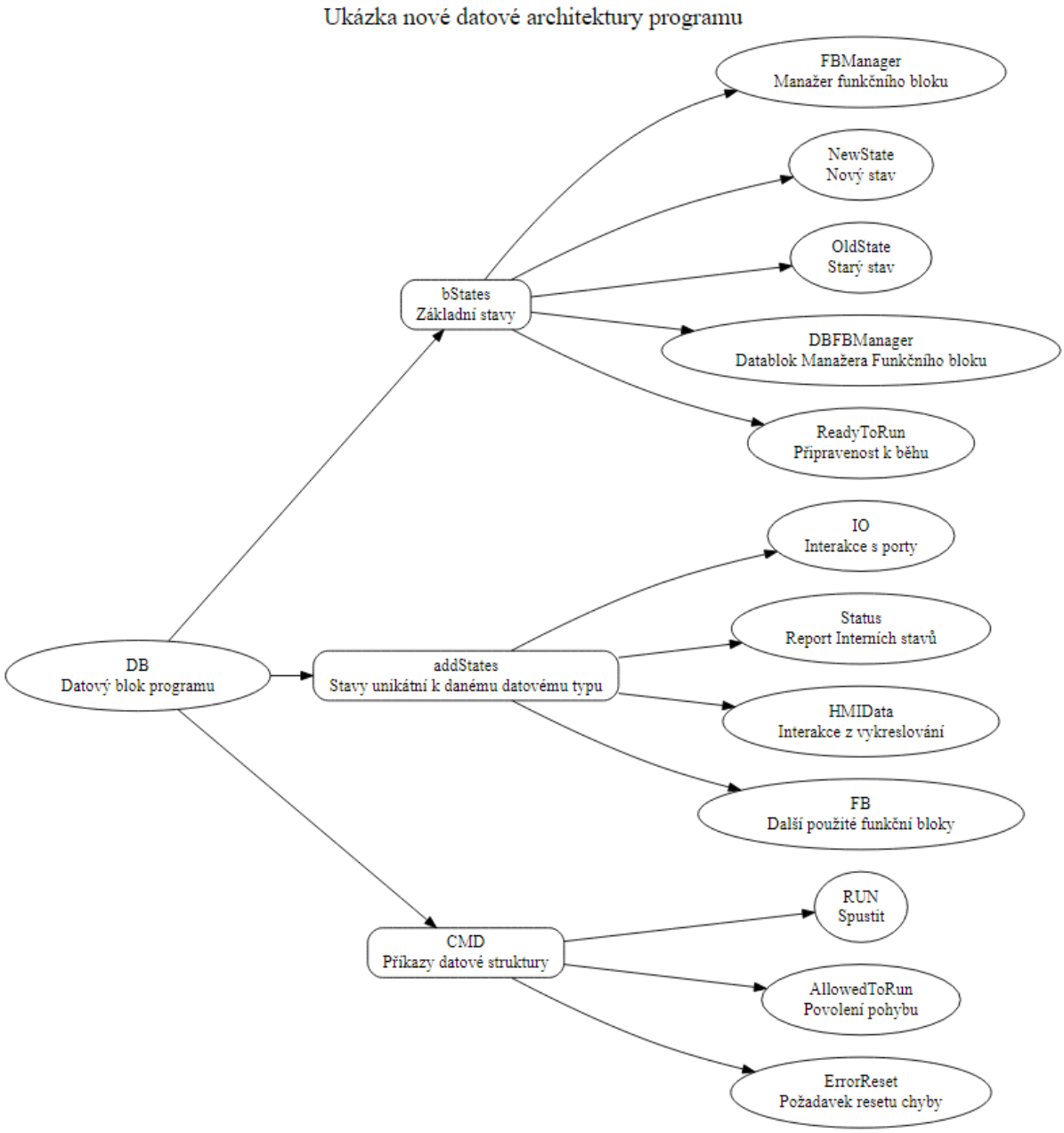
na úrovni programu je vyvolán přechod do nouzového stavu jen daného programu. Programy byly nově vybaveny 2 bezpečnostními okruhy. Okruhem primárním, kontrolujícím stav nouzového tlačítka. A okruhem sekundárním, který reaguje na interní bezpečnostní smyčku programu odpovídající přidělené části hardwaru. Softwarová bezpečnost byla vytvořena tak, že v případě záchytu stavu nebezpečí z nouzového tlačítka je jednoznačná a jednoduchá cesta k úkonům pro dosažení bezpečí stroje.



Obrázek 3.14: Diagram interakcí automatu, coorautu a manageru

3.4.6 Nová datová struktura programů

V rámci integrace programů do koordinačního automatu **COORAUT** a modularizace byla vytvořena nová forma obálky programů. Byl vytvořen datový typ **bStates**, který zaštiťuje základní proměnné potřebné pro provoz každého automatu. Pro proměnné specifické k danému automatu byla vytvořena datová struktura **addStates**, která pokrývá přenosy vstupů a výstupů hardwaru, interních proměnných a vykreslování. Dále je **addStates** místem řešícím instance funkčních bloků. Pro příkazy byla vytvořena proměnná **cmd** obsahující strukturu **CommandAbstract** a příkazy specifické pro daný program. Výsledkem tohoto zásahu byla homogenizace struktury všech programů. Tato nová struktura byla označena jako DataBlok (**DB**).



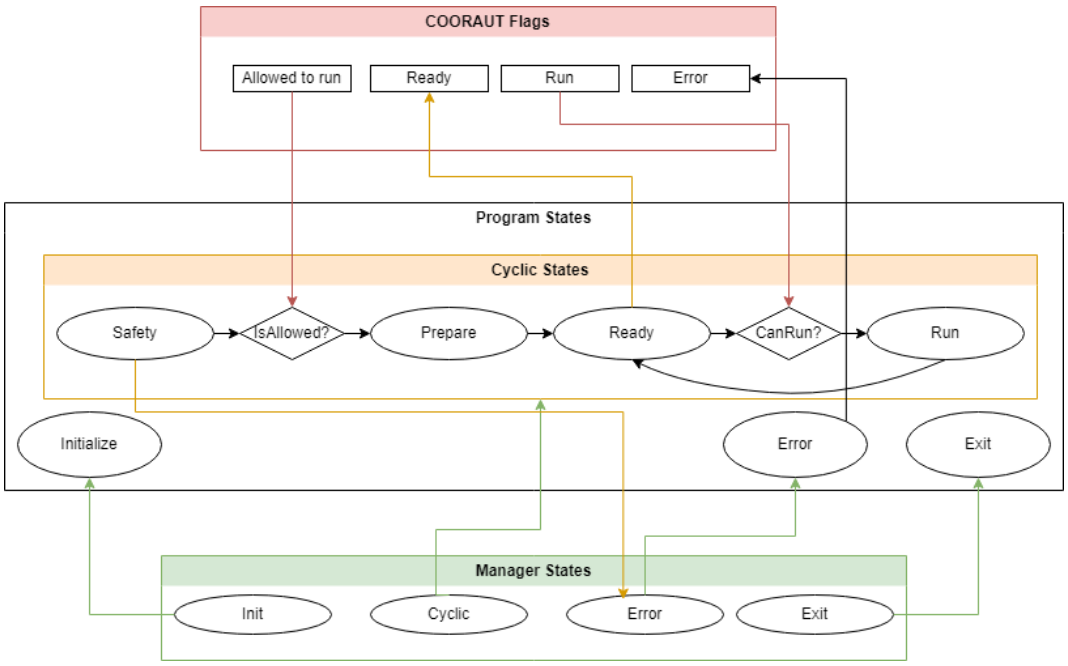
Obrázek 3.15: Diagram datové struktury programu

3.4.7 Redesign programů do funkčních bloků

Programy obsahující automaty pro manipulaci s fyzickými prvky se transformovaly do formy funkčních bloků s homogenizovanou obálkou. Původní automaty byly ve funkčních blocích rozděleny do úseků: **Init, Cyclic, Exit** a **Error**. Oproti původní struktuře programu tak došlo ke vzniku doplňkového stavu řešícího bezpečnostní činnost. Dále byly funkční bloky rozšířeny o proměnné řídící režii: **allowedToRun, isReady, isError**. Tyto

proměnné ve spolupráci s **COORAUTEM** umožnily detailní kontrolu nad procesem funkčního bloku.

Vrcholy programů nově slouží jako prostor implementace primární bezpečnostní smyčky, prostor přenosu dat z globální do lokální vrstvy a prostor volání funkčních bloků.



Obrázek 3.16: Diagram interakcí automatu, coorautu a manageru

3.5 FUNKČNÍ BLOKY

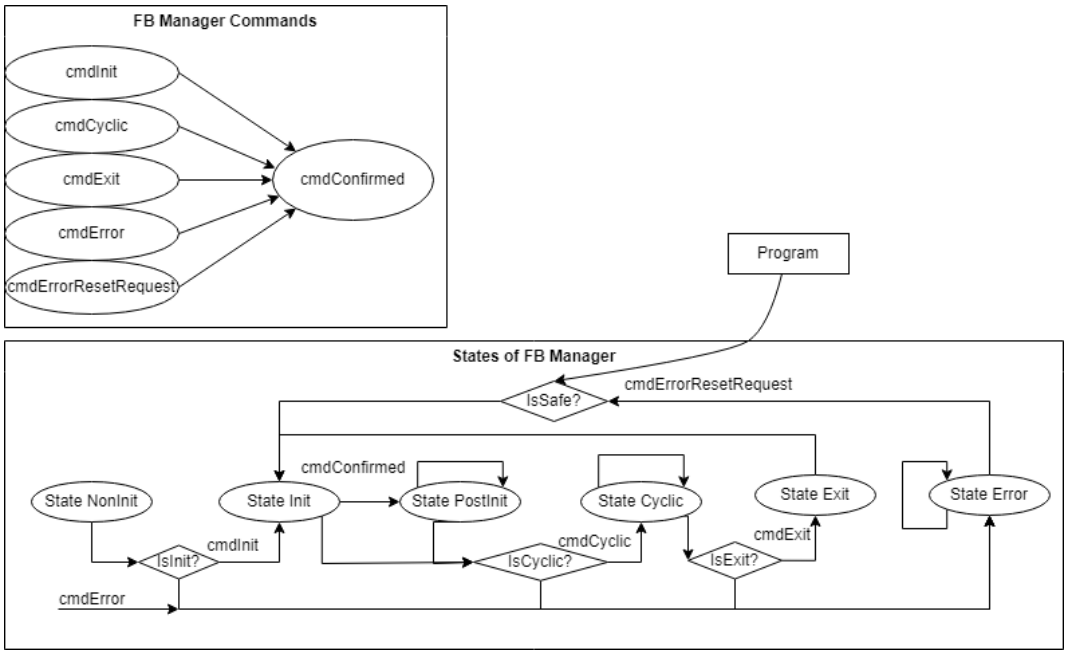
3.5.1 FBManager

Přepínání stavů automatů

Pro přepínání stavu automatů umístěných ve funkčních blocích byl využit jednoduchý stavový automat tzv. manažer funkčních bloků. Volba tohoto způsobu řešení byla iniciována původními pokusy o konverzi programů do funkčních bloků. V průběhu daných pokusů bylo zjištěno, že vzhledem k původní formě programu by musela vzniknout sada funkčních bloků, která by byla specifická pro každý úsek programu. Čili unikátní funkční blok pro každou oblast programu **Init, Cyclic, Exit**. Toto řešení by však výrazným způsobem omezilo udržitelnost a jednoduchost nasazení funkčních bloků, což nebylo žádoucí. Z tohoto důvodu byl pro přepínání stavových automatů vytvořen **FBManager**.

FBManager

FBManager je stavový automat řešící stavy **NonInit, Init, PostInit, Cyclic, Exit, Error**. Stal se řešením přepínání stavů funkčních bloků, resetu poruch a vyhlašování poruchových stavů. Ve vazbě s tím vznikl stav umožňující při poruše reagovat na daný problém bez většího narušení struktury funkčního bloku. **FBManager** se stal ústřední částí funkcionality celé linky, byl proto pokryt diagnostickými mechanismy a připraven na všechny možné eventuality a přechody. Tím byl zajištěn jednoznačný chod automatu. Pro účely diagnostiky byla vytvořena pomocná chybová proměnná uvádějící důvod poruchy a příznak poruchy. Později byl na tento systém napojen i další systém oznámení poruch z externích funkčních bloků.



Obrázek 3.17: Diagram stavových přechodů FBManageru

Popis sekvence

Z výše uvedeného diagramu vyplývá, že pro přechod ze stavu **NonInit** do **Init** musí dojít k interakci s příkazem **cmdInit**. Tento příkaz je vyvolán jen a pouze při inicializaci programu. Výsledkem je známý stav programu. Lze poznat, jestli byl program deaktivován, či nikoliv. Tomu se lze dále adaptovat. Stav **Init** následuje po stavu **NonInit** nebo **Exit** a probíhá vždy jen jednou. Pokud již došlo ke stavu **Init**, tak se přechází do stavu **PostInit**. Tento stav je stav „vyčkávací nebo nečinný“, kdy se očekává signál z koordinačního automatu a přechod do stavu **Cyclic**. Ve stavu **Cyclic** program zůstává, dokud není zavolán příkaz Exit. Všechny tyto stavy lze obejít příkazem **cmdError**, který má absolutní prioritu nad všemi stavy a uvede funkční blok do stavu **Error**. Pro návrat ze stavu **Error** je nutné splnit podmínku bezpečí **isSafe**, kde se

kontroluje například pohyb pohonů, přítomnost chyb na kontrolní jednotce nebo teplota topného tělesa.

3.5.2 Propisovač proměnných - FBVariableParser

V reakci na velmi omezené možnosti manipulace s proměnnými napříč vykreslováním byl vytvořen speciální funkční blok řešící konfiguraci blokace, limitních hodnot a nastavení výchozích parametrů. Za tímto účelem vznikla sada funkčních bloků s názvem **Parser**. Vrcholový funkční blok **Variable Parser** kontroluje limitní hodnoty proměnných. V případě, že proměnná poruší limit, tak je resetována do původního stavu a není zapsána hlouběji do programové paměti. Dále je možnost proměnnou resetovat na výchozí hodnotu za pomoci proměnné **RestoreDefaults**.

```
//Variable Parser
HMI42ConfigParser(
CmdRestoreDefault := gParams.HMI.RestoreDefaults,
Page := gParams.HMI.ManipulableData.ParamSpoolPage42,
HMIConfigParser := HMIConfigParser);
```

Zdrojový kód 3.1: Ukázka deklarace VariableParseru

```
//Config Parser
HMIConfigParser(
HmiMin := 0,
HmiMax := 0,
HmiDefault := 0,
HmiIsLimited := FALSE,
HmiInterface := Page.DynamicAngleDegreeMax.Interface,
cmdRestoreDefault := CmdRestoreDefault,
Var:= Page.DynamicAngleDegreeMax.MaxDegree
);
```

Zdrojový kód 3.2: Ukázka deklarace ConfigParseru

```
// Config Parser Code
HmiInterface.DefaultValue:= HmiDefault;

IF NOT HmiIsLimited THEN
  HmiInterface.MaximumValue:= REAL_MAX;
  HmiInterface.MinimumValue:= REAL_MIN;
ELSE
  HmiInterface.MaximumValue:= HmiMax;
  HmiInterface.MinimumValue:= HmiMin;
  //APPLY RANGE CHANGES
  IF Var < HmiMin OR Var > HmiMax THEN
    Var:= HmiInterface.DefaultValue;
  END_IF
END_IF

IF cmdRestoreDefault THEN
  Var:= HmiInterface.DefaultValue;
END_IF
```

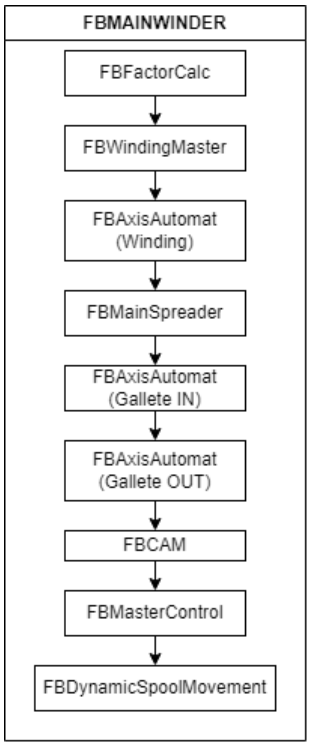
Zdrojový kód 3.3: Ukázka kódu ConfigParseru

V kombinaci s novým designem datové struktury vykreslování bylo dosaženo přehledného konfiguračního prostředí pro vizuální prostředí celého stroje. Vedlejším efektem této implementace bylo sdílení parametrizace proměnných pro vykreslování **VC4** a nové vykreslování **MappView**.

3.5.3 Hlavní návin - FBMainWinder

Původní program **MainWinder** pokrýval koordinaci a synchronizaci os stroje. Za pomoci referenční virtuální osy a převodových poměrů upravoval převodové rychlosti stroje tak, aby docházelo ke korektnímu přepínání a navíjení příze.

Design **MainWinderu** byl zvolen nevhodně. Jeho konstrukce spočívala v opakovaném kopírování kódu a modifikaci již dříve vytvořených automatů. V kombinaci se synchronizacemi více automatů vznikl kód, který měl přes 3000 řádků a byl obtížně modifikovatelný. Z tohoto důvodu byla provedena refaktorizace programu **MainWinder**. Program byl rozdělen do 12 funkčních bloků a hlavní smyčka zkrácena na 751 řádků při zachování původních funkcí.

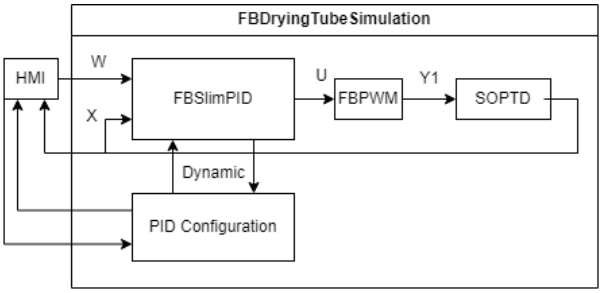


Obrázek 3.18: Diagram hlavního návinu

3.5.4 Sušící trubice - FBDryingTubes

V průběhu úprav a testování programu řízení sušících trubic vznikly potíže s načítáním paměti PID regulátoru. Při transformaci programu do funkčního bloku byla totiž narušena sekvence přístupu do dynamické paměti. Proto byla reimplementována simulační struktura dynamického systému sušících trubic. Z původní kódové dokumentace byl použit dynamický systém druhého řádu se zpožděním. Vznik problému nebyl ze začátku jednoduše patrný, protože se u regulátoru v režimu **LCRSLIMPID_REQU_OFF** nastavovaly výchozí hodnoty na zesílení Kp = 5 a integraci Tn = 60000, čili

PI regulátor. Celý problém spočíval v komplexnosti konfigurace regulátoru, který v některých režimech dané paměťové adresy ignoroval.



Obrázek 3.19: Zjednodušený diagram interakcí v FBDryingTubesSimulation

3.5.5 Vysoké napětí - FBHV

Program vysokého napětí je odlišný od většiny programů stroje. Je totiž nestavový. Z normy [11] se můžeme dočíst, že takový typ automatů je považován za bezpečnější. V rámci zachování již zmíněné bezpečnosti bylo při transformaci programu do funkčního bloku a integraci do nového systému provedeno několik úprav, které zaručují bezpečnost i po tak významné změně. Nový funkční blok v kombinaci s novým systémem si zachoval svoji funkcionalitu a všechny jeho možné stavy zůstaly plně pokryté.

3.5.6 Prototyp implementace UDP Serveru

UDP server vznikl na základě požadavku vytvoření ethernetové komunikace s PC a Rasberry Pi. Program byl pojmenován **Rasberry Server**. Tento server implementuje UDP komunikaci s JSON parserem dat. V rámci tohoto vývoje byla vytvořena nová globální proměnná **gRasberryServerOUT**, která má v současné chvíli malou strukturu dat, popisující interní stavy programů. Tato struktura je díky implementaci JSON enkodéru a dekodéru jednoduše adaptovatelná na veškeré vstupní proměnné a datové typy programů. Funkční blok JSON enkodéru a dekodéru byl převzat z [36] jako neoficiálního zdroje funkčního bloku. Vzhledem k tomu, že ke zmíněnému bloku nebyla dostupná aktuální dokumentace, bylo nutné při vývoji UDP serveru experimentovat. Samotná tvorba UDP Serveru byla už nenáročná, kdy bylo možné postupovat podle návodu [37].

Prototypová implementace nově vzniklé komunikace očekává spojení s klientem, kdy po získání zprávy od klienta vznikne dynamická proměnná obsahující pole znaků popisující aktuální stav **gRasberryServerOUT**. Tato data se následně propíší do paměti UDP serveru a odešlou. Po konci přenosu dynamická data zaniknou. Takto řešená implementace podporuje komunikaci s větším množstvím uživatelů, umožňuje výběr dalších datových struktur na základě dotazů.

```
IF JsonSerializerObj.Active AND NOT JsonSerializerObj.Error THEN
  JsonSerializerObj.PVname:= ADR('::gRasberryServerOUT');
  JsonSerializerObj.FileDevice:=0;
  JsonSerializerObj.FileName:=0;
  JsonSerializerObj.Execute:= TRUE;
END_IF

JsonSerializerObj();
recv();

IF recv.status = 0 AND JsonSerializerObj.Done AND NOT JsonSerializerObj.Error THEN
  brsmemset( ADR(receive_buffer), 0 , SIZEOF(receive_buffer));

  send.pData := JsonSerializerObj.JsonBuffer;
  send.datalen := brsstrlen( JsonSerializerObj.JsonBuffer );
  send.pHost := ADR(sender_ip_addr);
  send.port := recv.port;
  send.enable := TRUE;
END_IF

send();

IF send.enable AND send.status <> ERR_FUB_BUSY THEN
  send.enable := FALSE;
  IF JsonSerializerObj.Execute AND JsonSerializerObj.Done THEN
    JsonSerializerObj.Execute:= FALSE;
    JsonSerializerObj();
  END_IF
END_IF
```

Zdrojový kód 3.4: Ukázka kódu RasberryServer

```
 C:\Users\vascu\PycharmProjects\b&rtest
 \main.py
use port: 7010
Received: '{
    "::gArduinoServerOUT": {
        "ChamberHumidity": 36
         .979278564453125,
        "ChamberTemperature": 23
         .637195587158203,
        "ChamberVoltage": 0,
        "DryingTube1Temperature": 0,
        "DryingTube2Temperature": 0,
        "TensionCN": 0.65004426240921021
    }
}' from host ('192.168.1.2', 5000)
```

Obrázek 3.20: Snímek z testování komunikace

Testování proběhlo v prostředí PyCharm, kdy byl použit referenční kód aplikace ze zdroje [37].

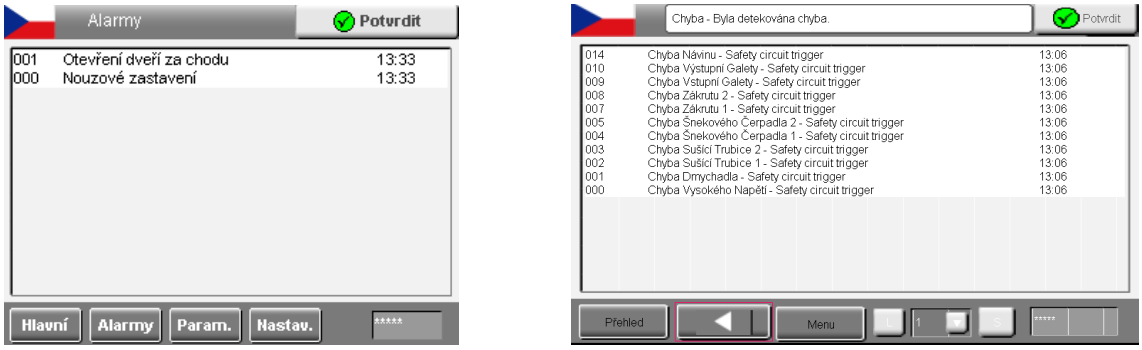
3.6 ÚPRAVY VC4 A TVORBA MAPPVIEW

3.6.1 VC4

Změny vizualizace **VC4** byly ovlivněny požadavkem na zachování původního designu.

Jako první krok při změně vizualizace **VC4** byla provedena transformace původní vizuální formy displeje z rozlišení 320×240 na rozlišení 800×480. Následovalo dohledávání vazeb vizualizace, programů a vznik **HMI** datové struktury. Jako druhý krok byla provedena změna vazeb proměnných z prostoru programů na **HMI**. Ve spojení s tím, byly navázány i manipulační proměnné pro nastavování blokačních a limitních stavů.

Vykreslovací prostředí **VC4** bylo nově vybaveno možností konfigurovat hardware a režimy provozu daného hardwaru. Konfigurace stroje byla implementována tak, aby při startu stroje došlo automaticky k načtení prvního profilu stroje, což umožňuje připravit stroj do provozu během 2 dotyků prstem. V rámci konfigurace byla vytvořena nápověda zapojení stroje a jistá forma predikce režimů stroje na základě vybraného hardwaru. Vykreslování bylo vybaveno možností uložit do paměti 5 dalších konfigurací stroje. Proměnné profilu lze resetovat za pomocí obnovení na výchozí hodnoty v záložce parametrů stroje. Pro uložení profilu stroje je nutný stav „**Připraven**“, tím je zajištěna bezpečnost při načítání profilu. Nově vznikla záložka parametrů stroje, která umožňuje v případě potřeby adaptovat konstanty pro výpočet polohování stroje (vzdálenost kladky, průměry galet atd.). Záložka alarmů byla významným způsobem vylepšena o přenos dynamických názvů poruch. Většina programů byla rozdělena do přehledných menu a byla doplněna o detailnější popis interních stavů.



(a) Původní (b) Nový

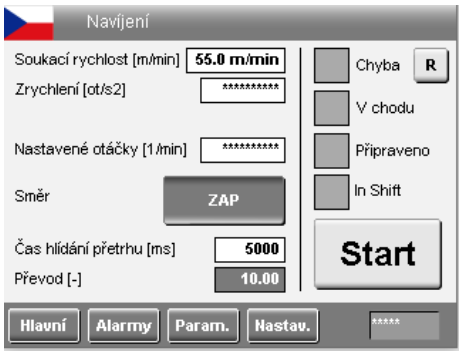
Obrázek 3.21: Ukázka vizuálních změn v alarmech

Design alarmů a upozornění se soustředil na 2 základní aspekty, a to vizuálně jednoznačně sdělit, že nastala porucha a umožnit uživateli alarm

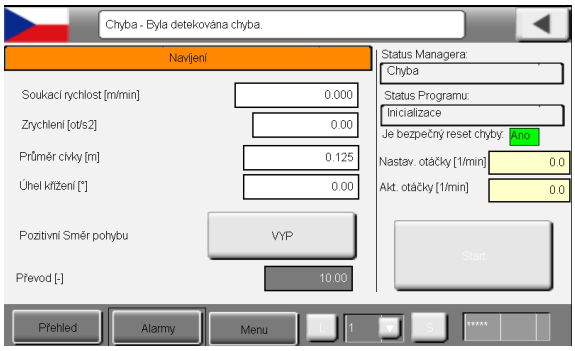
ignorovat. Ve vykreslování je nově zobrazován PopUp, místo původně použité záložky alarmů. Při transformaci záložky alarmů byla zachována původní architektura, která nutí uživatele vědomě potvrdit alarm v pravém horním rohu displeje.

Obrázek 3.22: Ukázka aktivace alarmu

Vzhled většiny programů byl upraven za účelem zlepšení čitelnosti. Každý úsek manipulace s hardwarem má nově výrazný nadpis. V menu je jednoznačně viditelné, jaké parametry jsou určeny jen ke čtení a se kterými lze manipulovat.



(a) Původní



(b) Nový

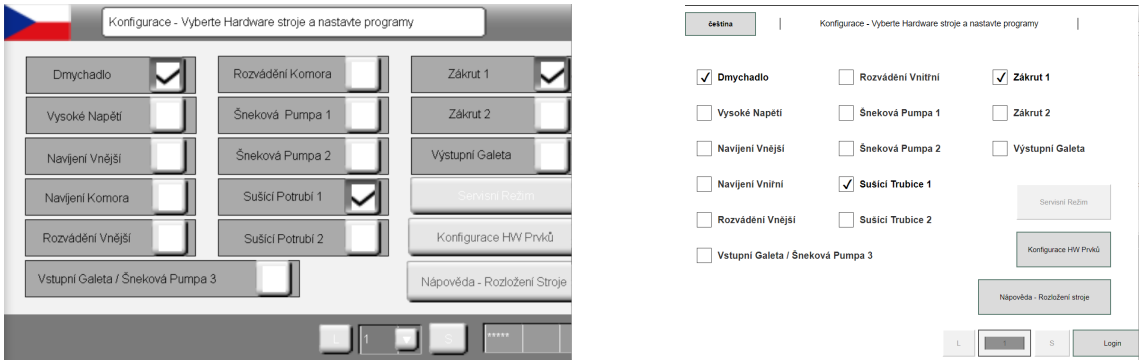
Obrázek 3.23: Ukázka vizuálních změn v návinu

Status manager nově zobrazuje aktuální stav funkčních bloků programů manipulujících s fyzickým hardwarem. Jeho výstup vychází ze stavů **FBManagera**, takže je možné dohledat, jestli je např. program připraven,

v chybě atd. To poskytuje zpětnou vazbu pro uživatele i vývojáře a vytváří další, vyšší úroveň diagnostického pokrytí **FBManagera**.

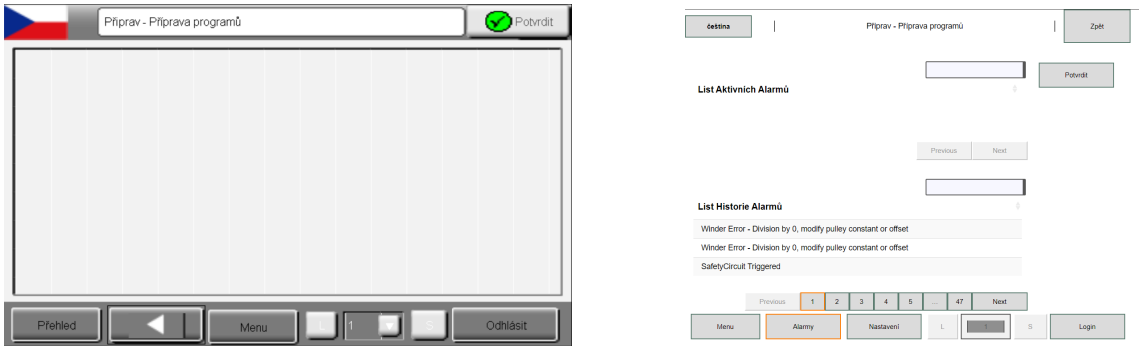
3.6.2 MappView

Pro účely konstrukce MappView byly vytvořeny již zmíněné funkční bloky **Parseru**, program **AlarmCtrl**, konvertor dat a datové struktury v **HMI**. Do vykreslovacího rozhraní byly integrovány MappService bloky **AlarmX**, **AlarmXHistory** a byl zprovozněn **OpcUA** server. Integrace MappView, vzhledem k časové tísni, byla provedena jen částečně. Byly ale položeny základy většiny funkcionalit a některých stránek, které lze použít jako referenci při dalším vývoji. Viz. níže uvedené snímky.



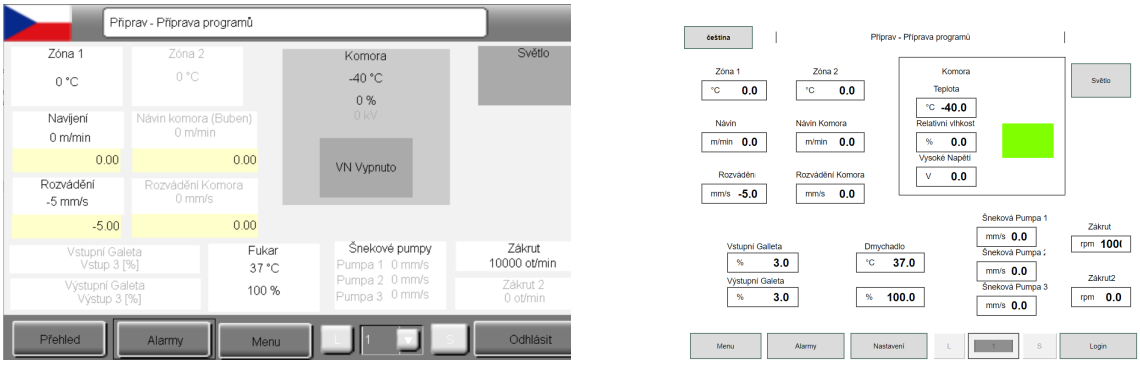
(a) Původní (b) Nový

Obrázek 3.24: Ukázka MappView vs VC4 Konfigurace



(a) Původní (b) Nový

Obrázek 3.25: Ukázka MappView vs VC4 Alarmy



(a) Původní (b) Nový

Obrázek 3.26: Ukázka MappView vs VC4 Alarmy

Faktory, které negativně ovlivnily proces tvorby MappView:

- „Rozsah stroje a nutnost funkčnosti“ - Datová struktura **HMI** obsahovala více jak tisíc jednotlivých vazeb. Zrekonstruování datové struktury vizualizace **HMI** bylo proto extrémně časově náročné. Z důvodu priority na testování a funkčnost linky KOPRIS1 byla preferována vizualizace **VC4**. Implementace **MappView** se tak stala z tohoto hlediska sekundární. V pozdější fázi vývoje stroje bylo jasné, že z důvodu rozsáhlosti vazeb **HMI** nedojde k plné a včasné realizaci všech stránek vizualizace dle požadavku zadání.
- „Nepohodlnost prostředí“ - Technologie **MappView** neumožnila rychlou tvorbu vizualizace. Funkce nápovědy neobsahovala návody pro komplexnější tvorbu scriptů pro manipulaci s webem. Vazby zdrojů, eventů a proměnných musely být sepsané v centrálním souboru manuálně. Chyběly funkcionality, které byly ve **VC4** samozřejmostí. Jako například tlačítko zpět. Případně některé funkcionality z výběru scriptů vůbec nefungovaly. Například detekce události **OnClick** na obrázku.
- „Udělej si sám“ - **MappView** nahrazuje omezený počet funkcionalit možností tvorby vlastních seskupení tzv. widgetů s vazbami a funkcionalitami za pomoci interních událostí nebo vlastností. Tyto vlastní widgety umožňují významným způsobem zvýšit komplexnost interakcí s webem. Tvorba takovýchto widgetů je ale v současnosti podmíněna manuálním vytvořením kódu.

3.7 TESTOVÁNÍ

Každý úsek nového softwaru byl nejdříve otestován a odladěn formou simulace pro zjištění a odstranění případných nedostatků. Následně byl proveden test funkčnosti softwaru praktickou formou, a to nahráním

softwaru do systému linky KOPRIS1. Praktická část testu probíhala za přímého odborného dohledu Ing. Martina Diblíka, Ph.D. V rámci fyzického testování byla provedena kontrola všech periferií stroje. Došlo k úpravě na pohonech linky KOPRIS1, jak na softwarové úrovni, tak hardwarové. Při testování se objevily menší komplikace s hardwarem, specificky s 4PPC70 displejem + CPU modulem. Tomuto modulu po jisté době selhala dotyková plocha. Tento problém byl vykompenzován použitím virtuálního monitoru (VNC). Pohon ACOPOS, zajišťující pohyb **CAM** osy na lince KOPRIS1 nebyl v průběhu testování nikdy na lince přítomen. Nebylo ho tak možné řádně otestovat a byl z daného důvodu vizuálně zablokován.

Kromě výše uvedených drobných nedostatků byla testováním v rámci provozu linky KOPRIS1 prokázána funkčnost a bezpečnost nových úprav v softwaru.

3.7.1 Finalizace projektu

Výsledný projekt linky KOPRIS1 získal sadu nových vlastních knihoven:

- **SourceLib** - Vrcholový úsek datové a softwarové struktury projektu.
- **HMILib** - Vizualizace.
- **MainWider** - Funkce hlavního návinu.
- **COORLib** - Funkce koordinačního automatu a podřízených programů.
- **AUTOMATLib** - Knihovna sekvenčních automatů.
- **MixedLib** - Knihovna nezařazených datových struktur.

Projekt byl dále přehledně rozdělen do složek podle účelu programu a datových struktur:

- **HW** - Hardware stroje.
- **AUTOMAT** - Sekvenční automaty.
- **COORAUT** - Koordinační automat.
- **HMI** - Programy ovlivňující vykreslování **VC4** a **MappView**.
- **DISABLED** - Deaktivované programy.
- **VHW** - Virtuální hardware stroje.
- **MAPPVIEW** - Programy manipulující jen s MappView.

V rámci finalizace projektu byly doplněny poznámky k fyzikálním parametrům proměnných a copyrighty. Pro některé programy byla doplněna dokumentace k logice daného úseku kódu formou komentáře **THEOREM** nebo **DOC**.

ZÁVĚR

Cílem této diplomové práce bylo seznámení se s aktuálním mechanickým, elektrickým a softwarovým řešením experimentální linky KOPRIS1. Na základě poznatků ze seznámení a analýzy vypracovat a implementovat kód, který by splnil požadavky zadání a uživatelů pracujících na lince KOPRIS1. V rámci implementace adaptovat a realizovat vykreslovací rozhraní s využitím technologie MappView.

Práce se v teoretické části zabývala programovatelnými logickými automaty 2.1.2, programovacími jazyky pro PLC 2.2.1, teorií designu datových struktur 2.2.3, řešením softwarové a hardwarové bezpečnosti 2.3 a nastíněním způsobu výroby nanovlákenných struktur 2.5. Došlo k seznámení s prostředím od firmy B&R Industrial Automation 2.4 a experimentální linkou pro výrobu nanovláken KOPRIS1 2.6.

V praktické části práce byla zpracována analýza původního softwarového vybavení experimentální linky KOPRIS1 3.2 a analýza využitelnosti IO. Na základě této analýzy a požadavků uživatelů byly provedeny a otestovány modifikace umožňující hardwarovou rekonfiguraci, komunikaci s ostatními zařízeními přes Ethernet a jednoduchou rozšířitelnost softwarového vybavení linky 3.4, 3.5. Důraz byl kladen na rozšířitelnost softwaru na další stroje a bezpečnost. Došlo k plné adaptaci původní vizualizace VC4 3.6.1 a vytvoření všech podkladů pro tvorbu plnohodnotné nové vizualizace za pomoci technologie MappView 3.6.2. Realizace MappView byla provedena formou funkčního konceptu, který lze použít jako referenci pro budoucí vývoj.

Vzhledem k tomu, že se povedlo stroj modularizovat a homogenizovat bez ztráty funkcionalit, můžeme považovat vedlejší cíl této práce, a to uvedení projektu do stavu blížícímu se průmyslovému stroji a zachování funkcí, za úspěšný.

Nově vzniklý software byl na stroji otestován 3.7 a byla ověřena jeho funkčnost a bezpečnost.

Diplomová práce splnila cíl zadání.

LITERATURA

[1] PIETRUSEWICZ, Krzysztof. Multi-degree of freedom robust control of the CNC X-Y table PMSM-based feed-drive module. Online. Archives of Electrical Engineering. 2012, roč. 61, č. 1, s. 15-31. ISSN 0004-0746. Dostupné z: https://doi.org/10.2478/v10171-012-0002-6. [cit. 2024-04-20].

[2] OSI Model. Online. In: Maturitní helpdesk. Duben 2017. Dostupné z: https://matureplus.4fan.cz/ pos/3-model-isoosi-vrstvy/. [cit. 2024-04-26].

[3] Ethernet Frame Format. Online. In: CBTnuggets. 2024, 2024-02-13. Dostupné z: https://www.cbtnuggets.com/blog/technology/networking/what-is-ethernet-frame-format. [cit. 2024-04-28].

[4] *Programmable controllers – Part 3: Programming languages*. Standard, International Electrotechnical Commission, Geneva 20, Switzerland, 2003.

[5] Channel Session Subscription. Online. In: Unified Automation. 2020. Dostupné z: https://documentation.unified-automation.com/uasdkhp/1.4.1/html/channel_session_subscription.png. [cit. 2024-05-05].

[6] LAMBERTS, Robert. Taylor Cone. Online. In: Cellkraft. 2016. Dostupné z: https://cellkraft.se/use-cases/electrospinning/. [cit. 2024-04-24]. CC BY 3.

[7] Peko Precision. Online. Březen 2021, leden 2024. Dostupné z: https://www.pekoprecision.com/blog/most-popular-PLC-brands/. [cit. 2024-04-26].

[8] B&R. Automation Runtime TM213. B&R Straße 1 5142 Eggelsberg, Austria: Industrial Automation, 2023. Interní výukové materiály.

[9] PLCOPEN. Motion Control. Online. 2019. Dostupné z: https://plcopen.org/technical-activities/motion-control. [cit. 2024-05-05].

[10] MATHWORKS. Simulink PLC Coder. Online. 2024. Dostupné z: https://www.mathworks.com/products/simulink-plc-coder.html. [cit. 2024-05-05].

[11] *Functional safety of electrical/electronic/programmable electronic safety-related systems – Part 3: Software requirements* . Standard, International Electrotechnical Commission, Geneva 20, Switzerland, 2010.

[12] ORTOLA VIDAL, Jeronimo. PLC Parts Service Life. Online. CERN. Indico.cern.ch. 2021. Dostupné z: https://indico.cern.ch/event/1072946/contributions/4565594/attachments/2328717/3968312/PLC_Parts_Service_Life.pptx. [cit. 2024-04-28].

[13] B&R INDUSTRIAL AUTOMATION. POWERLINK. Online. B&R INDUSTRIAL AUTOMATION. BR AUTOMATION. 2023. Dostupné z: https://www.br-automation.com/cs/technologie/powerlink/. [cit. 2024-04-28].

[14] B&R. POWERLINK configuration and diagnostics TM950. B&R Straße 1 5142 Eggelsberg, Austria: Industrial Automation, 2023. Interní výukové materiály.

[15] B&R INDUSTRIAL AUTOMATION. OPENSAFETY. Online. B&R INDUSTRIAL AUTOMATION. BR AUTOMATION. 2023. Dostupné z: https://www.br-automation.com/cs/technologie/opensafety/. [cit. 2024-04-28].

[16] BEKSI, William a PAPANIKOLOPOULOS, Nikolaos. Round-trip times (RTT) using the UDP, TCP, and Websocket protocols: A Cloud-based Object Recognition Engine for robotics. Online. In: Research Gate. 2015, 2015-09-28. Dostupné z: https://www.researchgate.net/figure/Round-trip-times-RTT-using-the-UDP-TCP-and-Websocket-protocols-for_transferring_fig5_308837835. [cit. 2024-04-28].

[17] Common PLC programing languages advantages. Online. Linkedin. 2024, 2024-03-06. Dostupné z: https://www.linkedin.com/advice/1/what-some-common-plc-programming-languages-advantages. [cit. 2024-04-28].

[18] ControlLogix in SIL 2 Applications: Rockwell Automation Publication 1756-RM001T-EN-P. Online. In: RockwellAutomation. 2023. Dostupné z: https://literature.rockwellautomation.com/idc/groups/literature/documents/rm/1756-rm001_-en-p.pdf. [cit. 2024-04-28].

[19] EDEMA, Victory. Function block instances in siemens tia portal. Online. SolisPLC. 2023, 2023-03. Dostupné z: https://www.solisPLC.com/tutorials/function-block-instances-in-siemens-tia-portal. [cit. 2024-04-28].

[20] VOJIR, Martin a BERAN, Leos. Global data structure for positioning machine controlled by PLC. Online. In: 2015 IEEE International Workshop of Electronics, Control, Measurement, Signals and their Application to Mechatronics (ECMSM). IEEE, 2015, s. 1-6. ISBN 978-1-4799-6972-2. Dostupné z: https://doi.org/10.1109/ECMSM.2015.7208699. [cit. 2024-05-05].

[21] B&R. *BR Coding Guidelines* [online]. Květen 04, 2023 [cit. 31.03.2024]. Dostupný z: https://www.br-automation.com/cs/akademie/skoleni-v-automation-academy/skolici-materialy/ridici-technika/tm231-br-coding-guidelines/.

[22] SIL, průmysl procesní techniky. Online. FESTO. Festo. 2024. Dostupné z: https://www.festo.com/cz/cs/e/reseni/odvetvi/prumysl-procesni-techniky/chemicky-prumysl/funkcni-bezpecnost-sil-id_4237. [cit. 2024-05-05].

[23] Emergency stop: compact guide. Online. In: Automation24. 2023. Dostupné z: https://media.automation24.com/shopsystem/infopages/not-halt/automation24-whitepaper-emergency-stop-systems-compact-guide-en.pdf. [cit. 2024-05-05].

[24] GODBOLEY, Sangharatna; KRISHNA, P. Radha a JHA, Ritesh Kumar. Atomic Condition Coverage Analysis for Structured Text Based Programmable Logic Controller (PLC). Online. In: Proceedings of the 17th Innovations in Software Engineering Conference. New York, NY, USA: ACM, 2024, s. 1-5. ISBN 9798400717673. Dostupné z: https://doi.org/10.1145/3641399.3641427. [cit. 2024-05-05].

[25] HAO, Li; SHI, Jianqi; SU, Ting a HUANG, Yanhong. Automated Test Generation for IEC 61131-3 ST Programs via Dynamic Symbolic Execution. Online. In: 2019 International Symposium on Theoretical Aspects of Software Engineering (TASE). IEEE, 2019, s. 200-207. ISBN 978-1-7281-3342-3. Dostupné z: https://doi.org/10.1109/TASE.2019.00004. [cit. 2024-05-05].

[26] OPC UA. Online. OPC Foundation. 2024. Dostupné z: https://opcfoundation.org/about/opc-technologies/opc-ua/. [cit. 2024-05-05].

[27] B&R. Introduction to motion control TM400. B&R Straße 1 5142 Eggelsberg, Austria: Industrial Automation, 2023. Interní výukové materiály.

[28] B&R. Basics of virtualization and simulation for control technology TM291. B&R Straße 1 5142 Eggelsberg, Austria: Industrial Automation, 2023. Interní výukové materiály.

[29] B&R. Programming and commissioning safety applications with mappSafety TM515. B&R Straße 1 5142 Eggelsberg, Austria: Industrial Automation, 2023. Interní výukové materiály.

[30] B&R. Working with SafeDESIGNER TM510. B&R Straße 1 5142 Eggelsberg, Austria: Industrial Automation, 2023. Interní výukové materiály.

[31] B&R. Diagnostics and service TM920. B&R Straße 1 5142 Eggelsberg, Austria: Industrial Automation, 2023. Interní výukové materiály.

[32] KONEČNÝ, Lukáš. Co je Nanovlákno. Online. NanoSPACE. 2022. Dostupné z: https://www.nanospace.cz/blog/co-je-nanovlakno/. [cit. 2024-05-05].

[33] CHVOJKA, Jiří. Speciální kolektory pro elektrostatické zvlákňování. Disertační práce. Liberec: Technická univerzita v Liberci, 2012.

[34] VALTERA, Jan; KALOUS, Tomas; POKORNY, Pavel; BATKA, Ondrej; BILEK, Martin et al. Fabrication of dual-functional composite yarns with a nanofibrous envelope using high throughput AC needleless and collectorless electrospinning. Online. Scientific Reports. 2019, roč. 9, č. 1. ISSN 2045-2322. Dostupné z: https://doi.org/10.1038/s41598-019-38557-z. [cit. 2024-05-05].

[35] MARTIN, Robert. Agile software development, principles, patterns, and practices. Harlow: Pearson, 2014. ISBN 978-1-292-02594-0.

[36] PRIES, Fabian. PvJson. Online. Github. 2023, 2023-07-18. Dostupné z: https://github.com/br-automation-com/PvJson. [cit. 2024-05-04].

[37] HILCHENBACH, Christoph. Demo AsTcp AsUdp. Online. Github. 2022, 2022-10-4. Dostupné z: https://github.com/hilch/demo-AsTcp-AsUdp. [cit. 2024-05-04].