FAKULTA MECHATRONIKY,
INFORMATIKY A MEZIOBOROVÝCH
STUDIÍ TUL

Diplomová práce

Tvorba řídicího software pro lanovou elektrodu pro výrobu plošných nanovlákenných produktů

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

Liberec 2024

FAKULTA MECHATRONIKY,
INFORMATIKY A MEZIOBOROVÝCH
STUDIÍ TUL

Zadání diplomové práce

Tvorba řídicího software pro lanovou elektrodu pro výrobu plošných nanovlákenných produktů

Jméno a příjmení:	**Bc. Jakub Kubát**
Osobní číslo:	M22000047
Studijní program:	N0714A270010 Mechatronika
Zadávající katedra:	Ústav mechatroniky a technické informatiky
Akademický rok:	2023/2024

Zásady pro vypracování:

1. Seznamte se s principem a praktickou realizací výroby plošných nanovlákenných produktů pomocí lanové zvlákňovací elektrody.
2. S využitím dostupného laboratorního hardware vytvořte funkční model lanové elektrody. Využijte dostupnou automatizační a pohonnou techniku BR-Automation.
3. Navrhněte a vytvořte software pro řízení pohybu lanové elektrody. Naprogramujte nezbytné pracovní režimy, vnitřní diagnostiku a chybové stavy.
4. Algoritmus řízení otestujte a uzavřete do podoby funkčního bloku. Rozhraní a způsob použití funkčního bloku zdokumentujte.

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] 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

[2] 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.

[3] MARTINÁSKOVÁ, Marie, Ladislav ŠMEJKAL. Řízení programovatelnými automaty. Praha: Vydavatelství ČVUT, 2004. ISBN 978-80-01-02925-1.

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. Dr. Ing. Jaroslav Hlava
garant studijního programu

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.

13. dubna 2024 Bc. Jakub Kubát

Tvorba řídicího software pro lanovou elektrodu pro výrobu plošných nanovlákenných produktů

Abstrakt

Tato diplomová práce se zabývá tvorbou řídicího software pro ovládání lanové elektrody, která je využita k výrobě plošných nanovlákenných produktů. Práce popisuje princip zařízení a laboratorního modelu, vývoj řídicího algoritmu pro řízení jedné lanové elektrody a následně se zabývá tvorbou funkčního bloku.

Klíčová slova: Programování PLC, řízení synchoronních servomotorů, řídicí software pro servomotory, B&R Automation, B&R Automation Software, Lanová elektroda

Abstract

This thesis deals with the development of control software for controlling a wire rope electrode used for the production of nanofibre printed products. The thesis describes the principle of the device and the laboratory model, followed by the development of a control algorithm for controlling a single wire rope electrode and then discusses the creation of a function block.

Keywords: PLC programming, control of synchronous servomotors, control software for servomotors, B&R Automation, B&R Automation Software, wire rope electrode

Poděkování

Rád bych poděkoval vedoucímu práce za rady a připomínky vyplývající z konzultací mé práce. Jsem vděčný za možnost zabývat se opět tématem výroby nanovláken s propojením tvorby softwaru. Velice děkuji za zodpovědné vedení a časovou flexibilitu v rámci konzultací.

Obsah

Seznam zkratek

TUL	Technická univerzita v Liberci
FM	Fakulta mechatroniky, informatiky a mezioborových studií Technické univerzity v Liberci
ST	Structured Text
true	logická jednička
false	logická nula
B&R	Bernecker & Rainer
VNC	Virtual Network Computing
USB	Universal Serial Bus
PLC	Programmable Logic Controler
HMI	Human Machine Interface
ST	Structured Text
IL	Indstruction List
SFC	Sequencional Function Chart
LD	Ladder Diagram
FBD	Function Block Diagram
FB	Function Block
ID	IDentification
CPU	Central Processing Unit

Úvod

Zařízení na výrobu nanovláken pomocí zvlákňovací elektrody je jednou z dostupných technologií pro výrobu nanovláken. Téma nanovlákem mi připadá velice zajímavé. Už při dokončování bakalářských studií jsem se zabýval návrhem a tvorbou řídicího systému laboratorního odstředivého zvlákňovacího stroje. Při výběru diplomové práce jsem měl zájem o téma podobného směru.

Před začátkem tvorby softwaru bylo nutné se seznámit s funkčností použitého zařízení, použitou technologií a vybavením celého stroje.

Dále jsem se věnoval návrhu algoritmu pro řízení jedné lanové elektrody tvořené dvěma servomotory. Pro účely programování byl využit laboratorní model, aby bylo zajištěno reálné mechanické spojení. Poté jsem algoritmus převedl do podoby funkčního bloku. Před jeho tvorbou je nutné si předem uvědomit vstupy a výstupy. Bylo potřeba si určit co bude mít uživatel k dispozici, jaké vstupní parametry bude potřebovat a jaké se očekávají výstupy z funkčního bloku. Toto vše je nutné, aby měl uživatel kompletní přehled o chodu stroje a mohl zadávat všechny parametry v požadovaném rozsahu a vhodných jednotkách.

Na závěr práce se věnuji testování funkčního bloku a reprezentaci jeho funkcí pomocí jednoduchého programu doplněného o vizualizaci.

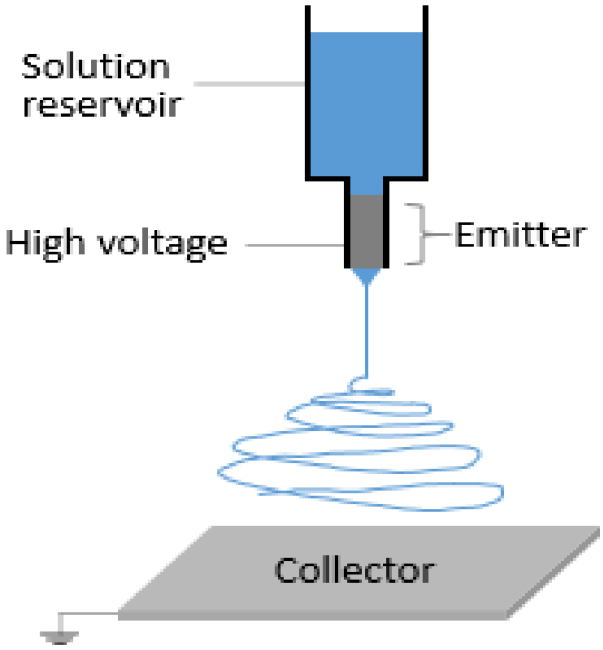
Ve své práci jsem čerpal ze zdrojů v práci uvedených, také ze zkušeností získaných v rámci navazujícího či bakalářského studia a zkušeností z praxe v průmyslu.

1 Technologie výroby vláken

Technologií výroby vláken je v praxi využíváno velmi mnoho. V této kapitole popíši technologii použitou ve výrobní lince a pro porovnání zmíním některé další principy.

1.1 Elektrospinning

Elektrospinning, neboli technika zvlákňování využívá k výrobě vláken elektrostatické síly vytvářené vysokým střídavým napětím.



Obrázek 1.1: Princip výroby – Elektrospinning [1]

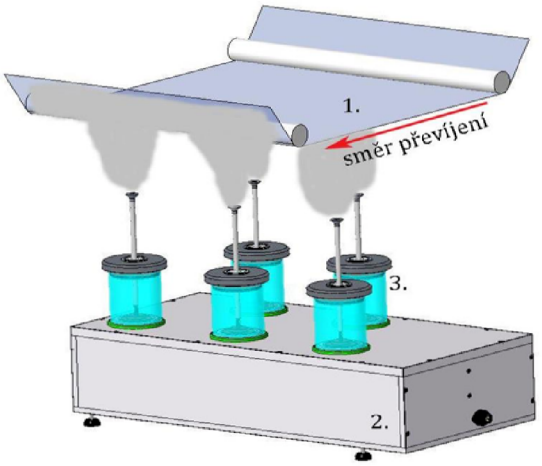
1.1.1 Elektrospinning – nanospider

Nanospider je založen na principu otáčejícího se válce, který je částečně ponořen do polymeru. Při otáčení na sebe nanáší tenkou vrstvu polymeru, který vynáší na vrchol válce. Na vrcholu válce vzniká tzv. Taylorův kužel, kde se vytváří Taylorovy proudy. Proudy roztoku jsou odpařením rozpouštědla přeměněna na pevná vlákna a poté jsou nanášena na nosič.

1.1.2 Elektrospinning – elektroda

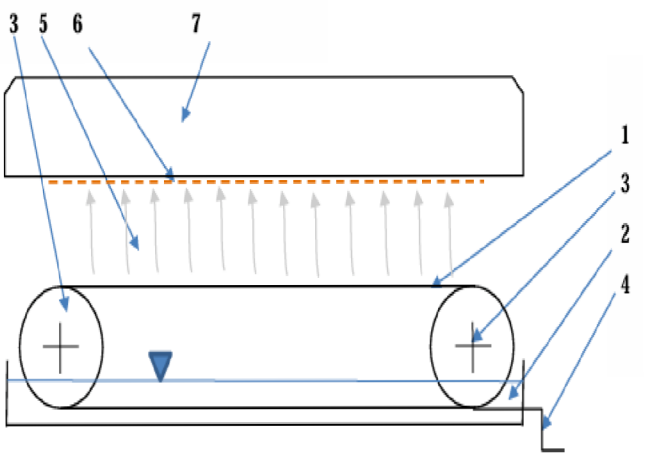
Na podobném principu jako nanospider funguje i zvlákňování pomocí elektrod. Silné odpudivé síly překonávají slabší síly povrchového napětí v nabitém polymeru. Pomocí tohoto principu dochází k uvolnění vlákna z elektrody. Uvolněná vlákna jsou následně nanesena na nosič, kterým je v případě výrobní linky plošná netkaná textilie.[10]

Pro vytváření plošných textilií byla využita prvně přeplavovací elektroda, která byla pro tuto technologii testována. Její využití je však nedostatečné, protože nevznikají celistvá homogenní vlákna.[9]



Obrázek 1.2: Technologie přeplavovací zvlákňovací elektrody [9]

Problém homogenity se podařilo odstranit tím, že se přešlo z přeplavovací elektrody k využití lanové elektrody, která je znázorněna na obrázku 1.3.

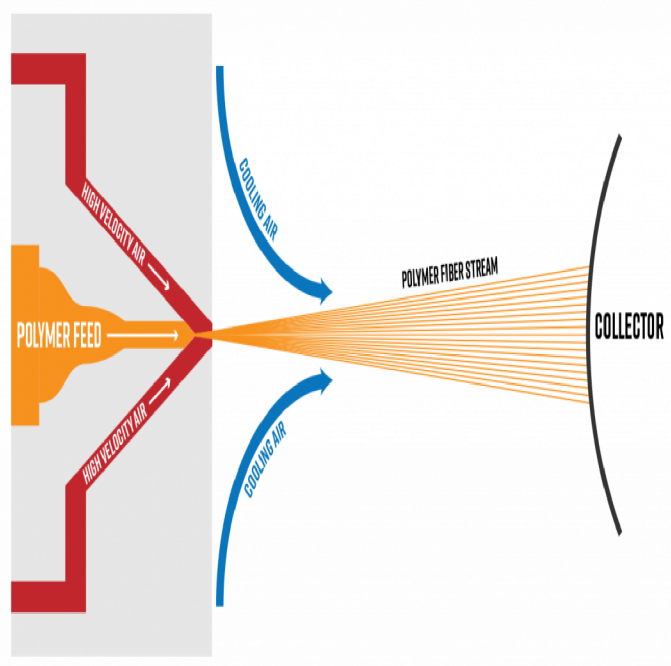


Obrázek 1.3: Technologie lanové elektrody [9]

1.2 Výroba vláken odstředivou silou

Technologie výroby vláken je založena na principu s využitím odstředivé síly. Polymer je nanášen na rotující válec a směřován tlakem vzduchu. Tím se z polymeru oddělují nanovlákna a mikrovlákna. Tento druh výroby vláken byl využit například u odstředivého zvlákňovacího zařízení vytvořeného na TUL. Princip výroby pomocí odstředivé síly je přiblížen na obrázku 1.4. Žlutou barvou znázorněný polymer uvolňující se z rotujícího válce je usměrňován pomocí tlaku vzduchu (modrá barva) a následně nanášen na nosič.

V případě zájmu o vidění této technologie v praxi, naleznete popis zařízení s využitím této technologie v mé bakalářské práci. [8]



Obrázek 1.4: Princip výroby – Odstředivá síla [1]

2 Popis zařízení

2.1 Princip funkce

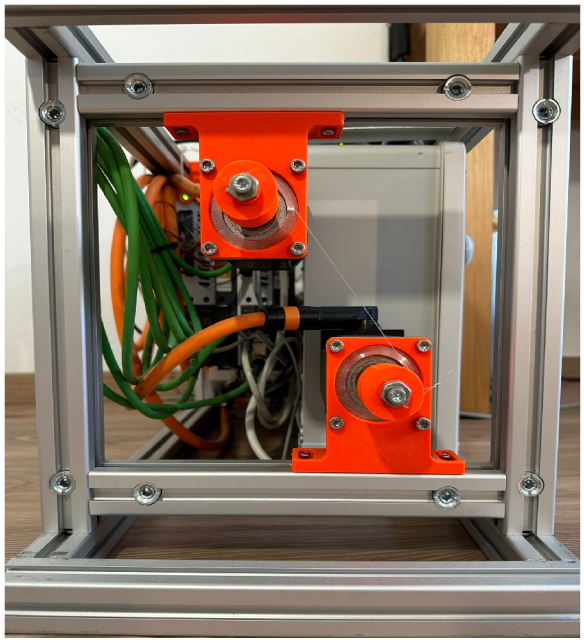
Zařízení pro výrobu plošných nanovlákenných produktů pracuje na principu nanášení polymeru na lanovou elektrodu smáčením. Kladka s navinutým lanem je z části ponořena do polymeru a dochází tak k nanesení polymeru na lano. Pomocí motorů je lano převíjeno z jedné strany na druhou, kde se vždy část lana navíjí na kladku. Kladky včetně lana jsou napojeny na vysoké napětí a pomocí elektrostatického pole dochází k tvorbě nanovláken, která jsou následně nanášena na plošný nosič. Převíjení lana je realizováno dvěma synchronními motory, které udržují lano stále napjaté. Motory jsou uloženy pod kryty, aby nedošlo k jejich zanesení polymerem nebo čisticím roztokem. Při chodu nesmí dojít k povolení tahu, jinak by došlo ke spadnutí lana z kladky a tím k přerušení výroby a zničení vláken. Stroj je využit k tvorbě plošných vláken až o šířce 1,6 metru a následně skládán do rolí. Rychlost výrobní linky se pohybuje v rozmezí 0,5 až 10 metrů za minutu. [6]

Obrázek 2.1: Linka na výrobu plošného kompozitního materiálu s obsahem nanovláken [9]

Jak můžeme vidět na obrázku 2.1, na výrobní lince jsou umístěny čtyři lanové elektrody za sebou. Každá lanová elektroda je poháněna dvěma servomotory. Lanko mezi servomotory představuje tahovou mechanickou vazbu. V případě laboratorního modelu je zařízení tvořeno pouze jedním párem motorů, neboli jednou elektrodou. Pro vytvoření funkčního bloku nebylo možné použít simulátor, protože bylo potřeba realizovat fyzickou vazbu mezi motory. Model zvlákňovací elektrody vidíme na obrázku 2.2.

Obrázek 2.2: Laboratorní model linky lanové elektrody

Pro tvorbu řídicího algoritmu byl využit zjednodušený laboratorní model, jehož kladky jsou upnuty přímo na motory. Toto provedení je snadnější a pro vytvoření algoritmu a poté funkčního bloku dostačující. Detailní provedení kladek je znázorněno na obrázku 2.3.



Obrázek 2.3: Náhled na uložení kladek laboratorního modelu

2.2 Technologie

Aktuální technologie převíjení lana z jedné kladky na druhou je velmi spolehlivá. Díky momentovému brzdění jsme schopni definovat a sledovat tah v lanku. Má ale i své nevýhody.

Zařízení potřebuje neustálý tah v lanu, jeho zajištění je realizováno pomocí momentového řízení a absolutního polohování. Při aktuálním provedení dochází k neustálému brzdění a rozjíždění či reverzaci, což jsou energicky velmi náročné části pohybu motorů. Oba motory jsou stále v činnosti a generují kroutící moment, to není z energetického hlediska optimální a projevuje se to na zvýšeném ohřevu. Konkrétní popis pohybů je popsán v sekci 5.2, kde je rozebrán celý software stroje.

Pomocí sledování momentů a poloh na jednotlivých motorech můžeme sledovat několik problémů, které mohou nastat. Ať už je jedná o případné přetržení lana či jiné problémy, pro které bylo nutné vytvořit diagnostické nástroje. Podrobněji zmiňuji možné chyby přímo v kapitole 7, která se zabývá ošetřením možných chyb na zařízení.

Dalším principem řešení lanové elektrody, která se aktuálně testuje, je tzv. nekonečné lanko. Technologie je založena na převíjení uzavřeného lana stále dokola a je tak z hlediska využití energie o dost výhodnější, protože kontinuálním převíjením lanka zamezíme neustálým reverzacím.

Vzhledem k tomu, že motory jsou umístěny v plastovém krytu, je při aktuální technologii problém s odvodem ztrátového tepla. Je nutné sledovat teploty motorů a včas zareagovat na zvyšující se teplotu. Aktuálně se při zvýšení teploty motor vypne a elektroda spadne z lanovice. Proto je vhodné zavést monitoring teploty, díky kterému bude poté mít nadřazený systém možnost zareagovat na její zvýšení.

2.3 Hardware výrobní linky

Výrobní linka pro svoji funkci využívá komponenty od B&R. Motory jsou tvořeny synchronními pohony typu 8LVA33.B8015S100–0, které jsou řízeny frekvenčními měniči ACOPOS. Celé zařízení je řízeno pomocí modulárního PLC typu X20CP1584. Na reálném stroji nebylo možné provádět testování, byl proto využit laboratorní model. Pro laboratorní model zvlákňovací elektrody bylo využito ekvivalentních komponent od B&R, aby bylo zařízení co nejvíce přiblíženo realitě. Avšak tento výběr komponent není nutnou podmínkou pro tvorbu modelu výrobní linky. Komponenty reálného stroje a modelu se nijak zásadně neliší. Pro porovnání jsem vytvořil v sekci 4 tabulku srovnávající parametry jednotlivých komponent.

3 Vývojové prostředí

3.1 Automation Studio

Pro vývoj softwaru jsem využil dostupného programu od společnosti B&R – Automation Studio verze 4.11. Jedná se o programovací nástroj pro tvorbu softwaru, který je navržen s využitím jazyků dle IEC 61131–3 a standardu PLCopen. Díky těmto normám není software limitován výběrem hardwaru. Detailnější popis programovacího softwaru viz manuál dostupný na stránkách Bernecker & Rainer [3].

3.2 Programovací jazyky

Pro tvorbu algoritmu má programátor na výběr několik jazyků a je prakticky pouze na jeho volbě, který si vybere. Norma IEC 61131–3 [7] definuje pět základních programovacích jazyků, lze vybírat z:

Tabulka 3.1: Tabulka programovacích jazyků

Programovací jazyk	**zkratka**
Ladder Diagram	LD
Function Block Diagram	FBD
Structured Text	ST
Instruction List	IL
Sequentional Function Chart	SFC

Pro torbu softwaru jsem se mohl rozhodnout při výběru programovacího jazyka dle dostupných, viz tabulka 3. Pro jednoduchou a přehlednou realizaci jsem zvolil strukturovaný text. Strukturovaný text vychází z jazyka Pascal či C a je jim velmi podobný. Jazyk pracuje s instrukcemi a příkazy, které mohou být prováděny podmíněně. Jazyk vykonává zadané instrukce v posloupnosti. Strukturovaný text je vhodný pro tvorbu mého funkčního bloku, protože zařízení neobsahuje veliké množství vstupů a výstupů, pro které by se mohl stát nepřehledným. [11]

4 Použité komponenty

Při tvorbě zařízení se zaobíráme volbou komponent především z hlediska vhodnosti dle požadavků, ceny, doporučení či navázané spolupráce v rámci organizace. Komponenty zvolené pro model elektrody jsou především vybrány v rámci doporučení fakulty. Volba komponent byla zaměřena na reálný stroj, kde jsou také použity komponenty od firmy B&R, ale v principu na hardwaru nezáleží. Snaha o návrh softwaru byla taková, aby nebylo nutné pro využití funkčního bloku dodržet typ hardwaru.

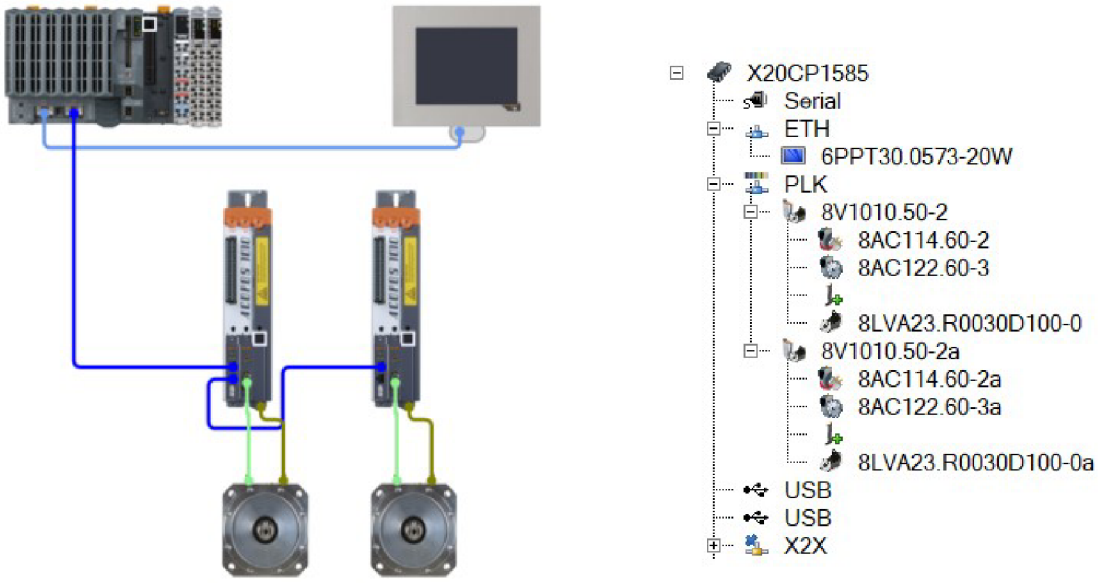
Rozdíly mezi výkony motorů na reálném zařízení v porovnání s motory na laboratorním modelu nejsou příliš velké. Pro porovnání parametrů jednotlivých komponent jsem využil tabulky.

Tabulka 4.1: Tabulka porovnání použitých komponent

typ	**označení**	**zařízení**	**parametry**
servomotor	8LVA33.B8015S100–0	výrobní stroj	1500 rpm, 2, 5 Nm, 2, 45 kg
servomotor	8LVA23.R0030D100–0	laboratorní model	3000 rpm, 1, 3 Nm, 1, 45 kg
PLC	X20CP1584	výrobní stroj	0.6 GHz, 400 µs
PLC	X20CP1585	laboratorní model	1 GHz, 200 µs

Tabulka porovnává hlavní komponenty, kterými jsou motory a PLC. Jak vidíme z tabulky 4.1 motory mají opravdu podobné výkony a dokáží vytvořit podobné momenty. Co se týče PLC, ty mají podobné vlastnosti také. Jsou vybaveny stejnými sběrnicemi a PLC laboratorního modelu je dokonce rychlejší než PLC aktuálně použité na výrobním stroji.

Laboratorní zařízení je tvořeno dvěma hlavními synchronními motory, které jsou řízeny pomocí příslušných frekvenčních měničů. Vše je řízeno hlavním PLC ze série X20, které je umístěno uvnitř rozváděče společně s rozšiřujícími kartami. Komunikace mezi jednotlivými prvky probíhá pomocí sběrnice PowerLink. Dotykový display, který slouží pro odzkoušení funkčního bloku, nastavování jednotlivých bitových proměnných a zadávání parametrů, komunikuje s PLC pomocí sběrnice Ethernet Powerlink. Jednotlivé prvky zařízení jsou viditelné z obrázku 4.1, který zobrazuje hardwarovou konfiguraci laboratorního modelu.



Obrázek 4.1: Hardwarová konfigurace laboratorního modelu

Díky využití standardu PLCopen a normy IEC 61131–3 je výběr hardwaru prakticky nezávislý na softwaru. S tímto přístupem jsem programoval a vytvářel algoritmus, který byl v závěru převeden na funkční blok.

4.1 Řízení procesu

Řízení výrobních procesů v průmyslu lze provádět pomocí PLC či průmyslových PC. Z hlediska bezpečnosti, rozšířitelnosti a modularity jsou v praxi více využívána PLC pro výrobní stroje. Nejčastěji se jedná o modulární PLC, která mohou být za svého provozu rozšířena při případných přestavbách či vylepšeních zařízení.

Pro model výrobní linky nanovláken je využito PLC od B&R ze série X20. Konkrétně se jedná o kompaktní PLC typu X20CP1585. Pro komunikaci lze využít zabudovaný Ethernet, Ethernet Powerlink a rozhraní USB. PLC je vybaveno kompaktní Flash pamětí, což je výhodou pro offline instalaci. Kompaktní paměť může být vyjmuta a lze na ni přes software Automation Studia nahrát přímo software pro námi řízený proces. Tato možnost je velice užitečná pro offline instalaci v případě nemožnosti komunikace či problému ve spojení mezi PLC a PC programátora. Konkrétní použité PLC je zobrazeno na obrázku 4.2.

Obrázek 4.2: Programovatelný automat B&R typ X20CP1585[12]

4.2 Řídicí jednotka pro servopohony

Motory je nutno nějak řídit, zadávat jim žádané hodnoty a zároveň od nich dostávat zpětnou informaci o jejich stavu a aktuálních parametrech. Pro splnění těchto požadavků a dosažení plynulé regulace potřebujeme frekvenční měniče se zpětnou vazbou. Pro náš laboratorní model jsou použity měniče ACOPOS typu 8V1010.50–2. Tyto měniče umožňují připojení jednoho motoru a až 3 plug–in moduly. Pro komunikaci využívá měnič modul 8AC114.60–2, který vytváří spojení pro PowerLink. Dalším důležitým modulem je 8AC122.60–3, který slouží pro detekci vstupních signálů enkodéru.

Obrázek 4.3: Frekvenční měnič ACOPOS 8V1010.50–2 [2]

4.3 Motory

Výkonovou část zařízení tvoří sycnhronní motory. Model využívá synchronní motory s permanentními magnety se čtyřmi pólpáry. Mohou dosáhnout rychlostí až 3000 ot/min a dokáží vytvořit moment o velikosti až 1,3 Nm. Synchronní motory jsou obecně dražší, ale vyplatí se vzhledem ke svým dobrým vlastnostem, kterými je vysoká účinnost a možnost konstantní rychlosti od nulové zátěže až po plnou zátěž. Tyto vlastnosti je činí vhodnými pro naši aplikaci.

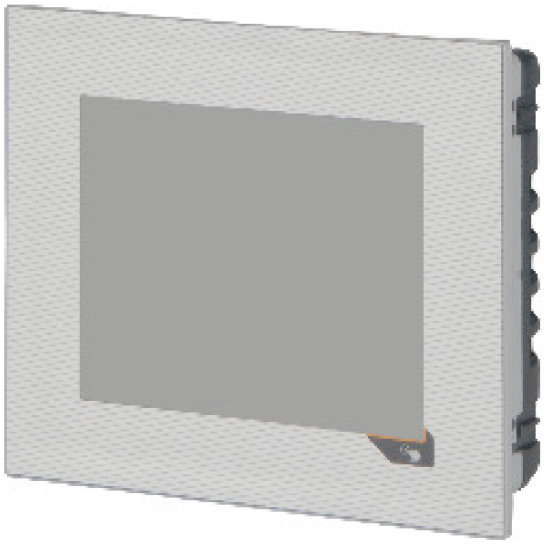
Obrázek 4.4: Synchronní motor 8LVA23.R0030D100–0 [5]

4.4 Ovládací grafický panel

Pro jednoduchou reprezentaci a otestování funkčního bloku nesmí chybět uživatelské rozhraní, které slouží k interakci s uživatelem. Operátorský panel je potřebný pro zadávání vstupů do funkčního bloku a reprezentaci aktuálních statusů zařízení.

Laboratorní model byl doplněn dotykovým panelem 6PPT30.0573–20W kvůli možnosti reprezentace funkčního bloku. Přes panel jsem provedl kompletní ovládání, tzn. zadávání všech parametrů a povelů. Zároveň jsem pomocí panelu mohl zobrazit kompletní přehled diagnostiky chyb, kterou jsem do funkčního bloku naprogramoval.

Dostupný panel od B&R má dva vstupy pro USB a dva vstupy pro Ethernet, pomocí kterého je také k zařízení připojen. Napájení jsem provedl z již využitého zdroje, který napájí interní komponenty v rozváděči. Pomocí stávajících průchodek jsem vyvedl napájení a ethernet kabel sloužící pro připojení programátorského PC. Náhled panelu je na obrázku č 4.5.



Obrázek 4.5: Dotykový panel 6PPT30.0573–20W [4]

5 Tvorba algoritmu pro řízení

Konečným cílem mé práce bylo vytvořit funkční blok, který bude ovládat jednu lanovou elektrodu. Tento blok bude sloužit pro další zařízení, která budou obsahovat více lanových elektrod. Výhodou je, že blok tak bude pro řízení zařízení s více lanovými elektrodami volán vícekrát.

Protože programování funkčního bloku přímo je kvůli hledání chyb příliš náročné, tak jsem nejdříve program vytvořil jako běžnou programovou jednotku ve strukturovaném textu. V B&R je nazvaná jako „Program All in One“. Po kompletním odladění chyb a odzkoušení jsem teprve program ucelil do funkčního bloku – více v kapitole 8.

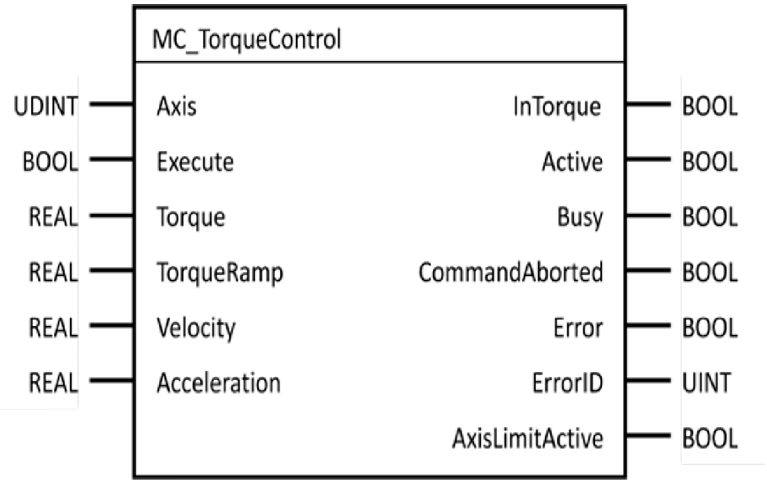
5.1 Řízení motorů – využité bloky

Pro řízení pohybu motorů jsem využil část `Basic` programů, které jsou připraveny jako vzorové přímo výrobcem B&R. Jejich využití je vhodné z důvodu snadného řízení a stejné výchozí pozice vstupních parametrů pro funkční blok.

Základní programy pro ovládání motorů jsem doplnil o momentové řízení (více v sekci 5.2), měření aktuálního momentu, snímání teploty motorů, snímání aktuálních momentů a indikaci dosažení požadovaných pozic. Snímání teploty jsem zavedl z důvodu aktuálního problému vyskytujícího se na zařízení. Problém spočívá v tom, že motory zařazené za sebou vydávají ztrátové teplo. Teplo se nestíhá odvádět a motory vytváří teplotní tunel. Motor, který je umístěn poslední v řadě, se tak často přehřívá, a stroj potom padá do chyby. Teplotu je tedy možné zobrazovat jako výstup, či programově monitorovat její hodnotu a nakládat s ní podle potřeb.
V této sekci popíši některé využité bloky a přidané ověřovací sekce, které jsou pro řízení nutné a je nutné o ně základní `Basic` programy motorů rozšířit.

5.1.1 Řízení momentu motoru

Pro dosažení řízení momentu jsem využil bloku `Mc_MoveTorque`, který hlídá nastavený moment motoru při pohybu konstantní rychlostí. Soustředil jsem se na výběr bloku z PLCopen standardu tak, aby byl software na hardwaru nezávislý.



Obrázek 5.1: Blok `MC_MoveTorque` [3]

Blok potřebuje definovat vstupní osu, kterou lze definovat pod parametrem `Axis`. Následně je nutné zadat rychlost `Velocity`, zrychlení `Acceleration`, moment `Torque`, rampu, se kterou má být momentu dosaženo `TorqueRamp`, a vyvolat jeho spuštění pomocí ovládacího signálu `Execute`. Výstupními hodnotami je dosažení momentu `InTorque`, zaznamenání stavu aktivní `Active`, zaznamenání stavu zaneprázdněného bloku `Busy`, zamítnutí příkazu `CommandAborted`, dosažení limitu osy `AxisLimitActive` a zaznamenání chyby `Error` a jejího identifikačního čísla `ErrorID`. Další informace o bloku a jeho využití lze nalézt v Helpu Automation studia [3].

Momentové řízení jsem zavedl jako součást `Basic` programu včetně všech parametrů a povelů. Doplnil jsem tak rozsáhlou strukturu programu pro řízení motorů. Jak vidíme z kódu, momentové řízení obsahuje zpracování povelů pro zastavení pomocí příkazů `HALT` a `STOP`. Ve spodní části vidíme, že momentové řízení obsahuje také čtení chybových textů a identifikačních čísel chyb. Čtení chyb jsem zavedl do kompletního programu pro řízení každého jednotlivého pohonu.

5.1.2 Indikace dosažení požadovaných pozic

Pro indikaci dosažení požadované pozice jsem zavedl jednoduchou zpětnou vazbu, založenou na rozdílu žádané a aktuální pozice motoru. Tímto rozdílem dokážu v případě shody ověřit dosažení motoru na požadovanou pozici. Tuto zpětnou vazbu využívám v rámci celého algoritmu, protože provedení absolutního pohybu neobsahuje kontrolu dosažení požadované pozice. Následně jsem tuto kontrolu zavedl jako součást funkčního bloku. Obecně kontrola slouží především jako ověření po dokončení procesu absolutního pohybu a ověřuje dojetí motoru na požadovanou pozici.

5.1.3 Snímání teploty

Teplotu je důležité kontrolovat kvůli zamezení přehřívání motorů. Snímání teploty je bohužel závislé na konkrétním hardwaru, proto nebylo možné využít PLCopen blok. Toto je jediná část programu, která se bohužel kvůli nutnosti vyčtení parametru z hardwaru, stala hardwarově závislá. Čtení teploty bylo realizování pomocí bloku `MC_BR_ReadParID`. Pomocí tohoto bloku lze vyčítat jednotlivé parametry z motorů. Pod parametrem 381 jsem vyčetl teplotu motoru.

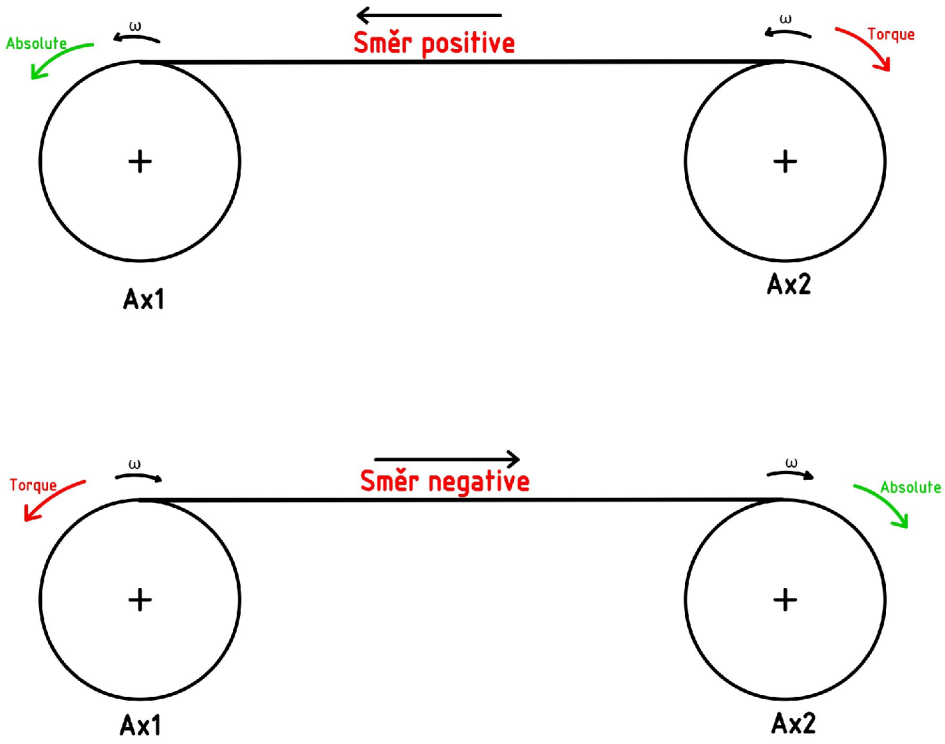
5.1.4 Snímání momentů

Uživateli jsem chtěl v rámci zobrazování stavů lanové elektrody zobrazovat i aktuální momenty vytvářené motory. Proto jsem doplnil programy k řízení motorů snímáním aktuálního momentu. Snímání aktuálního momentu jsem provedl pomocí bloku `MC_ReadActualTorque`. Jedná se blok z rodiny PLCopen, tudíž je jeho realizace možná na hardwaru předem neurčeném.

5.2 Řízení regulačních struktur motorů

Pro udržení lana v tahu při převíjení z jedné kladky na druhou potřebuji zajistit pro jeden motor polohové řízení a pro druhý motor momentovou brzdu. Momentová brzda je důležitá z hlediska dodržení napjatosti v lanu s předem definovaným momentem, který se odvíjí od typu použitého lana či nastavení vhodné nebo bezpečné hodnoty.

Mezi těmito typy pohybů motory vždy přepínají podle směru navíjení lana. Na obrázku 5.2 vidíme náčrt jedné osy výrobní linky nebo také osy laboratorního modelu.



Obrázek 5.2: Schematické uspořádání lanovic a orientace veličin

Na obrázku 5.2 je pro lepší pochopení naznačen pohyb, který vykonávají motory. Při pohybu v pozitivním směru se Ax2 nejprve nastaví do momentu působícího proti směru pohybu (na obrázku naznačeno červeně). Ax1 počká na dosažení momentu u Ax2 a následně se Ax1 nastaví do polohové vazby, respektive jede absolutně do požadované pozice (na obrázku naznačeno zeleně). Po dosažení požadovaných pozic se role obrátí, na obrázku naznačeno v dolní polovině jako „směr negative“. Při pohybu vpravo Ax1 působí momentem proti směru pohybu jako momentová brzda. Po dosažení momentu a tím udržení napnutí lana jede Ax2 absolutně do žádané pozice. Detailní popis pohybů, jejich zapnutí a všech vnitřních kontrol je popsán v sekci 5.3.

5.3 Algoritmus pro řízení

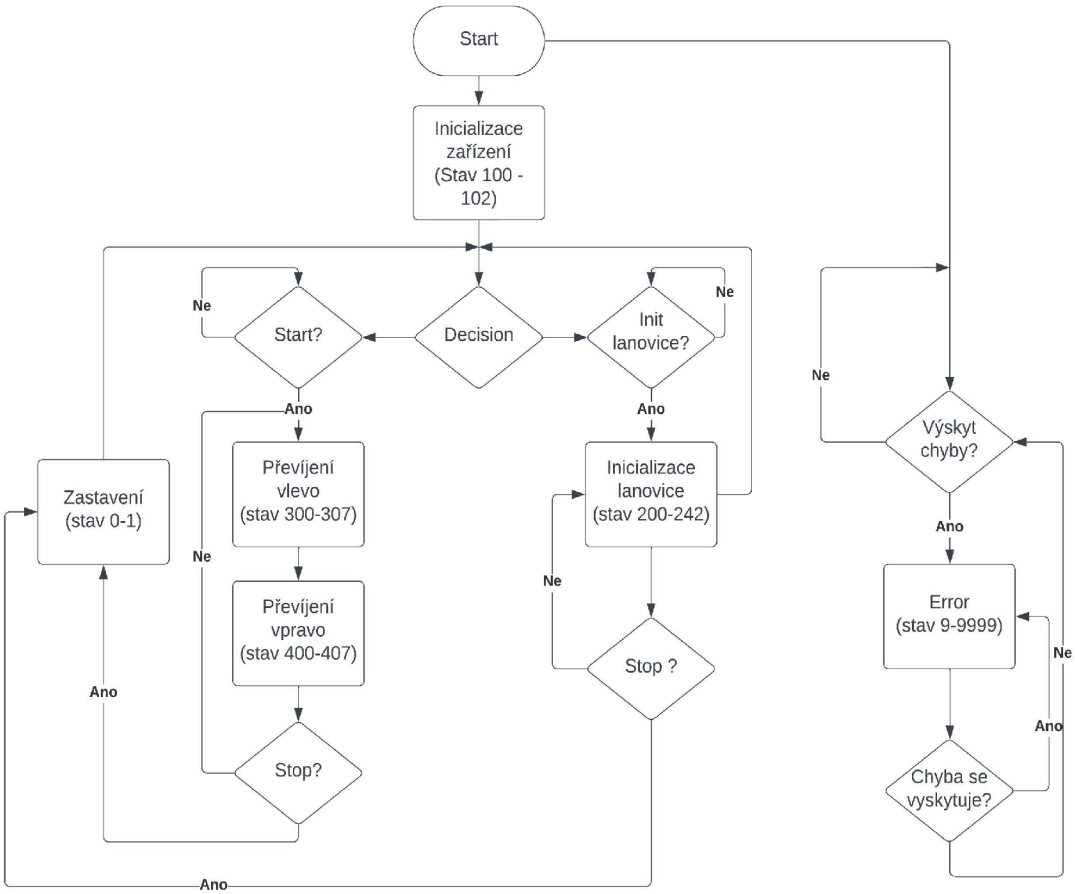
V této sekci popíšu především hlavní algoritmus stroje, ze kterého jsem následně vytvořil funkční blok. Přiblížím jednotlivé parametry, povely pro motory a kontroly jejich dosažení. Také zde popíši jednoduchý vývojový diagram.

Abych mohl zařízení dobře ovládat, zadávat mu parametry a sledovat jeho vnitřní stavy, potřebuji toto vše zavést jako proměnnou.

5.3.1 Vývojový diagram k algoritmu

Pro hlavní program jsem sestavil vývojový diagram, který lze nalézt na obrázku 5.3. Z vývojového diagramu lze vyčíst, jak zařízení funguje nebo jaký povel lze provést ze stavu po inicializaci. Z vývojového diagramu také vyčtem, kdy lze zařízení uvést do zastavení. Zastavení lze provést kdykoliv během inicializačního procesu, nebo také v rámci běhu programu v okamžiku, kdy je lano kompletně navinuto na pravou kladku. Samozřejmě požadavek na zastavení lze provést kdykoliv během chodu stroje, avšak v rámci výrobního procesu bude vykonán až při navinutí lana na pravou kladku.

Vývojový diagram popisuje základní stavbu programu, jeho stavy a příkazy, které lze provést pro změnu chodu stroje.



Obrázek 5.3: Vývojový diagram hlavního algoritmu

5.3.2 Proměnné

Zařízení potřebuje pro ovládání povely pro motory, které jsem si částečně převzal z `Basic` programu a doplnil k nim další. Ať už se jedná o povely pohybu, zapnutí napájení, či zastavení, jsou u obou motorů identické. Díky tomu bylo využito stejných datových typů, které jsem následně využil pro oba typy motorů.

Dalšími důležitými parametry jsou jak mechanické, tak i dynamické parametry. Mechanické parametry obsahují řadu vstupů, které se nastaví pouze jednou při prvním testování procesu a následně jsou pro další zapnutí stroje uloženy v paměti zařízení. Je nutné zmínit, že oba motory pracují se stejnými kinematickými parametry rychlostí, zrychlení a zpomalení. Stejně tak tomu je i v případě parametrů mechanických.

Při vytváření algoritmu jsem zavedl mnoho proměnných, všechny jsem následně popsal v kapitole funkčního bloku 8.3.

5.3.3 Algoritmus

Pro chod stroje jsem navrhl algoritmus popsaný ve vývojovém diagramu 5.3. Všechny stavy jsou očíslovány a jejich soupis je v tabulce 5.4. Číslování stavů je zvoleno, protože je při větším množství přehlednější a pro programátora jednodušší než jejich slovní popis.

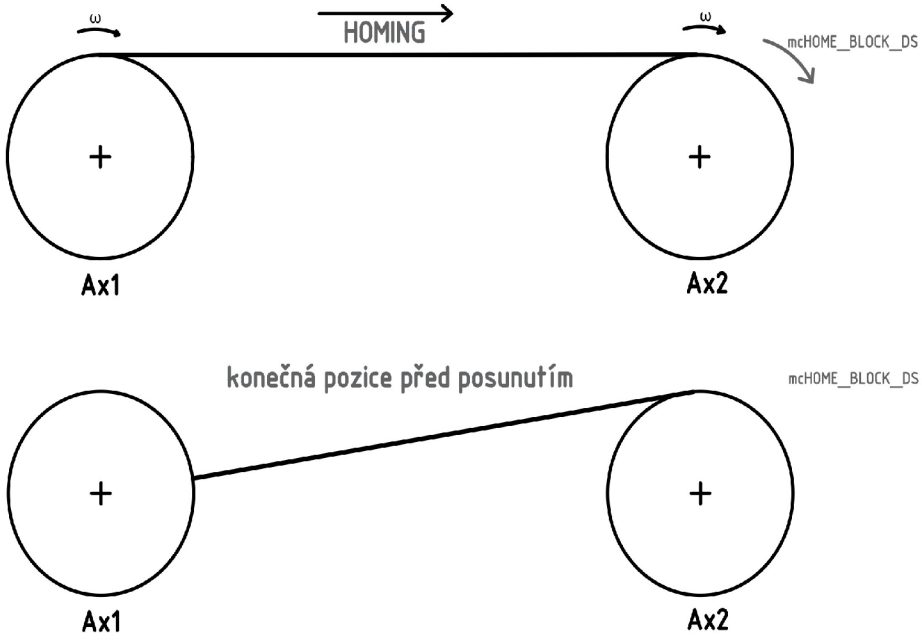
Po zapnutí napájení je zařízení v režimu výběru (`Vyber`), není vybrán žádný z výrobních režimů. Je tak provedeno z důvodů bezpečnosti po sepnutí zařízení, protože stav `Vyber` obsahuje nulové parametry zrychlení, zpomalení a rychlosti. Po otočení hlavním vypínačem se dostaneme do stavu `100`, kde dojde k ověření zablokování levého motoru. Pokud je levý motor zablokován, přecházíme do stavu `101`, kde dojde k ověření zablokování pravého motoru. Po těchto kontrolách dojdeme do stavu `102`, kde zařízení čeká na povel od uživatele. Jak lze vidět z vývojového diagramu, v tomto stavu může uživatel přejít do stavů `200` či `300`. Stav `200` značí inicializaci stroje, stav `300` značí spuštění stroje, které je však podmíněno předchozím nastavením parametrů a provedením inicializačního procesu. Bez provedení inicializačního procesu nelze zařízení spustit v žádném z dostupných režimů. Inicializační proces obsahuje svoje vlastní parametry, proto může být proveden bez předchozího výběru režimu, ale musí být zvolena rychlost inicializace a mechanické parametry.

Inicializace

Inicializačním procesem rozumíme stažení lana na pravou kladku a následné pootočení o zvolenou otáčku/otáčky nebo pouze o část otáčky zpět. Vytvoříme si tak rezervní část lana na levé kladce. Poté při provádění pohybu nebudeme nikdy dojíždět až na samotné upnutí lana ke kladce a nebude tak hrozit případné utržení při nepřesném polohování. Toto pootočení je vstupním parametrem pro zadání od uživatele.

Proces inicializace začíná přivedením signálu na napájení pravého motoru (stav `200`) a následnou kontrolu, zda se napájení provedlo. Pokud je pravý motor napájen, přecházíme do dalšího stavu `201`. Ve stavu `201` dáme povel pro vypnutí napájení

na levém motoru, je zde pouze pro případ, kdyby napájení bylo zapnuto. V tomto kroku provedeme na pravém motoru homing s mechanickým limitem. Tzn. že motor provede natočení lana na pravý motor a pokud se reálná pozice liší od pozice simulované, ve které by měl motor být, tato pozice je vyhodnocena jako pozice koncová. Koncové natažení lana po provedení homingu je viditelné na dolní části obrázku 5.4.



Obrázek 5.4: Homing s mechanickým limitem

Po spuštění tohoto pohybu čekáme až bude motor ve stavu `Standstill`, tzn. bude dokončen homing a tím bude motor zastaven. Poté budeme napájet i levý motor.

Napájení levého motoru znovu zkontrolujeme a ve stavu `203` provedeme pasivní homing. Pasivním homingem rozumíme nastavení aktuální pozice do stavu nula, bez pohybu motoru. Následně ve stavu `205` provedeme kontrolu pozice po pasivním homingu. Ověřím, zda se požadovaná pozice rovná pozici skutečné. Poté provedeme příkazy ve stejném sledu i na levém motoru. Uděláme pasivní homing a po jeho dokončení zkontrolujeme, zda je v požadované pozici. Stavem `210` začínáme sérii povelů, které vedou k posunu o požadovanou pozici. Naším cílem je vytvoření rezervy lana pomocí pootočení.

Ve stavu `211` nastavíme parametry na pravém motoru a nastavíme ho do momentové brzdy. Stavem `212`, ověříme zda pravý motor dosáhl požadovaného momentu a zároveň levý motor je ve stavu `Standstill`. Ověříme napájení levého motoru a následně provedeme pootočení vlevo o předem zadaný parametr. Levý motor jede absolutně na žádanou pozici, pravý motor táhne momentem proti a drží tak lano stá-

le napnuté. Následně ve stavu 220 zkontrolujeme, zda levý motor dosáhl požadované pozice. Pokud ano, provedeme pasivní homing a dostaneme tak obě osy do nulových výchozích pozic pro spuštění programu. Znovu ověříme dosažení pozic a následně se vracíme do stavu 102, který jak ukazuje vývojový diagram, je stavem výchozím. Tímto přechodem zpět do stavu 102 je proces inicializace lanové elektrody a tím vytvoření rezervního pootočení dokončen.

Spuštění

Pohyb stroje začínáme vždy pohybem vlevo, jelikož je inicializace nastavena tak, že celé lano (kromě rezervy vytvořené posunutím) je namotáno na pravé kladce. Stavem 300 začíná série povelů pro pohyb vlevo. Prvně nastavím pravý motor do momentové brzdy s předem definovanými parametry. Parametrem pro momentovou brzdu je uživatelem zadaný moment. Po dosažení požadovaného momentu na pravém motoru vyšlu příkaz do levého motoru na absolutní pohyb s předem zadanou pozicí, rychlostí, zrychlením a zpomalením. Po dosažení koncové pozice zkontroluji vypnutí příkazů na obou motorech a dosažení stavu `Standstill`. Pokud ano, pohyb vlevo je dokončen a provedeme stejné úkony pro směr převinutí lana na pravou kladku.

Převinutí lana vpravo začíná stavem 400. Nejprve nastavíme levý motor do momentové brzdy, aby držel během převíjení napnuté lano. V okamžiku, kdy dosáhne levý motor požadovaného momentu, pravý motor nastavíme do absolutního pohybu dle uživatelem nastavených parametrů. Následně čekáme, až motor dosáhne požadované pozice a dostane se do stavu `Standstill`. Nyní se nacházíme ve stavu rozhodování, pokud je v tento moment vyžadováno zastavení chodu, motory se zastaví a zařízení přejde do stavu 0. Pokud nevznikl požadavek na zastavení, zařízení přejde do stavu 300 a celý pohyb se cyklicky opakuje (tzn. převíjení lana vlevo a následně převíjení vpravo).

Zastavení

Tento stav je dovolen kdykoliv během inicializačního procesu, ale není dovolen kdykoliv během výrobního procesu. V rámci výrobního procesu je zastavení dovoleno pouze v době, kdy je lano namotáno na pravé kladce. Je to z důvodu možného opětovného zapnutí. Tím, že lano namotám vždy do stejné pozice, která je shodná s pozicí inicializace, dosáhnu možnosti opětovného zapnutí. Toto opětovné zapnutí je velmi důležité z hlediska úspory času. Uživatel nechce po každém zastavení provádět inicializaci lanové elektrody a to ani v případě, že ukončí výrobu a chce zařízení spustit s čisticím roztokem.

Stav zastavení obsahuje především vyvolání příkazu stop pro motory. Po provedení příkazu následuje čekání na klidový stav. V momentu, kdy jsou motory zastaveny a stroj je tak uveden do klidu, se vracíme do stavu 100. Zařízení následně po sekci úkonů popsaných v prvním odstavci přejde do stavu 102, kde čeká na další povely.

Chybový stav

Během chodu programu zařízení stále zkoumá, zda se nevyskytuje nějaká chyba. Chyb je na zařízení snímáno mnoho na synchronních motorech, ale také na lanové elektrodě. Pro správný běh programu odhaluji všechny možné zjistitelné chyby. Snímám, jestli nedošlo k chybě při výběru režimu nebo k prokluzu kladek a tím k povolení lana či jestli nedošlo k jeho přetržení. Přetržení lana pravděpodobně nebude problém na reálném stroji, ale v rámci laboratorního modelu k tomuto problému mohlo dojít. O těchto chybách a jejich snímání se dočtete v kapitole 7. V případě výskytu některé chyby zařízení okamžitě přechází do errorového stavu.

Errorové stavy začínají stavem `999` a konkrétně chybový stav je stav označen čísly `999,909,1999,2999`. Ve stavu `999` nejprve dojde k zastavení motorů, následně přejdeme do stavu `909`, kde se čeká až uživatel odstraní a poté potvrdí chybu pomocí tlačítka `ACK`. Pokud se chyba vyskytuje na lanové elektrodě, tzn. došlo k prokluzu kladek, přetržení lana či k chybě při výběru režimu, tak zařízení zůstává ve stavu `909`, dokud uživatel nepotvrdí chybu. Pokud se chyba vyskytuje na motorech, přejdeme do stavu `1999`. V tomto stavu zařízení setrvá, dokud pomocí tlačítka potvrzení (`ACK`) nedojde k odstranění této chyby na motorech. Chyb může na motorech nastat více najednou, proto je důležité také sledovat proměnnou `ErrorID`, která zobrazuje číslo chyby. Se sledováním čísla chyby zároveň musí dojít k opětovnému stisku tlačítka `ACK`, dokud nebudou všechny chyby odstraněny. Pokud se chyba již nevyskytuje, přejdeme do stavu `100`. V tomto stavu může zařízení setrvat a čekat na příkaz `STOP`. Tímto jsou všechny chyby na zařízení odstraněny a můžeme znovu zařízení provést procesem inicializace. Následně až po provedení procesu inicializace můžeme spustit výrobní proces pomocí tlačítka `RUN`.

5.4 Tabulka jednotlivých stavů zařízení

Každý stav zařízení má své identifikační číslo, jejich kompletní přehled je uveden v návodu k funkčnímu bloku. Stručný přehled číslování stavů je znázorněn v tabulce 5.4.

Tabulka 5.1: Tabulka jednotlivých stavů

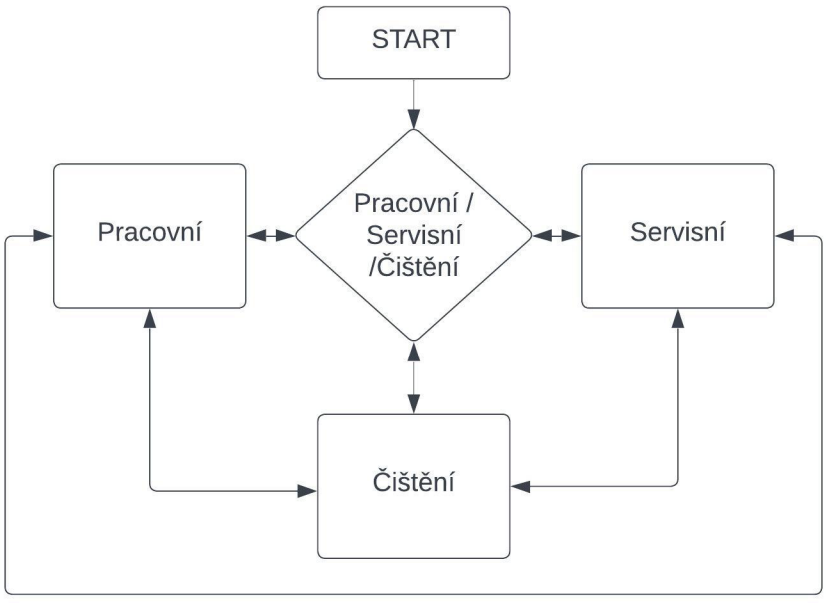
Číslo stavu	**Popis stavu**
100 – 102	inicializace bloku, stav čekání na povel
200 – 242	inicializace lanové elektrody
300 – 307	běh programu vlevo
400 – 407	běh programu vpravo
000 – 001	stav zastavení
999, 909, 1999, 2999	stav chyby

6 Režimy zařízení

Funkční blok (a také samotné zařízení) disponuje třemi dostupnými uživatelskými režimy, stejně jako je tomu v reálném provedení stroje. Jeden režim pro odladění parametrů pro výrobu, další režim pro běžný chod výroby a v poslední řadě režim, který nám zajistí odstranění zbytkového polymeru po ukončení výrobního procesu.

Všechny režimy dovolují stejné typy parametrů, jde pouze o tři sady parametrů. Z tohoto důvodu, jak je zobrazeno ve vývojovém diagramu 6.1, můžeme mezi režimy libovolně přepínat i během chodu programu. Vzhledem k tomu, že se při změně režimu jedná o změnu parametrů, která se vždy provede při volání dalšího povelu, je povolení přepnutí za běhu bezpečné. Sady parametrů pro každý režim jsou po nastavení uživatelem ukládány a díky tomu máme možnost s nimi znovu spustit výrobu nebo čištění i po delší odstávce stroje.

Výhodou přepnutí parametrů za chodu stroje je také to, že po ukončení výrobního procesu je ihned možné přejít na čisticí parametry. Pro uživatele to tak v praxi znamená změnu rychlosti procesu pomocí dvou stisknutí a následně může provést výměnu kádinky s polymerem za čisticí roztok.



Obrázek 6.1: Vývojový diagram režimů zařízení

Z obrázku 6.1 je viditelné, že po spuštění přechází zařízení do režimu výběru, režim je podle toho také pojmenován `Vyber`. V režimu výběru jsou automaticky v zařízení nastaveny nulové hodnoty a není tak možné stroj spustit. Proti spuštění s nulovými parametry jsem zavedl ošetření, které zamezuje přechod do dalšího kroku, pokud nejsou parametry větší než nula. Zároveň se tato hláška o nulových parametrech `Nulove parametry` zobrazuje ve výstupní proměnné text, která je popsána v sekci 8.3 v části výstupů.

U všech režimů lze nastavit odlišné limity zadávání hodnot. Učinil jsem tak z důvodu, že není vhodné nastavit stejný maximální limit pro čištění a pracovní režim, protože polymer nedisponuje zajisté stejnou viskozitou jako čisticí roztok.

6.1 Pracovní režim

Pro běžnou výrobní činnost jsem v zařízení naprogramoval pracovní režim, který dovoluje uživateli provádět běžný provoz pro výrobu nanovláken. Z hlediska provozu je mu dovoleno nastavovat řadu parametrů, pro dosažení ideálních podmínek při výrobě nanovláken. Parametry v režimu jsou zapamatovány, tzn. i po ukončení výroby mohou být opakovatelně využity.

6.2 Servisní režim

Před započetím výroby je nutné zjistit ideální mechanické a kinematické parametry. Pro dosažení kvalitní výroby je nutné nastavit vhodné rychlosti a zrychlení. To se může měnit při jiném typu polymeru či jiném lanu. Pro nalezení správných parametrů jsem zavedl servisní režim.

6.3 Čisticí režim

Pro opakovatelnost použití stoje je nutné provádět po každém výrobním procesu čištění elektrody od polymeru. Při nevyčištění stroje by mohly být jednotlivé výrobní komponenty trvale zaneseny nebo zničeny. Po výrobě je tedy nutné využít čisticí roztok a následně nechat elektrody pročistit.

Rozdílem tedy bude sada parametrů, kde budou především nižší rychlosti a zrychlení, protože čisticí roztok má jinou viskozitu než polymer a v případě stejné rychlosti by došlo k jeho nechtěnému úniku. Režim nedovoluje nastavení velkých výrobních rychlostí a zrychlení. Ovšem je těžké definovat, co je velká výrobní rychlost, pokud není předem známo všech využití funkčního bloku. Proto je nastavení maximálních a minimálních hodnot realizováno vstupním parametrem. Nastavování tohoto parametru lze naprogramovat tak, aby byl dostupný pouze uživateli s patřičným oprávněním. Více je popsáno v kapitole 8 v sekci 8.3.

7 Diagnostika lanové elektrody

Zařízení disponuje základními diagnostickými nástroji v podobě chybových hlášení, které jsou přeneseny ze synchronních motorů, konkrétně z programu `Basic`. Tyto chyby jsem následně ještě rozšířil o snímání dalších chyb, které mohou vzniknout na lanové elektrodě, viz sekce 7.2. Všechny chyby jsou definovány svým ID, neboli identifikačním číslem. Číslo je dále doprovázeno textem, který uživateli zobrazuje na displej popis vyskytujícího se problému. Identifikační ID těchto chyb jsou viditelná z tabulky 7.1.

7.1 Tabulka chyb na zařízení

Tabulka 7.1: Tabulka snímaných chyb na elektrodě s jednotnými idnetifikačními čísly

Typ chyby	**ID chyby**	**text chyby**
prokluz lanka vlevo	1	Prokluz kladky – vlevo
prokluz lanka vpravo	2	Prokluz kladky – vpravo
přetržení lanka vpravo	3	Pretrzeno lano – vlevo
přetržení lanka vpravo	4	Pretrzeno lano – vpravo
chyba výběr režimu	5	Chyba vyberu rezimu

Tabulku, která by popsala všechny chyby vznikající na motorech samotných, jsem nevytvářel. Všechny chyby týkající se pohonů a také toho, co je mohlo způsobit, lze nalézt v helpu Automation Studia. [3]

7.2 Chyby snímané na lanové elektrodě

7.2.1 Prokluz kladek

Pro udržení stálého napnutí lanka musím kontrolovat, zda nedošlo k prokluzu kladek. Prokluzem kladek by došlo k vytvoření větší mezery, tzn. k povolení lana nebo k posunu lana na některou ze stran. Posun lana by mohl později dělat problémy, pokud by se provedl několikrát.

Snímání této chyby je zajištěno pomocí snímání hodnoty absolutního rozdílu aktuálních pozic kladek. Pokud dojde ke zvětšení rozdílu nad nastavenou hodnotu

odchylky, je vyhodnocen prokluz kladek. Odchylka je zavedena v celých otáčkách. Tato nastavená odchylka např. pro hodnotu 0,5 odpovídá 1/2 otáčky. Prokluz kladek může nastat při nesprávném dojetí motoru, který navíjí lano či v případě, že dojde k mechanickému zásahu do lana. Mechanickým zásahem do lana rozumíme mechanické vytažení většího kusu lana mezi kladkami a tím vzniku větší mezery.

Při snímání prokluzu lana mohu i snímat, jestli k prokluzu došlo při pohybu navíjení lana na levou či pravou kladku. Díky tomu je v tabulce 7.1 rozeznán i směr prokluzu vlevo/vpravo. Rozeznání směru prokluzu jsem provedl pomocí snímání aktuálního stavu zařízení. Vím, v kterém konkrétním číselném stavu zařízení je, tudíž mohu rozpoznat, zda se zařízení pohybuje v daný moment vpravo či vlevo. Chyba je doplněna hláškou v proměnné `text`, kde se podle příslušného směru zobrazí „Prokluz kladky – vlevo“ nebo „Prokluz kladky – vpravo“.

Při tomto snímání prokluzu také sleduji směr pohybu jednotlivých motorů, je to důležité pro odlišení prokluzu a přetržení lana. Více v subsekci 7.2.2.

K prokluzu kladek může dojít např. nesprávným dotažením kladky upnuté přímo na pohon, což se mi přihodilo při testování na laboratorním modelu. Odstranění chyby je jednoduché. Musíme odhalit mechanickou chybu, tu následně odstranit a poté pomocí potvrzovacího příkazu chyby (`ACK`) potvrdit chybu v systému. Poté zařízení přejde postupně z chybového stavu až do stavu `102`, kde ho nebude možné znovu spustit předtím, než bude provedena inicializace lanové elektrody.

7.2.2 Přetržení lanka

Další možností, která se za určitých podmínek může přihodit i na reálném stroji, je přetržení lana. Jak jsem se zmínil v sekci prokluzu kladek, přetržení lana je od prokluzu kladek rozpoznáno pomocí sledování směru otáčení jednotlivých motorů. Pokud dojde k přetržení lana, znamená to, že motor, který tvořil momentový protipohyb, se po přetržení točí opačným směrem, než motor, který jel absolutně na požadovanou pozici. Zároveň je pro rozeznání změny pozic sledován, stejně jako u prokluzu kladek, absolutní rozdíl aktuální pozic jednotlivých kladek. Pokud je tento rozdíl větší než nastavená hodnota odchylky otáčky, tak je vyhodnocen jako prokluz/přetržení. Poté je rozhodnuto, zda došlo k prokluzu nebo přetržení právě na základě snímání směrů otáčení jednotlivých motorů. Také, stejně jako u prokluzu kladek, snímám, při kterém směru pohybu (vpravo/vlevo) k přetržení došlo. V proměnné text se objeví hláška „Pretrzeno lano – vlevo“ nebo „Pretrzeno lano – vpravo“. Výrazy vlevo a vpravo značí směr pohybu navíjení lana na kladku, při kterém k přetržení došlo.

Pokud by se lano přetrhlo, musí dojít k odstranění chyby fyzicky, tzn. výměnou lana. Následně v systému stisknutím příkazu potvrzení chyby (`ACK`) bude chyba odstraněna i v systému. Poté musí dojít znovu k inicializačnímu procesu lanové elektrody pomocí příkazu `INIT`.

7.2.3 Chyba při výběru režimu

Změna režimu je realizována pomocí proměnné typu `INT`. Chyba při výběru režimu může nastat při špatném zvolení hodnoty proměnné. Tato chyba může nastat prakticky jen při tvorbě programu s využitím funkčního bloku, pokud je blok správně použit, chyba později již nemůže vzniknout. Vznik chyby je doprovázen textovou hláškou zobrazenou v proměnné `text` s textem „Chyba vyberu rezimu“.

7.3 Chyby snímané na motorech

Samotné pohony mají diagnostiku pro širokou škálu chyb. Jejich konkrétní významy dle chybových identifikačních čísel lze nalézt v helpu Automation Studia. [3]

Texty chyb snímaných na motorech se propisují do jednoho hlavního textu ve funkčním bloku. Stejně tak i identifikační čísla chyb vzniklých na motorech se propisují do jedné hlavní proměnné `ErrorID` ve funkčním bloku. Díky tomu má uživatel možnost pomocí dvou proměnných sledovat stav celé lanové elektrody.

8 Funkční blok

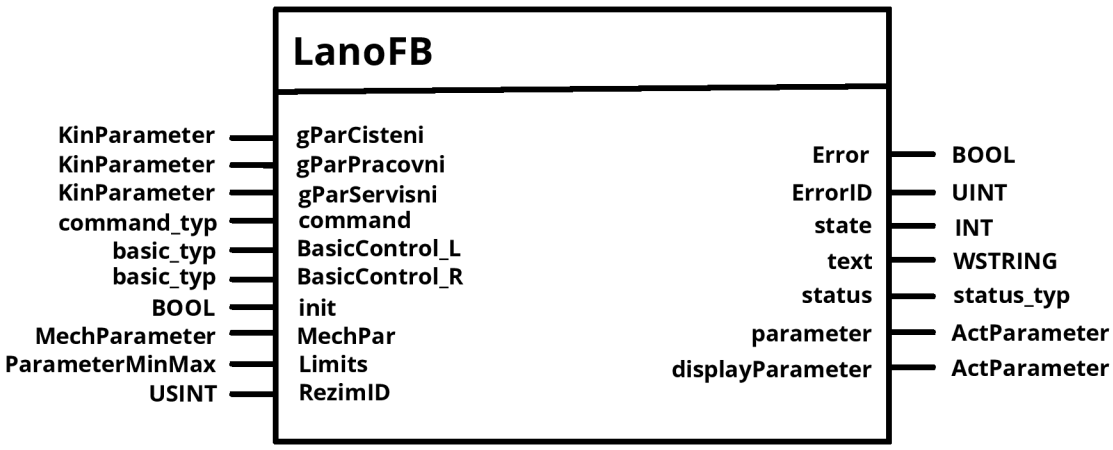
Po otestování celého algoritmu pro řízení lanové elektrody a všech diagnostických nástrojů jsem dospěl k závěru, že je algoritmus připraven pro tvorbu funkčního bloku. Rozhodl jsem se zamyslet se nad počtem vstupních, vnitřních a výstupních proměnných nutných pro řízení stroje a také zadávání povelů a parametrů pro pohyby. Přehled všech vstupních a výstupních proměnných jsem uvedl v sekci 8.3.

Tvorba funkčního bloku je v Automation Studiu poměrně uživatelsky přívětivá. Stačí si přidat funkční blok jako programovou sekci, vhodně ho pojmenovat, nastavit všechny proměnné dle požadovaného typu, vytvořit interní program a blok zavolat. Po vytvoření bloku jsem se rozhodl pro jeho testování, které je popsáno v kapitole 9.

8.1 Návrh funkčního bloku

Navržený algoritmus popsaný v sekci 5.3 jsem využil k tvorbě funkčního bloku. Algoritmus jsem upravil tak, aby mohl zpracovávat vstupní hodnoty, uvnitř předávat jiné a vytvářet statusové hodnoty. Stačilo provést malé úpravy, dle zvolených vstupů a výstupů provést přejmenování jednotlivých proměnných a přidat část pro inicializaci.

Funkční blok si můžeme představit následovně, viz obrázek 8.1. Funkční blok obsahuje 10 vstupních proměnných pro volbu parametrů, zadávání povelů a 7 výstupních proměnných pro sledování stavů zařízení a ovládání jednotlivých motorů.



Obrázek 8.1: Funkční blok – LanoFB

Jednotlivé proměnné a jejich význam je popsán v sekci 8.3, kde jsou také zobrazeny datové typy a jednotky k parametrům.

8.2 Předpoklady

Snažil jsem se udělat funkční blok co nejvíce uživatelsky nastavitelný, bohužel ale nastavení počtu jednotek na otáčku se nedá realizovat pomocí proměnné, ale pouze pomocí inicializační tabulky příslušného pohonu. Proto bylo nutné nastavení zvolit jednotně s ohledem na aktuální nastavení na reálném stroji a to z důvodu co největšího přiblížení se k reálnému zařízení a také s předpokladem, že funkční blok se bude na tomto či podobném zařízení také využívat.
Funkční blok předpokládá nastavení 1000 jednotek na jednu otáčku u každého z motorů, který bude funkční blok využívat. Tuto věc je nutné u každého zařízení zohlednit.

Při programování je nutné motory nastavit jako lineární osy.

8.3 Vstupní a výstupní parametry

Tabulka vstupních parametrů zobrazuje kompletní přehled vstupů a jejich požadovaný datový typ. Bližší specifikace vstupů a jejich datových typů jsou popsané níže pod tabulkou.

Tabulka 8.1: Tabulka vstupních parametrů funkčního bloku

parametr	datový typ	popis parametru
gParCisteni	KinParameter	sada parametrů pro čistící režim
gParServisni	KinParameter	sada parametrů pro servisní režim
gParPracovni	KinParameter	sada parametrů pro pracovní režim
command	command_typ	sada příkazů pro ovládání lanové elektrody
BasicControl_L	basic_typ	struktura pro ovládání levého motoru
BasicControl_R	basic_typ	struktura pro ovládání pravého motoru
init	BOOL	proměnná pro inicializaci CPU
MechPar	MechParameter	sada parametrů pro mechanické parametry
Limits	ParameterMinMax	sada parametrů pro nastavení limitů
RezimID	USINT	proměnná pro hodnotu při volbě režimu

8.3.1 Kinematické parametry

Sady parametrů pro jednotlivé režimy, tedy `gParCisteni`, `gParServisni` a `gParPracovni` využívají datového typu KinParameter. Tento datový typ se skládá z třech proměnných typu `REAL`. Jedná se o základní kinematické parametry sloužící pro pohyb motorů. Všechny tyto tři sady parametrů jsou nastaveny jako retainové proměnné, tzn. jsou zapamatovány i po vypnutí zařízení. Datový typ vypadá následovně:

```
KinParameter :  STRUCT
  zrychleni : REAL; (*zadání parametru zrychlení [m/min^2]*)
  zpomaleni : REAL; (*zadání parametru zpomalení [m/min^2]*)
  rychlost : REAL; (*rychlost pohybu [m/min]*)
END_STRUCT;
```

8.3.2 Příkazy pro ovládání lanové elektrody

Struktura pro povely `command` je datového typu `command_typ`. Tento datový typ v sobě obsahuje povely pro zastavení stroje, spuštění stroje, inicializaci stroje a potvrzení v případě vzniku chyby. Pomocí těchto příkazů lze zařízení ovládat. Tudíž tyto příkazy jsou uvnitř funkčního bloku využity k přepínání mezi jednotlivými stavy.

Pro využití funkčního bloku je tak nutné zapínání a vypínání těchto proměnných provést mimo funkční blok. Jak se k tomuto dá přistupovat jsem popsal v kapitole 9.

```
command_typ :  STRUCT  (*příkazy *)
  Stop : BOOL; (*Požadavek pro zastavení stroje*)
  Run : BOOL; (*Požadavek pro spuštění procesu*)
```

```
  Ack : BOOL; (*Požadavek pro potvrzení chyby*)
  Init : BOOL; (*Požadavek pro provedení inicializačního procesu*)
END_STRUCT;
```

8.3.3 Struktury pro ovládání motorů

Vstupní proměnná `BasicControl_L` a `BasicControl_R`, která je jako ukazatel na globální proměnnou, je typu `basic_typ`. Tento datový typ se skládá z dalších čtyř datových typů sloužících pro statusy, příkazy, parametry a stavy motorů. Datový typ `basic_typ` vypadá následovně:

```
basic_typ :  STRUCT  (*control structure*)
  Command : basic_command_typ; (*command structure*)
  Parameter : basic_parameter_typ; (*parameter structure*)
  Status : basic_status_typ; (*status structure*)
  AxisState : basic_axisState_typ; (*axis state structure*)
END_STRUCT;
```

Proměnné `BasicControl_L` a `BasicControl_R` jsou namapovány jako hlavní ovládací proměnné pro motory. Jak vidíme výše, obsahují zadávání příkazů, parametrů, statusů a stavů jednotlivých os.

Bližší specifikace datových typů viz. `Basic` program[3]. Tyto datové typy je nutné doplnit o následující: Datový typ command_typ musí být doplněn o:

```
MoveTorque : BOOL;
```

Datový typ parameter_typ musí být doplněn o:

```
TorqueRamp : REAL;
Torque : REAL;
```

Datový typ status_typ musí být doplněn o:

```
InPos : BOOL; (*TRUE if target pos reached*)
InTorque : BOOL; (*TRUE if set torque is reached*)
```

Datový typ basic_axisState_typ musí být doplněn o:

```
TorqueMotion : BOOL;
```

8.3.4 Inicializační proměnná

Každý algoritmus je po startu CPU je nutné nastavit do počátečního stavu. Tato proměnná slouží k provedení počáteční inicializace CPU. Při inicializaci CPU se proměnná nastaví na true. Následně po vstupu do funkčního bloku provede inicializační proces bloku. Nejedná se však o inicializaci lanové elektrody, ale pouze o nastavení počátečních parametrů ve funkčním bloku, např funkční blok je nastaven do stavu `100` a režim zařízení je nastaven do `Výběr`.

Proměnná je datového typu `INT`.

8.3.5 Mechanické parametry

Sada mechanických parametrů slouží k nastavení parametrů pro koncové polohy, průměr kladky, rychlost inicializace, maximální moment a zadání parametrů pootočení při inicializaci tak, aby došlo k vytvoření rezervy na laně. Jednotlivé parametry jsou různých datových typů, proto jsem zde vložil jejich deklaraci. Všechny mechanické parametry jsou zapamatovány i po vypnutí stroje a to z důvodu lepšího komfortu uživatele.

```
MechParameter :  STRUCT
    cilovaPoziceL : REAL;(*cílová pozice levé kladky [xxxx]*)
    cilovaPoziceR : REAL;(*cílová pozice pravé kladky [xxxx]*)
    Mmax : REAL; (*maximální tah v lanku [Nm]*)
    d : REAL; (*průměr kladky [m]*)
    rychlostInicializace : REAL; (*rych. pootočení při inic. procesu*)
    pootoceni : REAL; (*zadání parametru pootoceni v init stavu*)
    odchylka : REAL; (*zadání parametru dovolené odchylky*)
  END_STRUCT;
```

CilovaPoziceL, CilovaPoziceR:
Parametry cílových pozic zařízení se zadávají v jednotkách otáček, kde jednu otáčku počítáme jako 1000 jednotek. Tzn. že pokud chceme například zadat otočení o dvanáct otáček, musíme zadat hodnotu 12000. Na pravém motoru nastavujeme proměnnou `cilovaPoziceR` na nulovou hodnotu, pokud tedy nechceme po prvním převinu lana na levou kladku při převíjení zpět končit v jiném místě. Pokud chceme končit v jiném místě, musíme nastavit hodnotu menší než nula. Hodnota větší než nula by způsobila přetržení lana či zničení stroje. Nastavení nuly je vhodné pro využití celého rozsahu lana, protože při inicializaci dochází k pasivnímu homingu a oba motory jsou po inicializaci v poloze nula. Tyto hodnoty nejsou ve funkčním bloku nijak omezeny minimální či maximální hodnotou, omezení je na programátorovi vizualizace.
Mmax:
Maximální tah v lanku zadáváme pomocí hodnoty této proměnné. Jednotkami jsou maximálního momentu jsou [Nm]. Tuto hodnotu nastavujeme buď dle limitu lana, či po odzkoušení dle vhodné hodnoty pro udržení požadovaného tahu v lanu.
r:
Průměr kladky, jak je viditelné z části deklarační, zadáváme v metrech. Průměr slouží především k přepočtu rychlostí a zrychlení z jednotek otáčení motoru na metry za sekundu.
RychlostInicializace:
Při posunu u inicializačního procesu je vhodné volit rychlost kvůli proměnlivým parametrům. Každý stroj může mít jiné lano, jinou délku lana, jiné průměry kladek či chceme lanové elektrody při inicializaci napínat rukou. Proto je vhodné zavést tuto rychlost jako parametr.

Rychlost inicializace zadáváme stejně jako rychlost pro výrobní proces v metrech za minutu.

Pootočení:
Parametr pootočení u inicializačního procesu znamená vytvoření si rezervní mezery na laně. Pootočením dosáhneme toho, že lano nebude dojíždět na místa upnutí a nedojde tak při malé odchylce k případnému poškození lana.

Parametr pootočení zadáváme stejně jako cílové pozice v jednotkách, kde 1000 jednotek značí jednu celou otáčku.

odchylka:
Pro kontrolu prokluzu a přetržení lana je potřeba nastavit hodnotu, u které řekneme, že jsme se už pootočili o příliš velký kus lana a došlo tak k prokluzu/přetržení. Tuto hodnotu jsem zavedl jako vstupní parametr, který je součástí mechanických parametrů. Je to z důvodu, protože každý stroj má jiný průměr kladky a na malé kladce je půl otáčky o hodně menší vzdálenost než na kladce velké. Proměnná je datového typu `REAL` a je zadávána v jednotkách celých otáček. Tzn. že číslo 0.5 znamená dovolení odchylky otočení o půl otáčky.

Všechny tyto parametry byly při tvorbě funkčního bloku ošetřeny pouze v rámci vizualizace. Tzn. že jejich maximální a minimální hodnoty jsou při mém využití funkčního bloku ošetřeny pouze ve vizualizaci. Ošetření těchto parametrů musí být při využití funkčního bloku provedeno programátorem ve vizualizaci.

Programátor, který bude blok využívat, musí tyto parametry omezit nějakými maximy a minimy. Pokud by tak neučinil, mohl by budoucí uživatel zařízení zadávat i nevhodné či nesmyslné hodnoty.

8.3.6 Limitní hodnoty pro zadávání kinematických parametrů

Každá vstupní hodnota potřebuje vždy nastavení rozmezí hodnot maxima a minima. Tyto limity jsou využity pro nastavení rozmezí zadávání hodnot u rychlostí, zrychlení a zpomalení. U nastavení mechanických parametrů jsem nechal omezení limitů pouze na programátorovi, protože se často jedná o parametry, které se nastaví jednou při tvorbě algoritmu stroje. Ovšem nastavení rozmezí kinematických parametrů jsem zavedl jako proměnné. Tyto proměnné mohou být následně skryty a dostupné pouze pro uživatele s odpovídajícím přístupem, či pouze pro programátora. Mohou ale také třeba být dostupné pro správce zařízení, ale ne pro uživatele. Tyto limitní hodnoty jsou nastaveny jako retainové proměnné, tzn. jsou zapamatovány i po vypnutí stroje.

Nastavení těchto limitních hodnot může být např. provedeno jako Min a Max Datapoint u jednotlivých zadávání rychlostí, zrychlení a zpomalení.

`ParameterMinMax` jsou hodnoty datového typu `REAL` a jejich deklarace vypadá následovně:

```
ParameterMinMax :  STRUCT
  zrychleniMAX : REAL; (*zadání MAX parametru zrychlení [m/min]*)
  zpomaleniMAX : REAL; (*zadání MAX parametru zpomalení [m/min]*)
  rychlostMAX : REAL; (*zadání MAX rychlosti pohybu [m/min]*)
  zrychleniMIN : REAL; (*zadání MIN parametru zrychlení [m/min]*)
  zpomaleniMIN : REAL; (*zadání MIN parametru zpomalení [m/min]*)
  rychlostMIN : REAL; (*zadání MIN rychlosti pohybu [m/min]*)
END_STRUCT;
```

8.3.7 Proměnná pro výběr režimu

Jedná se o proměnnou datového typu `USINT`. Tato proměnná je určena pro volbu režimů. Pod číslem nula najdeme režim nazvaný jako `Vyber`. Pod číslem jedna se nachází `pracovni` režim. Pod hodnotou dvě je `servisni` režim a pod hodnotou tři se nachází režim `cisteni`.

Tabulka 8.2: Tabulka výstupních parametrů funknkčního bloku

parametr	**datový typ**	**popis parametru**
status	status_typ	sada statusů lanové elektrody
Error	BOOL	výskyt chyby
state	INT	aktuální stav zařízení
text	WSTRING[79]_typ	výstupní text pro uživatele
ErrorID	UINT	identifikátor chyby dle čísla
parameter	ActParameter	sada aktuálních použitých parametrů
displayParameter	ActParameter	sada aktuálních parametrů pro zobrazení

8.3.8 Status

Status je základní výstupní proměnná, která nám udává informace o aktuálních stavech lanové elektrody. Proměnná je datového typu status_typ, který obsahuje 6 proměnných, které se dále ještě dělí dle datových typů.

```
status_typ :  STRUCT
  error : error_typ; (*chybové hlášení*)
  rychlost : speed; (*rychlosti jednotlivých motorů*)
  teplota : temp;(*teploty motorů*)
  errorAx : axErr;(*chyby vyskytující se na motorech*)
  actPos : actualPosition;(*aktuální pozice motorů*)
  rezim : REAL; (*aktuálně zvolený režim*)
END_STRUCT;
```

error:
Proměnná error obsahuje datový typ error_typ, který dává uživateli kompletní přehled o chybách vyskytujících se na lanové elektrodě a na motorech samotných. Tzn. jedná se o proměnné typu `BOOL`, které vypovídají o výskytu jednotlivých chyb. Deklarace proměnných vypadá následovně:

```
error_typ :  STRUCT  (*jednotlivé chyby*)
  pretrzeni_lana : BOOL; (*přetržení elektrody / lana*)
  prokluz_kladky : BOOL; (*prokluz kladky*)
  chyba_rezim : BOOL; (*nedefinovaný stav při výběru režimů*)
  ax2 : BOOL; (*motor LEFT*)
  ax1 : BOOL; (*motor RIGHT*)
END_STRUCT;
```

rychlost:
Proměnná rychlost je tvořena datovým typem `speed`, který do sebe překopírovává jednotlivé rychlosti motorů a přepočítává je na metry za minutu. Proměnné rychlostí motorů jsou typu `REAL`.

Deklarace datového typu vypadá takto:

```
speed :  STRUCT
  ax2 : REAL; (*motor LEFT*)
  ax1 : REAL; (*motor RIGHT*)
END_STRUCT;
```

teplota:
Snímání teploty motorů je zajištěno pomocí proměnných uložených v datovém typu `temp`. Tento datový typ obsahuje dvě proměnné typu `REAL` pro měření teplot na jednotlivých motorech. Datový typ vypadá následovně:

```
temp :  STRUCT
  ax2 : REAL; (*motor LEFT*)
  ax1 : REAL; (*motor RIGHT*)
END_STRUCT;
```

errorAx:
Chyby jednotlivých os jsou uloženy v proměnné datového typu `axErr`. Proměnné pro snímání identifikačních čísel errorů na jednotlivých osách jsou typu `UINT`.

```
axErr :  STRUCT
  ax2 : UINT; (*motor LEFT*)
  ax1 : UINT; (*motor RIGHT*)
END_STRUCT;
```

actPos:
Tato proměnná pro snímání aktuálních pozic motorů je typu `actualPosition`. Tento datový typ obsahuje aktuální pozice jednotlivých motorů, jejichž proměnné jsou typu `REAL`.

```
actualPosition :  STRUCT
  ax2 : REAL; (*motor LEFT*)
  ax1 : REAL; (*motor RIGHT*)
END_STRUCT;
```

rezim:
Proměnná slouží k výpisu aktulního zvoleného režimu. Proměnná je typu `REAL`.

8.3.9 Error

Výstupní proměnná `error` je typu `BOOL` a označuje výskyt chyby na zařízení. Do této proměnné se propisují všechny chyby, které se na zařízení mohou vyskytnout. Její hodnota je tak hlavním ukazatelem toho, že se na zařízení žádná chyba nevyskytuje. Při výskytu chyby je tato proměnná nastavena na hodnotu `true`.

8.3.10 state

Výstupní proměnná `state` je důležitá z hlediska výpisu aktuálního stavu zařízení. Může být využita například pro zobrazení aktuálního stavu zařízení na displej či ke kontrole funkčnosti při tvorbě programu s využitím funkčního bloku.

8.3.11 text

V této výstupní proměnné je uživateli zobrazováno textové upozornění. Upozornění se mohou týkat chyb, které se na zařízení vyskytnou či upozornění na nulové parametry. Při výskytu chyby je do tohoto textu vypsán text příslušné chyby. Ať už se jedná o chybu na lanové elektrodě či přímo na některém z pohonů, text chyby se zobrazí v této proměnné při výskytu kterékoliv chyby.

Další možností výskytu obsahu textu v této proměnné je při volbě kinematických parametrů. Pokud uživatel vybere režim, v kterém má nastaveny nulové parametry, bude v tomto textu upozorněn hláškou „Nulove parametry“. Stroj v tomto případě nelze spustit, protože nulové parametry znamenají, že jedna z hodnot rychlosti, zrychlení nebo zpomalení, je nulová.

8.3.12 ErrorID

Tato proměnná slouží k výpisu identifikačního čísla vyskytující se chyby. Pokud je proměnná rovna nule, zařízení funguje bez chyby a výstupní proměnná `Error` bude také rovna nule. Pokud se na zařízení vyskytne chyba, v proměnné se objeví číslo této chyby a lze ji dle čísla dohledat. Proměnná je datového typu `UINT`. Více o těchto chybách je popsáno v kapitole 7.

8.3.13 parameter

Výstupní proměnná `parameter` je datového typu `parameter_typ`. Tento datový typ v sobě obsahuje aktuálně použité kinematické proměnné, se kterými zařízení právě provádí úkony. Kinematické proměnné jsou do této struktury překopírovány podle aktuálně vybraného režimu. Datový typ se skládá z rychlosti, zrychlení a zpomalení, všechny proměnné jsou typu `REAL`. Všechny proměnné skrývající se pod proměnnou `parameter` jsou v jednotkách vstupujících do motorů, tedy v units/s.

Datový typ je deklarován následovně:

```
ActParameter :  STRUCT  (*aktuální parametry*)
  zrychleni : REAL; (*zadání parametru zrychlení [m/min^2]*)
  zpomaleni : REAL;(*zadání parametru zpomalení [m/min^2]*)
  rychlost : REAL; (*rychlost pohybu [m/min]*)
END_STRUCT;
```

8.3.14 displayParameter

Tato datová struktura slouží ke zobrazení aktuálních parametrů uživateli na uživatelský panel. Proměnná je stejného datového typu jako výstupní proměnná `parameter`. Rozdíl mezi těmito proměnnými je pouze v tom, že všechny proměnné v `displayParameter` jsou přepočteny pro lepší zobrazení uživateli. Proměnné jsou tedy převedeny do m/min či m/min^2.

8.4 Využití algoritmu na zařízení s více lanovými elektrodami

Před započetím výroby je nutné na reálném stroji rozjet všechny lanové elektrody v požadované sekvenci. Tento úkol nebyl předmětem tvorby funkčního bloku a je již na programátorovi konkrétního zařízení.

8.5 Spuštění a ovládání FB

Ovládání funkčního bloku je možné pomocí sady povelů, které jsou obsaženy v proměnné `command`. Na zařízení musí být nejprve vyplněny mechanické parametry. V programu funkčního bloku je podmínka, že pokud nebudou zvoleny parametry pro zadání průměru kladky, rychlosti pootočení a maximálního momentu, tak nebude možné inicializační proces lanové elektrody spustit. Po jejich vyplnění může být vyvolán příkaz `INIT`, při kterém bude provedena inicializace a zařízení tak napne lano a provede pootočení, aby namotalo rezervní část lana na levou kladku.

Po provedení inicializace je možné zařízení spustit do výrobního režimu pomocí proměnné `RUN`. Před vyvoláním je nutné mít zvolen režim, jinak se v proměnné `text` zobrazí nápis „Nulove parametry“. Samozřejmě v režimu musí být navoleny příslušné parametry, ty se volí v proměnné `gParPracovni` a dalších dvou dle zvoleného režimu. Před spuštěním do výrobního režimu musí být také zadány koncové polohy kladek. O nastavení všech potřebných parametrů se zmiňuji v sekci 8.3.

Pro zastavení výroby lze využít příkazu `STOP`, které dokáže zařízení zastavit v případě inicializačního procesu kdykoli ihned po stisku tlačítka. Požadavek pro zastavení lze vyvolat i v rámci výrobního procesu, avšak zařízení bude zastaveno až v okamžiku, kdy je lano namotáno na levé kladce.

V případě výskytu chyby na lanové elektrodě je nutné provést její odstranění a následně chybu potvrdit příkazem `ACK`. Po potvrzení chyby bude zařízení vyžadovat ještě provedení zastavení pomocí příkazu `STOP`. Poté bude vráceno do stavu, kde je možné jej znovu inicializovat a následně spustit výrobní proces.

Zadávání parametrů, jejich datové typy a jednotky jsou popsány v sekci s parametry 8.3. Nebudou zde proto znovu popisovány.

9 Testování funkčního bloku

Pro otestování funkčního bloku jsem využil dostupného dotykového displeje pro vytvoření uživatelského rozhraní. Na tomto rozhraní jsem vytvořil několik velmi graficky jednoduchých základních obrazovek, abych otestoval všechny vstupní a výstupní parametry funkčního bloku.

Pro ovládání jsem na zařízení vytvořil jednoduchý stavový automat, který pracuje s ovládáním vstupních příkazů pro funkční blok.

Pomocí dotykového displeje jsem otestoval funkční parametry celého zařízení. Bylo důležité vyzkoušet chod zařízení ve všech režimech, uložení nastavených parametrů a jejich změny za chodu programu.

Pro zjištění stavů uvnitř zařízení jsem vytvořil i obrazovku, která zobrazuje chyby na zařízení a aktuální polohy motorů. Následně jsem otestoval zaznamenávání chyb a jejich odstranění.

9.1 Mapování proměnných

Pro práci s funkčním blokem bylo nejprve nutné přiřadit v hlavním programu proměnné pro funkční blok, viz tabulky 9.1 a 9.2.

Tabulka 9.1: Tabulka přiřazení vstupních parametrů pro FB

vstupní parametr	**proměnná**
gParCisteni	EX_gParCisteni
gParServisni	EX_gParServisni
gParPracovni	EX_gParPracovni
command	Motion.command
BasicControl_L	BasicControlL
BasicControl_R	BasicControlR
init	inicializace
MechPar	EX_MechPar
Limits	EX_Limits
RezimID	EX_RezimID

Tabulka 9.2: Tabulka přiřazení výstupních parametrů pro FB

výstupní parametr	**proměnná**
Error	EX_Error
ErrorID	EX_ErrorID
state	EX_state
text	EX_text
status	Motion.status
parameter	Motion.parameter
displayParameter	displayPar

9.2 Tvorba jednoduché vizualizace

Pro jednoduchost jsem vytvořil šest základních obrazovek, které mi umožnily vyzkoušet a zdokumentovat všechny funkcionality funkčního bloku.

9.2.1 Hlavní obrazovka

Hlavní obrazovka je důležitá z hlediska zobrazování všech aktuálních nastavených parametrů. Vlevo nahoře na obrázku 9.1 vidíme zobrazení aktuálního stavu zařízení. Všechny stavy zařízení mají své jednotné číslo, což je velmi výhodné pro přehled uživatele či programátora. Více informací o jednotlivých stavech je poskytnuto v návodu k funkčnímu bloku, který jsem k zařízení vytvořil a je přílohou práce. Tabulku popisující jednotlivé stavy naleznete také zde 5.4.

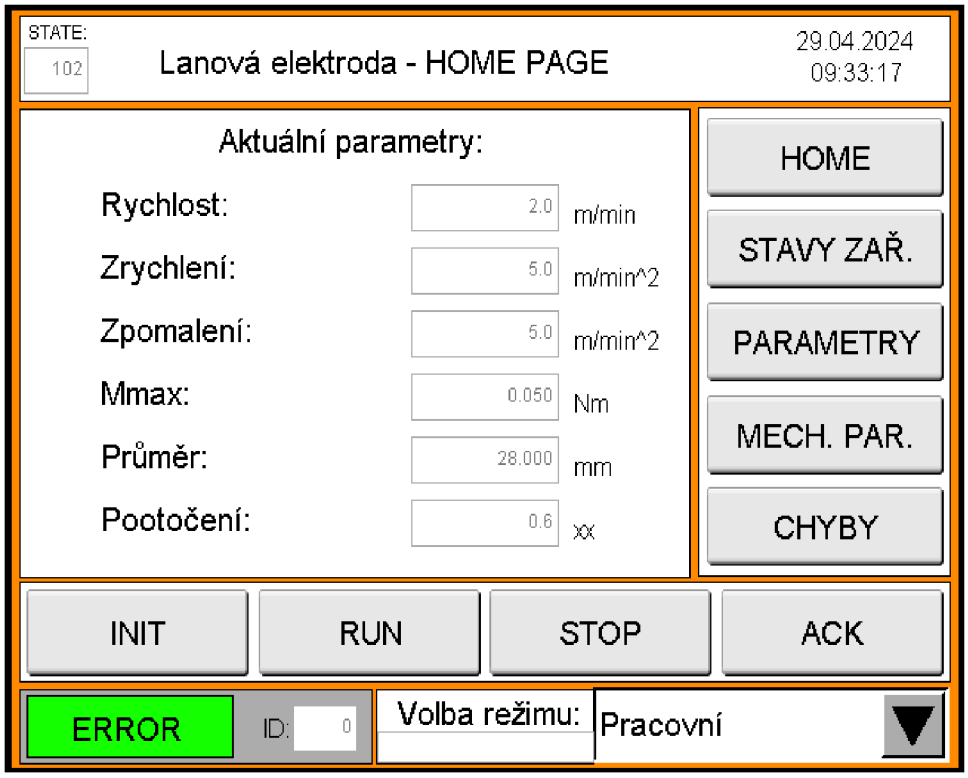
Na hlavní stránce uprostřed vidíme sadu aktuálních zvolených parametrů v zařízení. Jsou zde vidět hlavní mechanické parametry typu maximální moment, průměr kladky a pootočení při inicializaci. Po pravé straně vidíme jednotlivá tlačítka pro přepínání mezi jednotlivými obrazovkami laboratorního zařízení. Na dolní části obrazovky nalezneme tlačítka, která jsem využil k základním ovládacím povelům.

V levém dolním rohu jsem udělal zobrazení chybového stavu („`ERROR`“), zelená znamená bezchybový stav. Při případném výskytu chyby vidíme vedle zobrazení stavu erroru, který bude svítit červeně, také jeho identifikační číslo (`ErrorID`).

Na spodní liště uprostřed pod nápisem „Volba režimu“ se zobrazují hlášky pro uživatele, je zde využito proměnné `text`. V případě výskytu chyby zde bude popsáno, o jakou chybu se jedná. V případě, že bude vybrán režim a budou v něm nastaveny nulové parametry, zařízení se nespustí, a zde bude vypsán text „Nulove parametry“.

V pravém dolním rohu vidíme možnost volby režimu. Režim je volen na základě výběru z `Dropdownu` pomocí `USINT` hodnoty. Tato hodnota je pak ve funkčním bloku zpracována jako vstup a na základě této hodnoty je vybrán příslušný režim. Více o dostupných režimech v sekci 6.

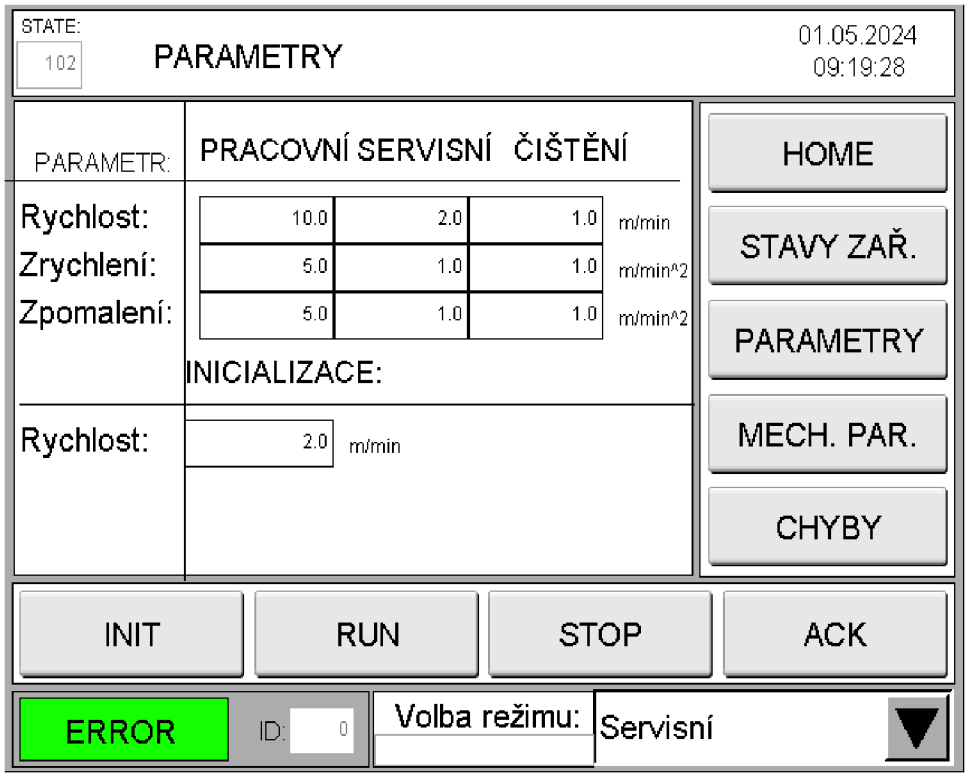
Ovládací tlačítka a obecně celý rámeček displajů je u všech obrazovek stejný, proto je popsán pouze zde u hlavní obrazovky.



Obrázek 9.1: Hlavní strana vizualizace

9.2.2 Obrazovka s kinematickými parametry

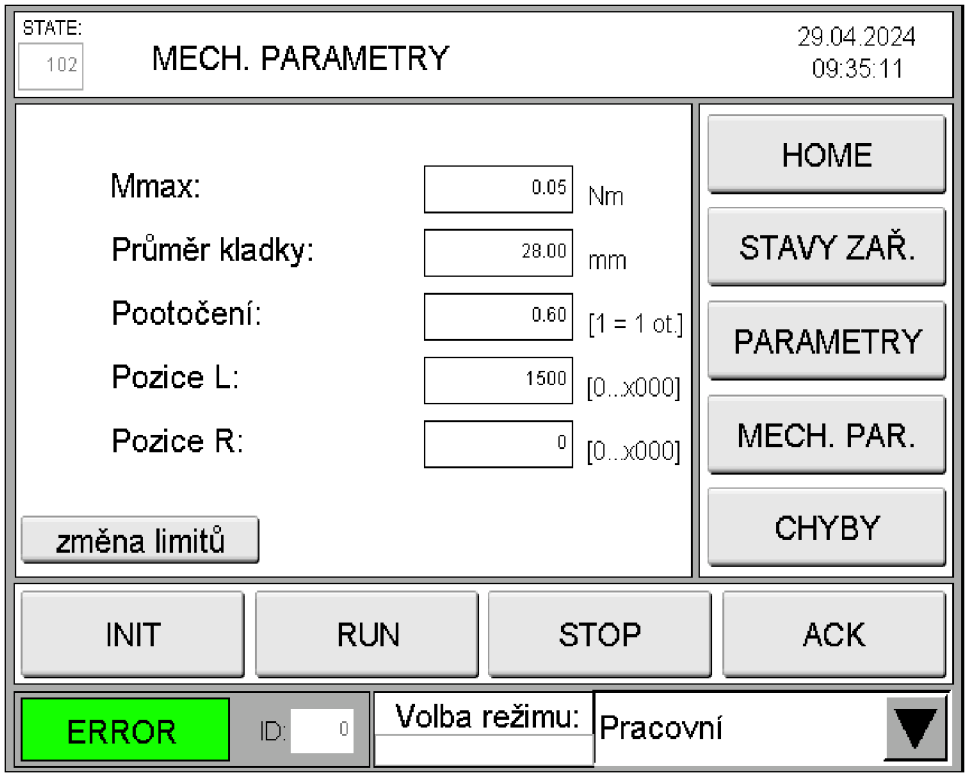
Na obrázku 9.2 můžeme nastavovat parametry pro jednotlivé režimy. Všechny režimy mají limity maximálních a minimálních hodnot, které se dají nastavit pomocí proměnných v proměnné `Limits`. Přepnutí na obrazovku s limitními hodnotami jsem umístil na straně s mechanickými parametry, viz obrázek 9.4. Je zde umístěno tlačítko s názvem „změna limitů“.



Obrázek 9.2: Obrazovka s kinematickými parametry

9.2.3 Obrazovka s mechanickými parametry

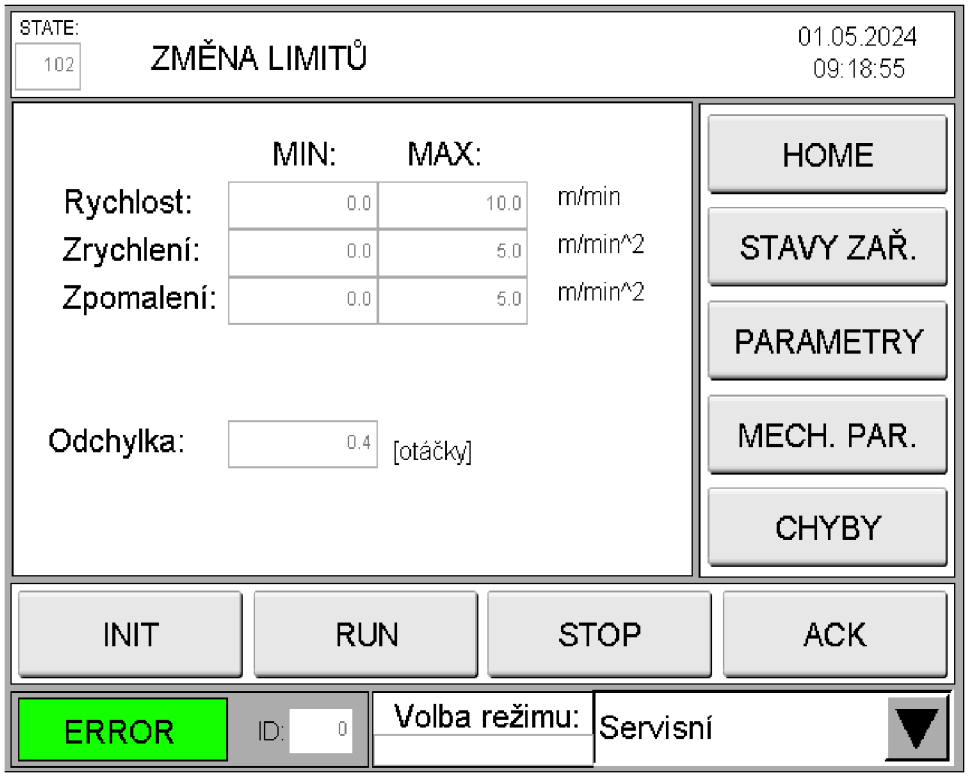
Na obrazovce pro zadávání mechanických parametrů vidíme možnost zadat všechny mechanické parametry, které jsou důležité pro rozběh či proces stroje. Prvním parametrem je maximální moment, který určuje tah v lanku, následně průměr kladky, pomocí kterého dopočítávám rychlosti a parametr `pootoceni` o kterou se má zařízení při inicializaci pootočit. Další dvě hodnoty jsou pro zadání koncových poloh dojezdu motorů. Více k těmto parametrům se dočtete v kapitole 8 v sekci 8.3.



Obrázek 9.3: Obrazovka s kinematickými parametry

9.2.4 Obrazovka s nastavením limitních hodnot

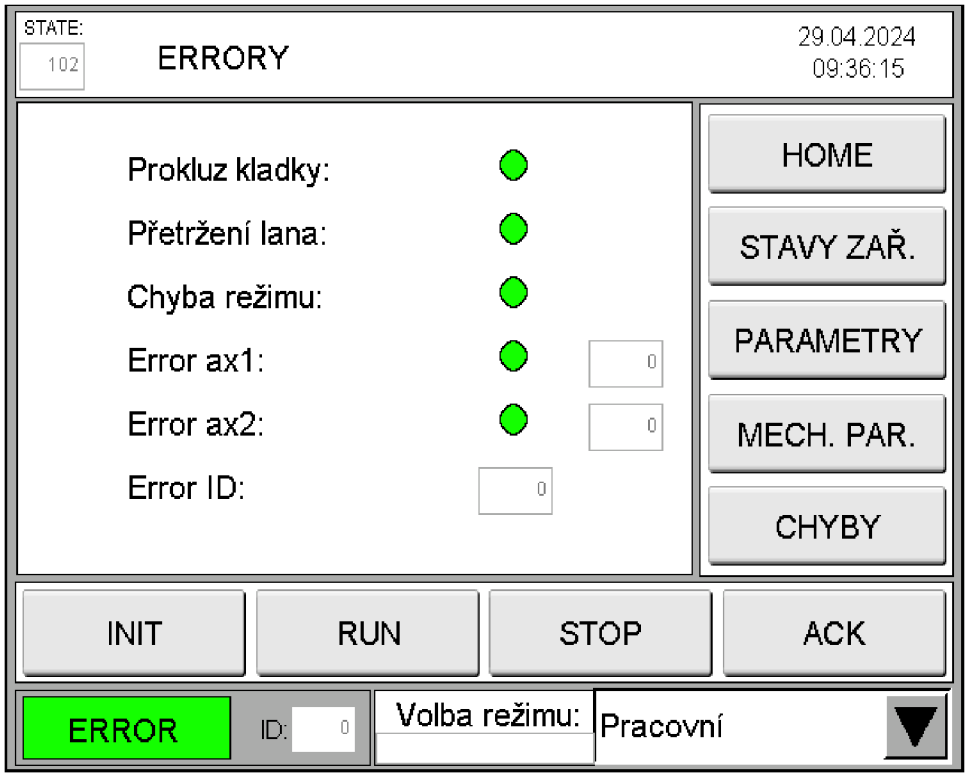
Na obrazovce změny limitů jsem chtěl předvést možnosti nastavení maximálních a minimálních hodnot pro kinematické parametry a také nastavení odchylky pro snímání chyb na lanové elektrodě. Více o těchto parametrech `Limits` se dočtete v sekci 8.3.



Obrázek 9.4: Obrazovka s nastavením limitních hodnot

9.2.5 Obrazovka s chybami

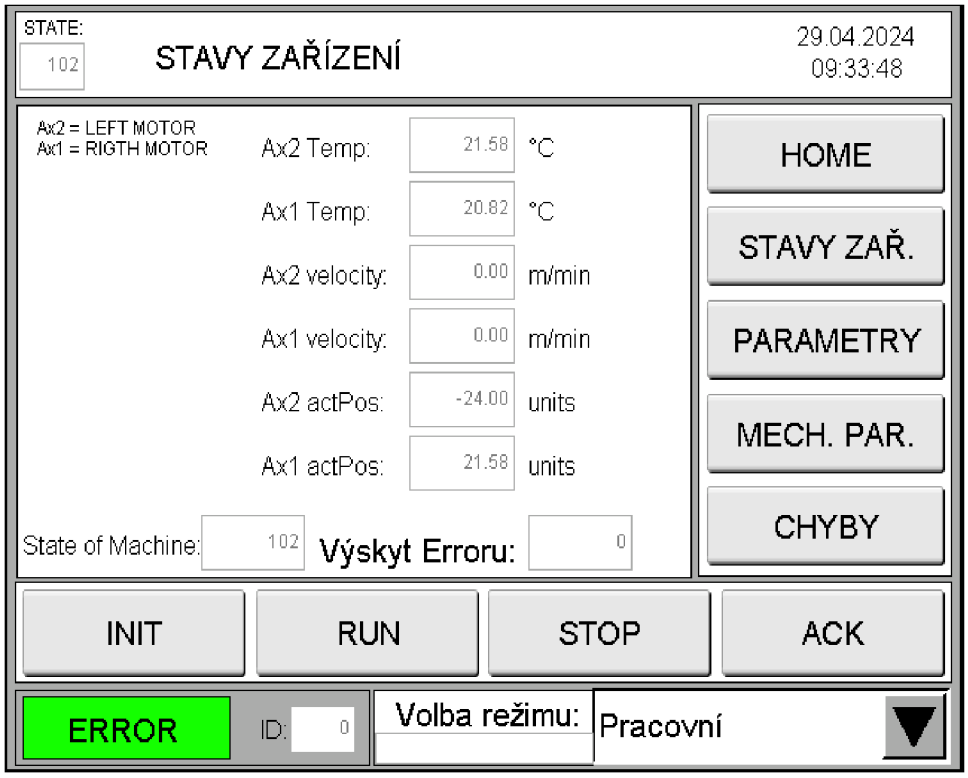
Abych mohl předvést a otestovat diagnostické nástroje zařízení, které jsem u funkčního bloku zavedl, vytvořil jsem obrazovku pro přehled výskytu chyb. Není to jediná obrazovka pro přehled výskytu stavu zařízení. Další výstupní parametry lze nalézt na obrazovce 9.6.



Obrázek 9.5: Obrazovka s chybovými hlášeními

9.2.6 Obrazovka s přehledem stavů zařízení

Pro interpretaci naprogramovaných výstupů jsem vytvořil obrazovku, která reprezentuje stavy jednotlivých motorů. Na obrázku 9.6 vidíme využití doprogramovaných funkčních bloků pro snímání teplot. Další výstupní hodnoty zobrazují aktuální rychlosti motorů. Poslední dva parametry se věnují aktuálním pozicím pohonů. Na spodní část obrazovky jsem ještě umístil zobrazení aktuálního stavu zařízení a identifikační číslo chyby, která se na zařízení právě vyskytuje. Z obrázku 9.6 vidíme, že pole pro výpis čísla chyby obsahuje nulu. To znamená, že zařízení se nachází v bezchybném stavu.



Obrázek 9.6: Obrazovka se stavy motorů

9.3 Testování funkcionalit

Po volání funkčního bloku jsem otestoval ovládání vizualizace a zadávání parametrů. Vizualizace fungovala velmi dobře pro zadávání parametrů a pro volání povelů pro funkční blok.

Při testování jsem začal s běžným provozem zařízení, provedl jsem inicializaci lanové elektrody a zadal jednotlivé mechanické parametry. Následně jsem provedl zadání kinematických parametrů pro jednotlivé režimy. Po provedení inicializace lanové elektrody a zadání parametrů pro režimy jsem zkusil pomocí tlačítka `RUN` spustit zařízení. Lanová elektroda se rozjela s požadovanou rychlostí a zrychlením. Během výrobního procesu jsem vyzkoušel změny režimů a změny parametrů. Předávání nových parametrů fungovalo dle očekávání, po změně parametrů zařízení dokončilo právě vykonávaný pohyb (navinutí lana na jednu kladku) a při rozjetí dalšího pohybu už pracovalo s novými parametry.

Pomocí tlačítka `STOP` jsem vyvolal povel pro zastavení stroje. Dle algoritmu udělalo zařízení přesně očekávaný pohyb, došlo k navinutí lana stroje na pravou kladku a s definovaným zpomalením k úplnému zastavení.

Závěrem jsem otestoval zaznamenávání chyb. Vyzkoušel jsem povolit kladku a nechal jsem zařízení jet v pracovním režimu. Povolená kladka se začala protáčet a okamžitě byl zaznamenán prokluz kladky. Zařízení tímto zaznamenáním přešlo do režimu chyby. Stejnou chybu jsem vyzkoušel snímat pomocí mechanického vytažení

lana mezi kladkami v průběhu převíjení. Laboratorní model okamžitě zaznamenal prokluz kladky. Chybu jsem vždy následně potvrdil a nechal jsem lanovou elektrodu znovu zinicializovat. Při inicializaci jsem pomáhal mechanicky lano napínat tak, aby došlo k jeho navinutí do závitů kladky.

Další chybou, kterou jsem vyzkoušel, bylo přetržení lana. Při výrobním procesu jsem vzal nůžky a lano jsem drze přestřihl. Zařízení okamžitě zaznamenalo přetržení lana. Následně jsem lano vyměnil, chybu jsem potvrdil a spustil znovu inicializaci.

Zkusil jsem také přes Automation Studio vyvolat chybu na motoru. Zařízení se dostalo do chybového stavu a následně jsem chyby odstranil pomocí tlačítka `ACK`. Odstranění více chyb vzniklých najednou fungovalo spolehlivě. Tímto jsem otestoval kompletně všechny chyby, které jsem na zařízení naprogramoval a odladil.

9.4 Výsledek testování

Při testování jsem dospěl k závěru, že se mi povedlo vytvořit funkční blok, který splňuje požadované funkcionality. Pro testování byla vytvořena vizualizace, pomocí které je zařízení možné ovládat a zadávat mu požadované parametry.

Ovládání funkčního bloku pomocí příkazů funguje velmi dobře a spolehlivě. Stroj se dokáže zastavit v okamžiku, kdy je mu to dovoleno. V případě vyvolání požadavku pro zastavení při převíjení lana se zastaví až po navinutí lana na pravou kladku.

Provedené testování spočívalo ve změně parametrů, změně režimů a testování základního ovládání. Dále jsem se věnoval testování jednotlivých chyb.

Testování chyb a jejich potvrzování fungovalo bezproblémově a dle očekávání. V rámci testování jsem vyzkoušel jak snímání prokluzu kladky pomocí dvou způsobů, tak snímání přetrženého lana i vzniklých chyb na motorech.

10 Návod k využití funkčního bloku

K funkčnímu bloku pro lanovou elektrodu jsem vytvořil návod popisující funkci vytvořeného funkčního bloku pro ovládání lanové elektrody. Návod popisuje základní ovládání a volání funkčního bloku. Dává uživateli přehled o jeho režimech a dostupných sadách parametrů. V návodu je popsána funkčnost zařízení v jednotlivých režimech a možnosti, které uživatel ve zvoleném režimu dostává. Pro kompletní přehled jsou popsány jednotlivé stavy zařízení, parametry a jejich význam, rozsahy a možnosti volby.

Pro správný chod zařízení jsou zde popsány chyby, ke kterým může v zařízení dojít. Uživatel dostává kompletní řešení chyb, návod popisuje jak se tyto chyby mohou vyskytnout a co je nutné udělat pro jejich odstranění.

Pro dokonalý přehled pro programátora jsem v návodu popsal všechny stavy zařízení, které se identifikují čísly. Podle jednotlivého čísla lze poté sledovat, v jakém stavu se zařízení nachází, či při tvorbě programu k zařízení s více lanovými elektrodami lze např. hledat chybu podle čísla stavu.

11 Shrnutí práce

Ve své diplomové práci jsem se zabýval tvorbou řídicího software pro lanovou elektrodu pro výrobu plošných nanovlákenných produktů. Po seznámením se s výrobním strojem a použitou technologií jsem začal uvažovat o řídicím algoritmu. Zároveň při rozmýšlení algoritmu jsem zprovoznil laboratorní model a začal jsem programovat pohony v jazyku strukturovaného textu. Bylo nutné seřídit regulátory a následně přidat několik nutných funkčních bloků, které jsem potřeboval pro dosažení zpětných vazeb či pohybů. Laboratorní model jsem doplnil o dotykový displej, pomocí kterého jsem prováděl zadávání hodnot a testování příkazů pro pohony.

Pro tvorbu řídicího algoritmu jsem využíval jazyka strukturovaného textu, v kterém jsem naprogramoval základní funkcionalitu a následně jsem tento algoritmus uzavřel do podoby funkčního bloku.

Na zařízení jsem zavedl stavový automat, navrhl tři pracovní režimy, kterým odpovídají tři sady pracovních parametrů. Dále jsem se zamyslel nad možným vznikem chyb, vytvořil jsem pro ně širokou diagnostiku a identifikační čísla. Diagnostiku jsem doplnil chybami vznikajícími na pohonech.

Po kompletním otestování jsem algoritmus ucelil do funkčního bloku, který jsem opatřil množinou vstupních a výstupních proměnných. Závěrem práce jsem funkční blok otestoval a upravil pro něho vizualizaci. Přidal jsem několik obrazovek pro reprezentaci všech diagnostických nástrojů a statusů.

Při tvorbě řídicího softwaru jsem narazil na problém špatného vypínání bloku pro řízení motoru s požadovaným momentem. Problém se mi podařilo odstranit až po dlouhém zkoumání všech vnitřních signálů pomocí Trace funkce v Automation studiu. Výsledkem zkoumání bylo nízké nastavení zpomalení při vypínání bloku pro řízení momentu, které se propsalo z předchozího kroku.

Technologie použitá na zařízení je velmi spolehlivá, ačkoliv není ekonomicky velmi výhodná. Zařízení stále přechází z režimů akcelerace do decelerace a je tak spotřebováno mnoho energie. Do budoucna by bylo vhodné využít technologii nekonečného lanka. Tato technologie by byla z pohledu řízení ekonomičtější, avšak z pohledu výroby vláken náročná kvůli nanášení polymeru na lana či tažení lana bez polymeru zpět.

Závěr

Po seznámení se s principem výroby plošných nanovlákenných produktů a po nahlédnutí na reálný stroj jsem vyhodnotil, na jakém principu je nutné sestrojit algoritmus pro řízení. Promyslel jsem si algoritmus, režimy zařízení, diagnostické nástroje, vstupní a výstupní hodnoty.

Následně jsem začal programovat laboratorní model a doplnil ho o dotykový displej. Vytvořil jsem programy pro motory, nastavil regulátory a doplnil jednotlivé bloky pro řízení momentu do programů. Dále jsem řídicí programy rozvinul o nutné kontroly pro dosažení pozic. Samotný algoritmus jsem vytvářel v jazyku strukturovaného textu. Vytvořený algoritmus jsem odladil, otestoval a následně jsem ho uzavřel do podoby funkčního bloku. Funkční blok jsem po uzavření otestoval na laboratorním modelu, vytvořil jsem k němu uživatelsky jednoduchou vizualizaci a vyzkoušel zadávání parametrů, příkazů a nástroje pro odstranění chyb, které jsem nasimuloval.

Přínosem tohoto funkčního bloku je jeho snadná interpretace do zařízení s více lanovými elektrodami. Funkční blok má sloužit pro jednodušší vytvoření programu s možností ho vícekrát volat. Vhodné využití společně s nastavením vstupních a výstupních parametrů dělá z funkčního bloku velmi vhodný celek pro řízení lanové elektrody.

Literatura

[1] *5 Different Methods Used to Fabricate Nanofibers* [online]. 2024. [cit. 2024-04-20]. Dostupné z: https://www.nanoscience.com/blogs/how-to-fabricate-nanofibers/.

[2] *ACOPOS 8V1010.50-2* [online]. B&R Industrial Automation, 2024 [cit. 2024-04-15]. Dostupné z: https://www.br-automation.com/cs/produkty/rizeni-pohybu/acopos/servo-drives/8v101050-2/?noredirect=1.

[3] *Automation Studio* [online]. B&R Industrial Automation, 2024 [cit. 2024-04-15]. Dostupné z: https://www.br-automation.com/en/products/software/automation-software/automation-studio/.

[4] *B&R 6PPT30.0573-20W* [online]. B&R Industrial Automation, 2024 [cit. 2024-04-16]. Dostupné z: https://www.br-automation.com/cs/produkty/hmi/power-panel-t-series-and-c-series/power-panel-t-series/power-panel-t30/6ppt300573-20w/.

[5] *B&R 8LVA synchronous motors* [online]. B&R Industrial Automation, 2024 [cit. 2024-04-15]. Dostupné z: https://www.br-automation.com/cs/produkty/rizeni-pohybu/8lva-synchronous-motors/.

[6] BERAN, Jaroslav. *Linka na výrobu plošných kompozitních nanovlákenných materiálů s využitím AC elektrospinningu*. Liberec, 2023. ODBORNÁ ČÁST ZÁVĚREČNÉ ZPRÁVY. Technická Univerzita v Liberci.

[7] JOHN, Karl-Heinz a Michael TIEGELKAMP. *IEC 61131-3:programming industrial automation systems: concepts and programming languages, requirements for programming systems, decision-making aids*. [online]. 2. vyd. 2010. [cit. 2022-04-11]. ISBN 978-3-642-12014-5.

[8] KUBÁT, Jakub. *Návrh a tvorba řídicího systému laboratorního odstředivého zvlákňovacího stroje*. Liberec, 2022. Bakalářská práce. Technická Univerzita v Liberci.

[9] *Linka na výrobu plošného kompozitního materiálu s obsahem nanovláken* [online]. Katedra textilních a jednoúčelových strojů TUL, 2022 [cit. 2024-04-15]. Dostupné z: https://www.kts.tul.cz/realizace.

[10] NANDANA BHARDWAJ, Subhas C. Kundu. Electrospinning: A fascinating fiber fabrication technique. *Science Direct*. 2010.

[11] URBAN, Luboš. *Programování PLC podle normy IEC EN 61131-3 – víc než jednotné jazyky* [online]. Automa – časopis pro automatizační techniku, s. r. o., 2024 [cit. 2024-04-15]. Dostupné z: https://automa.cz/cz/casopis-clanky/programovani-plc-podle-normy-iec-en-61131-3-vic-nez-jednotne-jazyky-2005_02_30310_1237/.

[12] *X20CP1585* [online]. B&R Industrial Automation, 2024 [cit. 2024-04-15]. Dostupné z: https://www.br-automation.com/en/products/plc-systems/x20-system/x20-plc/x20cp1585/?noredirect=1.

Seznam obrázků

Seznam tabulek

A Přílohy

- Program funkčního bloku (B&R Automation Studio) (lanoFB.zip)
- Návod k funkčnímu bloku, PDF formát (navodFB.pdf)
- Program laboratorního modelu lanové elektrody (B&R Automation Studio) (lanElektroda.zip)