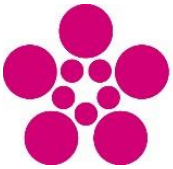


Ekonomická fakulta
Faculty of Economics

Jihočeská univerzita v Českých Budějovicích
University of South Bohemia in České Budějovice

Jihočeská univerzita v Českých Budějovicích

Ekonomická fakulta

Katedra aplikované matematiky a informatiky

Bakalářská práce

Návrh měřícího a zkušebního zařízení pro systém ochrany a řízení zdroje energie

Vypracoval: Ilia Pospelov

Vedoucí práce: Ing. Ludvík Friebel, Ph.D.

České Budějovice 2024

JIHOČESKÁ UNIVERZITA V ČESKÝCH BUDĚJOVICÍCH

Ekonomická fakulta

Akademický rok: 2022/2023

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Ilia POSPELOV**
Osobní číslo:	**E21571**
Studijní program:	**B0688A140010 Podniková informatika**
Téma práce:	**Návrh měřícího a zkušebního zařízení pro systém ochrany a řízení zdroje energie**
Zadávající katedra:	*****Katedra aplikované matematiky a informatiky**

Zásady pro vypracování

Bakalářská práce se zabývá výběrem vhodné řídící jednotky, která bude součástí měřícího a zkušebního zařízení pro zdroj energie. Pro vybraný programovatelný logický řadič (PLC) bude navržen software, který bude odpovídat požadavkům zákazníka.
Řídící jednotka bude za pomoci vhodného zdroje proudu generovat signály o požadovaných parametrech a bude měřit dobu od překročení kritických hodnot vstupního signálu měřící aparatury do doby reakce jejích výstupních bezpečnostních relé pro systém ochrany a řízení. Ovládání a monitorování stavu procesu řídícího PLC bude možné přes webové rozhraní.

Metodický postup:

1. Programovatelné logické řadiče a jejich využití v oblasti měření a regulace.
2. Definice požadavků uživatele a výběr vhodného PLC.
3. Zpracování návrhu řídícího systému pro danou technologii.
4. Implementace požadované řídící logiky včetně webového rozhraní v rámci zvolené technologie.
5. Zhodnocení a závěr.

Rozsah pracovní zprávy:	**40 – 50 stran**
Rozsah grafických prací:	**dle potřeby**
Forma zpracování bakalářské práce:	**tištěná**

Seznam doporučené literatury:

1. Historie PLC. *PLC AUTOMATIZACE*. [online]. Dostupné z: <http://plc-automati-zace.cz/knihovna/historie/historie-plc.htm>
2. MARTINÁSKOVÁ, M., & ŠMEJKAL, L. (2004). *Řízení programovatelnými automaty*. Praha: Vydavatelství ČVUT.
3. ROBENEK, J. Návrh PLC očima vývojáře. 8. část (Digitální vstupy / výstupy). *Vývoj.HW.cz* [online]. [cit. 15.03.2019]. Dostupné z: <https://vyvoj.hw.cz/teorie-a-praxe/navrh-plc-ocima-vyvojare-8-cast-digitalni-vstupy-vystupy.html>
4. ŠMEJKAL, L., & MARTINASKOVA, M. (1999). *PLC a automatizace: základní pojmy, úvod do programování*. Praha: BEN – technická literatura.
5. ŠMEJKAL, Ladislav. (2005). *PLC a automatizace*. 2. díl. Praha: BEN – technická literatura.

Vedoucí bakalářské práce: **Ing. Ludvík Friebel, Ph.D.**
***Katedra aplikované matematiky a informatiky

Datum zadání bakalářské práce: **20. ledna 2023**
Termín odevzdání bakalářské práce: **12. dubna 2024**

JIHOČESKÁ UNIVERZITA
V ČESKÝCH BUDĚJOVICÍCH
EKONOMICKÁ FAKULTA
Studentská 13
370 05 České Budějovice

doc. Dr. Ing. Dagmar Škodová Parmová
děkanka

doc. RNDr. Jana Klicnarová, Ph.D.
vedoucí katedry

V Českých Budějovicích dne 3. února 2023

Prohlášení

Prohlašuji, že svoji bakalářskou práci jsem vypracoval samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury. Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své bakalářské práce, a to v nezkrácené podobě vzniklé vypuštěním vyznačených částí archivovaných Ekonomickou fakultou – elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách, a to se zachováním mého autorského práva k odevzdanému textu této kvalifikační práce. Souhlasím dále s tím, aby toutéž elektronickou cestou byly v souladu s uvedeným ustanovením zákona č. 111/1998 Sb. zveřejněny posudky školitele a oponentů práce i záznam o průběhu a výsledku obhajoby kvalifikační práce. Rovněž souhlasím s porovnáním textu mé kvalifikační práce s databází kvalifikačních prací Theses.cz provozovanou Národním registrem vysokoškolských kvalifikačních prací a systémem na odhalování plagiátů.

12.04.2024 Ilia Pospelov

Poděkování

Děkuji svému vedoucímu **Ing. Ludvíku Friebelovi, Ph.D.**, za jeho vedení během psaní bakalářské práce. Jeho rady, podněty a inspirace byly pro mě klíčové. Oceňuji jeho trpělivost, ochotu a vstřícnost při konzultacích.

Také děkuji mým kolegům. Jejich účast, otevřenost a podpora byly neocenitelné pro úspěch této práce.

Obsah

1 Úvod

Ve světě dnes je čím dál tím víc jasné, jak důležité jsou spolehlivé a bezpečné technologie pro řízení a ochranu zdrojů energie. Vyvíjet a zavádět pokročilé systémy kontroly je klíčové pro zajištění bezpečného používání energetických zařízení. Tato bakalářská práce se zabývá vytvořením testovacího zařízení, které bude kontrolovat systémy ochrany a regulace zdrojů energie tak, aby bylo možné posoudit a vylepšit jejich reakci na vážné změny.

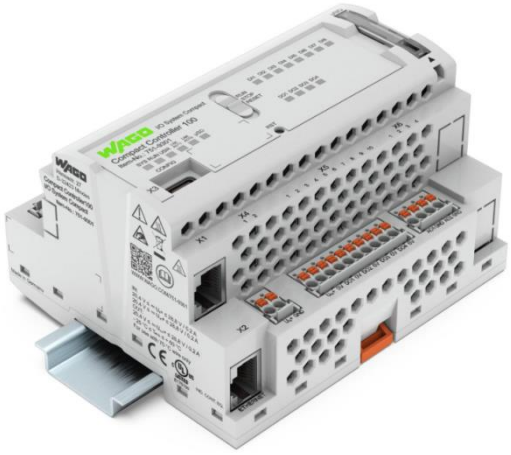
Potřebujeme takové zařízení k pravidelnému testování kontrolních systémů, kde hlavním parametrem je rychlost reakce na změny. Stávající testovací metody, které se opírají o ruční ovládání a měření, vyžadují hodně času a jsou náchylné k chybám kvůli lidskému faktoru. Použití moderních automatizovaných přístupů výrazně zvýší přesnost a spolehlivost výsledků.

V rámci této práce navrhujeme vyvinout systém založený na programovatelném logickém řadiči (PLC), který bude generovat signály s určitými parametry a automaticky měřit, jak rychle kontrolní systémy na tyto signály reagují. Použití PLC nejenže automatizuje testovací proces, ale také umožňuje dálkové ovládání zařízení přes webové rozhraní, což testování dělá flexibilnější a dostupnější.

Vývoj a zavedení takového zařízení do procesu testování systémů ochrany a regulace zdrojů energie zvýší přesnost a spolehlivost měření, což přispěje k zvýšení celkové bezpečnosti provozu energetických zařízení.

Cílem této bakalářské práce je vyvinout a ověřit systém testovacího zařízení založeného na programovatelném logickém řadiči Wago Compact Controller 100 (viz obrázek 1) pro kontrolu a zlepšení systémů ochrany a regulace zdrojů energie. Hlavním úkolem systému je automatizace testovacích procesů s větší přesností a spolehlivostí měření, stejně jako zjednodušení postupů monitorování a řízení prostřednictvím vyvinutého webového rozhraní. Cílem je také ověřit v praxi fungování PLC.

Obrázek 1: Wago Compact Controller 100.



Zdroj: WAGO (n.d.).

Pro dosažení stanoveného cíle byly určeny následující konkrétní úkoly:

1) Studium a analýza požadavků na testovací zařízení: určení klíčových funkčních a technických požadavků, které musí vyvíjený systém splňovat, na základě analýzy existujících řešení a specifikací energetických zařízení.
2) Výběr a zdůvodnění použití Wago Compact Controller 100 jako základu systému: prozkoumání charakteristik a možností tohoto řadiče pro použití v kontextu úkolu, včetně hodnocení jeho výkonu, komunikačních rozhraní a podpory softwaru.
3) Vývoj programového kódu v jazyce Structured Text v prostředí Codesys: programování logiky fungování testovacího zařízení, včetně algoritmů generování signálů a zpracování dat měření.
4) Vytvoření webového rozhraní pro monitorování a řízení testovacího procesu: vývoj uživatelského rozhraní na základě Codesys, které umožňuje pohodlný přístup k řízení testovacího zařízení a zobrazuje výsledky testů v reálném čase.
5) Testování a optimalizace systému: provedení série testů pro ověření funkčnosti a spolehlivosti vyvinutého řešení, analýza získaných dat a provedení potřebných úprav pro zlepšení fungování systému.

2 Teoretický přehled

2.1 Historie vývoje a princip fungování PLC

PLC jsou elektronická zařízení používaná v automatizaci průmyslových procesů. Jsou schopná provádět širokou škálu řídících, regulačních a monitorovacích funkcí. PLC hrají klíčovou roli v automatizaci díky své flexibilitě, spolehlivosti a možnosti rychlé změny řídícího programu v reakci na změnu výrobních požadavků.

Vývoj PLC začal dvěma paralelními směry: na jedné straně od reléové technologie, která se používala pro vytváření logických obvodů, a na druhé straně – od nepřetržitých nástrojových měření a pneumatických proporcionálně-integrálně-diferenciálních regulátorů, které byly později transformovány do programových realizací (Peterson, 2022).

První PLC bylo vytvořeno firmou Bedford Associates pro amerického výrobce aut General Motors v roce 1968 pod názvem Modulární Digitální Kontrolér (Modicon) (viz obrázek 2). Jednalo se o řešení problému vysokých nákladů na výměnu složitých reléových řídicích systémů při změně výrobních požadavků. PLC výrazně zjednodušilo proces adaptace řídicích systémů na nové podmínky, snížilo čas a náklady na změny (Olsson, 2005).

Obrázek 2: Modicon 084

Zdroj: (Peterson, 2022).

Od té doby PLC prošly významnými změnami: od zařízení postavených na elektromechanických relé a pneumatických komponentech po moderní mikroprocesorové systémy. Evoluce mikroprocesorů umožnila vytvořit PLC, schopné zpracovávat složité logické operace a řídit různé aspekty průmyslových procesů, včetně analogových signálů a matematických výpočtů (Amin & Mridha, 2020).

Programovatelné logické řadiče fungují na principu skenování svého řídicího programu, postupně čtou vstupní signály, provádějí zadanou logiku řízení a aktualizují výstupní signály pro řízení procesů. Software PLC umožňuje inženýrům vytvářet a modifikovat logiku řízení bez potřeby fyzického přepojování nebo výměny zařízení, což činí PLC ideálními pro použití v dynamicky se měnících výrobních podmínkách (Rohner, 1996).

Jednou z klíčových vlastností PLC je jejich spolehlivost a schopnost vydržet tvrdé průmyslové podmínky, včetně vibrací, prachu, špíny a extrémních teplot. PLC také nabízí vysokou míru flexibility v programování, podporují různé jazyky a metody programování, jako jsou schodové diagramy, diagramy funkčních bloků, seznamy instrukcí, stejně jako modernější programovací jazyky (Hallak & Bumiller, 2016).

Od svého vzniku se PLC staly nedílnou součástí moderní průmyslové automatizace, poskytují efektivní a flexibilní řešení pro řízení výrobních procesů v nejrůznějších odvětvích.

2.2 Přehled hlavních parametrů

Standardní průmyslový kontrolér obvykle zahrnuje centrální procesorovou jednotku, rozhraní pro připojení k sítím, moduly pro ukládání dat a různé komponenty pro realizaci funkcí vstupu a výstupu (viz obrázek 4).

PLC umožňují provádět řízení procesů a strojů na základě příchozích signálů a předem stanovených algoritmů. Nejdůležitějšími prvky každého PLC jsou jeho vstupy a výstupy, které zajišťují interakci kontroléru s vnějším prostředím. Vstupy PLC přijímají signály od různých senzorů a aktuátorů, zatímco výstupy posílají příkazy pro řízení těchto zařízení. Efektivní využití vstupů a výstupů PLC umožňuje realizovat složité automatizované systémy s vysokou přesností a spolehlivostí (Ye, 2024).

Diskrétní vstupy PLC jsou určeny pro přijímání signálů se dvěma stavy: zapnuto nebo vypnuto, což odpovídá logickým hodnotám 1 a 0. To umožňuje připojit k PLC širokému spektru zařízení, včetně tlačítek, spínačů, polohových senzorů a mnoha dalších. Některá zařízení, která mají více než dva stavy, mohou být připojena prostřednictvím několika diskrétních vstupů pro kódování různých stavů. Systémový software PLC automaticky čte stavy těchto vstupů, což usnadňuje práci programátora. Každý diskrétní vstup je vybaven filtry pro ochranu proti rušení a má individuální ochranné mechanismy, které zajišťují vysokou spolehlivost a přesnost práce (Bolton, 2015).

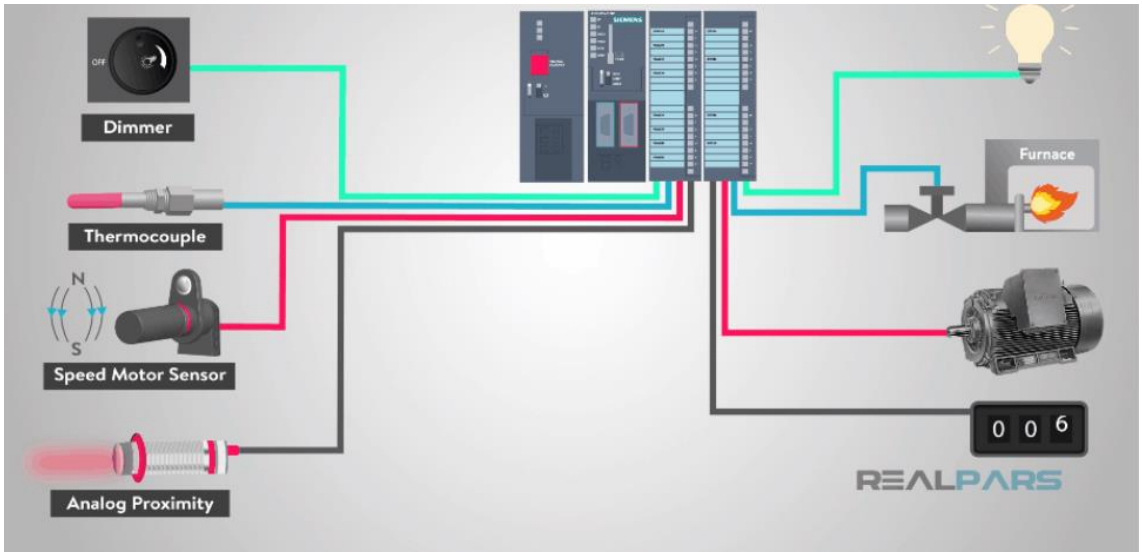
Analogové vstupy zpracovávají proměnlivé signály, převádějí je na digitální formu pomocí analogově-digitálních převodníků. To umožňuje PLC vnímat a zpracovávat široký rozsah signálů, reprezentovaných úrovní napětí nebo proudu. Standardní rozsahy signálů a vysoká přesnost převodu umožňují připojení různých senzorů, včetně odporových teploměrů a termočlánků. Důležitou vlastností analogových vstupů je potřeba kvalitní montáže vnějších obvodů pro minimalizaci rušení a dosažení přesných měření (Bolton, 2015).

Diskrétní výstupy PLC jsou určeny pro řízení aktuátorů, jako jsou ventily, relé a různé pohony. Mohou být realizovány jak na základě klasických relé, tak s použitím elektronických spínačů pro zvýšení rychlosti a spolehlivosti. Umístění spínacích prvků přímo u zátěže umožňuje snížit náklady na montáž a údržbu a také snížit úroveň rušení.

Analogové výstupy se používají pro řízení zařízení, která vyžadují proměnlivý signál, například pro regulaci rychlosti motoru nebo teploty.

Je důležité předvídat rezervu digitálních nebo analogových výstupů při návrhu systému, aby se zajistila flexibilita a možnost rozšíření systému v budoucnu (možností využit vstupy a výstupy viz obrázek 3).

Obrázek 3: Vstupy a výstupy PLC



Zdroj: (Richardson, 2018).

Do struktury procesorového bloku programovatelného logického kontroléru patří prvky jako mikroprocesor nebo centrální procesor, systém pro sledování reálného času, prvky pro ukládání informací a systém kontroly provádění známý jako "watchdog" (Borden & Cox, 2022).

Klíčovými charakteristikami centrálního procesoru jsou jeho pracovní frekvence, bitová architektura, možnosti připojení externích zařízení přes porty vstupu-výstupu, architektonické vlastnosti, charakteristiky stability provozu v různých teplotních podmínkách, schopnost provádět operace s čísly s plovoucí desetinnou čárkou a úroveň spotřeby elektrické energie (Bolton, 2015).

Efektivita procesorů založených na stejné architektuře přímo závisí na jejich taktovací frekvenci. Pro většinu průmyslových kontrolérů je typické použití procesorů s RISC-architekturou, které se vyznačují redukovanou sadou instrukcí. To umožňuje procesoru pracovat s pevným počtem příkazů pevné délky a používat přitom velké množství registrů. Redukovaná sada instrukcí usnadňuje vývoj efektivních kompilátorů a umožňuje realizaci procesorové pipeline, schopné zpracovat jeden příkaz za jeden takt cyklu (Hossameldin & Ihab, 2019).

Pro kontroléry zapojené do intenzivních matematických výpočtů je nezbytná přítomnost matematického koprocesoru nebo použití digitálních signálových procesorů, které

umožňují provádět složité operace během jednoho taktu. To umožňuje výrazně urychlit zpracování dat, včetně operací konvoluce nebo provádění rychlé Fourierovy transformace.

Objem paměti určuje množství informací, které může kontrolér zpracovat během svého provozu. Doba přístupu k paměti je kriticky důležitým parametrem, který ovlivňuje rychlost práce procesoru. Pro optimalizaci rychlosti práce je paměť rozdělena na úrovně s různou rychlostí přístupu a frekvencí použití dat, což umožňuje minimalizovat vliv pomalé paměti na výkon procesoru.

Mezi hlavní typy paměti v PLC patří:

1) ROM (Read-Only Memory) pro ukládání málo měněné informace;
2) RAM (Random Access Memory) pro dočasné ukládání dat aktivně se měnících během provozu;
3) Registry, které zajišťují vysokou rychlost přístupu a používají se pro mezioperace.

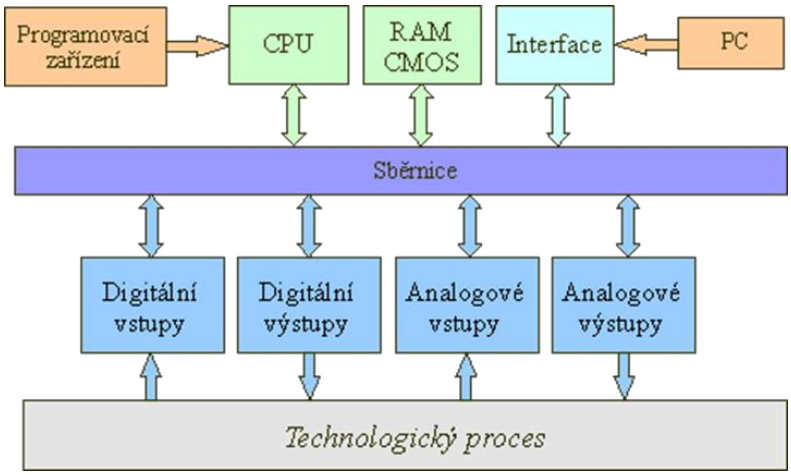
Registry představují nejrychlejší prvky paměti v architektuře procesoru, používané aritmeticko-logickou jednotkou (ALU) pro zpracování základních operací. Trvalá paměť (ROM) slouží pro dlouhodobé ukládání dat, která se mění extrémně zřídka, například operačního systému, zaváděcího kódu, ovladačů nebo spustitelných programů. Operační paměť (RAM) je určena pro dočasné ukládání dat, aktivně se měnících během provozu kontroléru, jako jsou diagnostické informace, proměnné uživatelského rozhraní, hodnoty tagů a výsledky mezioperací (Hossameldin & Ihab, 2019).

Jako ROM se obvykle používá elektricky stíratelná programovatelná paměť (EEPROM), která umožňuje změnit v ní uložená data pomocí elektrického signálu. Flash paměť, jeden z typů EEPROM, je založena na použití kondenzátorů, jejichž náboj se uchovává díky izolačním vlastnostem podložky unipolárních tranzistorů s plovoucí bránou. Hlavní výhodou flash paměti je její schopnost udržovat informace bez napájení, což zajišťuje nezávislost uložených dat na energii. Nicméně aktualizace informací ve flash paměti se provádí po blocích, což omezuje rychlost zápisu dat. Kromě toho se ROM vyznačuje poměrně nízkou rychlostí přístupu k datům.

Počet zápisových operací do flash paměti je fixní a nepřesahuje několik desítek tisíc cyklů. V arsenálu moderních procesorů pro roli operační paměti se používají různé typy: statická RAM (SRAM), dynamická RAM (DRAM) a synchronní dynamická RAM (SDRAM). SRAM funguje na bázi klopných obvodů, které mohou uchovávat data po dlouhou dobu, pokud je udržováno napájení. Dynamická paměť naopak uchovává informace v kondenzátorech, což vyžaduje jejich pravidelné přebíjení pro udržení dat. Hlavní nevýhodou použití klopných obvodových pamětí je jejich vysoká cena a nízká hustota ukládání dat, protože na jednom čipu se vejde omezené množství klopných obvodů. Na druhou stranu, výhodou SRAM je vysoká rychlost práce, dosahující gigahertzů, na rozdíl od paměti založené na kondenzátorech, jejíž rychlost je omezena stovkami hertzů. Nicméně je třeba vzít v úvahu, že všechny typy operační paměti ztrácejí uložené informace při odpojení napájení. Z tohoto důvodu jsou některé modely PLC vybaveny bateriovým napájením, což umožňuje udržet systém funkční i při dočasném přerušení dodávky elektřiny (Bolton, 2015).

V modulárních a monoblokových průmyslových kontrolérech se používá paralelní sběrnice, která umožňuje výměnu informací s moduly vstupu-výstupu, díky čemuž je rychlost dotazování výrazně vyšší ve srovnání se sériovou sběrnicí. Typy paralelních sběrnic jsou: VME, PCI, ISA, CXM, CompactPCI, PC/104. Sériová sběrnice, například RS-485 nebo RS-232, je nezbytná pro připojení vzdálených modulů vstupu-výstupu (Borden & Cox, 2022).

Obrázek 4: Bloková struktura PLC



Zdroj: (352Lab VŠB, n.d.)

2.3 Druhy PLC

V rámci rozlišení mnoha existujících PLC je nutné zvážit jejich hlavní rozdíly.

Základní charakteristikou PLC je počet vstupních a výstupních kanálů. Podle tohoto kritéria se PLC dělí do následujících skupin (Inst Tools, n.d.):

1) Nano-PLC (méně než 16 kanálů);
2) Mikro-PLC (od 16 do 32 kanálů);
3) Malé PLC (více než 32, ale méně než 128 kanálů);
4) Střední PLC (více než 64, ale méně než 1024 kanálů);
5) Velké PLC (více než 512 kanálů, ale méně než 5000 kanálů);
6) Velmi velké PLC (více než 5000 kanálů).

PLC se také mohou lišit podle umístění vstupně-výstupních modulů (International Atomic Energy Agency, 2018):

1) Monoblokové – kde nelze zařízení vstupu-výstupu odstranit nebo nahradit, protože konstrukčně je PLC součástí jednoho celku s vstupně-výstupními zařízeními (například jednoplatové PLC);
2) Modulární – kde jsou centrální procesor a vstupně-výstupní moduly umístěny v společném koši (šasi) a uživatel si vybírá složení modulů podle úkolu;
3) Distribuované – kde jsou vstupně-výstupní moduly umístěny v samostatných skříních a jsou propojeny s procesorem po síti (obvykle pomocí rozhraní RS-485), což umožňuje umístit je na vzdálenost až 1,2 km od procesorového modulu.

Často se PLC mohou kombinovat různými způsoby, například monoblokové PLC může mít vyměnitelné karty, a monoblokové a modulární PLC mohou být doplněny distribuovanými vstupně-výstupními moduly k zvýšení celkového počtu kanálů.

Mnoho PLC je vybaveno vyměnitelnými procesorovými moduly různé výkonnosti, což umožňuje rozšiřovat funkčnost systému bez změny jeho konstrukce.

PLC se také liší podle konstrukce a způsobu instalace:

1) Panelové – pro montáž na paneli nebo dveřích skříně;
2) Pro montáž na DIN lištu uvnitř skříně;

3) Pro instalaci na stěnu;
4) Stojanové – pro montáž ve stojanu;
5) Bez skříně (obvykle jednoplátové) - pro použití ve specializovaných konstrukcích zařízení (OEM - "Original Equipment Manufacturer").

Podle oblasti použití se PLC dělí na několik typů:

1) Univerzální průmyslové;
2) Pro řízení robotů;
3) Pro řízení pozicování a pohybu;
4) Komunikační;
5) PID řadiče;
6) Specializované (například pro řízení inteligentní domácnosti).

PLC se také mohou lišit způsobem programování:

1) Programovatelné z panelu řadiče;
2) Programovatelné pomocí přenosného programátoru;
3) Programovatelné pomocí displeje, myši a klávesnice;
4) Programovatelné pomocí osobního počítače.

Programovací jazyky pro PLC mohou zahrnovat:

1) Klasické algoritmické jazyky (C, C#, Visual Basic);
2) Jazyky podle normy IEC 61131-3.

Některé PLC mohou obsahovat vstupně-výstupní moduly, zatímco jiné ne. Například komunikační PLC plní funkci brány mezi sítěmi, zatímco některé PLC přijímají data z nižších úrovní hierarchie řízení procesů.

2.4 Standardy programování

Podle standardu IEC 61131-3 jsou programovací jazyky pro PLC rozděleny do dvou hlavních kategorií: textové a grafické. Textové jazyky, jako je strukturovaný text (Structured Text - ST) a seznam instrukcí (Instruction List - IL), umožňují vývojářům formulovat příkazy a algoritmy ve formě textových instrukcí, podobně jako tradiční programovací jazyky. Grafické jazyky, včetně schodišťových diagramů (Ladder

Diagram - LD) a diagramů funkčních bloků (Function Block Diagram - FBD), nabízejí vizuální způsob prezentace logiky programu, což usnadňuje jeho pochopení a analýzu (Bolton, 2015).

Kromě standardních programovacích jazyků existují také vysoce úrovňové modelovací jazyky, které umožňují reprezentovat strukturu a logiku PLC-programů na abstraktnější úrovni. Jedním z takových jazyků je sekvenční programování (Sequential Function Chart - SFC), který organizuje řídící program ve formě sekvence kroků a přechodů, usnadňující pochopení a vývoj složitých řídících procesů. Petriho sítě poskytující formální základ pro modelování paralelních procesů, jsou používány pro analýzu a verifikaci PLC-programů, umožňují vývojářům zachytit a vyhnout se potenciálním problémům se synchronizací a interakcí procesů (Bolton, 2015).

2.4.1 Textové programovací jazyky PLC

V průmyslovém standardu jsou vyděleny dva textové programovací jazyky: seznam Instrukcí (IL) a Strukturovaný Text (ST). Oba jazyky mají podobnost s tradičními nízkoúrovňovými programovacími jazyky a vycházejí z nejranějších etap vývoje technologií PLC (Parr, 2003).

IL je jedním z prvních programovacích jazyků pro PLC, jeho syntaxe připomíná assembler. IL se skládá z imperativních operací, které mohou mít parametry a používat registry pro ukládání hodnot. Programy v IL přímo interagují se základními komponenty hardwaru PLC (viz obrázek 5). Díky své nízkoúrovňové orientaci je IL často používán pro překlad do jazyků modelového ověřování, protože programy v jiných programovacích jazycích PLC mohou být převedeny do formátu IL.

Obrázek 5: Ukázka IL

```
LD %I1.1
R   %C8
LD  %I1.2
AND %M0
CU  %C8
LD  %C8.D
ST  %Q2.0
```

Zdroj: (Oliveira & Gaspar, 2020)

ST naopak vychází z programovacího jazyka Pascal a nabízí vyšší úroveň přístupu k programování PLC (viz obrázek 6). ST podporuje podmíněné a iterační operátory, což umožňuje implementovat složité řídící algoritmy bez nutnosti použití skokových příkazů, jak je tomu u IL. Syntaxe ST usnadňuje definici funkcí a bloků funkcí, čímž jej činí přívětivějším a srozumitelnějším pro vývojáře (Duranso, 2021).

Obrázek 6: Ukázka ST

```
IF Switch_1 AND Switch_2 THEN
        LAMP := 1;
ELSEIF Switch_3 AND NOT Switch_4 THEN
        LAMP := 1;
END_IF;
```

Zdroj: (Posdzi, 2021).

IL a ST nabízejí různé přístupy k programování PLC, odrážejíce evoluci jazyků z nízkoúrovňových na vyšší úrovně a strukturované. IL, který je bližší k hardwarové úrovni, poskytuje vysokou flexibilitu a kontrolu nad hardwarem PLC, což může být kriticky důležité v některých aplikacích. Jeho použití však vyžaduje hluboké znalosti o architektuře PLC a může být náročnější při vývoji složitých programů.

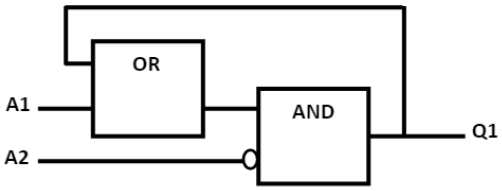
Na druhou stranu, ST poskytuje abstraktnější a snáze pochopitelnou syntaxi, zkracující dobu vývoje a usnadňující údržbu programů. To dělá ST preferovanou volbou pro implementaci komplexní řídící logiky a algoritmů, zejména při velkých objemech kódu.

2.4.2 Grafické programovací jazyky pro PLC

Diagramy Funkčních Bloků (FBD) jsou grafické struktury, které ukazují, jak jsou funkční bloky v programu PLC spojeny mezi sebou a jak probíhá tok dat (viz obrázek 7). Funkční bloky jsou podobné integrovaným obvodům a kombinují funkce poskytované PLC pro vykonávání konkrétních úkolů. Mohou být jak elementární, provádějící základní operace (například srovnání a přesun), tak i složité, vytvořené spojením sady funkcí. Díky jasně definovaným vstupům a výstupům mohou být funkční bloky používány jako "černé skřínky", což usnadňuje jejich použití programátory PLC. FBD umožňují snížit složitost systému pomocí abstrakce, spojující elementární funkce a

vzájemně propojené bloky, což je dělá oblíbenými při modelování a ověřování programů.

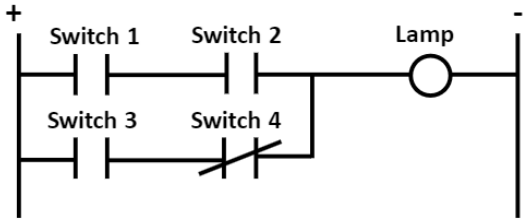
Obrázek 7: Ukázka FBD



Zdroj: (Posdzi, 2021).

Schodišťové Diagramy (LD), původně používané pro návrh reléových panelů, se časem vyvinuly v programovací jazyk pro PLC, také známé jako schodišťová logika nebo reléová schodišťová logika, schodišťové diagramy se skládají ze série pravidel, nazývaných stupně (rungs), které mohou být provedeny postupně během cyklu PLC (viz obrázek 8). Každý stupeň obsahuje prvky, které jsou provedeny sekvenčně zleva doprava, kde výstup každého prvku se stává vstupem pro další. V každém stupni mohou být přítomny takové důležité prvky jako cívky (coils) a kontakty (contacts), kde cívky slouží jako výstupní booleovské proměnné a kontakty jako vstupní. Sekvenční spojení prvků ve stupni tvoří logické "A", zatímco paralelní spojení tvoří logické "NEBO". Schodišťové diagramy mohou být snadno interpretovány jako formule výrokové logiky, což je často využíváno při ověřování PLC-programů (Borden & Cox, 2022).

Obrázek 8: Ukázka LD

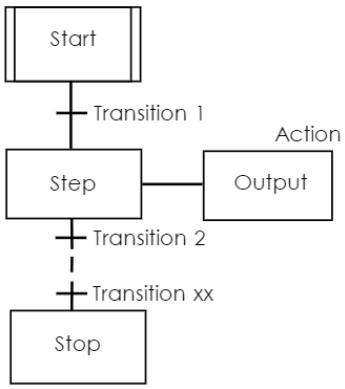


Zdroj: (Posdzi, 2021).

2.4.3 Modelovací jazyky

Sekvenční Funkční Schémata (SFC) jsou prvky, strukturované vnitřní organizaci programů PLC a bloků funkcí (viz obrázek 9). Často každý blok v SFC obsahuje schodišťový diagram, což ukazuje na nižší úroveň abstrakce v programu PLC. SFC poskytují široký přehled programu díky struktuře podobné blokovým diagramům a mohou reprezentovat několik toků provedení programu v rámci jednoho diagramu, zavádějící paralelismus. Inspirovány sítěmi Petri a standardem Grafcet, SFC by měly být vnímány odděleně od programovacích jazyků, jako modelování spolu se sítěmi Petri. SFC strukturují prvky uvnitř programu PLC, definujíci kroky spojené s bloky akcí a přechody, jasně představující pořadí provedení komponent programu, které mohou být uspořádány sekvenčně a/nebo paralelně.

Obrázek 9: Ukázka SFC

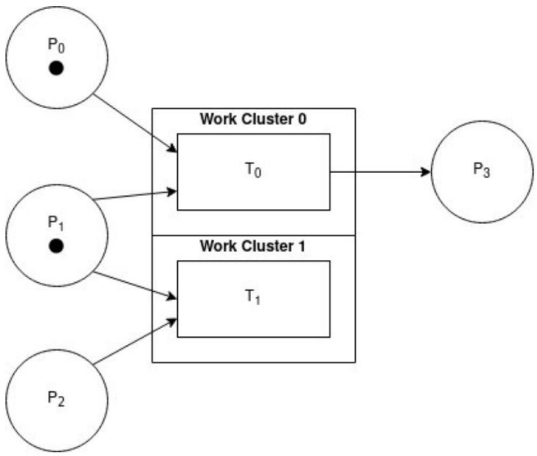


Zdroj: (Posdzi, 2021).

Petriho sítě jsou jedním z nejčastěji používaných formálních přístupů k modelování programů PLC. Petriho sítě se skládá z míst (places), přechodů (transitions) a oblouků, spojujících místa s přechody (viz obrázek 10). Místa mohou obsahovat tokeny, které se používají k modelování paralelismu ve vykonávání Petriho sítě, kde několik tokenů může přecházet mezi místy. Přechod tokenu z jednoho místa do jiného je omezen přechody a oblouky, spojujícími místa mezi sebou. Tokeny uvnitř míst představují základní prvky pro modelování paralelních vykonávání, přesouvající se přes přechody mezi místy. Petriho sítě jsou často používány pro modelování a ověřování programů PLC, protože je lze relativně snadno převést do PLC programů. Kromě toho Petriho sítě nacházejí široké uplatnění při ověřování modelů díky silné podpoře nástrojů a

analyzátorů v této oblasti. V procesu modelování programů PLC jsou používány různé varianty Petriho sítí, jako jsou signálně-interpretované Petriho sítě a barevné Petriho sítě (Rawson, 2022).

Obrázek 10: Ukázka Petriho sítě



Zdroj: (Rawson, 2022).

2.5 Komunikace

Ve současném průmyslu automatizace hrají komunikační protokoly klíčovou roli v zajištění plynulé interakce mezi systémy, rozhraními a strojním zařízením. Jsou základem pro vytváření hluboce integrovaných a vysoce technologických výrobních procesů. S nástupem éry digitalizace a formováním "chytrých" továren, vybavených stále složitějšími sítěmi vzájemně propojených zařízení, stále roste důležitost těchto protokolů. Umožňují okamžitý sběr a analýzu dat zařízení v reálném čase, otevírajíce výrobě nové horizonty v oblastech jako je prediktivní údržba a optimalizace procesů.

2.5.1 Modbus

Protokol Modbus je základním nástrojem v oblasti automatizace a řízení procesů. Byl vyvinut jako jednoduchý a spolehlivý mechanismus pro přenos dat mezi zařízeními a zůstává důležitým standardem v moderních průmyslových systémech.

Modbus využívá model klient/server pro komunikaci mezi zařízeními připojenými prostřednictvím různých typů sběrnic nebo sítí. Pro navázání komunikace je jedno zařízení konfigurováno jako server a druhé jako klient. Klient iniciuje spojení pomocí protokolu Modbus TCP tím, že pošle požadavek serveru, který následně odpovídá přenosem požadovaných informací (Sudhir, Mudhalwadkar, & Shah, 2016).

Přes svou spolehlivost a univerzálnost má Modbus TCP, jako jeden z nejstarších průmyslových protokolů, omezené vestavěné bezpečnostní funkce. Absence šifrovacích a autentizačních mechanismů z výchozího nastavení činí tento protokol zranitelným vůči neoprávněnému přístupu a manipulaci s daty. Pro zvýšení bezpečnosti se doporučuje použití dalších opatření, jako jsou virtuální soukromé sítě (VPN) a metody šifrování na úrovni aplikace, včetně bezpečnostních protokolů TLS nebo SSL.

V roce 2018 byl vyvinut bezpečnostní protokol Modbus s cílem posílit ochranu proti neoprávněnému přístupu. Výzkumy v této oblasti jsou zaměřeny na posílení bezpečnosti Modbus TCP, včetně návrhu metod autentizace a autorizace s využitím kontroly přístupu založené na uživatelském jménu a hesle (Sudhir, Mudhalwadkar, & Shah, 2016).

Modbus je široce využíván ve výrobní sféře pro přenos dat mezi různými zařízeními, jako jsou programovatelné logické kontroléry a rozhraní člověk-stroj (HMI). Výzkumníci navrhli různé přístupy k využití Modbus pro řízení procesů, včetně vytváření digitálních dvojčat pro detekci útoků typu "odmítnutí služby" na Modbus TCP a realizaci pokročilých systémů sběru a monitorování dat v reálném čase.

Modbus nadále hraje klíčovou roli v oblasti automatizace, poskytuje spolehlivý a osvědčený mechanismus pro výměnu dat mezi zařízeními. Zlepšení v oblasti bezpečnosti a rozvoj nových metod použití Modbus potvrzují jeho význam v arzenálu moderních technologií automatizace. Pokračující výzkum a vývoj v oblasti zlepšení bezpečnosti a funkčnosti Modbus napomáhá jeho přizpůsobení požadavkům průmyslového internetu věcí (IIoT), zdůrazňuje jeho důležitost pro budoucnost průmyslové automatizace.

Modbus přes sériovou linku (Modbus RTU a Modbus ASCII) je jeden z hlavních protokolů pro výměnu dat v průmyslové automatizaci. Slouží k přenosu informací mezi

různými zařízeními pomocí sériových rozhraní RS-232 nebo RS-485. Tento protokol se často používá pro komunikaci s programovatelnými logickými automaty (PLC), senzory, výstupními zařízeními a dalšími prvky systémů řízení a monitorování (Borden & Cox, 2022).

Jednoduchost a efektivita: Modbus RTU (Remote Terminal Unit) a Modbus ASCII (American Standard Code for Information Interchange) poskytují snadný a spolehlivý způsob výměny dat mezi zařízeními. Modbus RTU používá binární kódování dat, což jej činí efektivním pro přenos přes omezená sériová rozhraní. Naopak Modbus ASCII kóduje data ve formě textu, což usnadňuje ladění a pochopení přenášených zpráv.

Fyzická vrstva: Modbus přes sériovou linku pracuje na úrovni fyzického přenosu dat přes sériová rozhraní RS-232 nebo RS-485. To znamená, že je možné ho implementovat na různých typech hardwaru, včetně kabelů a konektorů, což jej činí flexibilním pro použití v různých průmyslových prostředích (Bolton, 2015).

Stejně jako Modbus TCP, standardní implementace Modbusu přes sériovou linku nezahrnuje šifrovací nebo autentizační mechanismy ve výchozím nastavení. To může způsobit zranitelnost komunikace vůči neoprávněnému přístupu a útokům (Borden & Cox, 2022).

Modbus přes sériovou linku zůstává populárním výběrem pro komunikaci v průmyslových systémech díky své jednoduchosti a spolehlivosti. Je široce využíván pro monitorování a řízení výrobních procesů, sledování zařízení, sběr dat a další automatizační úkoly.

2.5.2 Profibus a Profinet

Profibus (Process Field Bus) je otevřený standard digitální sítě, určený pro vzájemné propojení komponent procesní automatizace, jako jsou senzory, aktuátory a programovatelné logické kontroléry, v průmyslovém prostředí. Architektura Profibus je založena na modelu klient/server (Francisco, 2000).

Server v síti Profibus, který plní funkce řídicího kontroléru, řídí komunikaci s klienty, včetně pohonů, motorů, zařízení vstupu/výstupu a robotů. Pro připojení k zařízení pro Profibus je nutná konfigurovaná síť, nastavená adresa zařízení a komunikační kanál

se serverem Profibus, jako je PLC. Po připojení mohou zařízení bez problémů vyměňovat data a příkazy.

S rozvojem požadavků na průmyslové sítě začíná Profibus ustupovat modernějším protokolům, jako jsou Modbus TCP, Ethernet/IP a Profinet. Profinet, založený na Ethernetové sběrnici s otevřenou a standardizovanou architekturou, nabízí významné výhody oproti svému předchůdci, včetně vyšších rychlostí přenosu dat, zvýšené flexibility a lepší škálovatelnosti. Migrace z Profibus na Profinet umožnila průmyslovým systémům využít výhody Ethernetem orientované komunikace, což vedlo ke zvýšení efektivity, lepšímu propojení a zjednodušení integrace systémů. Profinet se prosadil jako standard spolehlivé a efektivní komunikace v průmyslovém prostředí (Francisco, 2000).

Profibus, jako starší protokol, původně nezahrnuje mechanismy šifrování a autentizace, což jej činí zranitelným vůči bezpečnostním hrozbám. Na rozdíl od toho Profinet nabízí pokročilé bezpečnostní funkce, včetně autentizace prostřednictvím certifikátů X.509 a ochrany heslem, stejně jako šifrování pomocí TLS nebo Secure Real-Time Transport Protocol (SRTP). Tato opatření zajišťují důvěrnost a integritu dat, efektivně chránící je před neoprávněným přístupem a manipulací s daty (Francisco, 2000).

2.5.3 Ethernet/IP

Ethernet/IP, který stojí v čele průmyslové automatizace, je pokročilý komunikační protokol založený na technologii Ethernet. Tento protokol umožňuje přenos dat v reálném čase mezi zařízeními různých výrobců a technologií, hrající klíčovou roli v řídících operacích na výrobních zařízeních vyžadujících rychlý a objemný přenos dat.

Protokol Ethernet/IP je postaven na architektuře klient/server a je široce používán v aplikacích řízení v rámci výrobních závodů. Pro implementaci Ethernet/IP musí být různá zařízení, jako jsou senzory, aktuátory a kontroléry, připojena ke společné síti a vyměňovat data mezi sebou pro koordinaci výrobních operací. To vyžaduje znalost IP adres a jmen těchto zařízení a jejich konfiguraci pro komunikaci v určených časových intervalech, po kterých se naváže TCP/IP spojení se zařízením pro výměnu vstupní/výstupní zprávy (Gal, n.d.).

Ethernet/IP nabízí řadu výhod, včetně rychlého přenosu dat, škálovatelnosti pro integraci široké škály zařízení a snadnosti nastavení a odstraňování závad. Jako standardizovaný protokol je také snazší udržovat. Ve srovnání s Modbus TCP, Ethernet/IP nabízí robustnější bezpečnostní funkce, podporující autentizační mechanismy, včetně autentizace založené na uživatelském jménu a hesle pro ověření pravosti zařízení a uživatelů. Kromě toho Ethernet/IP podporuje IPSec (Internet Protocol Security), který zajišťuje důvěrnost, integritu a autentizaci IP komunikací. Díky implementaci IPSec mohou být data vyměňována mezi zařízeními šifrována, což je chrání před neoprávněným přístupem (Gal, n.d.).

2.5.4 OPC-UA

OPC-UA (OPC Unified Architecture) je pokročilý multiplatformní komunikační protokol, vyvinutý pro bezpečný a spolehlivý přenos dat v oblasti průmyslové automatizace. OPC-UA je postaven na architektuře klient/server, zahrnující jeden nebo více OPC serverů a klientů. To umožňuje zajistit stálý tok dat mezi množstvím zařízení a řídícími aplikacemi, stejně jako sloužit jako komunikační prostředek mezi systémy SCADA a senzory. Dvoucestné spojení a trvalé relace jsou základem pro aktivní a nepřetržitou komunikaci mezi klienty a servery (Adl, 2023).

OPC-UA nabízí takové výhody, jako je vysoká úroveň bezpečnosti, schopnost přenášet velké objemy dat v reálném čase a vysokou škálovatelnost. Technologická nezávislost zajišťuje kompatibilitu s zařízeními a platformami od různých výrobců a operačních systémů. Nicméně, implementace OPC-UA v prostředích s velkým množstvím zařízení může být složitá a nákladná (Adl, 2023).

Bezpečnost byla základem OPC-UA, což zajišťuje komplexní bezpečnostní funkce. Protokol podporuje bezpečnostní protokoly transportní vrstvy, jako jsou TLS/SSL, což umožňuje šifrování a autentizaci pro bezpečný přenos dat. OPC-UA také zahrnuje mechanismy kontroly přístupu, umožňující administrátorům definovat detailní přístupové politiky pro uživatele a zařízení (Adl, 2023).

OPC-UA se ujal jako standard pro výměnu dat pro Průmysl 4.0, nabízející spolehlivý a bezpečný mechanismus pro integraci různých průmyslových zařízení a systémů. Díky

tomu mohou podniky implementovat pokročilé strategie automatizace a prediktivní údržby, výrazně zvyšující efektivitu a spolehlivost výrobních procesů (Adl, 2023).

3. Metodika

Metodika této práce definuje základní fáze a metody používané k dosažení stanovených cílů a řešení problémů. Předložený přístup zajišťuje vědecký charakter práce a umožňuje posoudit spolehlivost a reprodukovatelnost získaných výsledků.

Cílem této práce je vyvinout a otestovat systém testovacího zařízení založený na programovatelném logickém řadiči Wago Compact Controller 100 pro hodnocení a zlepšení funkcí ochrany a regulace zdrojů energie. Hlavním úkolem je automatizace procesu testování s cílem zvýšit přesnost a spolehlivost měření a zjednodušit monitorování a řízení prostřednictvím vytvořeného webového rozhraní. Součástí práce je také studium teorie fungování PLC.

Hypotézy práce:

• Použití programovatelného logického řadiče Wago Compact Controller 100 výrazně zlepší proces testování systémů ochrany a regulace zdrojů energie.

• Automatizace testování s využitím PLC a vývoj webového rozhraní pro monitorování a řízení zjednoduší a zrychlí proces testování.

Pro dosažení stanovených cílů a ověření hypotéz byly použity následující metody:

• Studium a analýza požadavků na testovací zařízení: identifikace klíčových funkčních a technických požadavků na základě analýzy existujících řešení a specifikací energetických zařízení.

• Výběr programovatelného logického řadiče Wago Compact Controller 100: zhodnocení vlastností a možností řadiče pro použití v rámci této práce, včetně hodnocení výkonu, komunikačních rozhraní a podpory softwaru.

• Vývoj programového kódu v jazyce Structured Text (ST) v prostředí Codesys: programování logiky testovacího zařízení včetně algoritmů generování signálů a zpracování měřených dat.

• Vytvoření webového rozhraní pro monitorování a řízení: vývoj uživatelského rozhraní na základě platformy Codesys umožňujícího snadný přístup k ovládání testovacího zařízení a zobrazování výsledků testů v reálném čase.

• Testování a optimalizace systému: provedení série testů pro ověření funkčnosti a spolehlivosti vyvinutého řešení, analýza získaných dat a úpravy pro zlepšení výkonu systému.

K ověření spolehlivosti výsledků práce slouží analýza získaných dat po provedení testů. Důvěryhodnost výsledků je potvrzena porovnáním s očekávanými výkonnostními ukazateli systému a srovnáním s analogickými řešeními.

Metodika práce představuje posloupnost kroků založených na teoretické analýze a praktické realizaci, což umožňuje dosažení stanovených cílů a ověření formulovaných hypotéz.

.

4 Praktická část

4.1 Analýza požadavků a výběr komponent

4.1.1 Funkční požadavky

Při identifikaci klíčových funkcí a možností pro navrhovaný systém testovacího zařízení založený na programovatelném logickém řadiči Wago Compact Controller 100 je třeba zohlednit následující konkrétní funkční požadavky, které budou sloužit jako základ pro návrh a implementaci systému.

1) Flexibilní generace signálů:
    - Systém musí umožňovat generování široké škály signálů s definovanými parametry pro simulaci různých provozních podmínek i závažných událostí, které mohou nastat u zdroje energie.
    - Důraz je kladen na flexibilitu nastavení parametrů generovaných signálů pro efektivní testování a simulaci různých scénářů.
2) Přesné měření reakčního času:
    - Implementace precizního měření času odezvy systému na změnu podmínek, zejména času od detekce signálu o překročení kritických hodnot do aktivace ochranných reakcí.
    - Zajištění vysoké přesnosti měření pro validní vyhodnocení výkonnosti a spolehlivosti systému řízení.
3) Intuitivní webové rozhraní pro testování:
    - Vytvoření uživatelsky přívětivého webového rozhraní umožňujícího operátorům pohodlné řízení testů, sledování procesů a získávání výsledků prostřednictvím webového prohlížeče.
    - Webové rozhraní musí být přístupné a responzivní, umožňující dálkové monitorování a řízení z libovolného místa v síti.

Tyto funkční požadavky reflektují klíčové požadavky identifikované při analýze systémových požadavků a slouží jako základ pro návrh a realizaci komplexního a účinného testovacího zařízení pro řízení zdroje energie.

4.1.2 Technické požadavky

Při stanovování technických specifikací navrhovaného testovacího zařízení založeného na programovatelném logickém řadiči Wago Compact Controller 100 je třeba zohlednit konkrétní technické požadavky, které zajistí spolehlivý a efektivní provoz systému.

1) Sériový přenos dat:
   - Implementace podpory pro sériový přenos dat pro komunikaci s dalšími zařízeními a systémy.
   - Použití protokolu Modbus pro výměnu dat s externími zařízeními, zajištění spolehlivého spojení a kompatibility.
2) Webové rozhraní založené na HTML5:
   - Vytvoření moderního a adaptivního webového rozhraní s využitím HTML5 pro správu a monitorování testů.
   - Zajištění možnosti přístupu k rozhraní z libovolného počítače v místní síti, což přináší pohodlí a flexibilitu při používání.
3) Interakce s binárními vstupy:
   - Přítomnost 8 binárních vstupů pro příjem signálů z každého vstupu, které odrazují reakce zdroje energie.
   - Správné nastavení a zpracování signálů z binárních vstupů pro zajištění správného fungování systému monitorování a řízení.
4) Bezpečnost a dostupnost:
   - Zajištění vysoké úrovně bezpečnosti dat a přístupu k systému prostřednictvím webového rozhraní.
   - Zaručení důvěrnosti informací a ochrany před neoprávněným přístupem.
5) Technická dokumentace a podpora:
   - Vypracování podrobné technické dokumentace popisující instalaci, konfiguraci a provoz systému.
   - Poskytnutí technické podpory a aktualizací softwaru pro zajištění stabilního a spolehlivého provozu systému.

Tyto technické požadavky definují hlavní charakteristiky a funkčnost navrhovaného testovacího zařízení, které zajistí jeho připravenost pro úspěšné nasazení a používání v rámci testování a řízení zdroje energie.

4.1.3 Výběr PLC: Kritéria výběru a zdůvodnění

Výběr vhodného PLC hraje klíčovou roli při vývoji testovacího zařízení. Tento krok je důležitým rozhodnutím, které ovlivňuje efektivitu a spolehlivost celého systému. Při výběru PLC je třeba zvážit řadu faktorů a kritérií, které zajistí úspěšné provedení stanovených úkolů.

PLC je základem automatizovaných systémů kontroly a řízení, a správný výběr zařízení určuje flexibilitu, spolehlivost a výkonnost celého testovacího systému. Pro efektivní rozhodnutí o výběru PLC je třeba analyzovat jeho technické specifikace, funkční možnosti a shodu s požadavky a cíli projektu.

V této části probereme hlavní kritéria výběru PLC pro testovací zařízení a vyjasníme důvody, proč bylo zvoleno konkrétní PLC - Wago Compact Controller 100. Podrobné zkoumání a zdůvodnění výběru umožní plně porozumět roli a významu PLC v kontextu našeho projektu.

Je důležité zohlednit následující kritéria při výběru vhodného PLC:

1) Technické specifikace:
    - Výkon a výkonnost kontroléru pro zajištění dostatečných prostředků k provedení všech úkolů.
    - Podpora potřebných komunikačních rozhraní (např. Ethernet, Modbus, CAN bus) pro integraci s ostatními systémy.
2) Spolehlivost a trvanlivost:
    - Spolehlivost a stabilita provozu PLC v různých provozních podmínkách.
    - Trvanlivost a odolnost vůči okolním podmínkám, včetně teplotních změn, vlhkosti a vibrací.
3) Softwarová podpora:
    - Přítomnost pohodlného a silného softwaru pro vývoj a ladění řídícího programu.
    - Podpora a aktualizace softwaru ze strany výrobce PLC.
4) Flexibilita a rozšiřitelnost:
    - Možnost rozšíření funkčnosti a připojení dalších modulů v budoucnosti.

- Podpora přidání nových vstupů/výstupů a rozšíření možností systému bez zásadních změn v hardwarové konfiguraci.

4) Poměr cena-kvalita:
   - Hodnocení poměru ceny kontroléru k jeho specifikacím a možnostem.
   - Volba optimálního řešení, které odpovídá rozpočtovým omezením projektu.

Během výběru PLC pro vývoj testovacího zařízení bylo zváženo mnoho řešení od různých výrobců a PLC. Nicméně konečné rozhodnutí padlo ve prospěch PLC Wago Compact Controller 100 z následujících důvodů.

Nejprve rozhodnutí ovlivnila zkušenost s kontroléry Wago, která již byla k dispozici. Tento faktor výrazně usnadnil proces integrace nového zařízení do stávající infrastruktury a umožnil vyhnout se zbytečným nákladům na čas a zdroje při zkoumání jiných alternativ.

Dále analýza výběrových kritérií ukázala, že Wago Compact Controller 100 plně splňuje všechny požadované specifikace a funkční možnosti nezbytné pro realizaci úkolu. Tímto způsobem bylo PLC Wago Compact Controller 100 vybráno jako optimální řešení, které splňuje všechna nezbytná kritéria.

4.2 Příprava na vývoj programu

4.2.1 PLC software

PLC Software jsou speciální programy pro tvorbu řídicích programů pro průmyslové regulátory. Jsou potřeba k automatizaci různých mechanismů a systémů ve výrobním prostředí. Pomocí PLC Software vývojáři vytvářejí programy, které řídí fungování regulátorů, přijímají data z senzorů a ovládají výstupní zařízení.

Tyto programy se často vytvářejí pomocí grafických programovacích prostředků, kde je snadné popsat řídící logiku, například pomocí logických schémat nebo textových algoritmů.

PLC Software také umožňuje integrovat regulátory s jinými systémy a zařízeními, vyměňovat data pomocí různých komunikačních protokolů, jako je Ethernet nebo

Modbus. Tyto programy mohou také vytvářet uživatelská rozhraní pro operátory k monitorování a ovládání procesů.

Existuje několik softwarových nástrojů určených pro vývoj softwaru pro PLC. Tyto programy poskytují pohodlné prostředí pro tvorbu, ladění a načítání řídicích programů do PLC. Zde jsou některé z nejběžnějších programů pro vývoj PLC:

CODESYS je jedním z nejoblíbenějších a nejsilnějších nástrojů pro programování PLC. Podporuje různé programovací jazyky, včetně logických schémat, textových jazyků (např. Structured Text) a grafických jazyků, jako je Function Block Diagram. CODESYS poskytuje široké možnosti pro tvorbu složitých řídicích programů.

TIA Portal je vyvinut firmou Siemens a je integrované prostředí pro vývoj PLC a dalších automatizačních zařízení od Siemens. Nabízí intuitivní rozhraní a nástroje pro tvorbu programů pomocí jazyků jako je Ladder Logic, Structured Text a Structured Control Language.

RSLogix je software vyvinutý pro programování PLC od společnosti Rockwell Automation. Podporuje různé programovací jazyky, včetně Ladder Logic, Structured Text a Sequential Function Chart. RSLogix poskytuje nástroje pro tvorbu složitých řídicích programů pro široké spektrum průmyslových aplikací.

Twincat je vyvinut společností Beckhoff a slouží k programování a řízení PLC od firmy Beckhoff. Podporuje různé programovací jazyky včetně IEC 61131-3 a umožňuje tvorbu programů pro širokou škálu průmyslových systémů a řešení.

Volba CODESYS jako softwarového nástroje pro vývoj softwaru pro PLC je podložena řadou faktorů, které dělají tento nástroj zvláště vhodným pro diplomovou práci.

Důležitým aspektem volby CODESYS je jeho široká kompatibilita s různými modely a výrobci PLC. To mi umožňuje snadno se přizpůsobit různým požadavkům a vybrat optimální zařízení pro realizaci vyvinutých programů.

Dále je CODESYS velmi populární a široce rozšířený, což zajišťuje přístup k množství výukových materiálů, dokumentace a komunit vývojářů. To usnadňuje osvojení si nástroje a poskytuje možnost získat podporu a rady od zkušených odborníků.

Takže volba CODESYS pro bakalářskou práci je odůvodněna jeho silnými funkcemi, snadným použitím a rozsáhlou podporou v oblasti vývoje SW pro PLC.

V CODESYS verzi 2 byly pro tvorbu webových rozhraní potřeba znalosti Javy, protože nástroje pro vývoj rozhraní byly omezené a vyžadovaly použití Java appletů k zobrazení webových stránek na PLC. To přinášelo určité obtíže a vyžadovalo specifické programovací dovednosti.

Nicméně s vydáním CODESYS verze 3 byl tento přístup podstatně zjednodušen a modernizován. V CODESYS 3 již vývojáři nepotřebují znalosti Javy pro tvorbu webových rozhraní. Místo toho CODESYS 3 poskytuje intuitivní nástroje pro tvorbu uživatelských webových rozhraní, které lze vytvářet pomocí standardních webových technologií, jako jsou HTML5, CSS a JavaScript.

To umožňuje vývojářům vytvářet moderní a uživatelsky přívětivá webová rozhraní bez potřeby dalších znalostí nebo specifických programovacích dovedností. Rozhraní vytvořená v CODESYS 3 lze snadno upravit a přizpůsobit konkrétním potřebám uživatelů a požadavkům projektu, přičemž zůstává jednoduché na použití a má vysokou funkčnost.

Takže s přechodem na CODESYS 3 se vývoj webových rozhraní pro PLC stal dostupnější a jednodušší, otevírají se nové možnosti pro tvorbu moderních řídicích a monitorovacích aplikací.

4.2.2 Upřesnění logiky fungování systému

Programový software pro řízení a monitorování testovacích procedur. Software se vyvíjí pro řízení a monitorování testovacích postupů, které zahrnují měření vstupních signálů, řízení časových intervalů a analýzu výsledků. Náš systém musí nabídnout možnost flexibilního nastavení testovacích parametrů prostřednictvím webového rozhraní a automatické zpracování výsledků na základě nasbíraných dat.

Komponenty systému:

1) PLC bude použit k řízení externích zařízení a vstupů-výstupů. PLC bude programováno pomocí specializovaného programovacího jazyka podporujícího logické operace a spouštěče.
2) Hlavní PLC program (PLC_PRG) bude hlavním řadičem, který řídí provádění testovacích procedur. V PLC_PRG budou deklarovány proměnné pro uchování stavů testů, spouštěče pro detekci signálních hran a časovače pro řízení časových intervalů.
3) Testovací program (Test_PRG) bude obsahovat logiku konkrétních testů, které chceme provést. V této programu budou definovány kroky testování včetně podmínek spouštění spouštěčů, nastavení časových prodlev a analýzy výsledků testu.
4) Systém bude mít webové rozhraní pro interakci s uživatelem. Webové rozhraní umožní nastavovat parametry testů (například počáteční podmínky, časové intervaly a rychlosti) a spouštět testovací procedury. Výsledky testů budou vizualizovány ve webovém rozhraní pro analýzu a další zpracování.

Očekávaný výsledek: získat fungující systém, který umožní provádět testování řízených procesů, ovládat testovací parametry prostřednictvím webového rozhraní a analyzovat výsledky pro optimalizaci výrobních procesů. Tento systém by měl být flexibilní pro nastavení a škálovatelný pro různé požadavky a pracovní podmínky.

4.2.3 Celkový algoritmus programu

Inicializace: program se inicializuje s počátečními podmínkami, včetně deklarace proměnných a vstupně-výstupních zařízení.

Inicializace spouštěčů: pro každý vstup se vytvářejí instance spouštěčů F_TRIG a R_TRIG, které slouží k detekci signálních hran (náběžných a sestupných).

Inicializace časovačů: vytváří se instance časovače (Timer), která bude sloužit k měření časových intervalů.

Cyklus provádění: program vstupuje do cyklu provádění, kde monitoruje stav vstupů a řídí testovací procedury v závislosti na nastavených podmínkách.

Zpracování vstupních signálů: program kontroluje stav každého vstupu (bi_DI1 až bi_DI8). Pro každý vstup se kontroluje spouštěče F_TRIG a R_TRIG. Pokud nastane náběžná hrana signálu (rising edge) na vstupu (F_TRIG.Q se stane TRUE), program zaznamenává aktuální čas do ET_TimerX (kde X je číslo vstupu). Pokud nastane sestupná hrana signálu (falling edge) na vstupu (R_TRIG.Q se stane TRUE), program zaznamenává aktuální čas do ET_TimerX.

Řízení testovacích procedur: při spuštění spouštěčů program provádí určité akce v závislosti na aktuálním stavu systému (State). Například, pokud dojde ke spuštění spouštěče pro jeden z vstupů (F_TRIG1.Q nebo R_TRIG1.Q), program uloží aktuální čas do ET_Timer1 a nastaví příznak WasTriggered1 na TRUE.

Řízení výstupů: v závislosti na aktuálním stavu systému (State) a spuštění spouštěčů program řídí výstupy (DO1 a DO2). Například, pokud je State rovno 1000, pak DO1 je nastaveno na hodnotu TimerPulse.Q a DO2 na NOT DO1. Pokud je State rovno 11, pak je DO1 nastaveno na TRUE a DO2 na FALSE.

Zastavení procesu: pokud nastane událost ZastavTest nebo uplyne čas časovače (Timer.Q), program nastaví State na 0 pro zastavení procesu.

4.2.4 Obecný přístup k tvorbě webového rozhraní

Rozhraní pro nastavení testovacích parametrů:uživatel by měl mít možnost zadávat počáteční podmínky, časy, rychlosti a další charakteristiky potřebné pro provádění testů. Na rozhraní by měla být pole pro nastavení počátečních hodnot proudu, času změn a rychlosti pohybu a dalších parametrů souvisejících s různými fázemi testovacích procedur.

Řízení testů: webové rozhraní by mělo poskytovat tlačítka nebo ovládací prvky pro spuštění různých testů. uivatel by měl mít možnost vybrat a spustit konkrétní testovací procedury, jako je testování podle trvání nebo podle rychlosti pohybu.

Zobrazení výsledků: po dokončení testu by rozhraní mělo zobrazovat výsledky a data shromážděná během zkoušky. Na rozhraní by mohly být zobrazeny grafy, tabulky nebo jiné vizuální prvky pro názorné zobrazení výsledků testování, včetně času odepnutí vstupních signálů (DI) pro každý test.

Interaktivita a jednoduchost použití: webové rozhraní by mělo být intuitivní a snadno použitelné. Uživatelská zkušenost by měla být optimalizována pro řízení testovacích procedur, zadávání parametrů a analýzu výsledků.

4.3 Vývoj programu

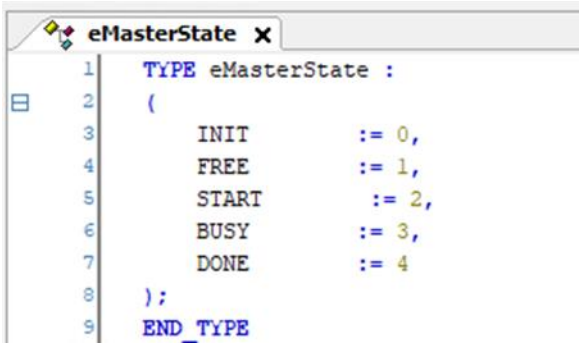
4.3.1 Vytváření pomocných prvků programu

1) **Enum eMasterState**

Typ eMasterState definuje stav master v programu. Obsahuje následující prvky (viz obrázek 11):

- INIT (0): Počáteční stav.
- FREE (1): Stav volnosti.
- START (2): Stav spuštění.
- BUSY (3): Stav zaneprázdnění.
- DONE (4): Stav dokončení.

Obrázek 11: Enum eMasterState



Zdroj: Zpracované autorem

Tento typ se používá k sledování aktuálního stavu mastera v systému.

2) **Union FourByteUnion**

Typ FourByteUnion je spojení dat umožňující reprezentovat 4-bajtové celé číslo různými způsoby. Obsahuje následující prvky (viz obrázek 12):

- (DWORD): 32-bitové celé číslo.
- bytes (ARRAY[0..3] OF BYTE): Pole 4 bajtů.
- words (ARRAY[0..1] OF WORD): Pole pro dva 16-bitových čísel).

Obrázek 12: Union FourByteUnion

```
FourByteUnion
1  TYPE FourByteUnion :
2  UNION
3      i: DWORD;
4      bytes: ARRAY[0..3] OF BYTE;
5      words: ARRAY[0..1] OF WORD;
6  END_UNION
7  END_TYPE
```

Zdroj: Zpracované autorem

Tento typ je užitečný pro převod mezi celými čísly a jejich bajtovým zobrazením.

3) Struktura typMasterQuery

Typ typMasterQuery reprezentuje dotaz master v programu. Zahrnuje následující prvky (viz obrázek 13):

- oQuery (typMbQuery): Struktura dotazu Modbus.
- oResponse (typMbResponse): Struktura odpovědi Modbus.
- eState (eMasterState): Stav provedení dotazu.
- sError (STRING): Zpráva o chybě.
- xError (BOOL): Příznak chyby (true = chyba, false = žádná chyba).

Obrázek 13: Struktura typMasterQuery

```
typMasterQuery
1  TYPE typMasterQuery :
2  STRUCT
3      oQuery        : typMbQuery;
4      oResponse     : typMbResponse;
5      eState        : eMasterState;
6      sError        : STRING;
7      xError        : BOOL;
8  END_STRUCT
9  END_TYPE
```

Zdroj: Zpracované autorem

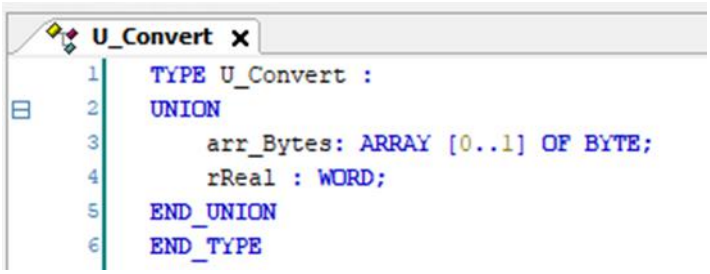
Tento typ se používá k provádění dotazů na zařízení Modbus a sledování stavu provedení.

4) Union U_Convert

Typ U_Convert představuje spojení dat pro převod mezi různými typy. Obsahuje následující prvky (viz obrázek 14):

- arr_Bytes (ARRAY [0..1] OF BYTE): Pole 2 bajtů.
- rReal (WORD): 16bitové celé číslo, které slouží k reprezentaci reálného čísla.

Obrázek 14: Union U_Convert



Zdroj: Zpracované autorem

Tento typ je užitečný pro konverzi mezi bajtovými poli a celými čísly.

4.3.2 Zápis přes Modbus

Tento příklad ukazuje proces zápisu dat přes Modbus s použitím určených parametrů a komunikačního zařízení.

Na obrázku 15 se nastavují počáteční parametry pro komunikaci pomocí protokolu Modbus. Pokud jde o první inicializaci, nastaví se rychlost přenosu dat, nastaví se adresa zařízení, funkční kód, adresa zápisu, počet registrů k zápisu a data k zápisu. Poté se příznak StartedInit nastaví na FALSE, aby se zabránilo opětovné inicializaci.

Obrázek 15: Nastavení komunikačních parametrů před zahájením operace

```
IF StartedInit THEN
    // Nastavení komunikačních parametrů před zahájením operace
    mySerialMaster.udiBaudrate := 57600;

    // Vyplnění struktury dotazu Modbus pro zápis dat
    ExtQuery.bUnitId := 60;
    ExtQuery.bFunctionCode := 16#10; // Kód funkce pro zápis do více registrů
    ExtQuery.uiWriteAddress := 0; // Počáteční adresa pro zápis
    ExtQuery.uiWriteQuantity := PocetRegistru; // Počet registrů k zápisu
    ExtQuery.awWriteData := Data; // Data k zápisu

    StartedInit := FALSE;
END_IF
```

Zdroj: Zpracované autorem

Na obrázku 16 se připravují data pro zápis do registrů Modbus. Každý prvek pole Dat se vyplní konkrétní hodnotou, jako je oddělovací část, amplituda, šířka pulzu, proud atd. Tato data se obvykle připravují na základě aktuálních parametrů nebo externích vstupů.

Obrázek 16: Sestavení dat k zápisu do registrů Modbus

```
CASE State OF
    0:
        // ... kód byl skryt pro snazší zobrazení ...
    1:
        // ... kód byl skryt pro snazší zobrazení ...

         // Sestavení dat k zápisu do registrů Modbus
        Data[0] := DivFirstPart;
        Data[1] := DivSecondPart;
        Data[2] := REAL_TO_WORD(PocatecniAmplitude*10000);
        Data[3] := SHL(Func3(PulseWidthValues[PocatecniPulseWidthItem], PolarityValues[PolarityItem]),8);
        Data[4] := 0;
        Data[5] := REAL_TO_WORD(10000 * CurrentHodnota);
        Data[6] := REAL_TO_WORD(10000 * PulseCharge);
```

Zdroj: Zpracované autorem

Obrázek 17 představuje volání funkce pro odeslání požadavku pomocí protokolu Modbus. Funkce mySerialMaster se volá s určitými parametry, včetně nastavení spojení, portu komunikace, dat požadavku, spouštěcí podmínky a obslužných rutin pro chyby. Tento volání spouští proces přenosu dat pomocí Modbus.

Obrázek 17: Volání funkce zápisu přes Modbus

```
511   // Volání funkce zápisu přes Modbus
512   mySerialMaster(
513       xConnect:= TRUE, // Navázání spojení
514       I_Port  :=  IoConfig_Globals.COM1, // Výběr komunikačního portu
515       utQuery :=  ExtQuery, // Předání struktury dotazu
516       xTrigger    := GVL.Start, // Spuštění operace
517       utResponse  :=  utResponse, // Předání struktury odpovědi (pokud je potřeba)
518       xIsConnected =>, // Získání informace o spojení (pokud je potřeba)
519       xError => ErrorBool, //  Získání informace o chybě
520       oStatus => // Získání stavu operace (pokud je potřeba)
521   );
```

Zdroj: Zpracované autorem

Popis parametrů:

1) **StartedInit:** flag, který označuje začátek inicializace před zápisem přes Modbus.
2) **mySerialMaster:** instance modbusového mastera pro odesílání dotazů.
3) **ExtQuery:** struktura dotazu obsahující parametry pro zápis.
4) **Stav:** proměnná určující aktuální stav pro sestavení dat zápisu.
5) **Data:** pole obsahující data k zápisu do registrů Modbus.
6) **IoConfig_Globals.COM1:** komunikační port, přes který probíhá výměna dat.
7) **GVL.Start:** globální flag, který označuje začátek testu.

V tomto příkladu se nastavují komunikační parametry (přenosová rychlost a další), sestavuje se dotaz s uvedením adresy, počtu registrů a samotných dat k zápisu a následně se volá funkce mySerialMaster k odeslání dotazu.

4.3.3 Vytváření cílového programu

Pro efektivnější řízení a podporu testovacího systému jsem se rozhodl rozdělit cílový program na dvě části: **PLC_PRG** a **Test_PRG**. To nám umožňuje lépe strukturovat kód a řídit testovací procesy s větší flexibilitou a jasností. Níže je popis každé části programu.

Program **PLC_PRG** je řídící program navržený pro interakci s externím zařízením prostřednictvím sériového portu. Hlavními úkoly programu je nastavení komunikačních parametrů, odesílání příkazů a dat na externí zařízení a zpracování událostí pro spuštění určitých operací.

Hlavní komponenty:

1) **State:** aktuální stav programu, který určuje aktuální fázi práce a operace.
2) **mySerialMaster:** komponenta pro řízení sériového portu a výměnu dat s externím zařízením.

Na obrázku 18 se inicializují parametry pro komunikaci pomocí protokolu Modbus. Nastavuje se rychlost přenosu dat, adresa zařízení, funkční kód dotazu, adresa zápisu, počet registrů k zápisu a data, která budou zapsána.

Obrázek 18: Inicializace

```
IF StartedInit THEN
    // Nastavení komunikačních parametrů před zahájením operace
    mySerialMaster.udiBaudrate := 57600;

    // Vyplnění struktury dotazu Modbus pro zápis dat
    ExtQuery.bUnitId := 60;
    ExtQuery.bFunctionCode := 16#10; // Kód funkce pro zápis do více registrů
    ExtQuery.uiWriteAddress := 0; // Počáteční adresa pro zápis
    ExtQuery.uiWriteQuantity := PocetRegistru; // Počet registrů k zápisu
    ExtQuery.awWriteData := Data; // Data k zápisu

    StartedInit := FALSE;
END_IF
```

Zdroj: Zpracované autorem

Na obrázku 19 dochází k resetování všech parametrů a vlajek před začátkem testu.

Obrázek 19: Resetování všech parametrů a vlajek

```
CASE State OF
    0:
        // Resetování všech parametrů a vlajek
        GVL.Start := FALSE;
        StartTimer:= FALSE;
        StartPulseTimer := FALSE;
        ZastavTest := FALSE;
        DO1 := FALSE;
        DO2 := NOT DO1;
        DO3 := FALSE;

        WasTriggered1 := FALSE;
        WasTriggered2 := FALSE;
        WasTriggered3 := FALSE;
        WasTriggered4 := FALSE;
        WasTriggered5 := FALSE;
        WasTriggered6 := FALSE;
        WasTriggered7 := FALSE;
        WasTriggered8 := FALSE;
```

Zdroj: Zpracované autorem

Na obrázku 20 připravují se data pro odeslání příkazu v závislosti na typu testu. Vypočítávají se počáteční hodnoty a shromažďují se data pro zápis do registrů Modbus.

Obrázek 20: Příprava dat k odeslání příkazu v závislosti na typu testu

```
1:
    // Příprava dat k odeslání příkazu v závislosti na typu testu
    FixedWrite := FALSE;

    DivBytes.i := REAL_TO_DWORD(20000000 / PocatecniFrequency);
    uConvert.arr_Bytes[0] := DivBytes.bytes[2];
    uConvert.arr_Bytes[1] := DivBytes.bytes[3];
    DivFirstPart := uConvert.rReal;
    uConvert.arr_Bytes[0] := DivBytes.bytes[0];
    uConvert.arr_Bytes[1] := DivBytes.bytes[1];
    DivSecondPart := uConvert.rReal;

     // Sestavení dat k zápisu do registrů Modbus
    Data[0] := DivFirstPart;
    Data[1] := DivSecondPart;
    Data[2] := REAL_TO_WORD(PocatecniAmplitude*10000);
    Data[3] := SHL(Func3(PulseWidthValues[PocatecniPulseWidthItem], PolarityValues[PolarityItem]),8);
    Data[4] := 0;
    Data[5] := REAL_TO_WORD(10000 * CurrentHodnota);
    Data[6] := REAL_TO_WORD(10000 * PulseCharge);

    WasTriggered1 := FALSE;
    WasTriggered2 := FALSE;
    WasTriggered3 := FALSE;
    WasTriggered4 := FALSE;
    WasTriggered5 := FALSE;
    WasTriggered6 := FALSE;
    WasTriggered7 := FALSE;
    WasTriggered8 := FALSE;
    StartTimer := FALSE;

    ExtQuery.uiWriteQuantity := 7;
    ExtQuery.uiWriteAddress := 0;

    StartMethod();
    State := 100;
```

Zdroj: Zpracované autorem

Na obrázku 21 se připravují data pro spuštění testu. Nastavují se cílové hodnoty, které budou použity během testu. Poté se spouští samotný test.

Obrázek 21: Příprava data k začátku testu

```
10:
    // Příprava dat k začatku testu
    SpustTestFixed := FALSE;

    DivBytes.i := REAL_TO_DWORD(20000000 / CilovaFrequency);
    uConvert.arr_Bytes[0] := DivBytes.bytes[2];
    uConvert.arr_Bytes[1] := DivBytes.bytes[3];
    DivFirstPart := uConvert.rReal;
    uConvert.arr_Bytes[0] := DivBytes.bytes[0];
    uConvert.arr_Bytes[1] := DivBytes.bytes[1];
    DivSecondPart := uConvert.rReal;

    // Sestavení dat k zápisu do registrů Modbus
    Data[0] := DivFirstPart;
    Data[1] := DivSecondPart;
    Data[2] := REAL_TO_WORD(CilovaAmplitude*10000);
    Data[3] := SHL(Func3(PulseWidthValues[CilovyPulseWidthItem], PolarityValues[PolarityItem]),8);
    Data[4] := 0;
    Data[5] := REAL_TO_WORD(10000 * CurrentHodnota);
    Data[6] := REAL_TO_WORD(10000 * PulseCharge);

    ET_Timer1 :=0.0;
    ET_Timer2 :=0.0;
    ET_Timer3 :=0.0;
    ET_Timer4 :=0.0;
    ET_Timer5 :=0.0;
    ET_Timer6 :=0.0;
    ET_Timer7 :=0.0;
    ET_Timer8 :=0.0;

    ExtQuery.uiWriteQuantity := 7;
    ExtQuery.uiWriteAddress := 0;

    StartWithTimerMethod();
    State := 100;
```

Zdroj: Zpracované autorem

Program Test_PRG je určen k zpracování událostí a řízení testovacích operací pomocí triggrů k detekci událostí a časovačů k řízení časových zpoždění. Hlavním úkolem programu je detekce událostí a řízení stavu testovacích operací.

Hlavní komponenty:

1) Triggery (F_TRIG1 až F_TRIG8, R_TRIG1 až R_TRIG8): používají se k detekci událostí.
2) Časovače (Timer, ET_Timer1): používají se k měření času a řízení zpoždění.
3) Stavové vlajky (WasTriggered1 až WasTriggered8): používají se k sledování stavu detekce událostí.

Příklad použití triggru (F_TRIG1) a časovače (Timer) k zpracování události a nastavení vlajky WasTriggered1 (viz obrázku 22):

Obrázek 22: Zpracování události detekce triggeru F_TRIG

```
26      // Zpracování události detekce triggru F_TRIG
27      IF (F_TRIG1.Q OR R_TRIG1.Q) AND NOT WasTriggered1 THEN
28          // Výpočet časového intervalu
29          ET_Timer1 := (TIME_TO_REAL(Timer.ET) / 1000);
30          // Nastavení vlajky označující, že událost byla zpracována
31          WasTriggered1 := TRUE;
32      END IF
```

Zdroj: Zpracované autorem

V tomto příkladu: F_TRIG1.Q kontroluje, zda byl triggr F_TRIG1 aktivován. WasTriggered1 se používá k zabránění opětovnému zpracování události. Když F_TRIG1.Q se stane TRUE (triggr je aktivován) a WasTriggered1 je FALSE (událost ještě nebyla zpracována), provede se blok kódu uvnitř podmínky IF.

V tomto bloku kódu:

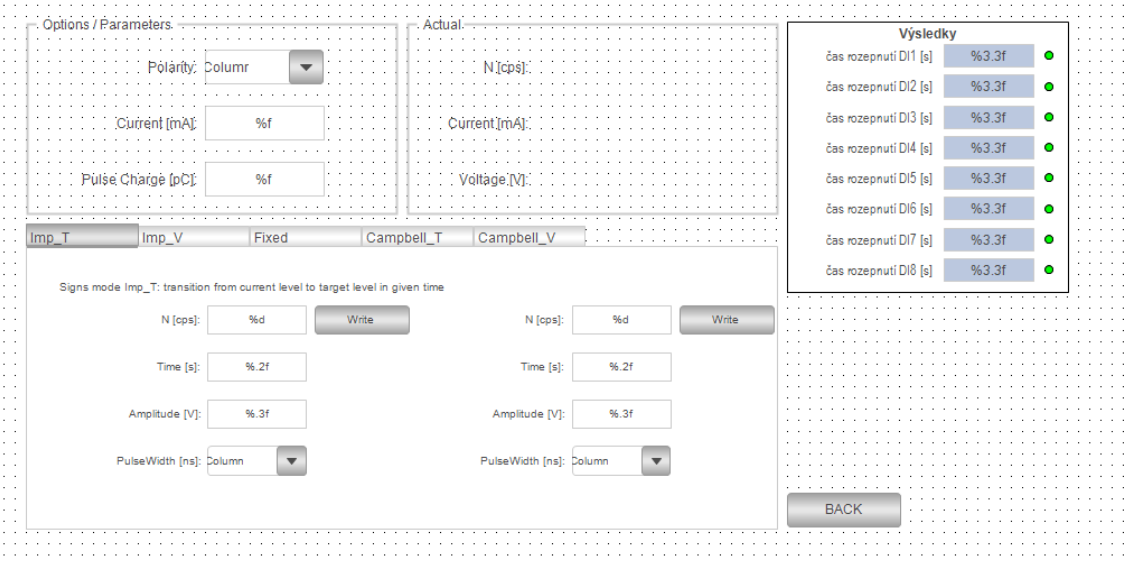
1) ET_Timer1 := (TIME_TO_REAL(Timer.ET) / 1000) ; vypočte aktuální hodnotu časového intervalu časovače Timer v sekundách.
2) WasTriggered1 := TRUE; nastaví vlajku WasTriggered1 na TRUE, aby bylo označeno, že událost byla zpracována.

Tento kód umožňuje zpracovávat události související s detekcí triggrů a měřením času pomocí časovačů v programu Test_PRG.

4.3.4 Vytváření webového rozhraní

Pro vytvoření webového rozhraní testovacího systému používáme nástroj WebVisu od Codesys (viz obrázek 23). Tento nástroj umožňuje integrovat standardní vizuální prvky, jako jsou tlačítka, textová pole, indikátory a další, což usnadňuje vytváření uživatelského rozhraní pohodlnějším a flexibilnějším.

Obrázek 23: Interface programu

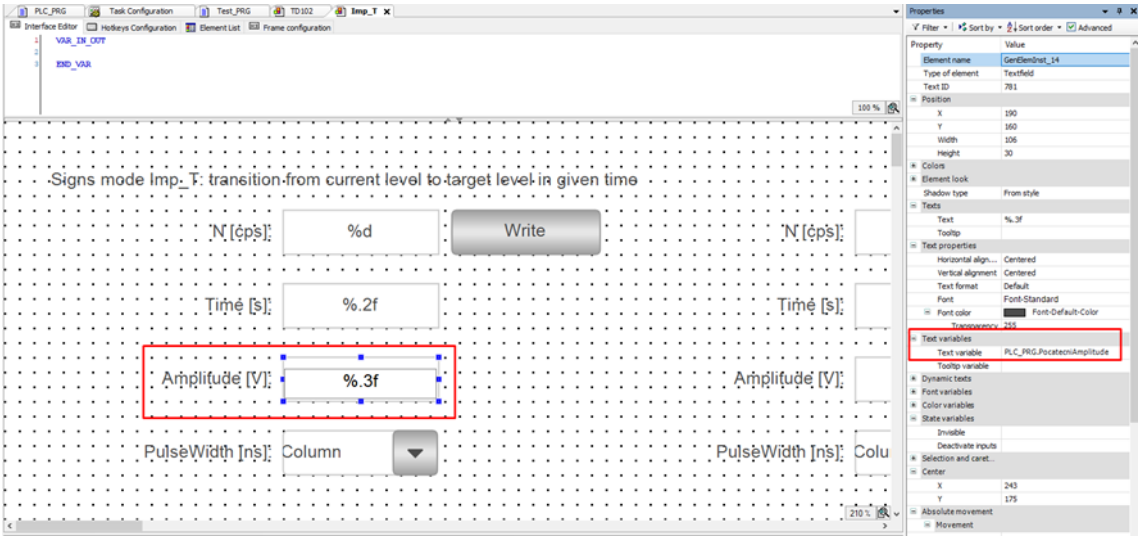


Zdroj: Zpracované autorem

Každý vizuální prvek lze snadno nastavit a propojit s konkrétní proměnnou nebo funkcí v programu. Nastavení probíhá prostřednictvím standardního editoru, kde definujeme vlastnosti prvků, jejich vzhled a chování.

Proces nastavení zahrnuje definici vlastností prvku, jako je text, barva, velikost a formát zobrazených dat. Například můžeme nastavit textové pole pro zobrazení hodnoty "Amplituda [V]:" (viz obrázek 24).

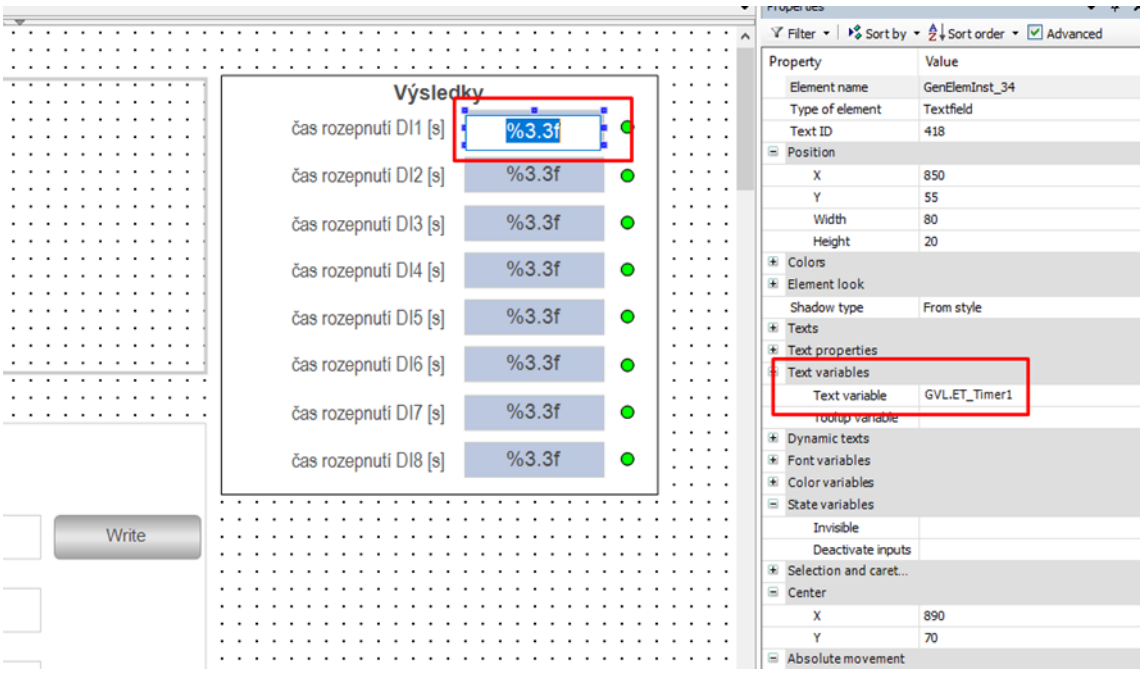
Obrázek 24: Nastavení vlastností textového pole



Zdroj: Zpracované autorem

Po provedení nastavení propojujeme prvky s proměnnými nebo funkcemi PLC. To umožňuje vizuálním prvkům zobrazovat aktuální data z programu PLC a dokonce s nimi interagovat (viz obrázek 25).

Obrázek 25: Nastavení vlastností textového pole



Zdroj: Zpracované autorem

Tento přístup zpřehledňuje webové rozhraní a zjednodušuje jeho používání, poskytuje efektivní monitorování a řízení testovacího systému. V tomto případě není třeba znalostí HTML5, protože veškeré vytváření probíhá prostřednictvím vizuálního editoru.

4.3.4 Testování

Před hlavním testováním zdroje energie jsme pečlivě připravili a provedli řadu předběžných testů, abychom zaručili bezproblémový průběh a úspěšné splnění cílů naší práce. Naše strategie testování byla založena na detailním zkoumání funkčnosti a spolehlivosti programovatelného logického kontroléru (PLC), který hrál klíčovou roli v řízení a regulaci zdroje energie.

Nejdříve jsme připojili PLC k našemu testovacímu zařízení a využili simulátor neutronové komory TD102. Tento simulátor nám umožnil provádět simulace různých scénářů změn výkonu zdroje energie, včetně skokových změn výkonu a změn výkonu s definovanou periodou reaktoru. Díky tomu jsme mohli ověřit schopnost PLC přizpůsobit se dynamickým podmínkám provozu a správně řídit zdroj energie.

Během předběžných testů jsme sledovali reakce PLC na různé změny a zajišťovali, aby PLC spolehlivě plnilo požadované úkoly. Důkladně jsme testovali jeho schopnost udržovat stabilní a přesný provoz zdroje energie v souladu s definovanými parametry. To zahrnovalo kontrolu jeho schopnosti správně reagovat na signály a provádět regulační operace v souladu s nastavenými parametry.

Pro vyhodnocení výstupů z neutronové instrumentace jsme použili speciální vyhodnocovací a měřicí přístroje, jako jsou N802S a N711C. Tyto přístroje nám poskytly možnost měřit klíčové parametry, jako je doba vystavení bezpečnostnímu signálu, což bylo nezbytné pro správné řízení zdroje energie v reálném provozu.

Dále jsme využili osciloskop k ověření nastavení simulátoru TD102 v impulzním režimu a monitorovali reakce PLC na vstupy a doby odezvy, včetně doby, kdy docházelo k rozepnutí bezpečnostního relé neutronové instrumentace. Tímto způsobem jsme získali důkladný a detailní pohled na funkčnost PLC a jeho integraci do procesu testování zdroje energie.

Díky předběžnému testování jsme měli jistotu, že PLC je připraveno na provádění základních funkcí řízení a kontroly během hlavních testů zdroje energie. Jeho spolehlivý provoz a schopnost plnit požadavky projektu nám poskytly důvěru v úspěšné dosažení cílů naší práce. Tato fáze předběžného testování byla klíčová pro zajištění bezproblémového průběhu hlavních testů a úspěšného dokončení celého projektu.

5 Závěr

V rámci této bakalářské práce byl proveden vývoj a testování systému testovacího zařízení založeného na programovatelném logickém kontroléru Wago Compact Controller 100 s cílem hodnotit a zlepšovat fungování systémů ochrany a regulace zdrojů energie. Během práce byly provedeny následující hlavní kroky:

1) Studium a analýza požadavků na testovací zařízení: byly stanoveny klíčové funkční a technické požadavky na základě analýzy existujících řešení a specifikací energetických zařízení. Toto umožnilo formulovat jasné úkoly a cíle práce.
2) Výběr programovatelného logického kontroléru Wago Compact Controller 100: byly studovány a zhodnoceny charakteristiky kontroléru pro použití v rámci práce. Zvláštní pozornost byla věnována výkonu, komunikačním rozhraním a podpoře potřebného softwaru.
3) Vývoj programového kódu ve strukturovaném textu v prostředí Codesys: bylo provedeno programování logiky testovacího zařízení včetně algoritmů generování signálů a zpracování měřících dat. To zajistilo automatizaci testovacího procesu s vyšší přesností a spolehlivostí měření.
4) Vytvoření webového rozhraní pro monitorování a řízení: byl vyvinut uživatelský rozhraní na základě Codesys pro pohodlný přístup k ovládání testovacího zařízení a zobrazení výsledků testů v reálném čase. Toto zjednodušilo proces monitorování a řízení systému.
5) Testování a optimalizace systému: byla provedena série testů pro ověření funkčnosti a spolehlivosti vyvinutého řešení. Získaná data byla analyzována a provedeny potřebné úpravy pro zlepšení funkčnosti systému.

Výsledkem práce bylo dosažení stanovených cílů, konkrétně vývoj funkčního systému testovacího zařízení na bázi programovatelného logického kontroléru Wago Compact Controller 100, který umožňuje automatizaci procesu testování a hodnocení fungování systémů ochrany a regulace zdrojů energie s vysokou přesností a spolehlivostí.

Další směry vývoje v této oblasti mohou zahrnovat rozšíření funkcionality systému, optimalizaci algoritmů a další zkoumání technických aspektů programování programovatelných logických kontrolérů pro zlepšení efektivity a spolehlivosti.

6 Seznam použitých zdrojů

352Lab VŠB. (n.d.). *Řídicí systémy*. Získáno 1. 4 2024, z 352Lab VŠB: http://352lab.vsb.cz/Podklady/03_PAR/R_PLC.html

Adl, A. R. (2023). Image processing and communication with PLC through OPC UA protocol. *The First National Conference on Research and Innovation in Artificial Intelligence.*

Amin, A., & Mridha, M. (2020). A mini view of PLC. *International Journal of Research in Advanced Engineering and Technology*, stránky 23-25.

Bolton, W. (2015). *Programmable Logic Controllers 6th Edition.* Newnes.

Borden, T., & Cox, R. (2022). *Technician's Guide to Programmable Controllers.* Cengage Learning, Inc.

Duranso, G. (4. Říjen 2021). *Basics of Structured Text (ST) Programming | Examples & Applications*. Získáno 5. 3 2024, z realpars.com: https://www.realpars.com/blog/structured-text

Francisco, V. (2000). Real-time fieldbus communications using Profibus networks. *IEEE Transactions on Industrial Electronics*. doi:10.1109/41.808018

Gal, N. (n.d.). *Remote Ethernet PLC Networking and Control.*

Hallak, G., & Bumiller, G. (2016). PLC for Home and Industry Automation. V *Power Line Communications: Principles, Standards and Applications from Multimedia to Smart Grid, Second Edition.* doi:10.1002/9781118676684.ch7

Hossameldin, E., & Ihab, A. (2019). RISC-V based implementation of Programmable Logic Controller. *FPGA for Industry 4.0*, 98-102.

Inst Tools. (n.d.). *PLC Sizes and Applications*. Získáno 10. 4 2024, z https://instrumentationtools.com/plc-sizes-and-applications/

International Atomic Energy Agency. (2018). *Advances in Small Modular Reactor Technology Developments: A Supplement to IAEA Advanced Reactors Information System.*

Oliveira, P. J., & Gaspar, J. (2020). *PLC Programming Languages.* Získáno 1. 04 2024, z http://users.isr.ist.utl.pt/~jag/aulas/api20b/docs/API_I_C3_2_IL.pdf

Olsson, G. (2005). Programmable Logic Controllers. V *Handbook of Networked and Embedded Control Systems.* Birkhäuser.

Parr, A. (2003). *Programmable controllers: an engineer's guide [3rd ed].* Newnes.

Peterson, D. (2. Březen 2022). *The Origin Story of the PLC*. Získáno 4. 4 2024, z https://control.com/technical-articles/the-origin-story-of-the-plc/

Posdzi, N. (2021). *PLC Programming and Applications.*

Rawson, M. (2022). *Petri Nets for Concurrent Programming.* doi:10.48550/arXiv.2208.02900

Richardson, D. (8. Říjen 2018). *PLC Analog Inputs and Signals*. Získáno 8. 4 2024, z realpars: https://www.realpars.com/blog/plc-analog-inputs

Rohner, P. (1996). *PLC Automation with Programmable Logic Controllers.* doi:10.1007/978-1-349-14267-5

Sudhir, A., Mudhalwadkar, R., & Shah, P. (2016). *Implementation of Modbus TCP communication between PLC and Matlab.*

WAGO. (n.d.). *Compact Controller 100*. Získáno 15. 3 2024, z wago.com: https://www.wago.com/global/automation-technology/discover-plcs/compactcontroller100

Ye, L. (Březen 2024). Design of PLC Electrical Control System. *Journal of Theory and Practice of Engineering Science*, stránky 134-162.

Seznam obrázků