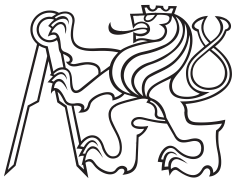
Bakalářská práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra řídicí techniky

Vývoj Knihovny a Testovacího Frameworku pro PLC od Schneider: Inovace v testování pro průmyslovou automatizaci

Jaroslav Künstler

Vedoucí: Ing. Ondřej Bělovský
Obor: Kybernetika a robotika
Květen 2024

Poděkování

Rád bych poděkoval vedoucímu práce Ing. Ondřeji Bělovskému, za pravidelné konzultace a užitečné rady při implementaci praktické části. Ing. Alexandru Osadcimu za pomoc při tvorbě zadání a další podporu během práce. Dále pak kolegům, kteří se formou konzultací podíleli na zlepšení praktcké části, jmenovitě Ing. Martinu Hunčovskému, Ing. Lukáši Kvardovi a Ing. Alexandu Zhigunovi.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne .....

.....
podpis autora práce

Abstrakt

Průmyslová automatizace je klíčovým prvkem moderních výrobních procesů, které se stále více spoléhají na technologické inovace pro dosažení efektivity a konkurenceschopnosti. Testování a ověřování funkcionality průmyslových řídicích systémů, zejména PLC (Programmable Logic Controller), je nezbytné pro zajištění jejich spolehlivosti a bezpečnosti.

Cílem této práce je vytvořit novou knihovnu pro rozšíření stávajícího testovacího frameworku, která umožní automatizované testování průmyslových systémů s využitím API Schneider PLC. Práce se zabývá prozkoumáním a porozuměním API, návrhem a implementací python balíčku pro integraci s testovacím frameworkem a realizací testovacích scénářů pro ověření funkcionality knihovny na reálném distribuovaném IO systému.

Klíčová slova: automatizace, distributed IO, PLC, testování

Vedoucí: Ing. Ondřej Bělovský
Siemens, s.r.o.,
Siemensova 2715/1,
Praha 13

Abstract

Industrial automation is a key element of modern manufacturing processes, which increasingly rely on technological innovations to achieve efficiency and competitiveness. Testing and verifying the functionality of industrial control systems, especially PLCs (Programmable Logic Controllers), are essential to ensure their reliability and safety.

This work aims to create a new library to extend the existing testing framework, enabling automated testing of industrial systems using the Schneider PLC API. The work involves exploring and understanding the API, designing and implementing a Python package for integration with the testing framework, and creating testing scenarios to verify the functionality of the library on a real distributed IO system.

Keywords: automation, distributed IO, PLC, verification

Title translation: Development of a library and Testing Framework for Schneider PLC: Innovation in testing for industrial automation

Obsah

1 Úvod **1**

1.1 Motivace 1

1.2 Cíl 2

1.3 Struktura práce 2

2 Technologie v průmyslové automatizaci **5**

2.1 PLC 5

2.1.1 Hardware 5

2.1.2 Software 8

2.1.3 Programování 11

2.2 Distributed IO 12

2.2.1 Hardware 12

2.2.2 Dělení 13

2.3 Komunikační standardy 16

2.3.1 Vybrané standardy 17

3 Analýza **21**

3.1 Popis testovací stanice 21

3.1.1 Zapojení 21

3.1.2 PLC 22

3.1.3 Testované zařízení 23

3.2 Nové funkcionality 24

3.3 Struktura testovacích scénářů . . 25

4 API Schneider PLC **27**

4.1 Použití 27

4.2 Struktura 28

4.3 Možnosti 28

4.4 Klíčové funkcionality 29

5 Implementace knihovny **31**

5.1 Logika 31

5.1.1 Rozbor konkrétního použití . 32

5.2 Struktura 34

6 Testovací scénáře **37**

6.1 Implementace 37

6.1.1 Start/Stop CPU 38

6.1.2 Změna RPI 39

6.1.3 Wirebreak 41

6.2 Funkčnost knihovny 42

7 Zhodnocení **43**

7.1 Navržené změny a řešení 43

7.2 Zhodnocení implementace 44

7.3 Měřitelné výsledky 45

8 Závěr **47**

8.1 Vyhodnocení 47

8.2 Možnosti rozšíření 48

A Literatura **49**

B Zadání práce **53**

Obrázky

2.1 PLC společnosti Siemens řady SIMATIC S7-1500. [21] 6

2.2 Diagram hardwaru PLC. [16] 7

2.3 Diagram výkonu programu v PLC. 9

2.4 Hierarchie volání POUs 10

2.5 Rozhraní programu Machine Expert 12

2.6 Porovnání Remote a Distributed IO. 14

2.7 Kompaktní vstupní/výstupní moduly. [20] 15

2.8 Vstupní/výstupní moduly ET 200AL s IP67/65. [19] 16

2.9 Ethernet/IP v kontextu OSI modelu [14] 18

3.1 Zapojení testovací stanice. 22

3.2 Použité PLC. 23

3.3 Použité zařízení SIMATIC ET 200SP. 24

6.1 Graf měření RPI 2 ms 40

Tabulky

6.1 Testovací scénář zastavení a spuštění PLC 38

6.2 Testovací scénář kontrola RPI . . 39

6.3 Testovací scénář wirebreak 41

7.1 Čas potřebný k provedení testů . 46

Kapitola 1

Úvod

1.1 Motivace

Moderní výrobní procesy se stále více opírají o technologické inovace a průmyslovou automatizaci jako klíčové prvky pro zajištění efektivity a udržení konkurenceschopnosti. Jedním z aspektů průmyslové automatizace je testování a validace funkcionality průmyslových řídicích systémů, zejména PLC (Programmable Logic Controller). Tato práce se zabývá rozšířením existujícího testovacího frameworku pro distributed IO zařízení o novou knihovnu, která umožní efektivní integraci s API od společnosti Schneider Electric.

Motivací pro tuto práci je potřeba automatizovat, a tím zefektivnit, testování průmyslových systémů, které je klíčové pro zajištění jejich spolehlivosti a bezpečnosti. Současný testovací framework je omezený v možnostech integrace s externími systémy a nedostatečně pokrývá potřeby současných aplikací. Rozšíření frameworku o novou knihovnu s podporou API Schneider Electric může významně zvýšit efektivitu a spolehlivost testovacích scénářů.

Tvorba nových testovacích scénářů zajistí pokrok v automatizaci testování a tím přispěje ke zvýšení efektivity práce. Sloužit budou také k testování funkčnosti knihovny a jako ukázka použití nového softwarového balíčku.

1.2 Cíl

Cílem této práce je vytvořit novou knihovnu pro rozšíření stávajícího testovacího frameworku pro PLC, která umožní integraci s API od společnosti Schneider Electric a tvorbu testovacích scénářů, využívajících tuto knihovnu. Konkrétně se práce zaměřuje na následující body:

- Prozkoumání a porozumění API Schneider PLC a identifikaci klíčových funkcionalit pro automatizované testování.
- Navržení a implementaci python balíčku využívajícího API Schneider PLC v rámci testovacího frameworku.
- Realizaci testovacího scénáře s využitím Schneider API na definovaném distributed IO systému a ověření očekávaných výsledků.

Knihovna by měla implementovat co možná nejvíce funkcí, zejména v oblasti hardwarové konfigurace, které se při testování využívají a měla by tak umožnit automatické vyhodnocení testovacích scénářů. Nicméně, zároveň je klíčové, aby použití této knihovny bylo pro uživatele co nejjednodušší. To znamená, že knihovna by měla být navržena s důrazem na jednoduchost použití a poskytovat intuitivní rozhraní. Uživatelé by měli být schopni snadno integrovat tuto knihovnu do svých existujících testů a využívat ji bez složité konfigurace nebo rozsáhlého učení.

1.3 Struktura práce

Práce je strukturována chronologicky dle postupu při implementaci požadovaného softwaru. V první části je uveden teoretický popis technologií průmyslové automatizace, následovaný analýzou technologií využítých v této práci. Poslední část práce je pak věnována samotné implementaci softwarového balíčku.

1. **Úvod:** Zde je představena problematika a cíle práce.
2. **Technologie v průmyslové automatizaci:** Poskytuje teoretický základ v oblasti průmyslové automatizace, čímž popisuje technologie využívané v této práci.

3. **Průzkum a Analýza:** Této části se věnují kapitoly Analýza a API Schneider PLC. Jejím cílem je prozkoumat současný stav testovacího frameworku a možnosti API od společnosti Schneider, sloužící k ovládání PLC.

4. **Implementační část:** Popisuje návrh a implementaci nové knihovny a testovacích scénářů, čemuž jsou věnovány kapitoly Implementace knihovny a Testovací scénáře.

5. **Závěr**: Společně s kapitolou Zhodnocení shrnuje dosažené výsledky a navrhuje možnosti budoucího rozvoje.

Kapitola 2

Technologie v průmyslové automatizaci

2.1 PLC

V oblasti průmyslové automatizace slouží programovatelný logický kontrolér (PLC) jako základní součástka, umožňující efektivní kontrolu a řízení různých průmyslových procesů. PLC je v podstatě specializované výpočetní zařízení pečlivě navržené k monitorování vstupů ze senzorů a provádění předdefinovaných řídících algoritmů pro manipulaci s výstupy, čímž automatizuje různé průmyslové úkoly v odvětvích jako průmyslová výroba, energetika a doprava.

2.1.1 Hardware

Hardware PLC zahrnuje několik základních součástek, navržených tak, aby zajistily spolehlivost v průmyslových prostředích. V následující části popíšeme základní strukturu hardwaru, která je obecně platná pro všechna PLC. Diagram této struktury je zároveň možno vidět na obrázku 2.2.

- **Procesor (CPU)**: V jádru každého PLC se nachází procesor, který je zodpovědný za provádění programovaných instrukcí a řízení výměny dat mezi různými moduly. Dále také zpracovává vstupy ze senzorů, provádí řídící algoritmy, vydává příkazy aktuátorům a dohlíží na komunikaci s externími zařízeními a rozhraními (například jiná PLC).

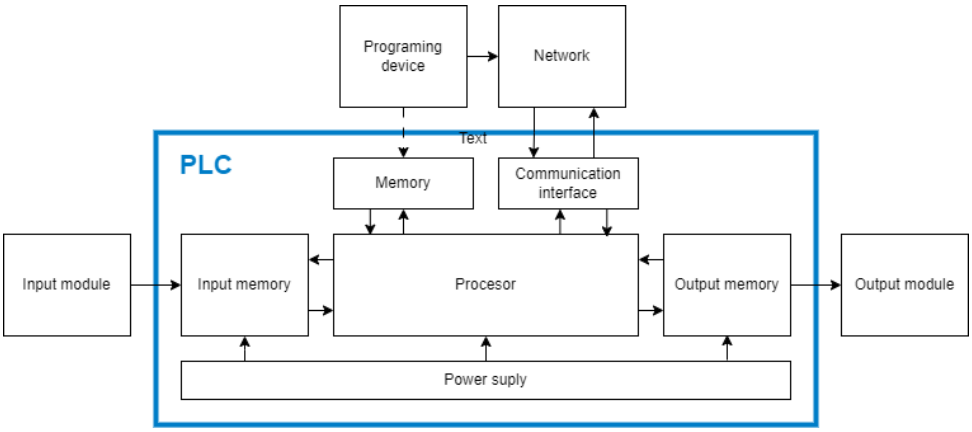
Obrázek 2.1: PLC společnosti Siemens řady SIMATIC S7-1500. [21]

- **Vstupní moduly**: Vstupní moduly se propojují se senzory a spínači k detekci změn fyzikálních veličin. Tyto moduly obvykle obsahují mnoho vstupních kanálů, z nichž každý je schopen monitorovat analogové nebo digitální signály. Analogové vstupní moduly převádějí kontinuální signály (napětí, proud) na digitální hodnoty, zatímco digitální vstupní moduly přímo monitorují binární signály (zapnuto/vypnuto, vysoké/nízké).
- **Výstupní moduly**: Výstupní moduly řídí aktuátory a zařízení na základě instrukcí poskytnutých CPU. Analogové výstupní moduly převádějí digitální signály z CPU na analogové řídící signály, umožňující přesnou kontrolu aktuátorů, jako jsou například motory, ventily a proměnné frekvenční měniče. Digitální výstupní moduly přímo spínají binární signály k řízení zařízení, jako jsou relé, solenoidy nebo signalizační světla.
- **Zdroj napájení**: Zdroj napájení poskytuje stabilní a spolehlivé elektrické napájení všem součástem PLC systému. Převádí přicházející střídavé nebo stejnosměrné napětí z hlavního zdroje do odpovídajících úrovní napětí vyžadovaných CPU, vstupních/výstupních modulů a dalších periferních zařízení.
- **Komunikační rozhraní**: PLC jsou vybaveny komunikačními rozhraními pro usnadnění výměny dat s externími zařízeními, řídicími systémy a nadřazeným softwarovým systémem. Běžné komunikační protokoly zahrnují Ethernet, sériovou komunikaci (RS-232, RS-485) a průmyslové fieldbus protokoly (Profinet, Modbus, DeviceNet). Tyto rozhraní umožňují monitorování v reálném čase, vzdálenou diagnostiku a integraci s řídícími systémy vyšší úrovně.
- **Moduly paměti**: Paměťové moduly uchovávají programový kód, data a konfigurační nastavení potřebná pro provoz PLC. Tyto moduly obvykle

zahrnují nevolatilní paměť (např. flash paměť) pro ukládání programu PLC a volatilní paměť (např. RAM) pro dočasné ukládání dat během provádění programu. Možnosti rozšíření paměti umožňují zvýšení kapacity úložiště pro zvládnutí větších a složitějších programů.

- **Sběrnice**: V kontextu PLC označuje pojem sběrnice ("bus") komunikační cestu, která umožňuje výměnu dat mezi různými součástmi vnitřního systému PLC. Sběrnic je několi druhů, přičemž každá zastává jinou funkci při komunikaci. Data bus přenáší data zpracovávaná procesorem. Address bus zajišťuje přenos adres při adresaci dat v paměti (zápis i čtení). System bus zprostředkovává komunikaci mezi vstupními a výstupními moduli. Nakonec Control bus zajišťuje časovou synchronizaci a řídící komunikaci CPU s jinými částmi hardwaru.[15]

Z hlediska hardwaru můžeme PLC dělit do dvou skupin, prostorově úsporných, kompaktních a škálovatelných modulárních. Kompaktní jednotky, mající v sobě integrovaný napájecí zdroj a I/O moduly, jsou ideální pro aplikace s omezeným prostorem nebo jednodušší automatizační úlohy, nabízející jednoduché řešení bez ztráty funkcionality. Naopak modulární PLC skládající se ze samostatných modulů, pro každou výše zmíněnou část, poskytují flexibilitu a škálovatelnost. Uživatelé mohou snadno konfigurovat a rozšiřovat modulární PLC přidáváním nebo odebíráním I/O modulů podle konkrétních požadavků aplikace. To činí modulární PLC vhodnými pro větší a složitější automatizační systémy, kde jsou potřebné rozsáhlé I/O moduly nebo kde systém může vyžadovat budoucí rozšíření.



Obrázek 2.2: Diagram hardwaru PLC. [16]

2.1.2 Software

Při programování PLC se stkáme s několika pojmy, jako jsou "POUs"- "program organization units"(organizační jednotky programu), nebo "tasks"(úlohy). V následující části bude vysvětlen význam těchto termínů, jejich přínos a využití pro programování PLC. Nejdříve je ale třeba porozumět tomu, jak je v PLC program vykonáván a v čem se tento přístup liší oproti běžným počítačům.

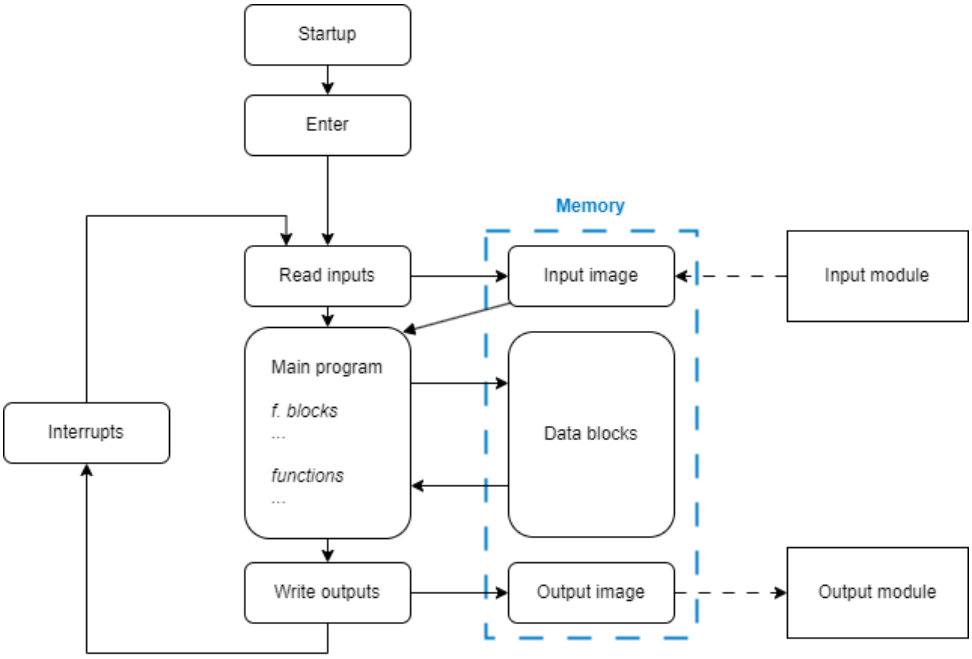
Vykonávání programu

Rychlý přehled procesu vykonávání programu v PLC může čtenáři poskytnout diagram 2.3. Můžeme vidět, že na začátku prochází program počáteční (vstupní - "enter") částí kódu, která je vykonána pouze jednou. Následuje vyčtení stavu vstupních hodnot z paměti, které jsou do tohoto místa zapisovány vstupním modulem připojeným k PLC. V tuto chvíli může začít hlavní blok programu, vykonávající výpočty a čtení/zápis do paměti, jehož výsledkem je nahrání výstupních hodnot do paměti. Tato data jsou vyčtena výstupním modulem daného PLC a tím jsou řízeny aktuátory. V tuto chvíli se může celý cyklus opakovat, pokud nedojde k žádné přerušující události (vysvětleno v sekci 2.1.2), jež by cyklus pozastavila a jeho vykonání by se tak po tuto dobu zdrželo.

POUs

Organizační jednotky slouží jako základní stavební bloky pro organizaci a strukturování řídící logiky systémů. POUs obalují specifické funkcionality a řídící úkoly v rámci programu PLC, usnadňují modulární programování, opakované využívání kódu a údržbu systému. "Program organization unit"je tedy souhrnný název pro funkci, funkční blok, data blok, nebo program, kterým je tato sekce věnována.

- **Programy**: Programy v PLC představují hlavní řídící logiku nebo sekvenci operací pro konkrétní úkol nebo proces. Obsahují primární logiku pro provádění požadovaných akcí na základě vstupních podmínek a stavů systému.



Obrázek 2.3: Diagram výkonu programu v PLC.

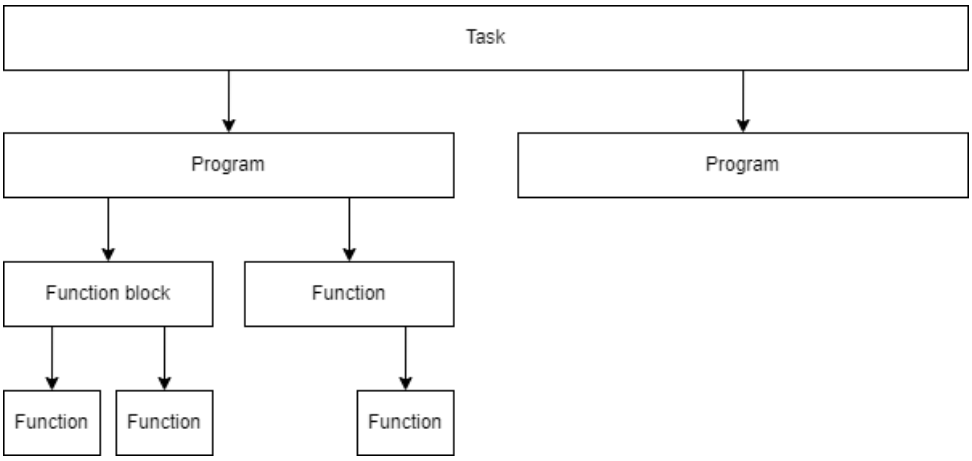
- **Funkční bloky**: Funkční bloky obalují opakovaně použitelné a parametrizované řídící algoritmy nebo rutiny v rámci programu PLC. Poskytují modulární přístup k programování tím, že umožňují vývojářům obalit složitou funkcionalitu do samostatných jednotek, které lze snadno integrovat do větších řídících strategií. Funkční blok také může mít svůj data blok, což je organizační struktura, ukládající proměnné využívané v daném funkčním bloku.
- **Funkce**: Funkce jsou podobné funkčním blokům, ale obvykle se používají pro menší, konkrétnější úkoly v rámci programu PLC. Umožňují vývojářům obalit běžně používané rutiny nebo výpočty, podporují opakované využití kódu a zjednodušují údržbu programu. Hlavním rozdílem oproti funkčním blokům je, že funkce nejsou parametrizovatelné a jejich chování je tedy neměnné. Příkladem funkce mohou být například aritmetické operace, logické operace, nenbo porovnávací operace.

Díky POUs mohou programátoři strukturovat složitou řídící logiku do menších a lépe spravovatelných jednotek, což podporuje modularitu a organizaci kódu. Dalším významným přínosem je možnost opakovaného využívání kódu: obalováním logiky do POUs mohou vývojáři kód opakovaně používat napříč různými programy nebo projekty, což vede ke snížení času a úsilí potřebného pro vývoj. Izolací specifických funkcionalit uvnitř programu usnadňují ladění a údržbu, což zjednodušuje lokalizaci a opravu chyb či aktualizaci kódu. Nakonec, modulární povaha POUs přispívá ke škálovatelnosti softwaru tím,

že umožňuje vývojářům přidávat nebo upravovat funkce bez významného dopadu na celkovou strukturu programu. [3]

Task

Úkoly v PLC představují souběžně výkonávatelná vlákna v rámci řídicího systému, umožňující současné zpracování více řídicích operací. Každý úkol v PLC pracuje nezávisle a může provádět specifickou řídicí logiku nebo vykonávat dedikované funkce v rámci celkového řídicího systému. Každý task pak může volat různé programy, tak jak je znázorněno na obrázku 2.4.



Obrázek 2.4: Hierarchie volání POUs

Task může být spuštěn cyklicky, nebo jako reakce na nějakou událost. Existují také volně běžící úkoly ("Freewheeling tasks"), což znamená, že úkol bude proveden co možná nejčastěji.

- **Cyklické činnosti**: Nazývané také jako hlavní činnosti ("main tasks"), provádějí činnost v pravidelných intervalech podle předdefinovaného časového plánu. Tyto úkoly fungují na neustálé smyčce, opakovaně provádějící stejnou sekvenci řídicí logiky v pevném časovém rámci. Cyklické činnosti jsou vhodné pro aplikace vyžadující konzistentní a předvídatelné řídicí operace, jako jsou nepřetržité výrobní procesy nebo systémy řízení pohybu.
- **Reakce na událost**: Známé také jako přerušující úkoly ("Interrupt tasks") jsou činnosti spouštěné událostmi provádějí operace v reakci

na specifické podmínky detekované v rámci systému. Tyto úkoly zůstávají nečinné, dokud nejsou spuštěny externím podnětem, jako je vstup ze senzoru, uživatelský příkaz nebo událost systému. Úkoly spouštěné událostmi jsou ideální pro zpracování časově kritických operací nebo dynamickou reakci na změny v prostředí.

- **Volně běžící činnosti**: Volně běžící činnosti, též známé jako úkoly na pozadí nebo neperiodické úkoly, provádějí svou činnost nezávisle na pevném časovém plánu nebo externích událostech. Tyto činnosti fungují "volně běžícím" způsobem, neustále běží na pozadí a vykonávají časově nekritické řídicí operace nebo úkoly údržby. Volně běžící úkoly jsou běžně využívány pro činnosti, jako je protokolování dat, diagnostika, komunikace s externími zařízeními nebo údržba systému.

Činnosti hrají klíčovou roli při návrhu a implementaci řídicích systémů založených na PLC, umožňují efektivní a flexibilní provádění řídicí logiky. Správou souběžnosti, časově orientovaného provádění a přidělování zdrojů úkoly přispívají k optimalizaci výkonu, odezvy a spolehlivosti systému v průmyslových automatizačních aplikacích. [11]

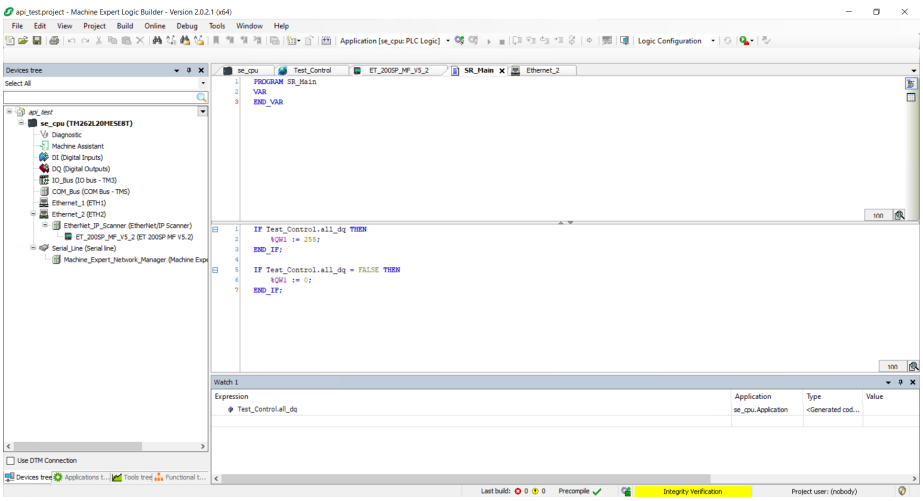
2.1.3 Programování

Programování PLC zahrnuje široké spektrum jazyků, každý přizpůsobený konkrétním potřebám a preferencím. Reléová logika (LADDER), připomínající drátové diagramy obvodů, oslovuje ty, kteří jsou obeznámeni s tradičními řídícími systémy. Blokové diagramy funkcí (FBD) nabízejí grafickou reprezentaci podobnou elektronickým obvodovým diagramům, zatímco strukturovaný text (ST) používá textový přístup podobný běžným programovacím jazykům. Sekvenční funkční grafy (SFC) poskytují vizuální reprezentaci sekvenční logiky řízení, ideální pro složité sekvence a stavové programování. [13]

K základním změnám programu lze využít rozhraní samotného PLC, které je však k větším změnám naprosto nevhodné a běžně se tedy využívá pouze pro vizualizaci a kontrolu procesu. Ve výjovém prostředí jsou však běžně používány počítače, které z hlediska uživatelského rozhraní a nabízených možností výrazně převyšují předešlé varianty.

V rámci prostředí vývoje softwaru pro PLC existuje mnoho možností, každá nabízející svou jedinečnou směs funkcí a schopností. TIA Portal od Siemensu, Studio 5000 od Rockwell Automation a Machine Expert (obrázek 2.5) společnosti Schneider Electric dominují trhu, chlubí se uživatelsky přívětivými

rozhraními, rozsáhlými knihovnami přednastavených komponent a robustními simulačními nástroji.



Obrázek 2.5: Rozhraní programu Machine Expert

2.2 Distributed IO

Distribuovaná zařízení pro vstup/výstup ("input/output"- I/O) jsou hardwarové komponenty, které jsou rozloženy nebo distribuovány po celém systému nebo síti místo toho, aby byly umístěny centrálně. Tato zařízení jsou obvykle používána k propojení se senzory, aktuátory a dalšími externími zařízeními v průmyslové automatizaci, systémech řízení budov a dalších aplikacích, kde je vyžadováno vzdálené monitorování a řízení.

K IO modulu lze připojit množství senzorů, se kterými navazuje komunikaci. Tímto způsobem může modul vyčítat/zapisovat požadované informace od/do těchto senzorů/aktuátorů a ukládat tyto hodnoty do své paměti. Propojení s PLC může být realizováno přes fyzickou vrstvu jako je například Ethernet, za použití různých komunikačních protokolů, jako je například ProfiNet.

2.2.1 Hardware

V této části je uveden obecný popis hardware distributed IO zařízení, který má čtenáři poskytnout základní představu o fungování a možnostech těchto zařízení.

- **Vstupně/Výstupní Kanály**: Primární funkcí distribuovaných I/O zařízení je propojení s externími senzory a aktuátory. Tyto zařízení obvykle disponují více vstupními a výstupními kanály, z nichž každý je schopen zpracovávat signály od senzorů (například teploty, tlaku, hladiny atd.) nebo odesílat signály k aktuátorům (například motorům, ventily, relé a jiné).

- **Mikrokontrolér nebo Procesor**: Distribuovaná I/O zařízení obsahují mikrokontrolér nebo procesor, který je zodpovědný za řízení vstupně/-výstupních kanálů a provedení jakéhokoli vestavěného softwaru nebo logiky nezbytné pro zpracování dat nebo provoz zařízení. Na starost má také zpracování komunikačního protokolu, díky kterému může docházet k výměně informací s PLC přes komunikační rozhraní.

- **Komunikační Rozhraní**: Distribuovaná I/O zařízení jsou vybavena komunikačními rozhraními, která jim umožňují výměnu dat s centrálním řadičem nebo jinými zařízeními v síti. Běžná komunikační rozhraní zahrnují Ethernetové porty, sériové porty (RS-232/RS-485) nebo konektory pole sběrnice (EtherNet/IP, Modbus, DeviceNet atd.).

- **Paměť**: Stejně jako v případě PLC slouží paměť pro uchovávání konfiguračních nastavení, aktualizací firmware/software a dočasných datových bufferů.

- **Zdroj Napájení**: Distribuovaná I/O zařízení vyžadují energii k provozu. V závislosti na aplikaci a umístění instalace mohou být napájena z externího zdroje energie (například AC nebo DC zdroj energie) nebo přes komunikační síť (například Power over Ethernet).

- **Diagnostické LED/Diodové Indikátory**: Mnoho distribuovaných I/O zařízení disponuje diagnostickými LED nebo indikátory, které poskytují vizuální zpětnou vazbu o stavu zařízení, aktivitě komunikace a poruchách, což usnadňuje diagnostiku a údržbu.

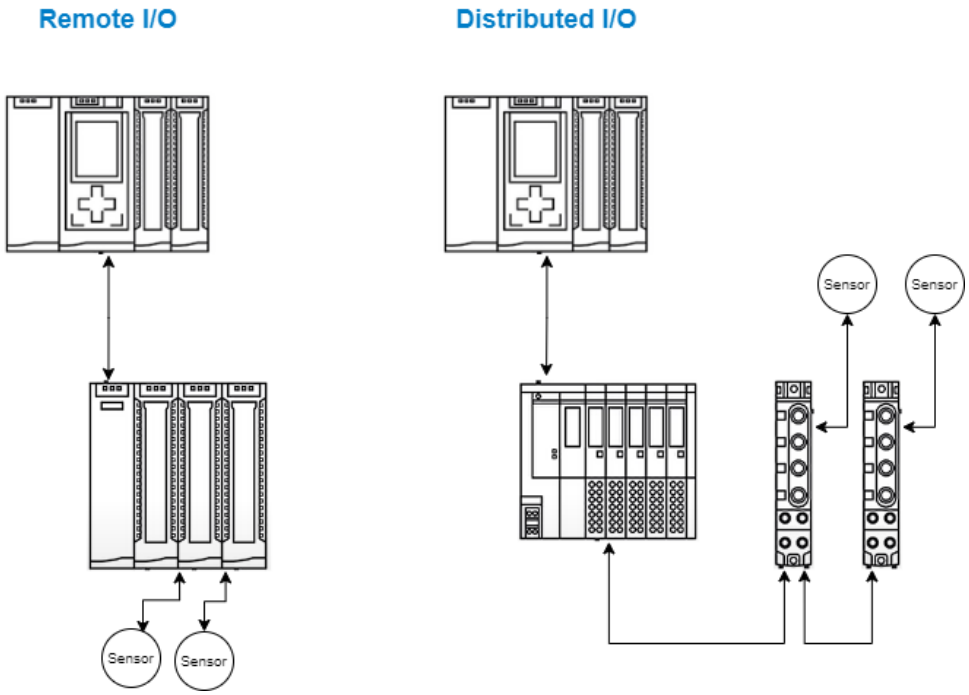
2.2.2 Dělení

Tato sekce nabídne několik pohledů na dělení distributed IO zařízení do skupin, dle jejich vlastností a funkcionalit. Toto dělení čtenáři umožní lépe se zorientovat v množství možností, které distributed a remote IO zařízení nabízejí.

Distributed/remote IO

Na obrázku 2.6 můžeme vidět, že PLC je vždy přopojeno k interface modulu, který zajišťuje ovládání připojených vstupních a výstupních modulů a komunikaci s PLC. Z obrázku je také možné vidět, že remote IO nabízí možnost připojení mnoha senzorů v relativně malé oblasti, zatímco distributed IO je pomocí jednoho interface modulu schopno pokrýt rozsahlejší prostor.

Pro lepší porozumění rozdílům demonstrujme nyní tento obecný popis na konkrétních příkladech. Uvažme inteligentní budovu a automaticky řízený průmyslový závod, kde každý stroj je ovládán jedním PLC. Je zřejmé, že v relativně malé oblasti, kde se nachází stroj bude vhodnější využít technologie remote IO a všechny senzory připojit k jednomu centrálnímu bodu. Naopak u rozsáhlé budovy bude vhodné připojit například senzory a aktuátory u brány k jednomu modulu a senzory a akční členy střešních oken ke druhému, to vše za možnosti ovládání jedním PLC.



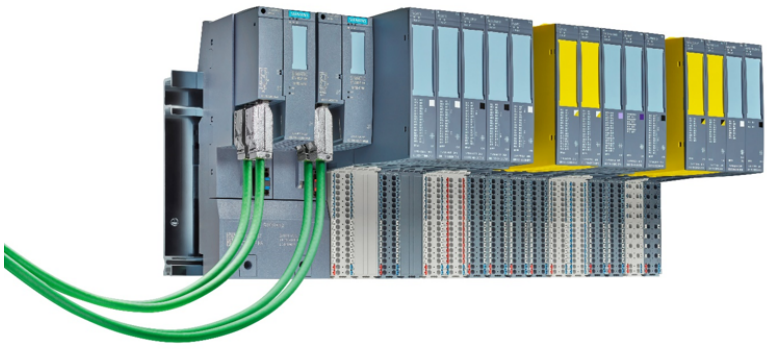
Obrázek 2.6: Porovnání Remote a Distributed IO.

Stupeň ochrany

Distribuovaná zařízení je možné dělit také dle prostředí, ve kterém je vhodné je použít. Uvažme nyní dva hypotetické příklady nasazení, kancelářskou budovu a průmyslový výrobní závod.

Čisté prostředí

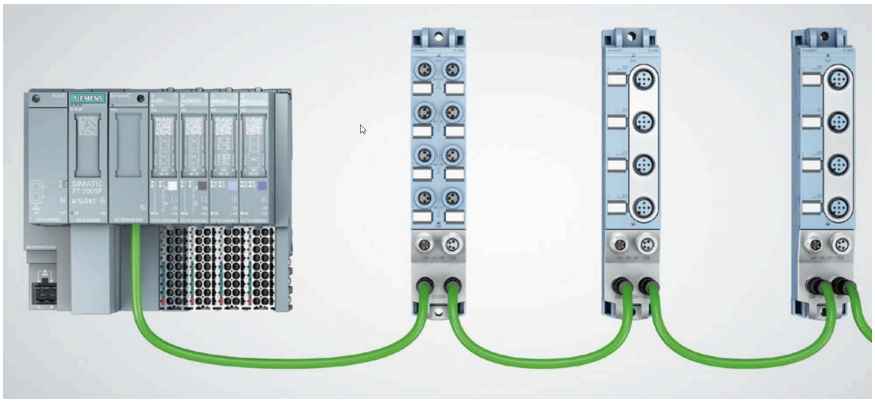
V čistém prostředí není třeba vyšší stupeň ochrany (například IP67 2.2.2 (Ingress Protection)) a je tedy možné použít naopak kompaktnější a snadněji zapojitelné vstupní/výstupní moduly. Na obrázku 2.7 je možné vidět vstupní a výstupní moduly ET200SP od společnosti Siemens společně s rozhraním pro Ethernet komunikaci pro připojení k PLC (připojeno zelenými ethernet kabely).



Obrázek 2.7: Kompaktní vstupní/výstupní moduly. [20]

Průmyslová výroba

V prostředí průmyslové výroby, kde může být velké množství prachu, či agresivní látky, které by mohly moduly poškodit je zapotřebí vyššího stupně ochrany. Na obrázku 2.8 jsou k vidění vstupní a výstupní moduly s ochranou IP67, opět i s modulem pro Ethernet komunikaci.



Obrázek 2.8: Vstupní/výstupní moduly ET 200AL s IP67/65. [19]

- **Stupně ocharny IP** IP65: Stupeň ochrany IP 65 znamená, že zařízení je chráněno proti prachu a proti vodě z trysky pod nízkým tlakem ze všech směrů. Pro distributed a remote I/O moduly to znamená, že jsou obvykle uzavřeny v krytu, který je odolný vůči prachu a stříkající vodě. To zajišťuje, že elektronika uvnitř modulu není poškozena.

  IP67: Tento stupeň ochrany zajišťuje, že zařízení je chráněno proti prachu a proti vodě při ponoření do vody pod určitým tlakem a po určitou dobu. Pro průmyslové I/O moduly to znamená, že jsou schopny odolat podmínkám, jako je těžký déšť, stříkající voda nebo krátkodobé ponoření do vody. To je obzvláště užitečné v prostředích, kde hrozí vystavení vlhkosti a vodě.

2.3 Komunikační standardy

Komunikační standardy v průmyslové automatizaci představují soubor specifikací a pravidel určujících způsob, jakým zařízení a systémy v průmyslovém prostředí komunikují a výměňují data mezi sebou. Tyto standardy jsou klíčové pro zajištění interoperability a kompatibility různých zařízení od různých výrobců, což umožňuje efektivní integraci a správu automatizačních systémů. Mezi hlavní aspekty komunikačních standardů patří definice formátu dat, metodika přenosu dat, síťová topologie a protokoly pro zajištění spolehlivé a bezpečné komunikace. [24]

Díky komunikačním standardům mohou průmyslové zařízení, jako jsou PLC, senzory, pohony a další, efektivně vyměňovat informace v reálném čase, což je zásadní pro řízení a monitorování výrobních procesů. Standardizace komunikace v průmyslové automatizaci snižuje náklady spojené s integrací

nových zařízení, minimalizuje riziko chyb a umožňuje jednodušší správu a údržbu systémů. Mezi známé komunikační standardy v průmyslové automatizaci patří například ProfiNet, Modbus, Ethernet/IP, CANopen a další, které poskytují rozmanité možnosti pro propojení zařízení a optimalizaci výrobních procesů.

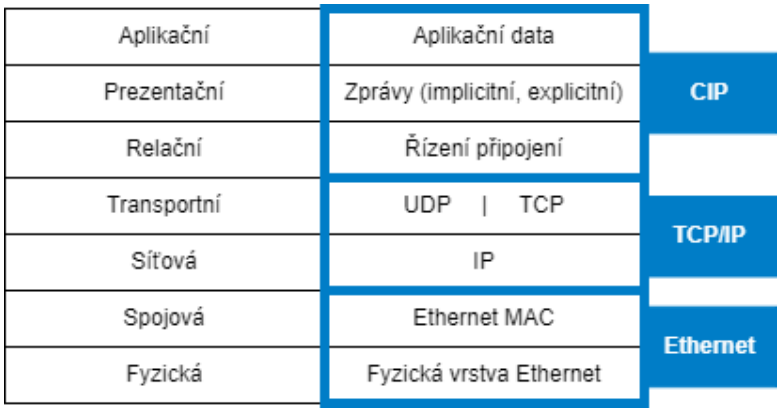
2.3.1 Vybrané standardy

V této sekci jsou detailněji popsány vybrané komunikační standardy a uvedeny příklady jejich využití. Zaměříme sa na některé z nejznámějších standardů, jako je Ethernet/IP, Modbus TCP, Profinet a CANopen, a prozkoumáme jejich klíčové vlastnosti a využití v průmyslovém prostředí.

EtherNet/IP

Ethernet/IP je komunikační standard vycházející z Ethernetu a TCP/IP protokolu, který je široce využíván v průmyslové automatizaci pro propojení zařízení a systémů. Jedná se o otevřený standard, který umožňuje efektivní a spolehlivou komunikaci mezi různými zařízeními od různých výrobců. Ethernet/IP poskytuje vysokou rychlost přenosu dat a podporuje širokou škálu aplikací v průmyslovém prostředí, včetně řízení procesů, monitorování stavu zařízení, sběru dat a diagnostiky. Tento standard je charakterizován jednoduchou integrací do stávajících Ethernetových sítí a podporou pokročilých funkcí pro zajištění bezpečnosti dat a správu sítě.

Výhoda Ethernet/IP spočívá v jeho schopnosti kombinovat standardní Ethernetové sítě s průmyslovým protokolem IP. Na obrázku 2.9 můžeme vidět protokol Ethernet/IP v kontextu OSI modelu. Tato kombinace poskytuje vysokou rychlost přenosu dat a dobrou kompatibilitu s existujícími IT infrastrukturami. Díky tomu je Ethernet/IP ideální volbou pro průmyslové aplikace, které vyžadují vysoký výkon a spolehlivost přenosu dat. [23]



Obrázek 2.9: Ethernet/IP v kontextu OSI modelu [14]

Modbus TCP

Modbus TCP je varianta Modbus protokolu, který využívá Ethernetovou síť jako komunikační médium pro přenos dat mezi průmyslovými zařízeními. Jedná se o jednoduchý a spolehlivý standard, který poskytuje efektivní možnost komunikace mezi různými zařízeními v průmyslovém prostředí. [4] Modbus TCP je často používán pro řízení a monitorování zařízení, jako jsou PLC, pohony, senzory a regulátory, a nabízí jednoduchou implementaci a podporu širokého spektra zařízení od různých výrobců.

Jednou z hlavních výhod Modbus TCP je jeho jednoduchost a snadná implementace. Tento lehký a spolehlivý protokol poskytuje efektivní možnost komunikace mezi různými zařízeními v průmyslovém prostředí bez potřeby složitých konfigurací nebo pokročilých funkcí. [26] To ho činí ideální volbou pro menší aplikace nebo prostředí, kde je potřeba rychlého nasazení a spolehlivé komunikace.

Profinet

Profinet je komunikační standard vyvinutý společností Siemens pro průmyslovou automatizaci, který využívá Ethernetovou síť jako komunikační prostředek. Jedná se o otevřený a standardizovaný protokol, který poskytuje vysokou rychlost přenosu dat, nízkou latenci a deterministické chování sítě. Profinet umožňuje integraci různých zařízení a systémů v průmyslovém prostředí a podporuje pokročilé funkce pro diagnostiku, konfiguraci a bezpečnost

dat. Tento standard je často využíván pro komunikaci mezi PLC, pohony, roboty a dalšími zařízeními v automatizovaných výrobních linkách.

Jednou z hlavních výhod Profinetu je jeho podpora deterministického chování sítě a nízké latence. Díky tomu poskytuje Profinet spolehlivou a přesnou komunikaci mezi zařízeními v průmyslovém prostředí, což je klíčové pro aplikace, které vyžadují synchronizaci a řízení v reálném čase. Profinet je také charakterizován pokročilými funkcemi pro diagnostiku, konfiguraci a bezpečnost dat, což ho činí ideální volbou pro náročné průmyslové aplikace. [17]

CANopen

CANopen je komunikační standard složený ze 7 vrstev, založený na sběrnici CAN (Controller Area Network), který je široce používán v průmyslové automatizaci pro propojení zařízení v reálném čase. CANopen poskytuje spolehlivý mechanismus pro výměnu dat mezi různými zařízeními prostřednictvím standardizovaných protokolů a profilů aplikací. Tento standard je vhodný pro distribuované systémy, kde je potřeba spolehlivého propojení mezi různými moduly a periferiemi, například v průmyslových robotických systémech nebo v automobilovém průmyslu, zejménka k připojení nekritických částí systému.

Jednou z hlavních výhod CANopen je jeho robustnost a spolehlivost v reálném čase. Tento standard poskytuje efektivní mechanismus pro výměnu dat mezi různými zařízeními v průmyslovém prostředí prostřednictvím sběrnice CAN, což je ideální pro distribuované systémy, kde je potřeba spolehlivého propojení mezi různými moduly a periferiemi. CANopen je také charakterizován jednoduchou implementací a podporou širokého spektra zařízení od různých výrobců. [2] [12]

Kapitola 3

Analýza

V rámci této kapitoly je věnována pozornost zapojení testovací stanice a zařízením v ní obsažených. Bude tedy popsán význam jejího zapojení a zmíněny základní informace o použitých zařízeních.

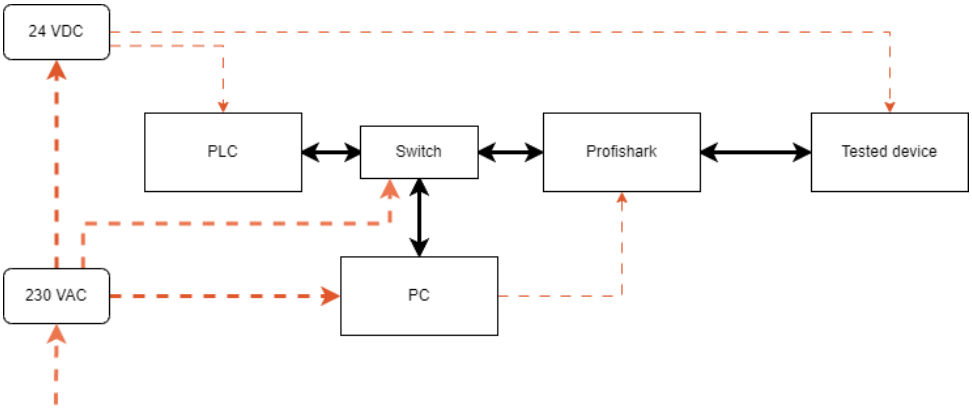
V následující části kapitoly je uveden přehled využití PLC od společnosti Schneider v testovacích scénářích v rámci využívaného frameworku spolu s možnostmi automatizace těchto scénářů. Jsou navrženy také nové testovací scénáře, inspirované již existujícími testy na jiných zařízeních, které nebylo doposud možné implementovat, kvůli potřebě změny konfigurace zařízení, nebo přehrání projektu, což zajišťuje nově implementovaná knihovna.

3.1 Popis testovací stanice

V této sekci je popsána využitá testovací stanice, tedy použité PLC a k němu připojené vstupní/výstupní moduly a další prvky připojené k síti.

3.1.1 Zapojení

Schéma 3.1 ilustruje zapojení testovací stanice, kde jsou všechny komponenty propojeny pomocí fyzické vrstvy Ethernet. Můžeme vidět, že v síti se mimo



Obrázek 3.1: Zapojení testovací stanice.

PLC a modulů nachází také počítač, využívaný k programování PLC (program MachineExpert) a spouštění testovacích skriptů. Povšimnout si můžeme také zařízení Profishark, které slouží k monitorování přenášených datových paketů a nabízí možnost jeho využití v rámci testovacích scénářů.

Napájení v této síti je zajištěno několika zdroji. Zařízení Profishark je napájeno pomocí USB 3.0 z počítače a přijímá napětí 5V. Moduly a PLC jsou napájeny z externího zdroje se stejnosměrným napětím 24V. Pro switche je využíváno střídavé napětí 230V. Tento systém rozdílných napájecích specifikací je pečlivě navržen s ohledem na specifické požadavky jednotlivých zařízení a zajišťuje spolehlivý provoz celé testovací stanice.[18] [25]

3.1.2 PLC

Pro účely testování je nasazeno PLC od společnosti Schneider Electric, konkrétně model Modicon M262 ve verzi TM262L20MESE8T, které je k vidění na obrázku 3.2. Toto PLC podporuje různé způsoby připojení, včetně sériové linky, USB rozhraní typu mini B a další možnosti. V rámci našeho testování je využito pouze připojení prostřednictvím rozhraní Ethernet, které slouží k nahrávání programů a konfigurací do PLC, stejně jako k vzájemné komunikaci s připojenými moduly.

Pro uchovávání dat ze vstupních modulů je v PLC vyhrazeno úložiště o velikosti 32 GB, zatímco pro zálohování uživatelských programů a konfigurací je k dispozici 1 GB prostoru. Konfigurace PLC probíhá přenesením projektu z řídícího počítače (označeného na diagramu jako PC) do PLC. V rámci tohoto projektu jsou specifikovány jak hardwarové konfigurace, tak i kód aplikace,

Obrázek 3.2: Použité PLC.

který bude na PLC vykonáván. [8]

3.1.3 Testované zařízení

Testovaným zařízením je SIMATIC ET 200SP, konkrétně verze IM 155-6 MF HF společnosti Siemens, které je k vidění na obrázku 3.3. Toto zařízení je vybaveno širokým spektrem komunikačních standardů, mezi něž patří Profinet, Modbus TCP a EtherNet/IP. Z této palety standardů bude v rámci testování využíván komunikační protokol EtherNet/IP.

Zařízení ET 200SP nabízí široké možnosti rozšíření, neboť umožňuje připojení až 64 modulů ET 200SP, což výrazně přesahuje potřeby provedení našich testů. Kromě toho je, pro protokol profinet, implementována systémová redundance, což představuje významnou funkční výhodu. Mechanismus systémové redundance umožňuje automatické přepnutí na záložní zařízení v případě detekce chyb v primárním zařízení, což zajišťuje nepřetržitý provoz a udržuje kontinuitu komunikace s nadřazeným systémem, tj. PLC. Důležitým faktorem je rovněž rychlost komunikace, v případě EIP se jedná o RPI ("Rate

Obrázek 3.3: Použité zařízení SIMATIC ET 200SP.

of Process Information"), která pro EIP dosahuje minimální periody 2 ms, což představuje nejrychlejší zařízením podporovanou frekvenci. [22] Tato perioda se liší dle komunikačního standardu, například u profinetu je to 250 μs. [1]

3.2 Nové funkcionality

Tato sekce popisuje a odůvodňuje funkcionality, které bude nová knihovna implementovat. Důraz při volbě těchto funkcionalit je kladen na možnost konkrétních využití v testovacích scénářích a tím garantovaný přínos.

Knihovna bude primárně zaměřena na možnost změn v hardwarové konfiguraci zařízení a přehrávání projektů, jelikož se jedná o úkony, které nenabízí žádná doposud využívaná knihovna. Tyto funkce umožní automatizaci testů, během kterých byl nutný lidský zásah při provádění těchto změn. Přehrávání projektů umožňuje jednak stažení aktuálního projektu ze zařízení, tak i nahrání jiného projektu uloženého v počítači.

Jedním z úkonů, které bylo doposud nutné provádět manuálně, je například restart PLC, ketrý umožní jeho automatický reset při detekci chybového stavu. Knihovna by také měla umožnit zastavení (a opětovné spuštění) PLC, kdy dojde k přerušení cyklicky běžící aplikace a tedy přerušení všech procesů, které PLC obstarává. Tímto způsobem bude možné ověřit, zda připojená zařízení reagují spravně na změnu stavu (start/stop).

Mezi změny v nastavení a konfiguraci se řadí například změna IP adresy PLC, změna a vyčtení RPI, nebo nastavení "communication path"(komunikační cesty). Možnost provádění těchto změn opět umožní ověřit, zda reakce připojených zařízení odpovídá očekávanému chování.

Poslední funkcionalitou, kterou by knihovna měla obsahovat je nastavení hodnot proměnných v PLC a případně také přidání (resp. odebrání) částí vykonávaného programu. Jelikož je program dělen na jednoduché organizační bloky (POUs 2.1.2), je možné přidat (resp. odstranit) vždy jeden takový blok.

3.3 Struktura testovacích scénářů

V této sekci je popsána využívaná struktura testovacích scénářů, spolu s pravidly, které je při jejich implementaci vhodné dodržovat.

V každém testovacím scénáři se nachází několik globálních proměnných popisujících jeho jméno a další relevantní informace. V rámci struktury souboru je dále obsažena třída, pojmenovaná podle konvence TestCase[název testu], obsahující povinně implementovanou funkci run_tc. Po definici třídy následují příkazy pro spuštění daného testovacího scénáře, které umožňují snadné a efektivní provedení testů v rámci testovacího prostředí. Taková struktura souboru poskytuje přehledný a systematický rámec pro organizaci a provádění testů.

Ke stadnardním postupům patří systematické rozdělování kódu v rámci testovacích scénářů do jednotlivých funkcí, které jsou následně volány. Tento přístup umožňuje strukturovat testovací scénáře do posloupnosti testovacích kroků ("test steps"), které jsou soustředěny v rámci funkce run_tc.. Takto psaný test je modulární, což umožňuje snadné přidávání (resp. odebírání) jednotlivývch kroků testu, usnadňuje testování a zlepšuje čitelnost a přehlednost kódu.

Mezi jednotlivými testovacími kroky je také vhodné informovat o aktuálním stavu testu. Tento aspekt přispívá k uživatelskému pohodlí a celkové transparentnosti procesu testování. Zahrnout do testovacího scénáře průběžné zprávy o stavu testu, včetně informací o současném test pointu a úspěšnosti dosavadních výsledků, přináší jasnější přehled o tom, jak skript prochází testovacím procesem. Tato funkce poskytuje uživateli přesnější informace a umožňuje mu snadněji identifikovat potenciální problémy.

Kapitola 4

API Schneider PLC

V následující kapitole je uveden popis API a možností, které nabízí, diskutována je také její struktura a využití knihoven umozňujících komunikaci s PLC.

4.1 Použití

Společnost Schneider Electric dodává k programu Machine Expert také soubor LogicBuilderShell.exe, který umožňuje spuštění IronPythonu verze 2.7 a obsahuje veškeré knihovní funkce potřebné pro komunikaci s PLC, jak je podrobně popsáno v následující části práce. Tento soubor otevře Python konzoli, prostřednictvím které lze provádět potřebné příkazy.

Pro otevření tohoto souboru lze využít příkazovou řádku, kde bude IronPython spuštěn, a umožní tak zadávání příkazů. Dále je možné soubor spustit s argumentem obsahujícím cestu k Python skriptu, který se poté vykoná v prostředí IronPythonu. Tento přístup umožňuje rychle a efektivně provádět předem definované úkony. Nicméně, jeho nevýhodou je, že skript je v této podobě obtížněji debugovatelný než při postupném načítání a provádění jednotlivých příkazů. [5]

4.2 Struktura

Využité API dovoluje provádět hardwarovou konfiguraci PLC v rozsahu, který umožňuje aplikace Machine Expert v online režimu, z čehož plynou jistá omezení (viz. sekce 4.3). Přístup ke konfiguraci je realizován pomocí dvou knihoven, které se nazývají "online"a "projects". Každá z těchto knihoven umožňuje provádět různé změny v nastavení nejen hardwarové konfigurace, ačkoli ta je hlavním předmětem zájmu. Knihovna "projects" umožňuje například vyhledávání objektů v projektu, u kterých je následně možné měnit konfiguraci. Jedná se například o CPU, komunikační rozhraní, vstupní (resp. výstupní) moduly a další. "Online" knihovna naopak umožňuje vyhledávání a změnu hodnoty proměnných v aplikaci běžící na PLC.

Objekty v projektu mají stromovou strukturu, která odpovídá jejich logické souvislosti a úrovni funkční hierarchie. Stejným způsobem je k nim tedy přistupováno prostřednictvím API a je možné na objektech volat metodu "get_children()"(resp. "get_parent()"), nebo objekty vyhledávat metodou "find()". [6]

```
1 inspectapi.dir(<object>)
```

Tento příkaz lze použít ke zjištění, jaké má objekt metody a parametry. [7] Je možné jej volat na libovolnou proměnnou, což bylo velice užitečné při provádění prvotní investigace.

4.3 Možnosti

Díky znalosti základní struktury projektu a práce s API je nyní možné investigovat její možnosti z hlediska změn hardwarové konfigurace a úprav aplikace bežící na PLC.

Jelikož úpravy konfigurace (změna projektu) se provádí až v samotném PLC, jsou všechny změny prováděny v online režimu a není tedy možné měnit veškerá dostupná nastavení. Příkladem může být například RPI, které je možné měnit pouze před nahráním projektu, tedy pouze z konfigurační aplikace. U každého z požadavků je tedy třeba ověřit jak jeho dostupnost přes API, tak i možnost jeho změn v online režimu.

Toto ověření lze provést jednak přes aplikaci Machine expert, která po nahrání projektu znepřístuní parametry, které jsou v online režimu neměnné. Další možností je ověření parametru is_accessible_online, který nabízí většina objektů a také parametrů, které tyto objekty obsahují a udává jejich dostupnost (z hlediska změn) přímo v PLC.

4.4 Klíčové funkcionality

Na základě použití v testech a možností zjištěných investigací API, jsou v této sekci identifikovány klíčové funkcionality, které je možné implementovat. Knihovna by měla ke své plné funkčnosti a dosažení co největší automatizace poskytnout co možná nejvíce těchto možností.

První, a naprosto klíčovou, funkcionalitou je možnost otevření a následného nahrání libovolného projektu. To umožňuje přechod mezi testovacími scénáři, které využívají odlišný projekt bez nutnosti lidského zásahu. Je také třeba implementovat možnost zastavení a opětovného spuštění hlavní aplikace, k čemuž je možné přistoupit pomocí knihovny online, konkrétně pomocí metody "stop()" objektu online aplikace. Objekt CPU (vyhledatlený pomocí knihovny projects) pak obsahuje vlastní metodu, umožňující provést restart celého PLC.

Z pohledu konfigurace PLC je třeba umožnit změnu nastavení síťových komunikačních parametrů, jako je IP, nebo MAC adresa, což umožní pozorovat chování systému po neočekávané změně těchto parametrů. Postup změny těchto parametrů závisí na možnosti zápisu tohoto parametru v režimu online. Některé parametry, jako například IP adresa možnost zápisu v online režimu dovolují.

Změnu IP adresy PLC je tedy možno provést v libovolném projektu vyhledáním objektu komunikačního rozhraní za pomoci knihovny projects. Tento objekt v sobě ukrývá, mimo jiné, seznam svých parametrů. V tomto seznamu je možné nalézt, IP adresu (a mnoho dalších parametrů), kterou lze jak vyčíst, tak zapsat. Obdobný postup lze využít pro čtení (resp. zápis) většiny parametrů.

Zápis parametru RPI však není v online režimu (tedy za pomoci knihovny online) možné zapsat a bylo tedy nutné nalézt jiné řešení pro jeho změnu. Nejschůdnější cestou se ukázalo být přehrání projektu, kdy oba projekty jsou identické, až na parametr RPI, který je tímto způsobem možné upravit. Z

hlediska rychlosti se nejedná o nevýhodu, jelikož i při změně v aktuálním projektu by muselo dojít k jeho opětovnému nahrání, jedinou nevýhodou je tedy potřeba ukládání několika duplicitních projektů. Stejně jako v případě IP adresy, i zde je možné tento postup využít pro zbývající parametry bez možnosti online zápisu.

Kapitola 5

Implementace knihovny

Tato kapitola popíše jakým způsobem bylo přistupováno k implementaci knihovny a návrhu její struktury. Dále také odůvodní princip jejího fungování a vysvětlí jej na konkrétních příkladech.

5.1 Logika

Prvním přístupem ke komunikaci s projektem a samotným PLC bylo otevření LogicBuilderShell.exe pomocí konzole a za následného využití knihovny subprocess s ní komunikovat. Investigace ovšem odhalila, že tímto způsobem nebude možné knihovnu implementovat, jelikož nebylo možné komunikovat s konzolí spuštěného iron pythonu.

Následovalo tedy spuštění souboru logical-builder-shell.exe s argumentem, obsahujícím cestu k python souboru, který definoval posloupnost příkazů k provedení. Bylo zjištěno, že tímto způsobem je možné zadat příkazy k provedení a pomocí knihovny subprocess následně vyčíst výtisk z této konzole.

Funkčnost knihovny je tedy taková, že uživatel definuje příkazy, které je třeba vykonat a k jejich spuštění následně využije metodu flush(). Příkazy jsou při vykonávání kódu poté postupně ukádány do pole a metoda flush vygeneruje python soubor. Následuje spuštění logical-builder-shell.exe, které je také vyvoláno metodou flush, s argumentem obsahujícím cestu k vygenerovanému

python souboru. Tímto je zahájeno vykonávání skriptu a uživatelský program po jeho dokončení pokračuje dále.

Pro pohodlí uživatele jsou implementovány dvě třídy, Control, které obsahuje veškeré metody na nejnižší možné úrovni využití API a je následně použita při implementaci třídy Tasks, umožňující jednoduše volat složitější předdefinované příkazy. Můžeme předpokládat, že pro většinu uživatelů budou příkazy nabízené třídou Tasks dostačující, což nám umožní rychleji provádět případně požadované změny a implementaci nových funkcionalit za použitít již připravené třídy Control.

5.1.1 Rozbor konkrétního použití

V této části je podrobněji vysvětlena logika funkčnosti knihovny na praktickém příkladu s ukázkami kódu. Pro účely pochopení práce knihovny a demonstarci nabízených funkcionalit zvolíme příklad otevření projektu, úpravy a vyčtení IP adresy a následného uložení projektu zpět do počítače.

Aby bylo možno knihovnu použít, je nejprve třeba inicializovat objekt třídy Control, který nabízí veškeré funkce knihovny.

```
1 control = Control(path=r"<path for temp files>",
2                    shell_path=r"<path to the shell>")
```

V tuto chvíli nám již nic nebrání otevřít projekt a přihlásit se (čímž nahrajeme projekt do PLC). Tyto úkony provedeme pomocí metod download_project() a login_project(), které se však v tuto chvíli nevykonávají, ale pouze přidají do seznamu příkazů, které tyto kroky později provedou. Můžeme si povšimnout, že metody přijímají argumety názvů proměnných, čímž je umožněno definovat i specifické chování programu, které při tvorbě knihovny nemuselo být nutně zohledněno.

```
1 control.download_project(r'<path to project>', "proj")
2 control.login_project("proj")
```

Nyní je možné pomocí metody find_in_project() nalézt komunikační rozhraní, obsahující seznam svých parametrů, který obdržíme zavoláním metody get_parameters().

```
control.find_in_project("<project device name>", "dut", "proj")
control.get_parameters("params", "dut")
```

IP adresu je nyní možné zapsat metodou change_ip() která adresu zapíše. Následně pak metoda get_ip() uloží hodnotu IP adresy do proměnné se specifikovaným názvem, v tomto případě ip_var. Příkaz show() následně tuto hodnotu vytiskne

```
new_ip = "<IP>"
control.change_ip("params", new_ip)
control.get_ip("params", "ip_var")
control.show("ip_var")
```

Nyní můžeme projekt uložit zpět do počítače pomocí metody upload|_project(), která jako argumenty přijmá název proměnné obsahující otevřený projekt, cestu k souboru, do kterého bude tento projekt uložen, heslo k projektu a jako poslední nastavení, zda má dojít k přepsaní případného již existujícího soubroru pod definovanou cestou.

```
control.upload_project('proj',
         r'<Path to save project>',
         '<Password>',
        True)
control.flush()
```

Na závěr je zavolána metoda flush(), při čemž dochází k uložení definované posloupnosti příkazů a vygenerování python souboru, který je jako argument předán při spuštění exe souboru, námi nazývaného shell, který tento skript spustí a vykoná v prostředí IronPythonu 2.7.

Nyní demonstrujme způsob vykonání stejné úlohy, tentokrát ale za použití třídy Tasks, které třídu Control obaluje a za cenu parametrizovatelnosti a flexibility usnadňuje její použití.

```
tasks = Tasks(shell_path="<path to shell>"
            project="<Path to project>")
tasks.set_ip("192.168.0.101", "dut2")
tasks.load_ip("<Variable name>", "<Communication device name>")
tasks.show("<Variable name>")
tasks.upload_project("<Path to save project>")
tasks.download_project()
```

Třída tasks přímo neimplementuje žádné nové funkcionality, ale pouze spojuje příkazy připravené třídou Control, čímž se vytváří snadno využitelné

sekvence příkazů. V této ukázce můžeme vidět použití knihovní třídy Tasks, které také odpovídá standardu pojmenování metod využívané pro kontrolu PLC jiných výrobců, čímž je zvýšena čitelnost kódu a usnadněno její použití pro nového uživatele.

5.2 Struktura

Knihovna obsahuje balíček "core"(jádro), ve kterém se nachází celá struktura knihovny využívaná koncovým uživatelem. Dále se zde nachází složka "examples"(příklady) a další pomocné soubory, jako je readme, nebo soubory definující proces tvorby balíčku po nahrání na serverové úložiště. Postupně nyní rozebereme tyto jednotlivé části struktury.

- **Jádro:** Obsahuje celou knihovnu, ke které uživatel přistupuje. Nacházejí se zde jak složky obsahující potřebná data a pomocné soubory, tak i python soubory se skripty tříd Control a Tasks.
  - Templates: (šablony), je složka která obsahuje textové soubrory s parametrizovatelnými částmi skriptů, které jsou využívány pro skládání a generování python souboru zmíněného v sekci 5.1.
  - Helpers: adresář obsahující pomocné skripty, jako jsou script_runner.py a template_loader.py, které slouží k vytvoření a spuštění generovaného python souboru.
  - Temp: Je místem k ukládání dočasných souborů, vytvářených knihovnou, jako je například generovaný python soubor, nebo json soubor využívaný při načítání proměnných z prostředí IronPythonu do prostředí řídícího skriptu.

  V souboru tasks.py je implementována třída Tasks, využívající třídu Control, která dědí od svých jednotlivých podtříd nacházejících se ve zbylých souborech.
- **Příklady:** Tato složka obsahuje ukázkové kódy jak ke třídě Control, tak ke třídě Tasks. Kódy ukazují možnosti základního použití a předvádí uživateli práci se třídou Control. Pro třídu Tasks je pokryta většina nabízených funkcionalit.
- **Pomocné soubory:** Mezi pomocnými soubory se zaměřím hlavně na pyproject.toml a azure_pipelines.yml. Soubor pyproject.toml definuje

vlastnosti balíčku, jako jeho název, obsažené soubory, nebo verze. Azure-pipelines.yml specifikuje postup, který se provede při vytváření balíčku. Automatické sestavení balíčku je spuštěno po nahrání commitu do větve release, po němž následuje jeho zařazení mezi artefakty.

```
schneider_plc_control
├── core
│   ├── templates
│   │   ├── open_project.txt
│   │   ├── login_project.txt
│   │   └── ...
│   ├── helpers
│   │   ├── script_runner.py
│   │   └── template_loader.py
│   ├── temp
│   │   ├── control_script.py
│   │   ├── data.json
│   │   └── shell.txt
│   ├── base.py
│   ├── user.py
│   ├── cpu.py
│   ├── project.py
│   ├── app.py
│   ├── control.py
│   └── tasks.py
├── examples
│   ├── control
│   │   └── set_ip.py
│   ├── tasks
│   │   └── set_ip.py
│   └── test_cases
│       ├── common
│       │   ├── config.yml
│       │   ├── status_parser.py
│       │   └── tshark.py
│       ├── cycle_times.py
│       ├── start_stop.py
│       └── wirebreak.py
├── README.md
├── requirements.txt
├── pyproject.toml
├── azure-pipelines.yml
└── manifest.in
```

Kapitola 6

Testovací scénáře

Následující kapitola je věnována popisu implementovaných testovacích scénářů, testovacích skriptů a funkčním vylepšením knihovny, ke kterým na základě jejího využití došlo.

6.1 Implementace

Testovací scénáře jsou implementovány tak, aby splňovaly pravidla zmíněná v sekci 3.3 a demonstrují využití nové knihovny. Jedná se tedy o scénáře, které bylo doposud nutné provádět manuálně, kvůli potřebě změny konfigurace za běhu testu. Je tedy možné implementovat nové testy, nebo spojovat již implementované testy (detailněji rozebráno v sekci 3.2).

Každý test není tvořen pouze testovacím skriptem, ale jeho součástí je také jeho definice. Za tímto účelem je běžně využívána tabulka popisující předpoklady ("preconditions") pro správnou funkčnost testu a jednotlivé kroky testu. Každý krok by pak měl obsahovat stručný popis a očekávaný výsledek. Tedy například vyčtení definovaného parametru bude mít jako výseledek předpokládaná vyčtená data.

6.1.1 Start/Stop CPU

Implementace testu při kterém dojde k zastavení a opětovnému spuštění aplikace na PLC byla provedena pomocí nově vyvinuté knihovny, která umožňuje manipulaci s PLC. Provedení testu zahrnuje čtení aktuálních dat z testovaného zařízení a následné porovnání s očekávanými hodnotami, což slouží jako indikátor úspěšného zastavení aplikace a správného chování tohoto zařízení. Vzhledem k nezbytnému zastavení aplikace v průběhu testu byla dosavadní realizace testování prováděna manuálně. Avšak díky nově implementované knihovně je nyní možné tento test automatizovat.

Test		Start/Stop CPU
Předpoklady		
Zařízení komunikuje pomocí EtherNet/IP		
Probíhá komunikace s PLC		
Krok		**Očekávané výsledky**
1	Zastavení PLC	
2	Vyčtení s následujícími parametry **Class 0x01, instance 0x01, attr. = 0x05** [9]	Návratová hodnota: 0, extended device status: 7, odpovídající bit: 0
3	Spuštění PLC	
4	Vyčtení s následujícími parametry **Class 0x01, instance 0x01, attr. = 0x05**	Návratová hodnota: 0, extended device status: 6, odpovídající bit: 1

Tabulka 6.1: Testovací scénář zastavení a spuštění PLC

Tabulka 6.1 popisuje předpoklady definující nastavení komunikace mezi PLC a testovaným zařízením před zahájením testu. Uvedeny jsou jednotlivé kroky prováděné během testu, které již přesně definují prováděné akce a jejich očekávané výsledky. Kroky 1 a 3 jsou prováděny za pomoci nové knihovny a zajišťují již zmíněné zastavení a spuštění aplikace. Za povšimnutí stojí také kroky 2 a 4, které zajišťují kontrolu chování zařízení při náhlém zastavení PLC.

Parametry definované při vyčítání dat z PLC skládají adresu, která definuje, jaká data chce uživatel vyčíst. K samotnému čtení je využita interní knihovna, která vrací návratovou hodnotu 0, pokud nedošlo k chybě, a dále pak vyčtená data. V tomto testu se zajímáme primárně o "extended device status", který definuje aktuální stav zařízení. Jeho hodnoty 6 (alespoň jedno I/O spojení běží) a 7 (alespoň jedno I/O spojení navázáno, všechna v idle módu) definují stav komunikace s PLC.

Idle mód je možné chápat jako stav zařízení, kdy probíhá komunikace s nadřazeným systémem, tedy s PLC, ale data vyměňovaná v rámci této komunikace se ze strany zařízení nemění. Současně je signalizováno zastavení aplikace, z čehož lze ověřit, že zařízení je správně připojeno a komunikuje, ale jeho aplikace neběží.

6.1.2 Změna RPI

V tomto testovacím scénáři dochází ke změně RPI a následnému ověření jeho délky využitím konzolové aplikace tshark. Pomocí knihovny schneider_plc_control dojde k přehrání projektu v PLC, kde jedinou odlišností mezi nimi je právě nastavené RPI. Následně je zaznamenána komunikace, ze které je následně možné určit intervaly v jakých komunikace probíhá a ověřit tak nastavený RPI.

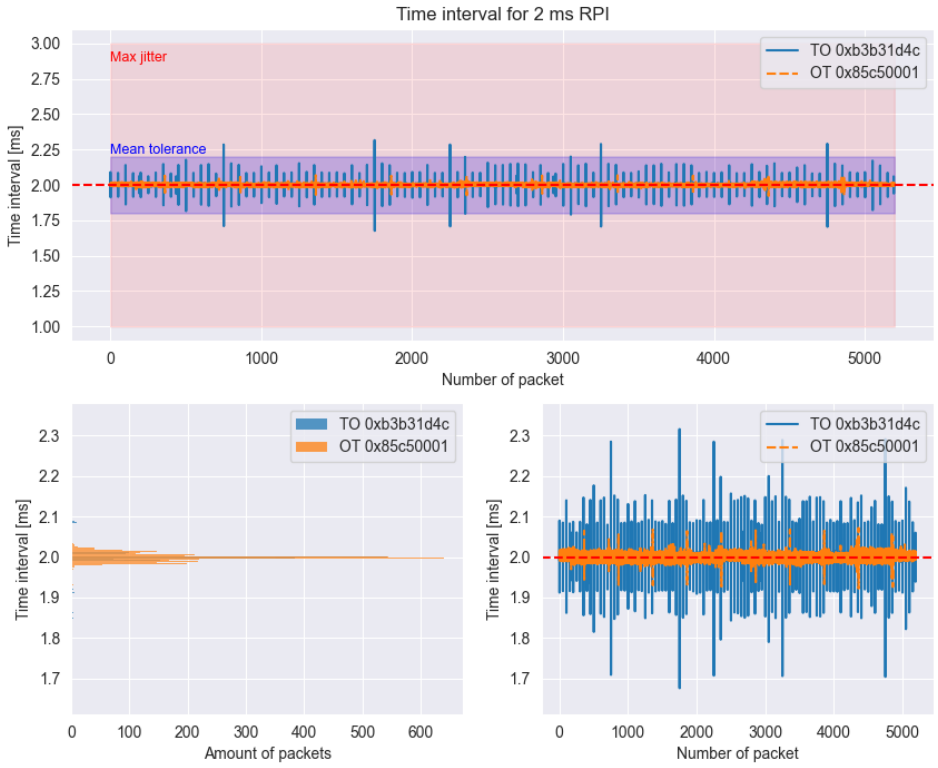
Test		Kontrola RPI
Předpoklady		
Zařízení komunikuje pomocí EtherNet/IP		
Probíhá komunikace s PLC		
Krok		**Očekávané výsledky**
1	Stažení projektu s **RPI 2 ms**	
2	Kontrola **RPI: 2 ms** na 1 min dlouhém záznamu komunikace.	Pakety jsou zaznamenány pomocí tsharku, je spočten interval a vyhodnocen. Standardní deviace **10%**, maximální zpoždění **50%** a průměr v toleranci **10%**
3	Stažení projektu s **RPI 10 ms**	
4	Kontrola **RPI: 10 ms** na 1 min dlouhém záznamu komunikace	Pakety jsou zaznamenány pomocí tsharku, je spočten interval a vyhodnocen. Standardní deviace **10%**, maximální zpoždění **50%** a průměr v toleranci **10%**
5	Stažení projektu s **RPI 50 ms**	
6	Kontrola **RPI: 50 ms** na 1 min dlouhém záznamu komunikace	Pakety jsou zaznamenány pomocí tsharku, je spočten interval a vyhodnocen. Standardní deviace **10%**, maximální zpoždění **50%** a průměr v toleranci **10%**

Tabulka 6.2: Testovací scénář kontrola RPI

Opět si můžeme povšimnout, že nově implementovaná knihovna je využita v krocích 1, 3 a 5, kdy dochází ke změně RPI, tedy k přehrání projektu (viz sekce 4.4). V krocích 2, 4 a 6 je využita konzolová aplikace tshark, která je ovládána pomocí třídy Tshark, která tuto aplikaci obaluje. Třída umožňuje zahájit sniffování s přednastvenou délkou trvání na definovaném zařízení profishark následujícím způsobem.

```
interface_list = ["Profishark <mac address>"]
cap_file = r"<Path to capture file>"
dump_t = Tshark(r"<Path to tshark.exe>", interface_list)
dump_t.capture(cap_file, sniff_time=10, logg=True)
```

Po uložení záznamu komunikace jsou data ze souboru nahrána testovacím skriptem a zpracována. Jsou spočteny časové rozdíly mezi jednotlivými pakety, které jsou následně porovnány s požadovanou hodnotou tak, jak uvádí tabulka 6.2. Na obrázku 6.1 můžeme vidět rozložení naměřených intervalů mezi pakatey pro nastavené RPI 2 ms. Legenda značí "communication ID", tedy unikátní kód relace. V tomto případě je navázáno pouze jedno spojení a tato ID tedy můžeme chápat jako směr komunikace mezi PLC a testovaným zařízením.



Obrázek 6.1: Graf měření RPI 2 ms

6.1.3 Wirebreak

Tento testovací scénář pokrývá situaci, kdy očekáváme hodnotu 1 (zapnuto) na nějakém vstupu a v případě, kdy tato podmínka není splněna, je hlášena chyba. Test tedy zkoumá, zda dojde ke správnému vyhodnocení situace a signalizaci chybového stavu.

Test		Wirebreak
Předpoklady		
Zařízení komunikuje pomocí EtherNet/IP		
Probíhá komunikace s PLC		
Diagnostika Wirebreaku je povolena		
Na zařízení neběží Grayův kód		
Krok		**Očekávané výsledky**
1	Nastavení výstupů do 0	Výstupní, a korspondující vstupní, kanály jsou v 0
2	Vyčtení s následujícími parametry **Class 0x01, instance 0x01, attr. = 0x05**	Návratová hodnota: 0, extended device status: 5, major recoverable fault: 1
3	Nastavení výstupů do 1	Výstupní, a korspondující vstupní, kanály jsou v 1
4	Vyčtení s následujícími parametry **Class 0x01, instance 0x01, attr. = 0x05**	Návratová hodnota: 0, extended device status: 6, major recoverable fault: 0
5	Nastavení výstupů do 0	Výstupní, a korspondující vstupní, kanály jsou v 0
6	Vyčtení s následujícími parametry **Class 0x01, instance 0x01, attr. = 0x05**	Návratová hodnota: 0, extended device status: 5, major recoverable fault: 1

Tabulka 6.3: Testovací scénář wirebreak

Zde se povšimněme, že byly přidány dodatečné předpoklady pro správný běh testu. Zapnutá diagnostika wirebreaku a zastavení spínání výstupů dle Grayova kódu. Zastavení běhu kódu je nezbytné, aby mohlo dojít k pevnému nastavení 0 (vypnto) (resp. 1 (zapnuto)) na výstupu, které je prováděno v krocích 1, 3 a 5. Za tímto účelem je opět využita knihovna schenider_plc_control a to následujícím způsobem.

```
tasks = Tasks(project="<path>", shell_path="<path>")
tasks.set_variables(["Test_Control.all_dq"], ["TRUE"])
tasks.start_cpu_app()
tasks.download_project()
```

Tato část kódu provede nastavení proměnné v projektu nazvané "all_dq"na hodnotu logické 1, čímž zajistí přenastavení výstupů. V následující ukázce můžeme vidět část kódu aplikace běžící na PLC, která zajišťuje změnu výstupů v závislosti na nastavení booleovské proměnné.

```
IF Test_Control.all_dq THEN
    %QW1 := 255;
END_IF;

IF Test_Control.all_dq = FALSE THEN
    %QW1 := 0;
END_IF;
```

Postup vyhodnocení je již obdobný jako v testu Start/Stop CPU 6.1.1. Dojde tedy k vyčtení dat z definované adresy a ta jsou následně porovnána s očekávanými hodnotami dle tabulky 6.3.

6.2 Funkčnost knihovny

V průběhu implementace testů došlo ke zjištění drobných nedostatků, zejména pak k potřebě implementovat nové funkcionality ke zpříjemnění práce s knihovnou. Jednalo se například o možnost dodatečných nastavení, jako jsou

- **Logging**: Nastavením parametru "logg"(výpis) je možno ovládat množství informací, které knihovna tiskne. Mezi těmito informacemi se nachází například i výstup z běhu generovaného skriptu v prostředí souboru shell.
- **Uložení cesty**: Jelikož umístění shell souboru se běžně nemění, stačí tuto cestu při inicializaci uvést pouze při prvním spuštění knihovny. Knihovna si tuto cestu uloží a při dalších použitích tedy již parametr shell_path může zůstat prázdný. Uloženou cestu přepíšeme jednoduše novým nastavením parametru shell_path.
- **Inteligentní otevření projektu**: Jelikož při volání metody run_commands dojde k vymazání pole, které ukládá posloupnost příkazů, bylo by nutné znovu inicializovat objekt. Třída Tasks však po provedení příkazů automaticky nastaví první příkazy tak, aby při dalším spuštění došlo ke znovuotevření daného projektu.
- **Přidání výpisů** - Přidání podrobnějších výpisů, které značí například špatnou cestu k shell souboru, neexistující projekt a další. Tato úprava umožňuje uživateli rychleji a přesněji určit, kde se vyskytuje chyba.

Kapitola 7

Zhodnocení

V této kapitole jsou popsány závěrečné změny, které byly provedeny po code review a testování knihovny v praktickém nasazení. Je zde také zhodnocena funkčnost knihovny jak vzhledem k nastaveným cílům, tak její praktické využitelnosti.

7.1 Navržené změny a řešení

V této sekci jsou diskutovány navržené změny kódu vzhledem k jeho čitelnosti, organizaci i praktickému použití. U každé změny je uveden důvod k jejímu provedení a stručně popsán způsob realizace.

- **Type hints**: Z důvodu jednosuššího ovládání knihovny a zpřehlednění kódu byly přidány typy argumentů funkcí.
- **Složka temp**: Pro lepší organizaci knihovny je přidána složka temp pro ukládání dočasných souborů, aby se tak zabránilo přidávání a odebírání souborů v kořenovém adresáři knihovny.
- **Odstranění uživatelského vstupu**: Jelikož by měla být knihovna využívána v automatizovaných testech, čekání na uživatelský vstup je nepřípustné. Bylo tedy třeba provést kontrolu, zda žádná metoda nevyžaduje akci uživatele. Tento nedostatek byl nalezen pouze v metodě

ukládající projekt ze zařízení do počítače a nahrazen novým parametrem této metody.

- **Změna importů**: Na mnoha místech v knihovně se vyskytovalo importování funkcí či tříd ve tvaru

```
1    from file import *
```

Za účelem zlepšení čitelnosti kódu byly tyto importy nahrazeny specifickými ve tvaru

```
1    from file import class, function
```

- **Stažení a uložení projektu přes tasks**: Třída Tasks nově implementuje metodu download_project, která umožňuje pouze stažení projektu do PLC. Metoda je využívána v případě, kdy je třeba pouze přehrát projekt.
- **Parametrizovatelnost testů**: Testy musí být při volání parametrizovatelné, tedy je třeba poskytnout uživateli možnost změnit libovolnou využitou hodnotu, jako jsou cesty k projektům, IP adresy, ale například i nastavení percentilu v testu ověřujícím RPI. Bylo tedy třeba provést drobné úpravy a testy plně parametrizovat předáním argumentů při inicializaci objektu daného testu.
- **Interakce s GUI**: Problémy nastaly při automatickém spouštění testů přes Azure Pipelines, jelikož nebylo možné interagovat s GUI a načíst názvy oken otevřených na počítači. Tento proces je proveden při kontrole, zda není projekt již otevřen v programu Machine Expert. V případě, kdy tedy není interakce s GUI možná, není tato kontrola vykonána a program pokračuje bez jejího provedení.

7.2 Zhodnocení implementace

Z pohledu udržitelnosti a rozšířitelnosti je knihovna navržena s ohledem na budoucí potřeby a možné rozšíření funkcí. Architektura je modulární a dobře dokumentovaná, což usnadňuje jak další vývoj nových funkcionalit, tak i údržbu stávajícího kódu. Díky této struktuře je možné efektivně přidávat nové prvky a komponenty bez zásadních změn v existujícím kódu, což přispívá k dlouhodobé životnosti knihovny a minimalizaci technického dluhu.

Dalším aspektem, který je důležitý z hlediska funkčnosti a udržitelnosti, je průběžné testování a ověřování kvality kódu. V rámci vývoje byly prováděny pravidelné testy, což přispívá k odhalení potenciálních chyb již v rané

fázi vývoje a umožňuje jejich rychlé opravy. Tímto způsobem je zajištěna nejen stabilita a spolehlivost knihovny, ale také minimalizováno riziko vzniku nedostatků a chyb.

Z funkčního hlediska lze konstatovat, že knihovna nedává prostor pro kritické nedostatky, které by bránily jejímu využití či zapříčiňovaly nepřiměřenou složitost jejího použití. Komplikace ve formě nemožnosti změny RPI a dalších parametrů, které jsou k zápisu nepřístupné bylo možné obejít pomocí přehrávání identických projektů s upravenými parametry. Toto řešení není zcela ideální, ale jelikož se jedná o důležitou funkcionalitu, je nutné využívat tento postup.

Knihovna bude převážně využívána pro automatizované testy, jejichž část byla již v rámci testování knihovny implementována. V případě, kdy dojde k negativnímu výsledku testu, je možné chybu snadno odhalit a rozhodnout o dalším postupu. Z hlediska samotných testů však nedochází k žádným chybám a lze tedy prohlásit, že nově implementované testy prokázaly očekávanou funkcionalitu a připravenost k integraci do existujícího testovacího frameworku.

Klíčovým prvkem těchto testů je využití nedávno implementované knihovny schneider_plc_control, která slouží jako základ pro komunikaci s PLC. Tímto způsobem je prakticky ověřena nejen funkčnost nových testovacích scénářů, ale také integrita a spolehlivost samotné knihovny. Tyto testy představují důležitý krok směrem k zajištění kvality a spolehlivosti knihovny a poskytují užitečnou zpětnou vazbu pro další vývoj a optimalizaci. Testy rovněž poskytují inspiraci pro uživatele knihovny a ukazují možnosti její integrace do testovacích scénářů.

7.3 Měřitelné výsledky

Tato sekce se věnuje vyhodnocení práce a to konkrétně v oblasti měřitelných výstupů. Postupně je diskutováno několik kritérií, v rámci čehož je zmíněn konkrétní přínos, který je zároveň interpretován a uveden do kontextu.

Prvním přínosem je zrychlení prováděných testů tak, jak uvádí tabulka 7.1, která popisuje změnu v množství potřebného času k provedení definovaného testu. Tyto časy budou relevantní, pouze pokud budou automatické testy postupně spouštěny, obecně je však možné říci, že automatické testy budou spouštěny přes noc a čas potřebný k jejich vykonání, z pohledu lidské práce,

bude tedy téměř nulový.

Test	Manuální [min]	Automatický [min]
Start/Stop CPU	2:20	0:50
RPI	9:35	2:32
Wirebreak	2:55	1:21
Součet	**14:50**	**4:43**

Tabulka 7.1: Čas potřebný k provedení testů

Z tabulky můžeme vidět, že manuální výkon testů zabere alespoň trojnásobek času, potřebného k provedení automatických testů. Je ovšem nutné zdůraznit, že množství odvedené práce při obsluze automatizovaných testů je v porovnání s provedením všech kroků manuálně výrazně nižší.

Druhým výrazným přínosem je snížení nákladů na potřebný hardware. Jelikož nebylo doposud možné přehrávat projekty, byla využívána dvě totožná PLC společnosti Schneider, aby mohlo dojít ke spuštění alespoň nějakých automatizovaných testů. K plné automatizaci by ale bylo zapotřebí 4 PLC. S novou knihovnou je již možné spouštět testy postupně na jednom PLC, čímž se ušetří nákup dvou nových a dojde k uvolnění jednoho PLC. Plná automatizace by tedy při ceně okolo 45 000 Kč za kus stála okolo 135 000 Kč. [10]

Knihovna umožnila implementaci dalších automatických testů, což pomohlo zvýšit pokrytí automatizovanými testy z původních 5 na současných 22. Automatizace je však stále v rané fázi vývoje, a s jistotou lze říci, že v budoucnu dojde k výraznějšímu navýšení, jelikož bude automatizováno mnoho dalších testů.

Kapitola 8

Závěr

8.1 Vyhodnocení

Prvním krokem k implementaci funkční knihovny bylo zjištění požadavků budoucích uživatelů, jejichž analýza je součástí kapitoly 3. Po vyhodnocení možností samotného API, popsaných v kapitole 4, bylo možné přistoupit k implementaci knihovny.

Knihovna je navržena s ohledem na udržitelnost a rozšířitelnost, což se projevuje modulární architekturou a dobrou dokumentací, umožňující efektivní přidávání nových funkcí a komponent, jak je uvedeno v kapitole 5. Pravidelné testování a ověřování kvality kódu zajišťuje stabilitu a spolehlivost knihovny. Nově implementované testy, popsané v kapitole 6, prokázaly funkčnost a připravenost k integraci do existujícího testovacího frameworku, poskytující užitečnou zpětnou vazbu pro další vývoj a optimalizaci.

Celkově práce přináší možnost pohodlnějšího vykonávání testů za současného snížení nákladů, jak časovývch, tak i finančních. Detaily tohoto zhodnocení je možné nalézt v kapitole 7, která uvádí konkrétní přínosy knihovny.

Nedostatkem je nemožnost změny určitých parametrů, jako je například RPI, což je možné řešit přehráním projektu s přenastaveným parametrem. Tyto problémy vzniklé při implementaci jsou detailněji popsány v části 4.4.

Na základě interního code review a dodatečného testování byly také provedeny změny uvedené v sekci 7.1, kde jsou tyto úpravy odůvodněny.

8.2 Možnosti rozšíření

- **Podpora jiných zařízení** V rámci této knihovny se nebude vyvíjet přímá podpora nových zařízení. Namísto toho bude knihovna integrována do většího balíčku, což umožní ovládání PLC od různých výrobců, jako jsou Siemens, Mitsubishi, Allen-Bradley atd. Tímto způsobem se zajistí, že uživatelé budou mít jednotnou strukturu pro snadnější používání, aniž by bylo nutné rozšiřovat schopnost knihovny komunikovat s dalšími typy PLC a zařízeními přímo.
- **Přidání funkcionalit** Rozšíření funkcionalit zahrnuje přidání nových možností pro nastavení projektu a konfiguraci, které poskytují uživatelům širší spektrum možností při práci s knihovnou. Tato rozšíření budou závislá na nově vznikajících požadavcích uživatelů a návrzích na nové způsoby využití.
- **Rozšíření testovacích scénářů**: Budoucím cílem v oblasti testování je nejen vytvoření nových testů, ale i jejich kontinuální vylepšování tak, aby stále efektivněji pokrývaly testované vlastnosti zařízení. Tvorba automatických testů přináší nejen zvýšení pokrytí funkcí a vlastností testovaných zařízení, ale také přináší širší využitelnost testů.

  Zobecnění testovacích scénářů přináší také několik významných výhod, které lze aplikovat v různých kontextech. Jedním z těchto výhodných aspektů může být schopnost testovat více zařízení současně v rámci jediného testovacího scénáře, což zvýší praktickou využitelnost a usnadní použití testů.

Příloha A

Literatura

[1] PI North America. Profinet rt: Real-time performance, 2024. URL: https://us.profinet.com/profinet-rt-real-time-performance.

[2] CiA. Canopen – the standardized embedded network, 2024. URL: https://www.can-cia.org/canopen/.

[3] D. Collins. What are program organization units (pous) in plc programming?, 2024. URL: https://www.motioncontroltips.com/what-are-program-organization-units-in-plc-programming/.

[4] Deos. Modbus: Einfach erklärt, 2023. URL: https://www.deos-ag.com/de/blog/modbus-einfach-erklaert/.

[5] Schneider Electric. Accessing the python interpreter in ecostruxure machine expert, 2023. URL: https://product-help.schneider-electric.com/Machine%20Expert/V2.2/en/SoMProg/index.htm#t=SoMProg%2FD-SE-0083834.html.

[6] Schneider Electric. Codesys script engine examples, 2023. URL: https://product-help.schneider-electric.com/Machine%20Expert/V2.2/en/SoMProg/index.htm#t=SoMProg%2FD-SE-0083864.html.

[7] Schneider Electric. Using the logic builder shell, 2023. URL: https://product-help.schneider-electric.com/Machine%20Expert/V2.2/en/SoMProg/index.htm#t=SoMProg%2FD-SE-0083835.html.

[8] Schneider Electric. Data sheet modicon m262, 2024. URL: https://www.se.com/us/en/product/download-pdf/TM262L20MESE8T?filename=Schneider+Electric_Logic-Motion-controller-Modicon-M262_TM262L20MESE8T.pdf.

[9] Schneider Electric. Ethernet/ip device, 2024. URL: https://product-help.schneider-electric.com/Machine%20Expert/V1.2/en/lmcprg/lmcprg/Ethernet_Configuration/Ethernet_Configuration-17.htm#XREF_D_SE_0002963_6.

[10] Schneider Electric. logic controller, modicon m262, 2024. URL: https://www.se.com/us/en/product/TM262L20MESE8T/logic-controller-modicon-m262-3ns-per-instruction-ethernet.

[11] Idekit. Definice činnosti regulátoru plc, 2024. URL: https://www.idekit.cz/cs/dokumentace/definice-cinnosti-regulatoru-plc.

[12] Vector Informatik. Canopen - höherschichtiges can-protokoll, 2024. URL: https://www.vector.com/de/de/know-how/protokolle/canopen.

[13] F. Lamb. Plc programming language fundamentals for improved operations, maintenance, 2023. URL: https://www.controleng.com/articles/plc-programming-language-fundamentals-for-improved-operations-maintenance/.

[14] machinemetrics.com. Enabling industrial automation with ethernet/ip, 2024. URL: https://www.machinemetrics.com/connectivity/protocols/ethernet-ip.

[15] Dr. M.Chakravarthy. *Documentation on programmable logical controller (PLC)*. Vasavi college of engineering, 1994.

[16] M. Phan. What is a plc made of? basic plc hardware guide, 2021. URL: https://www.bostontech.net/what-is-a-plc-made-of-basic-plc-hardware-guide-for-beginner/.

[17] Profinet. Profinet explained, 2024. URL: https://www.profinet.com/profinet-explained.

[18] Profitap. Profishark 1g installation and configuration manual, 2016. URL: https://www.intex.com.pl/wp-content/uploads/2016/06/ProfiShark-1G-Manual.pdf.

[19] Siemens. Simatic et 200al – for effortless installation anywhere, 2017. URL: https://assets.new.siemens.com/siemens/assets/api/uuid:6b752dd5-01d2-41e5-a1ff-5322a0d90c5d/et-200al-leaflet.pdf.

[20] Siemens. Sales release: Fail-safe i/o modules for simatic et 200sp ha distributed i/o system, 2020. URL: https://support.industry.siemens.com/cs/document/109780827/sales-release-fail-safe-i-o-modules-for-simatic-et-200sp-ha-distributed-i-
dti=0&lc=en-WW.

[21] Siemens. Simatic s7-1500 tm, 2023. URL: https://www.siemens.com/at/de/produkte/automatisierung/systeme/industrie/sps/simatic-s7-1500.html.

[22] Siemens. Data sheet 6es7155-6mu00-0cn0, 2024. URL: https://assets.alliedelec.com/v1652782306/Datasheets/8c5aa9728f0e89b382c9e1cfa8159d32.pdf.

[23] Pyramid Solutions. What is ethernet/ip?, 2020. URL: https://pyramidsolutions.com/network-connectivity/blog-nc/what-is-ethernet-ip.

[24] Jiří Sál. Porovnání průmyslových komunikačních sběrnic, 2018. URL: https://dspace.tul.cz/server/api/core/bitstreams/deaa72d0-5bad-4dad-a5bb-1245ed387aa4/content.

[25] tp link. Tl-sg1024 specification, 2024. URL: https://www.tp-link.com/de/service-provider/unmanaged-switch/tl-sg1024/#specifications.

[26] Wachendorff. Industrial ethernet modbus tcp. URL: https://www.wachendorff-prozesstechnik.de/technologie/industrial-ethernet/modbus-tcp/.

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Künstler** Jméno: **Jaroslav** Osobní číslo: **508496**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra řídicí techniky**

Studijní program: **Kybernetika a robotika**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Vývoj Knihovny a Testovacího Frameworku pro PLC od Schneider: Inovace v testování pro průmyslovou automatizaci

Název bakalářské práce anglicky:

Development of a library and Testing Framework for Schneider PLC: Innovation in testing for industrial automation

Pokyny pro vypracování:

Cíl práce:
Vytvoření nové knihovny pro rozšíření stávajícího testovacího frameworku, umožňujícího efektivní integraci s API od společnosti Schneider Electric pro ovládání PLC. Rozšíření rozsahu stávajících testovacích scénářů s využitím nové knihovny.
Průzkum a Analýza:
Prozkoumání a porozumění API Schneider PLC, identifikace klíčových funkcionalit pro automatizované testování. Analýza existujícího testovacího frameworku a související technologie (PLC, distributed IO. zařízení pro průmyslovou automatizaci, průmyslové komunikační standardy)
Kreativní komponenta:
Navržení a implementace python balíčku s využitím API Schneider PLC v rámci testovacího frameworku. Navržení a realizace testovacího scénáře s využitím Schneider API na definovaném distributed IO systému, spouštění scénářů a ověření očekávaných výsledků.
Hodnocení výsledků:
Knihovna projde vnitřním code review a bude k dispozici na interním serveru. Definované funkcionality jsou ověřeny využitím v automatických testovacích scénářích. Srovnání výsledků před a po implementaci nového řešení z hlediska časové náročnosti a stability definovaných testovacích scénářů.

Seznam doporučené literatury:

[1] Schneider Electric – EcoStruxure Machine Expert Script Engine User Guide - 12/2018
[2] Jiří Sál – Porovnání průmyslových komunikačních sběrnic – Technická univerzita Liberec – 2019
[3] Siemens s.r.o. – SIMATIC ET 200SP System Manual – Publikováno na support.industry.siemens.com - 11/2023
[4] Dr. M.Chakravarthy – Documentation on programable logical controller – Vasavi college of engineering

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Ondřej Bělovský Siemens s.r.o., Siemensova 2715/1, 155 00 Praha 13

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Ing. Martin Hlinovský, Ph.D. katedra řídicí techniky FEL

Datum zadání bakalářské práce: **18.01.2024** Termín odevzdání bakalářské práce: **24.05.2024**

Platnost zadání bakalářské práce:
do konce letního semestru 2024/2025

Ing. Ondřej Bělovský	prof. Ing. Michael Šebek, DrSc.	prof. Mgr. Petr Páta, Ph.D.
podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

. _____ _____

Datum převzetí zadání Podpis studenta

