Univerzita Pardubice
Fakulta elektrotechniky a informatiky

Příprava úloh pro výuku v laboratoři průmyslové automatizace

Bakalářská práce

2025 Leoš Preisler

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: 2024/2025

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Leoš Preisler**
Osobní číslo:	**I22026**
Studijní program:	**B0714A150008 Automatizace**
Téma práce:	**Příprava úloh pro výuku v laboratoři průmyslové automatizace**
Zadávající katedra:	**Katedra automatizace a matematiky**

Zásady pro vypracování

Cíl:
Zpracovat studentské úlohy pro výuku v laboratoři průmyslové automatizace. Úlohy musí využívat HW konfiguraci v laboratořiprůmyslové automatizace.
Teoretická část:
1. Vypracujte rešerši na téma: programovatelné automaty používané v průmyslu. V rešerši se zaměřte na automaty současně dostupné na trhu a na budoucnost programovatelných automatů.
2. Popište použitý programovatelný automat Siemens Simatic S7-1200, jeho zapojení a vývojové prostředí TIA Portál použité při řešení.
Praktická část:
3. Vypracujte studentské úlohy na ovládání výukového modelu motoru v prostředí TIA Portál a v prostředí Matlab.
4. Porovnejte získané výsledky z vypracovaných úloh z PLC automatů s výsledky získanými měřením v aplikaci Matlab Simulink.

Rozsah pracovní zprávy: **cca 40 stran**
Rozsah grafických prací:
Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

KARBAN, Pavel. *Výpočty a simulace o programech Matlab a Simulink*. Brno: Computer Press, 2006. ISBN 80-251-1301-9.
KOBRLE, Pavel a PAVELKA, Jiří. *Elektrické pohony a jejich řízení*. 3. přepracované vydání. V Praze: České vysoké učení technické, 2016. ISBN 978-80-01-06007-0.

Vedoucí bakalářské práce: **Ing. Mgr. Václav Horčic, Ph.D.**
Katedra automatizace a matematiky

Datum zadání bakalářské práce: **15. prosince 2024**
Termín odevzdání bakalářské práce: **16. května 2025**

L.S.

prof. Ing. Petr Doležel, Ph.D. v.r.
děkan

Ing. Libor Kupka, Ph.D. v.r.
vedoucí katedry

V Pardubicích dne 24. ledna 2025

Prohlašuji:

Práci s názvem Příprava úloh pro výuku v laboratoři průmyslové automatizace jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Beru na vědomí, že v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, a směrnicí Univerzity Pardubice č. 7/2019 Pravidla pro odevzdávání, zveřejňování a formální úpravu závěrečných prací, ve znění pozdějších dodatků, bude práce zveřejněna prostřednictvím Digitální knihovny Univerzity Pardubice.

V Pardubicích dne 16. 5. 2025

Leoš Preisler v.r.

PODĚKOVÁNÍ

Děkuji vedoucímu bakalářské práce Ing. Mgr. Václavu Horčicovi, Ph.D. za podnětné rady, metodickou a odbornou pomoc při zpracování mé práce.

ANOTACE

Tato bakalářská práce se zabývá návrhem a realizací výukových úloh v laboratoři průmyslové automatizace s využitím programovatelného automatu Siemens S7-1200. V teoretické části je provedena rešerše současných programovatelných automatů, jejich klasifikace a možností využití v automatizační technice. Praktická část se zaměřuje na tvorbu konkrétních úloh ve vývojovém prostředí TIA Portal, komunikaci s výkonovou jednotkou přes RT-DAC/USB2 a následné porovnání dat získaných z prostředí MATLAB/Simulink. Výsledkem práce je funkční výukový modul pro realizaci měření statické charakteristiky a odezvy na řídicí impuls, včetně vyhodnocení přenosové funkce systému.

KLÍČOVÁ SLOVA

programovatelný automat, TIA Portal, Siemens S7-1200, statická charakteristika, impulsní odezva, průmyslová automatizace, RT-DAC/USB, přenosová funkce, Matlab, výuková úloha

TITLE

Design of Educational Tasks for a PLC-Based Automation Laboratory

ANNOTATION

This bachelor's thesis focuses on designing and implementing educational tasks in the industrial automation laboratory using the Siemens S7-1200 programmable controller. The theoretical part includes a literature review of currently available PLCs, their classification, and applications in automation technology. The practical part deals with developing specific tasks in the TIA Portal environment, communication with the power interface via RT-DAC/USB2, and comparing acquired data with MATLAB/Simulink measurements. The outcome of the thesis is a functional training module for measuring static characteristics and control impulse response, including the evaluation of the system’s transfer function.

KEYWORDS

programmable logic controller, TIA Portal, Siemens S7-1200, static characteristic, impulse response, industrial automation, RT-DAC/USB, transfer function, Matlab, educational task

OBSAH

SEZNAM ILUSTRACÍ A TABULEK

SEZNAM ZKRATEK A ZNAČEK

Seznam použitých zkratek

CPU – Central Processing Unit (centrální procesorová jednotka)

CSV – Comma-Separated Values (textový formát dat)

DAC – Digital-to-Analog Converter (digitálně-analogový převodník)

DB – Data Block (datový blok v PLC)

DC – Direct Current (stejnosměrný proud)

FBD – Function Block Diagram (programovací jazyk podle normy IEC 61131-3)

FB – Function Block (funkční blok v PLC)

FPGA – Field-Programmable Gate Array (programovatelné hradlové pole)

HMI – Human Machine Interface (rozhraní člověk–stroj)

Hz – hertz (jednotka frekvence)

I/O – Input/Output (vstup/výstup)

IEC – International Electrotechnical Commission (mezinárodní elektrotechnická komise)

IL – Instruction List (programovací jazyk podle normy IEC 61131-3)

IPC – Industrial PC (průmyslový počítač)

LAD – Ladder Diagram (programovací jazyk – žebříkový diagram)

MATLAB – Matrix Laboratory (výpočetní prostředí)

OB – Organization Block (organizační blok v PLC)

OPC UA – OPC Unified Architecture (standardizovaný komunikační protokol)

PC – Personal Computer (osobní počítač)

PLC – Programmable Logic Controller (programovatelný logický automat)

PWM – Pulse Width Modulation (pulzně šířková modulace)

RT-DAC – Real-Time Data Acquisition and Control (rozhraní pro reálný čas)

S7 – řada PLC automatů Siemens

SCADA – Supervisory Control and Data Acquisition (systém pro sběr a řízení dat)

SCL – Structured Control Language (textový programovací jazyk pro PLC)

Simulink – nástroj pro simulační modelování v MATLABu

SM 1223 – Signal Module 1223 (rozšiřující digitální I/O modul)

SM 1231 – Signal Module 1231 (rozšiřující analogový vstupní modul)

ST – Structured Text (textový programovací jazyk podle normy IEC 61131-3)

TIA Portal – Totally Integrated Automation Portal (vývojové prostředí Siemens)

V – volt (jednotka napětí)

WinCC – Windows Control Center (vizualizační software Siemens)

Seznam použitých značek a veličin

$G(s)$ – přenosová funkce v Laplaceově tvaru

ω – úhlová rychlost [rad·s^{-1}]

K – zesílení systému (bezrozměrné)

T – časová konstanta [s]

t – čas [s]

f – frekvence [Hz]

Δ – změna hodnoty

R^2 – koeficient determinace (míra shody)

π – matematická konstanta (3,14159…)

TERMINOLOGIE

Přenosová funkce – matematický model, který popisuje chování systému v Laplaceově tvaru. Vyjadřuje vztah mezi vstupem a výstupem systému a využívá se pro analýzu dynamických vlastností.

Cyklické přerušení (OB) – speciální blok programu v PLC, který se vykonává periodicky s pevně definovanou periodou. Používá se pro časově deterministické operace, jako je vzorkování nebo řízení.

Vzorkovací doba – časový interval mezi jednotlivými odečty nebo zápisy signálu. Určuje, jak často je systém aktualizován nebo měřen.

RT-DAC/USB2 – rozhraní s FPGA obvodem, které slouží k řízení a sběru dat v reálném čase. Umožňuje propojení výkonné jednotky (servo systému) s počítačem přes USB a přímou spolupráci s MATLAB/Simulink.

PWM signál – pulzně šířková modulace, metoda řízení výkonu motoru pomocí střídání zapnutého a vypnutého stavu s proměnnou délkou impulsu.

SM 1231 – analogový signálový modul PLC Siemens, který slouží ke čtení analogových vstupů, např. z tachogenerátoru nebo potenciometru.

TIA Portal – vývojové prostředí společnosti Siemens pro programování a konfiguraci PLC, HMI a dalších zařízení v rámci systému průmyslové automatizace.

Servo systém – pohonný systém využívající elektrický motor, zpětnou vazbu a řízení. Používá se pro přesné nastavení polohy, rychlosti nebo síly.

Enkodér – snímač, který převádí mechanický pohyb (např. otáčení hřídele) na digitální signál, obvykle ve formě impulsů.

Tachogenerátor – senzor, který generuje napětí úměrné úhlové rychlosti otáčení motoru. Slouží jako analogová zpětná vazba pro řízení.

Statická charakteristika – graf nebo tabulka, která popisuje ustálený vztah mezi vstupní a výstupní veličinou systému bez ohledu na čas.

Identifikace systému – proces určení matematického modelu systému na základě měřených dat. V této práci probíhá pomocí MATLAB/Simulink na základě impulsní odezvy nebo statických charakteristik.

Simulinkový model – grafická reprezentace matematického modelu nebo algoritmu vytvořená v nástroji Simulink. Slouží pro simulaci chování systémů a jejich řízení.

ÚVOD

Programovatelné automaty (PLC) tvoří základní prvek současné průmyslové automatizace. Jejich schopnost spolehlivého řízení technologických procesů, modulární struktura a dostupnost vývojových nástrojů z nich činí důležitý prostředek pro výuku v technických oborech. Vzdělávání v této oblasti proto nevyžaduje pouze zvládnutí teoretických principů a programovacích jazyků, ale také praktické dovednosti spojené s návrhem, realizací a vyhodnocením reálných úloh.

Tato bakalářská práce se zaměřuje na návrh a realizaci výukových úloh pro laboratorní model servomotoru. Pro řízení byla využita řídicí jednotka Siemens S7-1200, programovaná v prostředí TIA Portal. Cílem bylo realizovat měření statické charakteristiky a impulsní odezvy systému a následně získaná data porovnat s výsledky naměřenými v prostředí Matlab/Simulink, kde byly použity předem připravené referenční úlohy.

Matlab byl dále využit ke zpracování naměřených dat nebo k identifikaci přenosové funkce systému. Porovnání výsledků z obou přístupů umožnilo ověřit správnost implementace a zhodnotit přesnost měření. Práce tak spojuje programování PLC, praktické měření a následné datové zpracování do uceleného výukového celku využitelného při výuce automatizace a řízení.

1 TEORETICKÁ ČÁST

1.1 Programovatelné automaty používané v průmyslu

1.1.1 Úvod do PLC – účel a princip činnosti

Programovatelný logický automat (PLC – Programmable Logic Controller) je specializované elektronické zařízení určené k řízení technologických procesů. Hlavní výhodou PLC systémů je možnost rychlé a flexibilní změny řídicí logiky bez zásahu do fyzického zapojení, a to pouhou úpravou uživatelského programu. PLC nahrazují dříve používané reléové, pneumatické nebo elektronické řídicí systémy a výrazně zjednodušují implementaci i změny v řízení (Šmejkal, 1999).

Základním principem funkce PLC je tzv. cyklické zpracování. V každém cyklu automat postupně provádí tři hlavní kroky: nejprve načte aktuální stav vstupních signálů, poté vykoná uživatelský program uložený v paměti, a nakonec nastaví výstupy podle výsledků výpočtu. Doba jednoho cyklu se běžně pohybuje v řádu milisekund a závisí na výkonu řídicí jednotky i složitosti samotného programu.

PLC systémy jsou konstruovány tak, aby byly odolné vůči elektromagnetickému rušení, vibracím, prachu a výkyvům teplot. Díky tomu se staly běžnou součástí průmyslové automatizace v oblastech jako je výroba, doprava, skladové hospodářství, chemický a potravinářský průmysl, ale i v energetice a stavebnictví. Standardní součástí PLC jsou vstupní a výstupní moduly, programovatelná CPU jednotka a často také komunikační rozhraní pro propojení s dalšími systémy.

1.1.2 Historický vývoj PLC a jejich význam v automatizaci

Programovatelné logické automaty (PLC) vznikly koncem 60. let 20. století jako reakce na potřebu flexibilnějšího a efektivnějšího řízení výrobních procesů. Před jejich zavedením bylo řízení strojů a výrobních linek realizováno pomocí rozsáhlých reléových systémů, časovačů a pevných vodičových propojení. Jakákoli změna logiky řízení v těchto systémech byla velmi náročná a časově zdlouhavá (AutomationDirect, 2015).

V roce 1968 vydala divize Hydramatic společnosti General Motors specifikaci pro nový řídicí systém, který by nahradil reléové systémy v automobilové výrobě. Na základě této specifikace vznikl první programovatelný automat – Modicon 084, který byl uveden do provozu v roce 1969. Nově vyvinutý systém umožňoval programování řídicí logiky pomocí jednoduchého jazykového zápisu, přístupného i pro techniky bez hlubších znalostí programování (AutomationDirect, 2015).

V následujících dekádách došlo k prudkému rozvoji této technologie. V 70. a 80. letech se začaly objevovat PLC s podporou analogových signálů, komunikačních rozhraní a různých programovacích jazyků. Důležitým mezníkem byla standardizace programovacích jazyků podle normy IEC 61131-3, která umožnila širší a efektivnější využití PLC systémů v praxi.
Dnes jsou programovatelné automaty neoddělitelnou součástí moderní automatizace. Nacházejí uplatnění jak v jednoduchých jednoúčelových aplikacích, tak i v rozsáhlých distribuovaných systémech. Jejich modularita, rychlá odezva, spolehlivost a možnost komunikace s nadřazenými systémy z nich činí klíčový nástroj současného průmyslového řízení.

1.1.3 Klasifikace PLC a typické průmyslové použití

Trh s programovatelnými automaty nabízí širokou škálu zařízení od různých výrobců, které se liší Programovatelné logické automaty (PLC) lze klasifikovat podle různých kritérií, přičemž nejběžnější je rozdělení podle konstrukce na kompaktní (integrované) a modulární jednotky.

1.1.3.1 Kompaktní (integrované) PLC

Kompaktní PLC integrují všechny základní komponenty – napájecí zdroj, centrální procesorovou jednotku (CPU) a vstupně-výstupní (I/O) moduly – do jednoho zařízení. Tato konstrukce je vhodná pro menší aplikace s omezeným počtem I/O bodů, kde není požadována rozšiřitelnost systému. Výhodou je jednoduchá instalace a nižší pořizovací náklady. Nevýhodou je omezená flexibilita, protože počet a typ I/O nelze snadno měnit (GeeksforGeeks, 2023).

1.1.3.2 Modulární PLC

Modulární PLC se skládají z jednotlivých modulů, které lze snadno přidávat nebo odebírat podle potřeb konkrétní aplikace. Tato architektura umožňuje vysokou flexibilitu a škálovatelnost systému, což je ideální pro komplexní a rozsáhlé průmyslové procesy. Modulární PLC často podporují různé typy I/O modulů a komunikačních rozhraní, což usnadňuje integraci s dalšími systémy (Empowered Automation Solutions LLC, 2023).

1.1.3.3 Typické průmyslové použití PLC

PLC nacházejí široké uplatnění v různých odvětvích průmyslu díky své spolehlivosti, flexibilitě a schopnosti pracovat v náročných podmínkách. Mezi typické aplikace patří:

- **Výrobní linky**: Automatizace výrobních procesů, řízení montážních linek a sledování kvality.
- **Dopravní systémy**: Řízení dopravníkových pásů, výtahů a eskalátorů.

- **Potravinářský průmysl**: Kontrola balicích strojů, plnicích linek a sledování hygienických podmínek.
- **Energetika**: Řízení distribučních sítí, monitorování spotřeby energie a integrace obnovitelných zdrojů.
- **Stavebnictví a infrastruktura**: Automatizace osvětlení, klimatizace a bezpečnostních systémů v budovách (RS Components, 2023).

1.1.4 Programovací jazyky a normy pro PLC

Programování programovatelných logických automatů (PLC) bylo standardizováno mezinárodní normou IEC 61131-3, která definuje syntaxi a sémantiku jednotné sady programovacích jazyků pro programovatelné řadiče (IEC, 2013). Tato norma zahrnuje dva textové jazyky, Instruction List (IL) a Structured Text (ST), a dva grafické jazyky, Ladder Diagram (LD) a Function Block Diagram (FBD). Kromě toho definuje Sequential Function Chart (SFC) pro strukturování vnitřní organizace programů a funkčních bloků (PLCopen, 2016).

1.1.4.1 Textové jazyky

- **Instruction List (IL)**: Textový jazyk podobný assembleru, který je v nejnovější verzi normy označen jako zastaralý a nedoporučuje se pro nové projekty (IEC, 2013).
- **Structured Text (ST)**: Vysokoúrovňový jazyk podobný Pascalu, vhodný pro komplexní výpočty a algoritmy (IEC, 2013).

1.1.4.2 Grafické jazyky

- **Ladder Diagram (LD)**: Grafický jazyk připomínající elektrické schéma reléové logiky, oblíbený mezi elektrotechniky díky své vizuální podobnosti s tradičními obvody (IEC, 2013).
- **Function Block Diagram (FBD)**: Grafický jazyk využívající funkční bloky pro reprezentaci logických a aritmetických operací, umožňující snadné propojení funkcí a vhodný pro složitější systémy (IEC, 2013).

1.1.4.3 Strukturování programů

- **Sequential Function Chart (SFC)**: Grafický jazyk pro strukturování programů do sekvenčních kroků a přechodů, užitečný pro řízení procesů s definovanými sekvencemi operací (PLCopen, 2016).

Použití těchto standardizovaných jazyků přináší výhody jako je snazší údržba kódu, lepší přenositelnost programů mezi různými systémy a zjednodušené školení personálu. Adopce normy IEC 61131-3 přinesla jednotnost v programování PLC, což umožňuje inženýrům a technikům pracovat efektivněji a s menším rizikem chyb (PLCopen, 2016).

1.1.5 Vývojová prostředí a nástroje pro programování PLC

Programování programovatelných logických automatů (PLC) vyžaduje specializovaná vývojová prostředí, která umožňují efektivní tvorbu, testování a ladění řídicích programů. Moderní integrovaná vývojová prostředí (IDE) podporují standardizované programovací jazyky podle normy IEC 61131-3 a nabízejí širokou škálu funkcí pro různé fáze vývoje a údržby automatizačních systémů (CODESYS Group, 2023).

1.1.5.1 TIA Portal (Siemens)

Totally Integrated Automation Portal (TIA Portal) je komplexní inženýrské prostředí vyvinuté společností Siemens, které integruje různé nástroje pro automatizaci do jednotného rozhraní. Umožňuje programování PLC řady SIMATIC, konfiguraci HMI panelů, frekvenčních měničů a dalších zařízení. TIA Portal podporuje programovací jazyky dle IEC 61131-3 a nabízí pokročilé funkce pro simulaci, diagnostiku a správu energetiky (Siemens AG, 2023).

1.1.5.2 CODESYS

CODESYS je nezávislé vývojové prostředí, které implementuje normu IEC 61131-3 a je kompatibilní s řadou zařízení různých výrobců. Nabízí pokročilé funkce pro vizualizaci, simulaci a ladění aplikací, a je široce používán v evropském i světovém měřítku (CODESYS Group, 2023).

1.1.5.3 GX Works2 (Mitsubishi Electric)

GX Works2 je vývojové prostředí od společnosti Mitsubishi Electric určené pro programování PLC řady MELSEC. Nabízí možnosti konfigurace systému, programování, ladění a diagnostiky. Uživatelé oceňují integrovanou simulaci a přehledné rozhraní (Mitsubishi Electric Corporation, 2023).

1.1.5.4 Studio 5000 (Rockwell Automation)

Studio 5000 je vývojové prostředí společnosti Rockwell Automation, navržené pro PLC Allen-Bradley. Kombinuje různé nástroje do jednotného rámce, což zvyšuje efektivitu vývoje a zkracuje čas potřebný pro uvedení systému do provozu (Rockwell Automation, 2023).

1.1.6 Budoucnost programovatelných automatů (PLC)

Programovatelné logické automaty (PLC) procházejí významnou transformací, která je poháněna technologickým pokrokem a rostoucími požadavky na flexibilitu, konektivitu a inteligenci v průmyslové automatizaci. Následující trendy naznačují, jakým směrem se bude vývoj PLC ubírat v následujících letech.

1.1.6.1 Integrace umělé inteligence a strojového učení

Moderní PLC systémy začínají využívat algoritmy umělé inteligence (AI) a strojového učení (ML) pro prediktivní údržbu, optimalizaci procesů a adaptivní řízení. Očekává se, že do roku 2028 bude 25 % nových nasazení PLC virtualizováno a 60 % bude mít integrované AI schopnosti (Cloud Studio IoT, 2025).

1.1.6.2 Virtualizace a cloudová konektivita

S rostoucím důrazem na flexibilitu a škálovatelnost se PLC systémy přesouvají do virtualizovaného prostředí a cloudových platforem. Tento přístup umožňuje snadnější aktualizace, vzdálený přístup a integraci s dalšími systémy v rámci Průmyslu 4.0 (Control Engineering, 2024).

1.1.6.3 Edge computing a decentralizace

Implementace edge computingu umožňuje zpracování dat přímo na místě, kde jsou generována, což snižuje latenci a zvyšuje efektivitu. PLC systémy se tak stávají součástí decentralizovaných architektur, které podporují rychlé a autonomní rozhodování (RL Consulting Inc., 2024).

1.1.6.4 Kybernetická bezpečnost a standardizace

S rostoucí konektivitou PLC systémů roste i potřeba robustní kybernetické bezpečnosti. Výrobci a průmyslové organizace kladou důraz na implementaci bezpečnostních standardů a protokolů, které chrání systémy před potenciálními hrozbami (PowerTechMax, 2024).

1.1.6.5 Udržitelnost a energetická efektivita

Budoucí PLC systémy budou navrženy s ohledem na udržitelnost a energetickou efektivitu. Integrace s obnovitelnými zdroji energie a optimalizace spotřeby energie se stanou klíčovými aspekty při vývoji nových automatizačních řešení (MDPI, 2024).

1.2 Popis použitého PLC systému a jeho konfigurace v laboratorní úloze

1.2.1 Přehled PLC Siemens S7-1200

Programovatelné logické automaty řady Siemens S7-1200 jsou kompaktní a modulární řídicí jednotky určené pro automatizaci menších a středně rozsáhlých technologických celků.

Vyznačují se jednoduchou instalací, podporou průmyslových standardů a integrovanými funkcemi pro komunikaci, řízení i diagnostiku (Siemens AG, 2023).

V laboratorním systému byl použit konkrétní typ CPU 1215C DC/DC/DC, napájený 24 V DC. Tato jednotka nabízí 14 digitálních vstupů, 10 tranzistorových výstupů, integrované rozhraní PROFINET, 125 kB paměti pro uživatelský program a možnost rozšíření až o osm I/O modulů. Je vhodná pro aplikace, kde je požadována vysoká rychlost zpracování signálů a možnost připojení dalších zařízení prostřednictvím komunikačních modulů.

Ve školní laboratoři je CPU 1215C DC/DC/DC trvale zapojena v rozváděči R0 jako centrální jednotka systému. K ní jsou připojeny digitální a analogové rozšiřující moduly – konkrétně například SM 1223 a SM 1231 – které rozšiřují počet dostupných vstupních a výstupních signálů. Systém obsahuje také oddělené napájení, připojení k ovládacím tlačítkům, LED indikaci, servopohon a analogová čidla.

Celé zapojení systému odpovídá reálné sestavě dodané firmou ProjectSoft HK a.s. jakožto zhotovitelem laboratorního vybavení pro Univerzitu Pardubice (ProjectSoft HK a.s., 2024 interní dokumentace dostupná na pracovišti). Řídicí jednotka je programována ve vývojovém prostředí TIA Portal, které poskytuje nástroje pro konfiguraci, programování, simulaci a správu zařízení řady S7-1200.

1.2.2 Zapojení řídicí jednotky a konfigurace modulů

Řídicí jednotka Siemens S7-1200 CPU 1215C DC/DC/DC je ve školní laboratoři instalována jako hlavní řídicí prvek v rozváděči R0. Je napájena stejnosměrným napětím 24 V a obsahuje 14 digitálních vstupů a 10 tranzistorových výstupů. Tyto jsou dále rozšířeny pomocí přídavných modulů, které zajišťují kompatibilitu systému s různými typy vstupně-výstupních signálů.

V sestavě jsou zapojeny následující moduly:

- SM 1223: rozšiřující digitální vstupně-výstupní modul,
- SM 1231: analogový modul pro připojení čidel s napěťovým nebo proudovým výstupem.

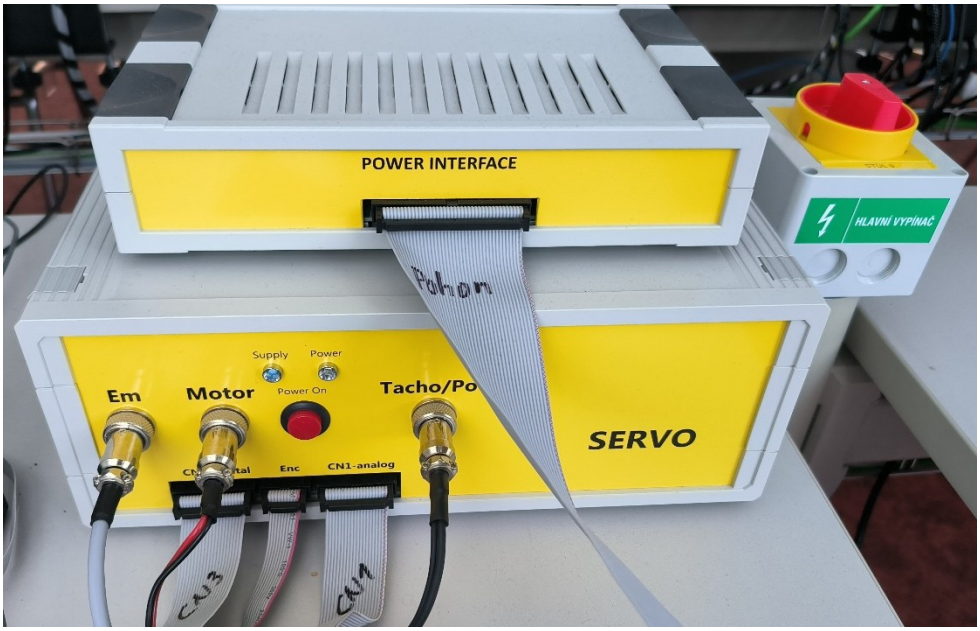
Digitální vstupy jsou připojeny k ovládacím tlačítkům, spínačům a koncovým snímačům. Tranzistorové výstupy slouží k řízení aktuátorů, jako jsou LED indikátory, reléové obvody nebo servopohony. Analogové signály jsou přiváděny například z potenciometrů nebo měřicích čidel s výstupem 0–10 V nebo 4–20 mA.

Všechna vedení jsou vyvedena přes svorkovnice v rozváděči, které umožňují přehledné připojení signálů a zjednodušují servisní zásahy. Systém je doplněn o resetovací a hlavní odpojovací tlačítko, jištění a další bezpečnostní prvky.

Součástí sestavy je také výkonová jednotka servosystému Inteco, která je propojena s PLC pomocí svorkovnicového pole označeného jako PLC Connectors. Signály mezi řídicí jednotkou a výkonovou částí jsou vedeny pomocí IDC páskových kabelů. Digitální řídicí signály zahrnují například Enable, PWM řízení, Brake, a analogová část obsahuje vstupy jako Potentiometer nebo Tacho. Toto zapojení je detailně dokumentováno v technických materiálech výrobce (Inteco, 2024a, interní dokumentace dostupná na pracovišti).

Fyzické provedení tohoto zapojení je znázorněno na Obrázek 1 a Obrázek 2. Na druhém obrázku je zachycen zadní pohled na výkonovou jednotku, kde jsou viditelné konektory PLC Connectors a přívod napájení. Na prvním je přední pohled s hlavním vypínačem a rozhraním Power Interface.

Celé zapojení odpovídá dokumentaci vypracované firmou ProjectSoft HK a.s., která dodala kompletní systém jako součást vybavení laboratoře automatizace (ProjectSoft HK a.s., 2024, interní dokumentace dostupná na pracovišti).



Obrázek 1 - Přední strana servosystému s rozhraním Power Interface

Obrázek 2 - Zadní panel výkonové jednotky Inteco s konektory PLC Connectors

1.2.3 Připojené periferie a signály

Při řešení laboratorní úlohy bylo řízení servopohonu realizováno prostřednictvím PLC Siemens S7-1200 a odpovídajících analogových a digitálních signálů. V této kapitole jsou popsány pouze ty periferie, které byly skutečně připojeny a aktivně využívány v průběhu experimentu.

1.2.3.1 Digitální výstupy

Z řídicí jednotky byly použity čtyři digitální výstupy pro ovládání výkonové části servosystému Inteco:

- **Enable** – aktivace napájení výkonové části,
- **Brake** – odblokování mechanické brzdy,
- **Direction** – nastavení směru otáčení motoru,
- **PWM** – šířkově modulovaný signál pro řízení výkonu motoru.

Výstupy byly zapojeny ze svorek PLC přes svorkovnice do zadního rozhraní výkonové jednotky označeného jako PLC Connectors. Propojení bylo realizováno pomocí IDC páskového kabelu dle specifikace výrobce (Inteco, 2024a, interní dokumentace dostupná na pracovišti).

1.2.3.2 Analogové vstupy

Analogové měřicí signály byly přivedeny do analogového modulu SM 1231, připojeného k řídicí jednotce. Konkrétně se jednalo o:

- **signál z tachogenerátoru**, který poskytuje napětí úměrné otáčkám motoru,
- **signál z potenciometru**, který reprezentuje žádanou hodnotu nebo simulovanou polohu.

Veškeré signály byly připojeny podle projektové dokumentace dodané firmou ProjectSoft HK a.s. a technických specifikací výrobce výkonové části. Ostatní periferní zařízení, která jsou

součástí laboratorního systému, nebyla v této úloze využita (ProjectSoft HK a.s., 2024; Inteco, 2024a, interní dokumentace dostupná na pracovišti).

1.2.4. Vývojové prostředí TIA Portal a struktura programu

K naprogramování a konfiguraci řídicí jednotky Siemens S7-1200 bylo využito vývojové prostředí Totally Integrated Automation Portal (TIA Portal). Tento integrovaný nástroj od společnosti Siemens slouží k návrhu hardwarové konfigurace systému, programování logiky řízení, správě vstupních a výstupních signálů, i k diagnostice a testování systému v reálném čase (Siemens AG, 2023).

TIA Portal umožňuje vytvářet projekty, ve kterých je možné definovat strukturu sestavy, adresaci jednotlivých signálů a programové bloky. Podporuje programovací jazyky definované v normě IEC 61131-3, konkrétně grafické jazyky LAD (ladder diagram) a FBD (function block diagram), a textový jazyk SCL (structured control language). V rámci konfigurace řídicí jednotky v této úloze byly vytvořeny výhradně datové struktury a symbolická pojmenování vstupů a výstupů, logika řízení nebyla realizována v TIA Portal, ale v prostředí MATLAB/Simulink.

Nastavení v TIA Portal sloužilo primárně k:

- přiřazení správných adres signálům v modulech SM 1223 a SM 1231,
- základnímu otestování funkčnosti vstupů a výstupů,
- ověření komunikace mezi PLC a výkonovou částí servosystému.

Kompletní řídicí algoritmus byl následně implementován v modelu vytvořeném v prostředí MATLAB/Simulink, který komunikoval s PLC přes podporu real-time přenosu dat.

Struktura projektu v TIA Portal tak sloužila jako podklad pro správnou interpretaci signálů a správné propojení mezi hardwarem a softwarovým řízením během experimentu.

1.2.5 Komunikace mezi PC a výkonovou částí pomocí RT-DAC/USB2

V rámci laboratorního systému bylo pro přímé řízení a měření fyzikálních veličin z výkonové části servopohonu využito rozhraní RT-DAC/USB2 (Obrázek 3) od společnosti Inteco. Toto zařízení slouží jako datové a řídicí rozhraní mezi osobním počítačem a výkonovým modulem a umožňuje realizaci řízení v reálném čase v prostředí MATLAB/Simulink.

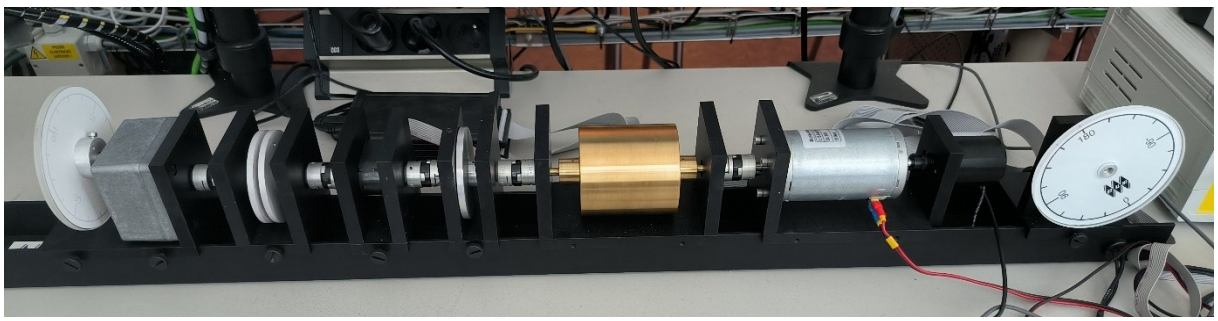
RT-DAC/USB2 je vybaveno FPGA obvodem, který zajišťuje vysokorychlostní obsluhu analogových a digitálních I/O signálů. Mezi jeho klíčové funkce patří:

- generování PWM signálu pro řízení otáček motoru,
- čtení analogových signálů z tachogenerátoru a potenciometru,

- hardwareové čítače pro práci s inkrementálními enkodéry,
- rozhraní pro Real-Time Windows Target, které umožňuje přímé propojení se Simulinkem bez nutnosti použití externího PLC.

Rozhraní komunikuje s výkonnou částí systému prostřednictvím více žilového páskového kabelu „Analog I/O“, který je vyveden ze zadní strany zařízení, který odpovídá specifikaci uvedené v dokumentaci rozhraní výkonové části (Inteco, 2024a, Interní dokumentace dostupná na pracovišti). Napájení RT-DAC je zajištěno externím 12 V DC adaptérem, na čelním panelu se nachází napájecí konektor, signální LED a přepínač napájení. Celkové mechanické uspořádání systému, včetně motoru, setrvačníku a fyzického propojení, je znázorněno na obrázku (Obrázek 4).

Obrázek 3 - Čelní pohled na RT-DAC/USB2



Obrázek 4 - Celkový pohled na propojenou sestavu s motorem a zátěží

1.2.6 Porovnání dat z MATLAB/Simulinku a TIA Portal

Jedním z cílů laboratorní práce bylo porovnání výstupních dat z různých zdrojů řízení – konkrétně:

- MATLAB/Simulink + RT-DAC/USB2,
- TIA Portal + PLC Siemens S7-1200.

V případě MATLAB/Simulink probíhala komunikace přímo s RT-DAC/USB2, přičemž byla nastavena vzorkovací frekvence 10 Hz, což odpovídá periodě 0,1 s. Vzorkování a řízení bylo realizováno prostřednictvím připravených Simulinkových bloků z Modular Servo Toolboxu, které byly dodány spolu s laboratorní sestavou (Inteco, 2024b, Interní dokumentace dostupná na pracovišti).

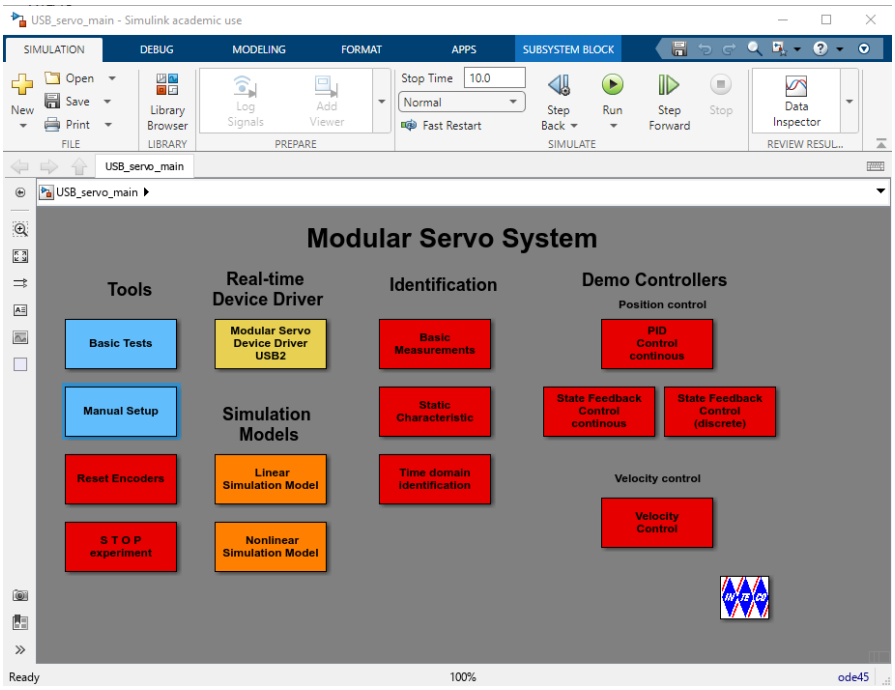
Naopak v systému TIA Portal byla data získána pomocí nástroje Trace, který zachytával průběh signálů z PLC řízeného v cyklickém přerušení s periodou 100 ms. Výsledky byly exportovány do CSV souboru a dále analyzovány.

Cílem porovnání bylo:

- ověřit kvalitu odezvy systému při řízení z různých prostředí,
- posoudit dopad vzorkovací frekvence a typu řízení na kvalitu dat,
- připravit podklad pro identifikaci přenosové funkce, která byla následně provedena a popsána v další části práce.

2 PRAKTICKÁ ČÁST

Jedním z cílů praktické části bylo vypracování studentských úloh pro výuku v laboratoři průmyslové automatizace. Studenti v nich budou využívat HW prostředky učebny, konkrétně PLC automaty Siemens Simatic S7-1200, Simatic WinCC Unified PC System a výukový model motoru „Modular Servo System“ od firmy Inteco. Naučí se základní postupy pro práci v prostředí Siemens TIA Portal a pro programování PLC automatů. Studenti se nadále seznámí i s ovládáním výukového modelu motoru pomocí úloh v prostředí Matlab/Simulink dodaných výrobcem modelu (Obrázek 5).



Obrázek 5 - Main Panel pro řízení Servo motoru v Matlabu

Dalším cílem bylo porovnání získaných výsledků z vypracovaných úloh v TIA Portalu s výsledky získanými pomocí vypracovaných úloh od výrobce Inteco v prostředí Matlab/Simulink. Proto bylo potřeba zvolit takové úlohy pro vypracování, které budou vhodné pro následnou analýzu a zisk dat a zároveň vhodné pro vysvětlení základních principů programování PLC automatů.

Proto jsem zvolil následující úlohy: Manual Setup, Control Impulse Response, Static Characteristic.

Úloha Manual Setup se zabývá základním vysvětlením pojmů ovládání servo motoru v prostředí TIA Portal, jako jsou: řízení otáček motoru, zpracování dat ze vstupů pro čtení rychlosti otáčení, změna směru otáčení, ošetření změny směru otáčení. Dále zde probíhá prvotní seznámení se s HMI Panelem, pomocí kterého je celé ovládání zprostředkováno.

V úloze Control Impulse Response se již využívá programový základ z úlohy Manual Setup, úloha se snaží napodobit průběh získaný z Matlabu. V TIA Portalu bylo cílem zpracovat program, který poběží automaticky po jeho spuštění, bez zásahu uživatele, provede sekvenci přednastavených úkonů a následně se zastaví. Celý průběh pohybu motoru v čase se bude vykreslovat do grafu na HMI Panelu. Úloha se dále zabývá ziskem dat z TIA Portalu a Matlabu pro následné porovnání.

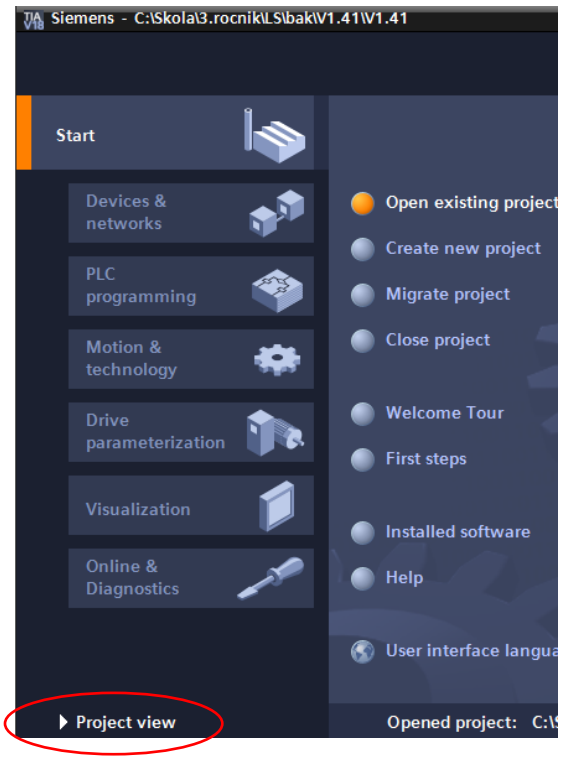
V úloze Static Characteristic se jako u předchozí úlohy vychází z předchozího programového základu. Cílem úlohy je získat data pro vykreslení statické charakteristiky motoru, a to v několika režimech: Ascending, Descending, Reversed. Uživatel před měřením zadá, v jakém rozsahu hodnot chce měřit a kolik chce mít měřících bodů, poté zvolí, v jakém režimu bude měření probíhat a následně jej spustí. Program následně ukládá data do určeného Data Bloku, odkud jsou následně přístupná pro další zpracování.

2.1 Vytvoření projektu, přidání zařízení

Za předpokladu, že student se s vývojovým prostředím TIA Portal setkává poprvé, je zapotřebí v rámci přípravy úloh zmínit i samotné zakládání projektu a následnou správnou konfiguraci používaných zařízení.

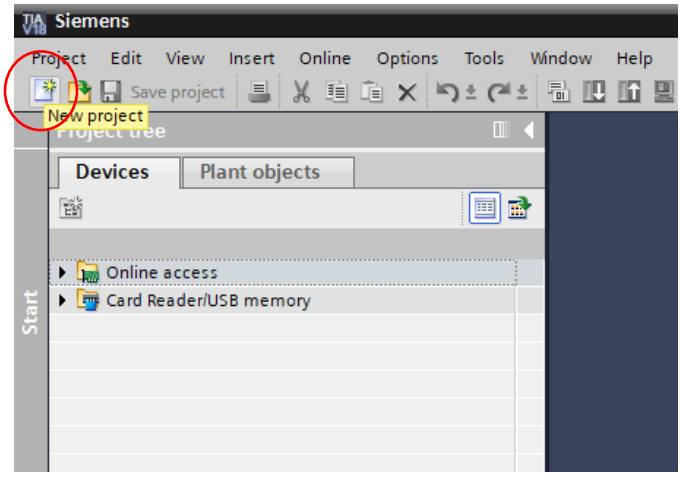
2.1.1 Vložení PLC a základní konfigurace

Po otevření TIA Portalu V18, verze dostupná na školních zařízeních, klikneme v levém dolním rohu na „Project View“ (Obrázek 6).



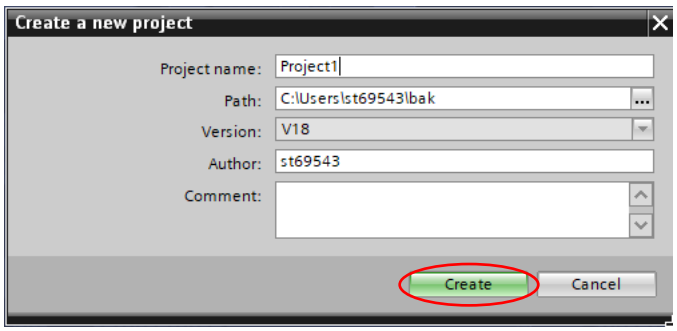
Obrázek 6 – Portal View po spuštění TIA Portalu

Po přepnutí náhledu do režimu „Project View“ stiskneme v levém horním rohu ikonu pro vytvoření nového projektu (Obrázek 7). Alternativně můžeme rozkliknout záložku „Project“ a zde zvolit možnost „New…“



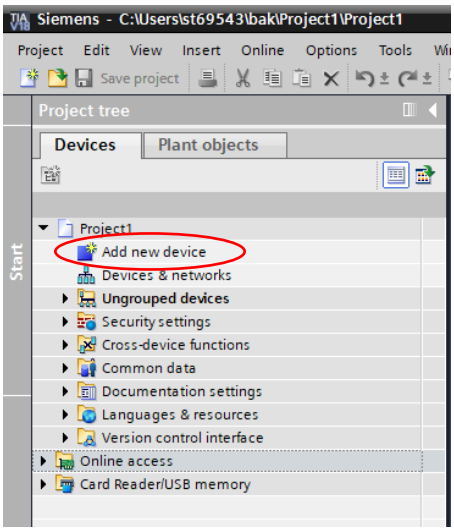
Obrázek 7 – Ikona pro vytvoření projektu v TIA Portalu V18

Otevře se nám okno (Obrázek 8), kde si projekt pojmenujeme, zvolíme jeho místo uložení, popř. můžeme změnit jméno autora nebo přidat komentář. Po vyplnění údajů stiskneme tlačítko „Create“.



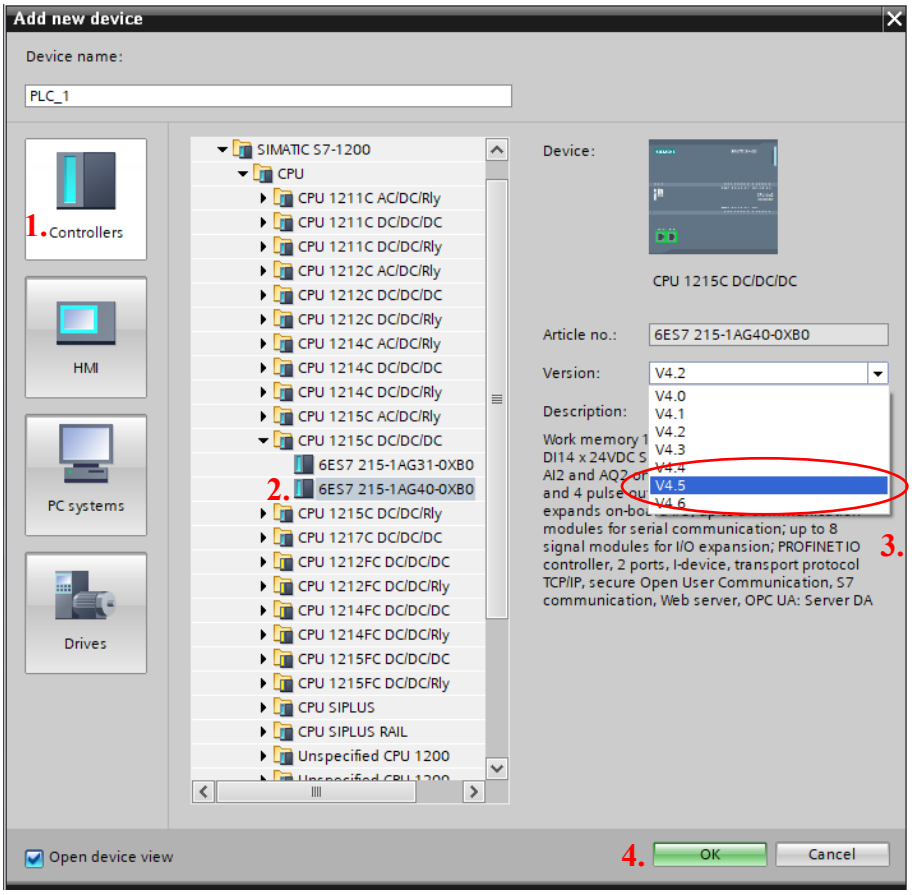
Obrázek 8 - Vytvoření samotného projektu

V dalším kroku, již máme vytvořený projekt, na levé straně obrazovky vidíme náš „Project tree“, neboli stromovou strukturu projektu, kde jsou viditelné hlavní konfigurační možnosti projektu, jako je přidání zařízení, síťové nastavení a bezpečnostní konfigurace. Pro přidání nového zařízení klikneme na první odrážku „Add new device“ (Obrázek 9).



Obrázek 9 - Project tree po vytvoření projektu

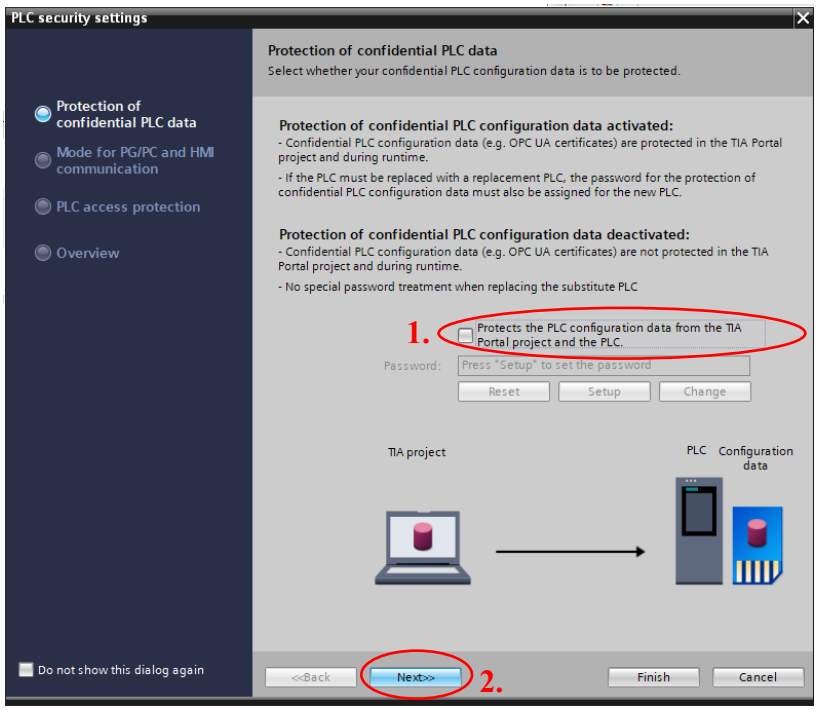
Otevře se nám okno s možností volby nového zařízení, zde zvolíme námi používané PLC podle obrázku níže (Obrázek 10). Po zvolení parametrů PLC klikneme na tlačítko „OK“. Tyto údaje je možné odečíst z reálného HW (Obrázek 11).



Obrázek 10 - Okno pro vložení nového zařízení

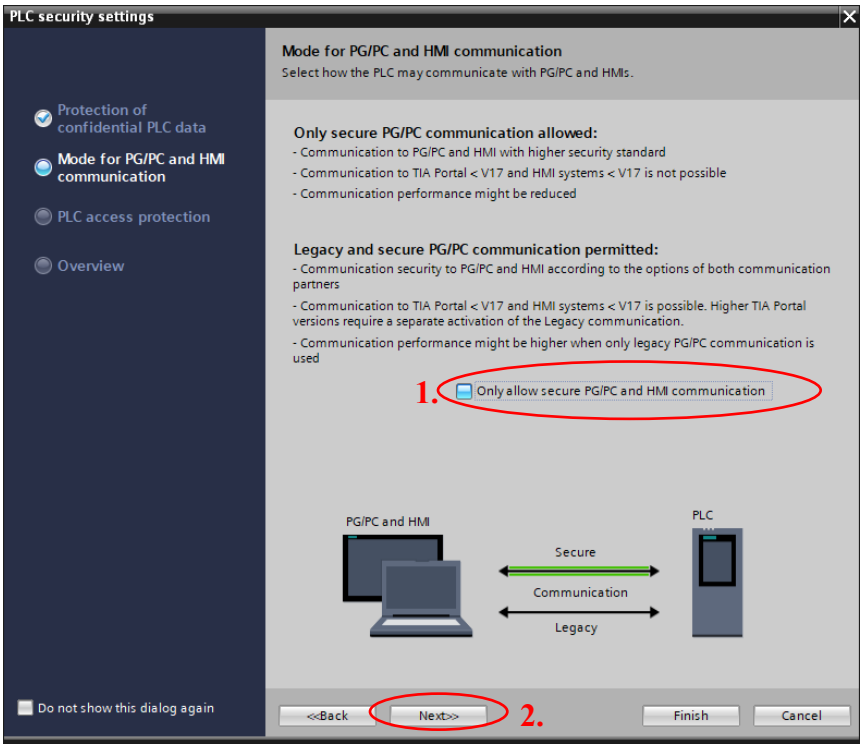
Obrázek 11 - S7-1200 na pracovišti č.8, učebna PL404

V nastavení zabezpečení PLC jsme zvolili deaktivaci ochrany důvěrných konfiguračních dat (Obrázek 12). Tím pádem nebudou například OPC UA certifikáty chráněny heslem v projektu TIA Portal ani při běhu programu. Jedná o zjednodušení následné práce v rámci konfigurace PLC automatu.



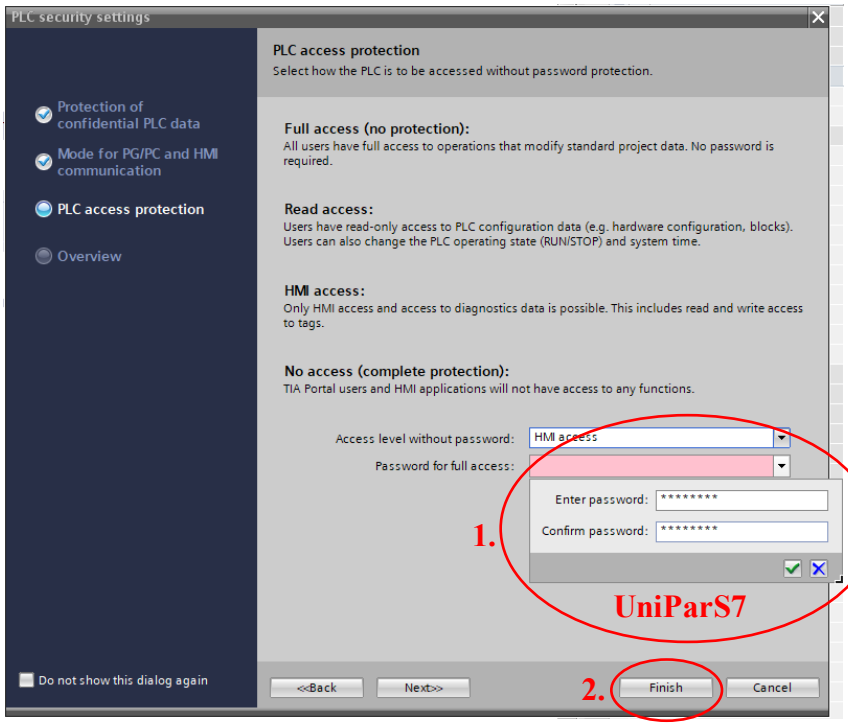
Obrázek 12 - 1. Konfigurace PLC

V tomto kroku ponecháme možnost jak bezpečné, tak i starší (legacy) komunikace (Obrázek 13), přestože používáme TIA Portal verze V18. Tento režim zjednodušuje časté nahrávání do PLC a opakované připojování, protože není nutné opakovaně navazovat plně zabezpečené spojení. Volba je tedy vhodná pro aplikace, kde je kladen důraz na plynulost a rychlost práce.



Obrázek 13 - 2. Konfigurace PLC

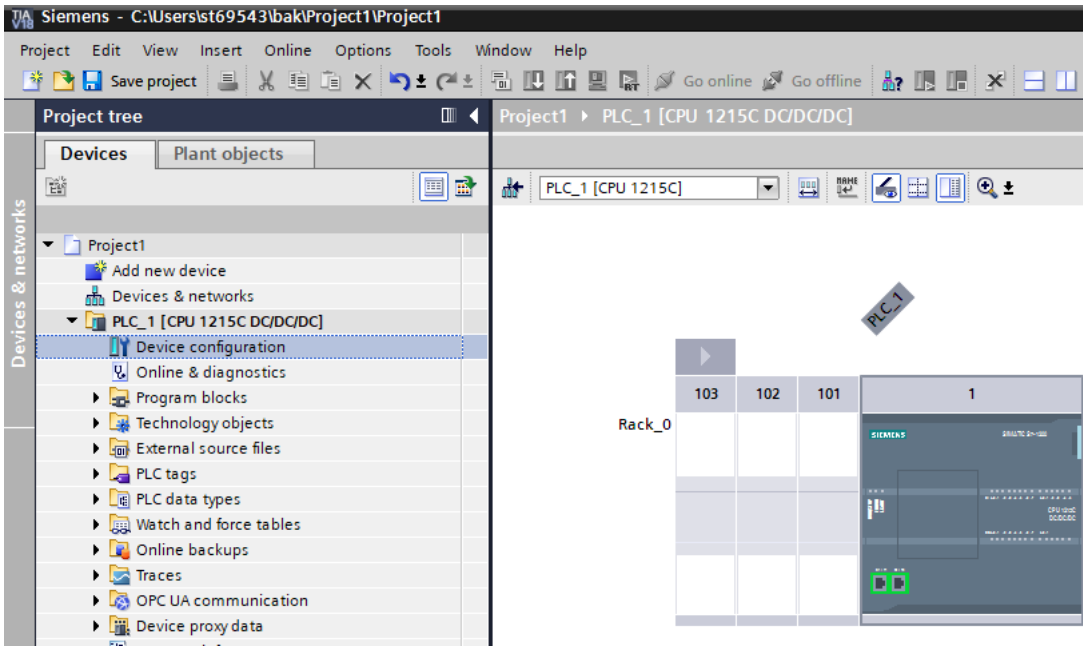
Dále nastavíme přístup k PLC tak, že bez zadání hesla je povolen přístup mezi HMI panelem a PC (Obrázek 14). Pro plný přístup k PLC jsme zvolili ochranu heslem, aby bylo možné řídit, kdo může upravovat konfiguraci a nahrávat programy. Nastavené heslo je: „UniParS7“. Tímto jsme dokončili konfiguraci PLC a můžeme kliknout na „Finish“ (Obrázek 14).



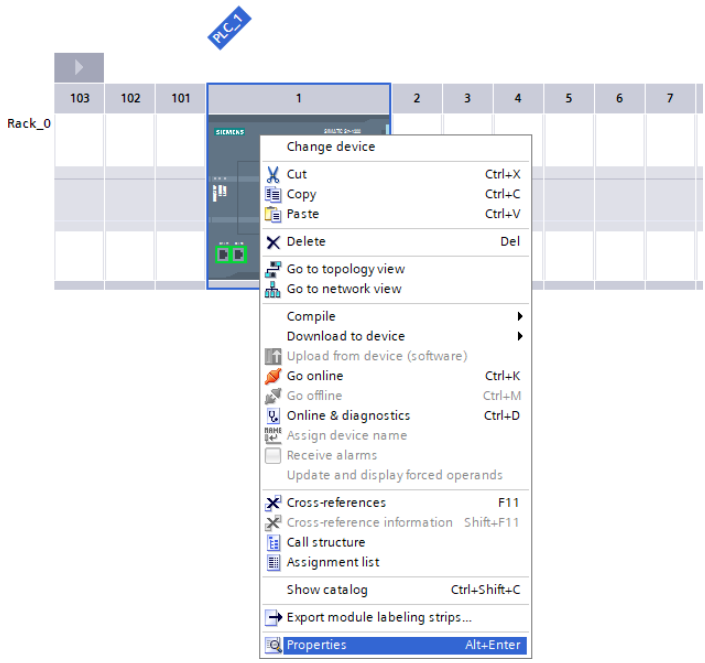
Obrázek 14 - 3. Konfigurace PLC

2.1.2 Podrobné nastavení konfigurace PLC

Nyní již vidíme naše PLC v projektovém stromě, musíme však ještě nastavit podrobnou konfiguraci. Proto v projektovém stromě klikneme u PLC na „Device Configuration“ (Obrázek 15), poté klikneme pravým tlačítkem myši na ikonu PLC a zvolíme kolonku „Properties“ (Obrázek 16).

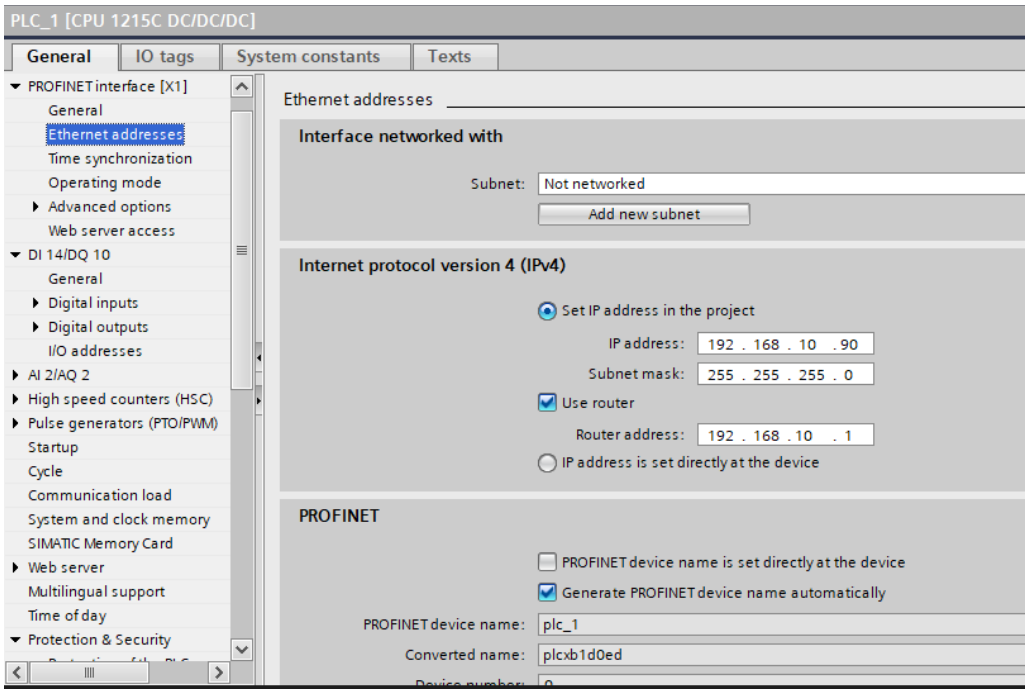


Obrázek 15 – PLC Device Configuration

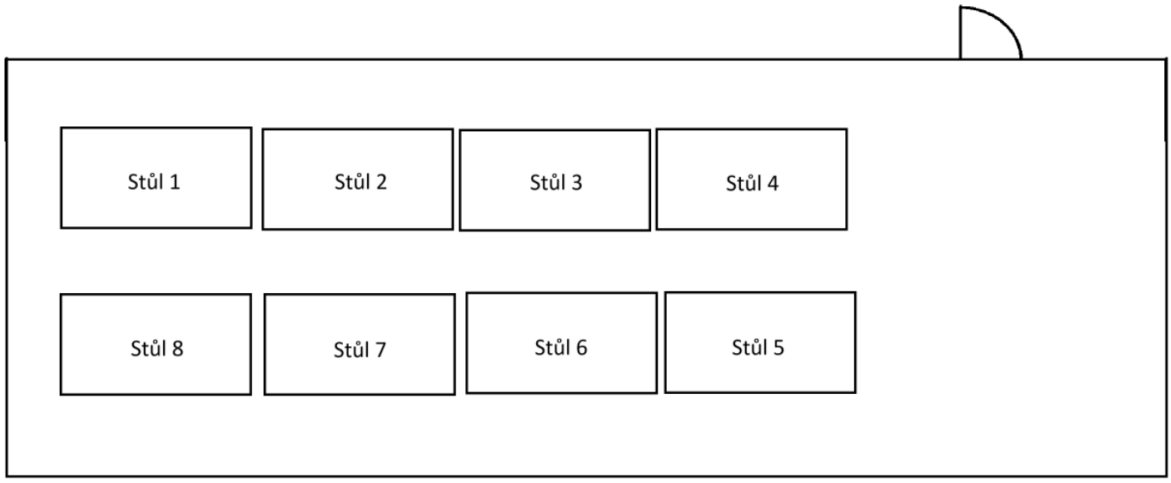


Obrázek 16 – PLC Device Configuration – Properties

V této části projektu nastavujeme pevnou IP adresu PLC, dosadíme zde požadované číslo, podle čísla pracoviště v učebně PL404 (Obrázek 18), za „xx“ do 192.168.10.xx (Stůl 1 PLC .20, PC System .21 … Stůl 8 PLC .90, PC System .91), a aktivovali jsme komunikaci přes router, aby zařízení mohlo komunikovat i mimo svou podsíť. PROFINET jméno zařízení bylo ponecháno automaticky generované, což zjednodušuje konfiguraci (Obrázek 17).

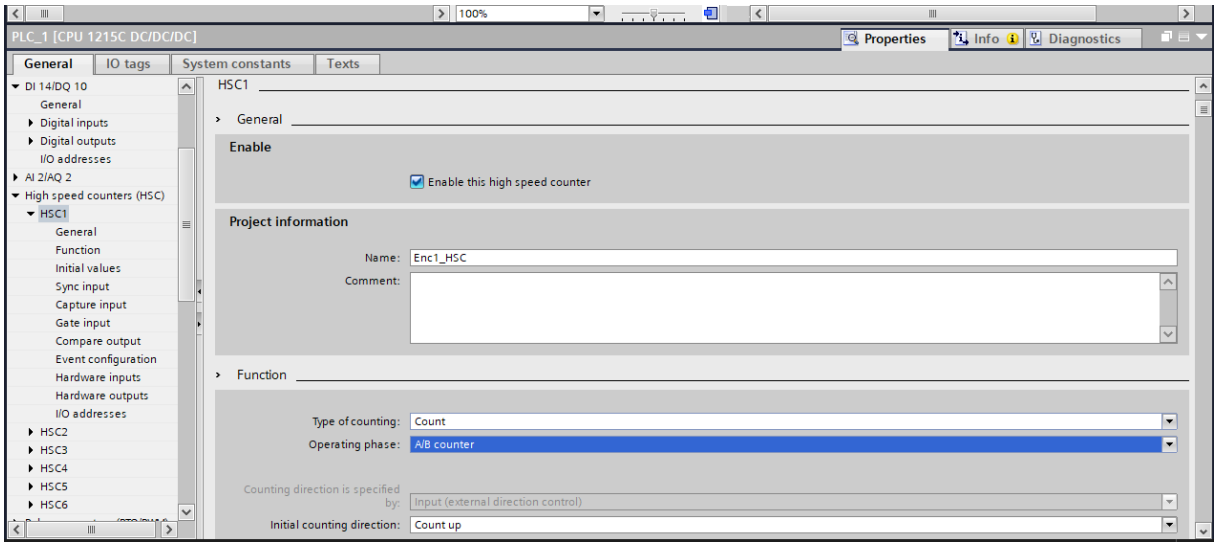


Obrázek 17 - Nastavení ethernetové adresy PLC



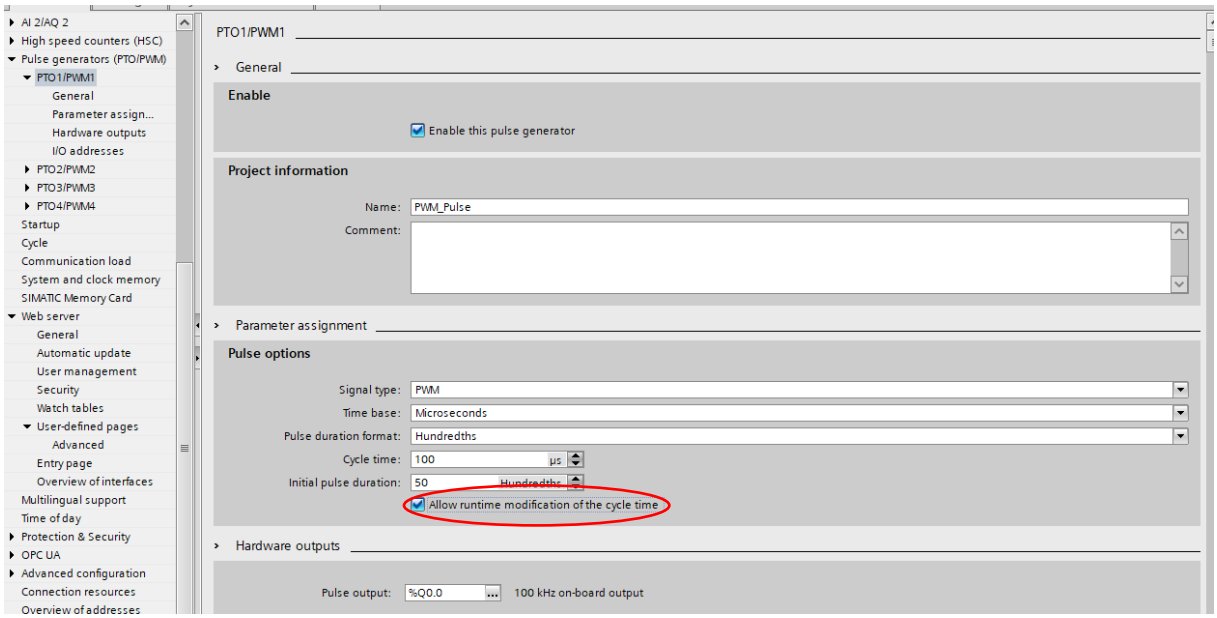
Obrázek 18 - Rozložení učebny PL404

V následujícím kroku jsme aktivovali rychlý čítač HSC1 pro čtení impulsů z kvadraturního enkodéru, který generuje dva fázově posunuté signály A a B (Obrázek 19). Čítač byl nastaven do režimu A/B counter, což umožňuje určování směru otáčení i přesného počtu impulzů. Následně si ještě v kolonce HSC1 -> Hardware inputs zkontrolujeme, že vstupy A a B jsou nastaveny jako I0.0 a I0.1.

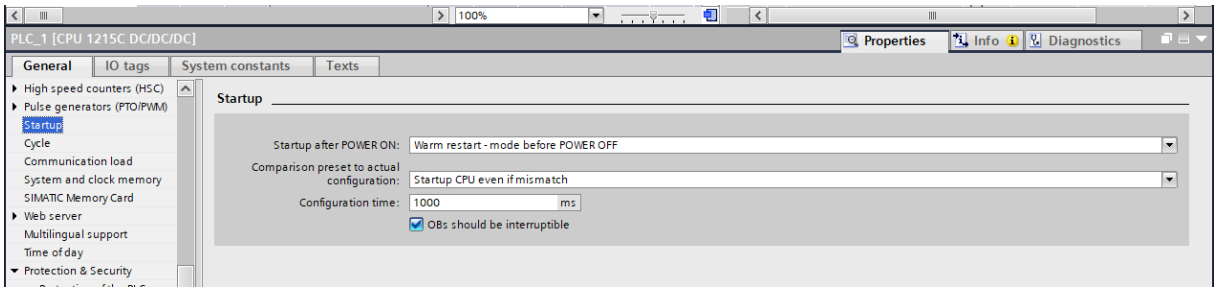


Obrázek 19 - PLC Konfigurace – HSC

Dále nakonfigurujeme PWM generátor s výstupem na Q0.0 pro generování pulzního signálu. Zkontrolujte nastavení cyklu: 100 µs s počáteční střídou 50 %. Zaškrtněte pole „Allow runtime modification of cycle time“. (Obrázek 20) Tím jsme zajistili možnost dynamicky upravovat rychlost motoru. Dále nastavíme PLC do „Warm Restart“ modu (Obrázek 20), díky čemuž si automat uchová stav před vypnutím. Povolením přerušení OB bloků jsme zároveň zajistili možnost rychlé reakce systému na vyšší prioritní úlohy.

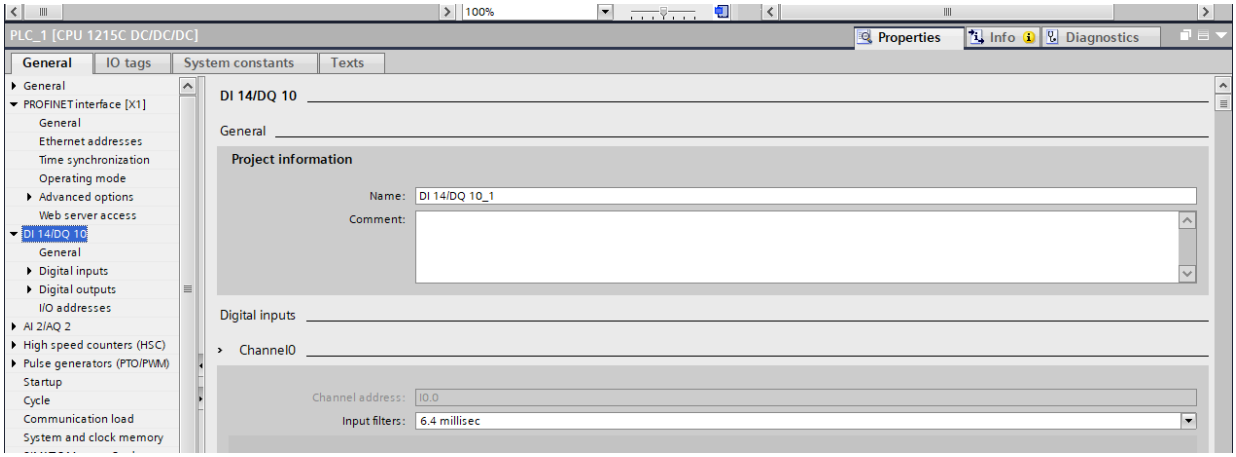


Obrázek 20 - PLC Konfigurace – Povolení PWM

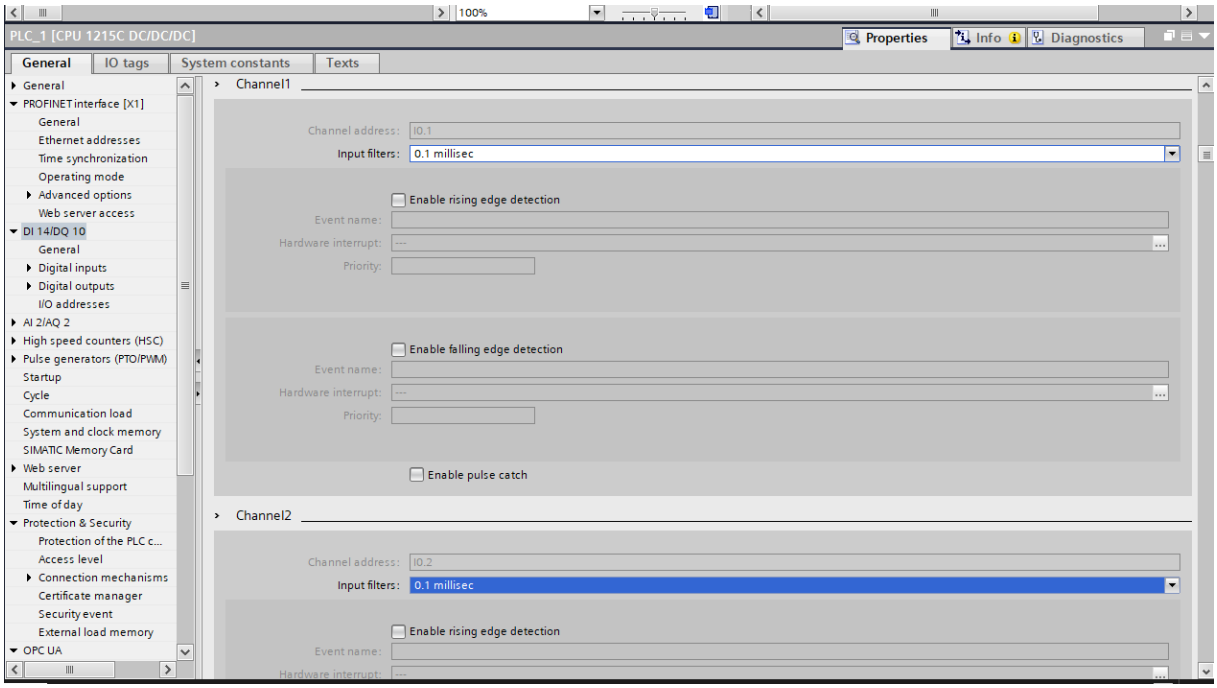


Obrázek 21 - PLC Konfigurace – PWM Startup

U jednotlivých digitálních vstupů, v záložce pro konfiguraci I/O portů, nastavíme různé hodnoty vstupního filtru podle očekávané rychlosti signálů. Pro vstup I0.0 byl použit delší filtr 6,4 ms (Obrázek 22) pro eliminaci nežádoucího rušení, zatímco pro vstupy I0.1 a I0.2 byl nastaven minimální filtr 0,1 ms (Obrázek 23) kvůli zachycení rychlých impulsů (HSC).

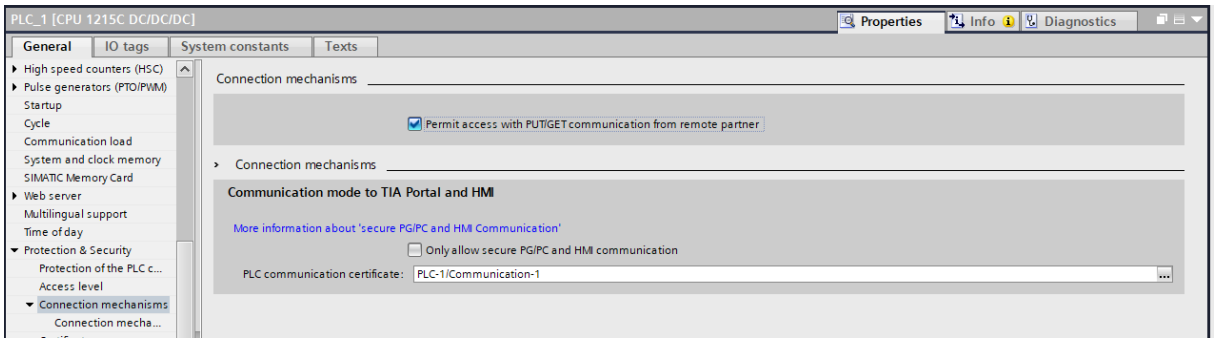


Obrázek 22 - PLC Konfigurace – I/O 1



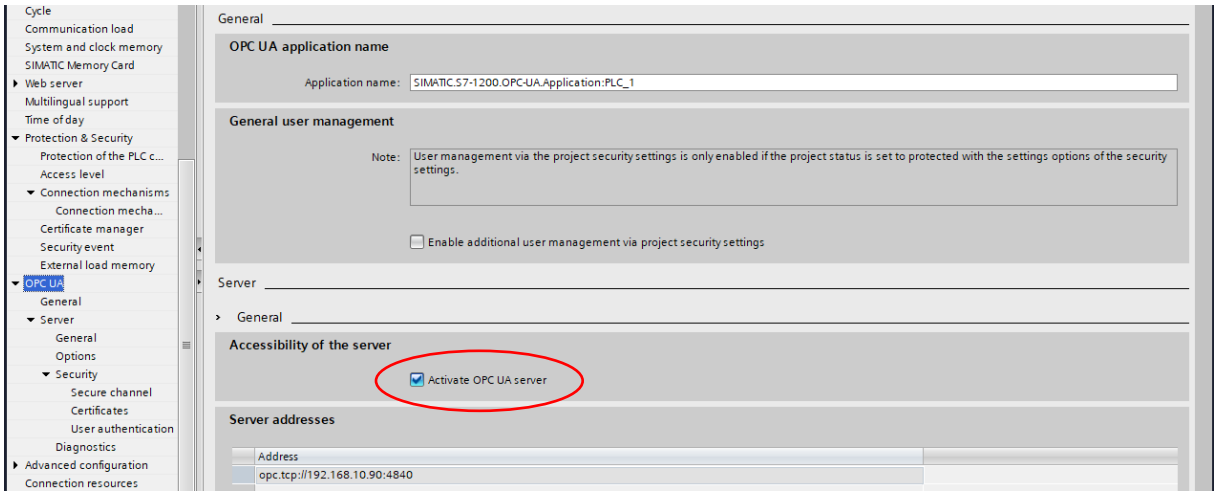
Obrázek 23 - PLC Konfigurace – I/O 2

V tomto kroku povolíme přístup k PLC pomocí komunikačního protokolu PUT/GET (Obrázek 24) z jiných zařízení, což umožňuje výměnu dat mezi PLC a HMI. Neomezujeme komunikaci pouze na zabezpečené kanály, aby byla zachována kompatibilita a jednoduchost. Toto nastavení je vhodné pro laboratorní podmínky, kde je síť pod kontrolou a nevyžaduje se vysoká úroveň zabezpečení.

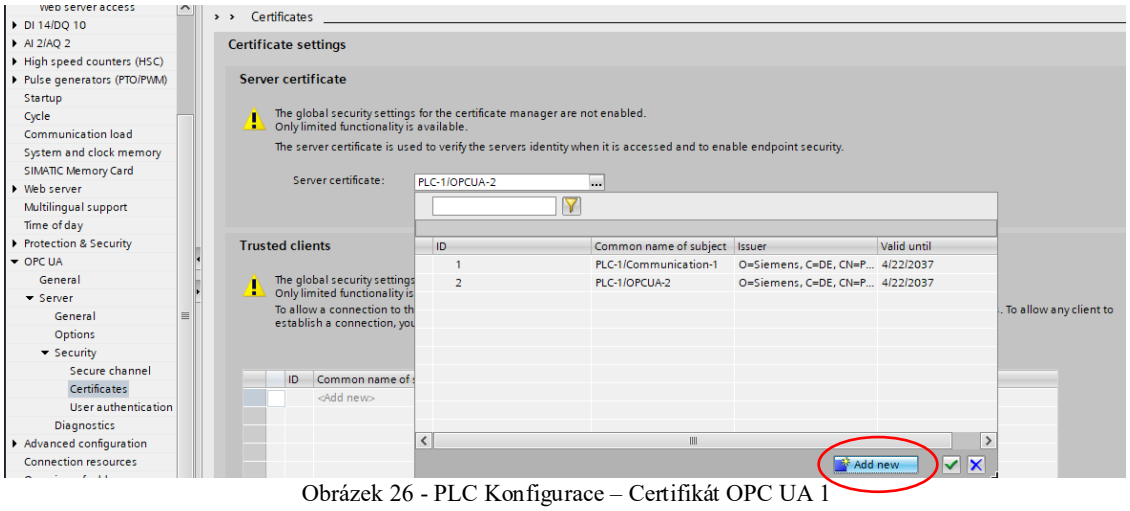


Obrázek 24 - PLC Konfigurace – PUT/GET

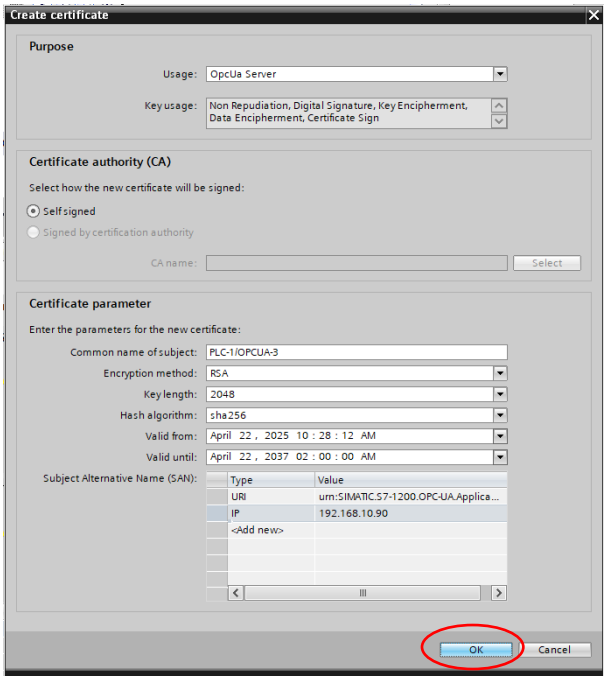
Jako další aktivujeme OPC UA server (Obrázek 25). Server slouží k bezpečné výměně dat mezi PLC a nadřazenými systémy. V našem případě není potřeba aktivovat správu uživatelů ani vyšší úroveň zabezpečení. Musíme si však vytvořit vlastní OPC UA certifikát, stiskneme „Add new“ podle obrázku (Obrázek 26). Certifikát bude použit pro šifrovanou a autentizovanou komunikaci mezi PLC a OPC UA klienty. Certifikát obsahuje název serveru, IP adresu a URI, což zajišťuje správné ověření totožnosti serveru během připojení. Následně jen potvrdíme stisknutím „OK“ (Obrázek 27).



Obrázek 25 - PLC Konfigurace – Aktivace OPC UA

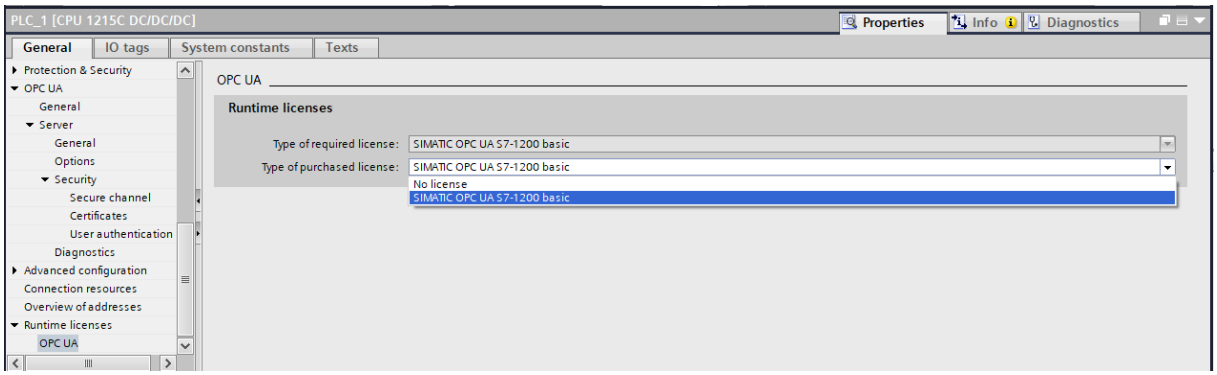


Obrázek 26 - PLC Konfigurace – Certifikát OPC UA 1



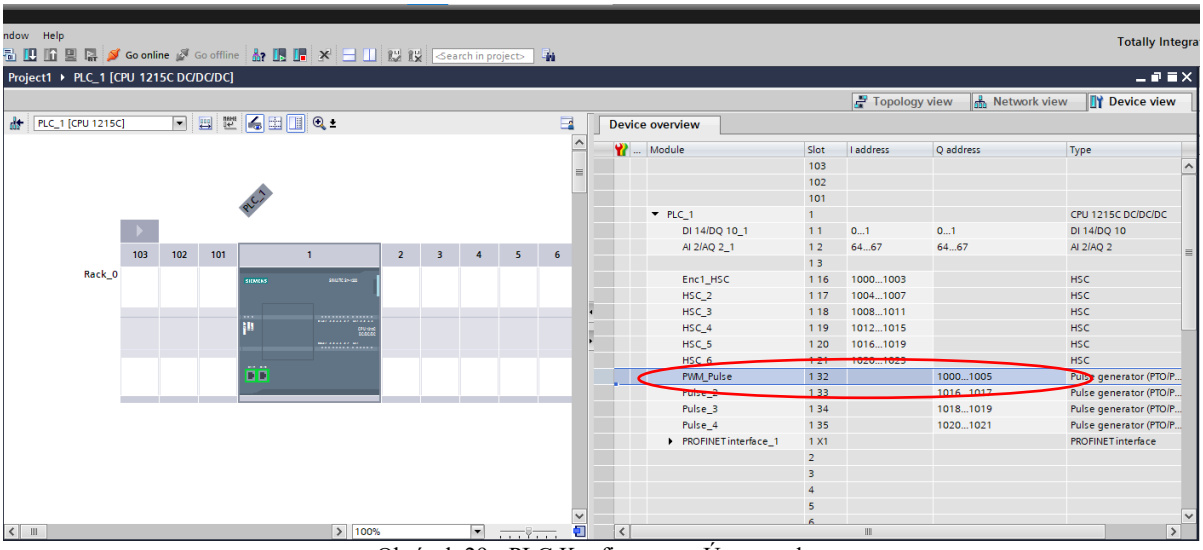
Obrázek 27 - PLC Konfigurace – Certifikát OPC UA 2

Pro správný běh OPC UA serveru na PLC jsme přiřadili runtime licenci „SIMATIC OPC UA S7-1200 basic“. Tato licence umožňuje plnohodnotné využití OPC UA funkcionalit, jako je komunikace s SCADA systémem nebo jinými klienty přes zabezpečený OPC UA protokol. Výběr licence (Obrázek 28) je nutný pro aktivaci serveru v reálném provozu.



Obrázek 28 - PLC Konfigurace – Certifikát OPC UA 3

Ještě se nakonec vrátíme do „Device Configuration“ do záložky „Device view“ (Obrázek 29). U objektu PWM_Pulse musíme ručně upravit výstupní adresy z původního rozsahu 1008...1013 na 1000...1005 a s tím spojené i další výstupní adresy, protože při ponechání původní adresace docházelo ke konfliktu, přepisování hodnot, a nesprávné funkci. Po této úpravě bylo chování výstupu stabilní a PWM generátor fungoval správně.

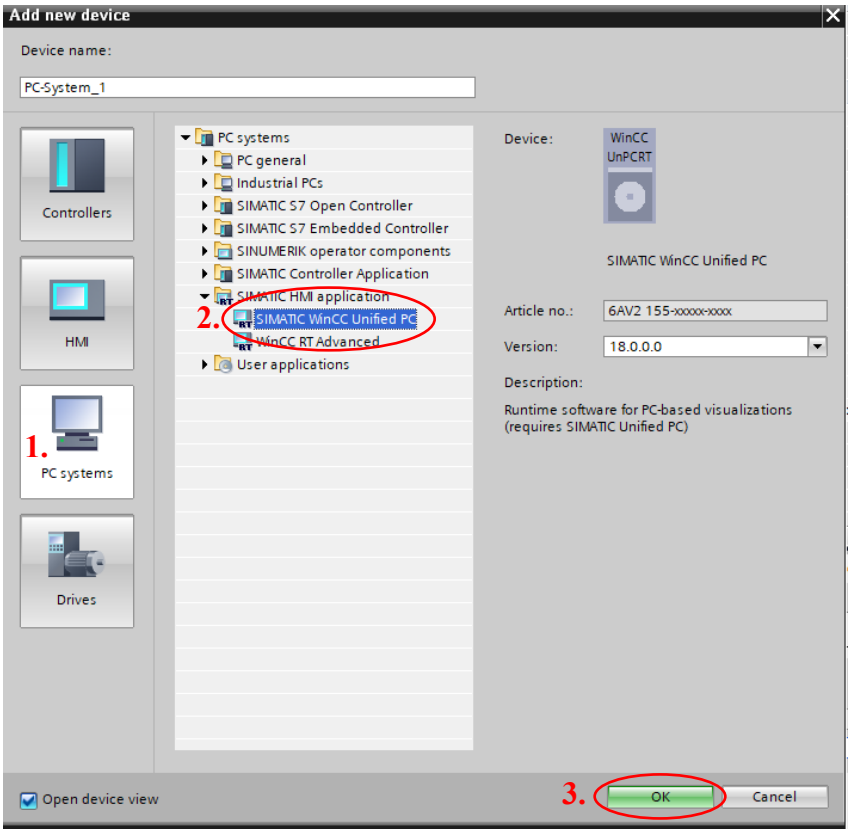


Obrázek 29 - PLC Konfigurace – Úprava adres

Tímto jsme dokončili konfiguraci PLC.

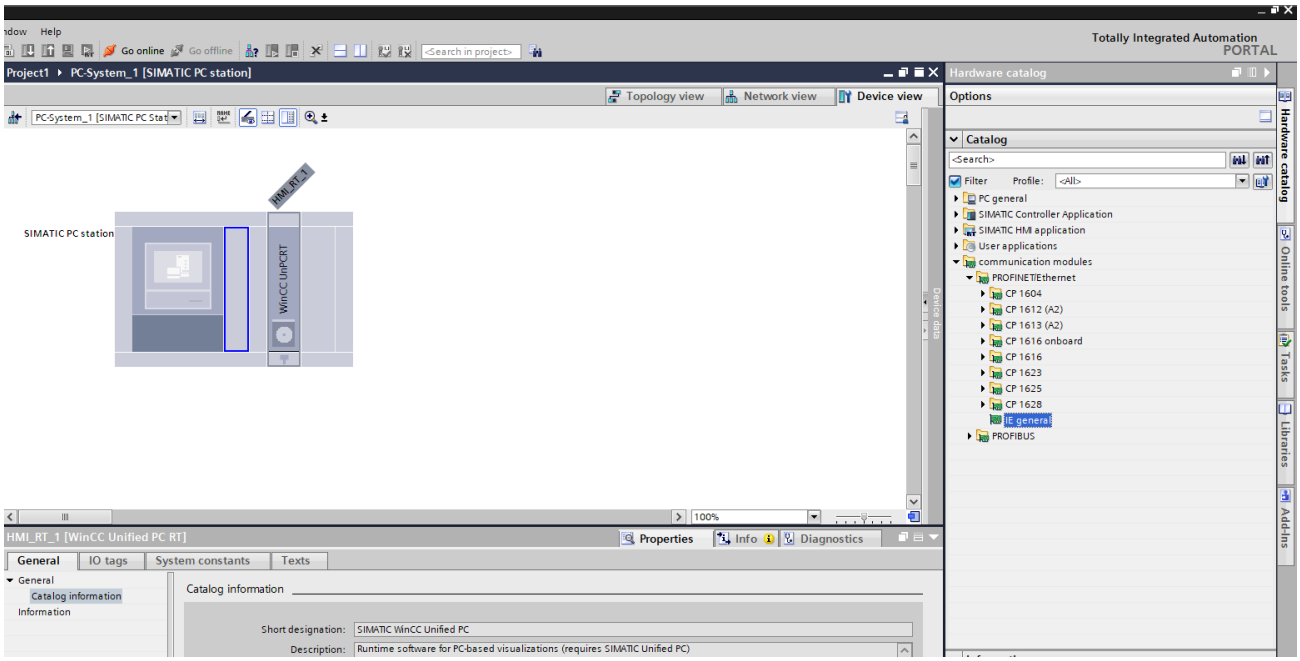
2.1.3 Vložení PC Systému (IPC)

Budeme pokračovat vložením našeho PC Systému. Otevřeme okno pro přidání nového zařízení (Obrázek 9), zde zvolíme PC Systém podle obrázku níže (Obrázek 30).



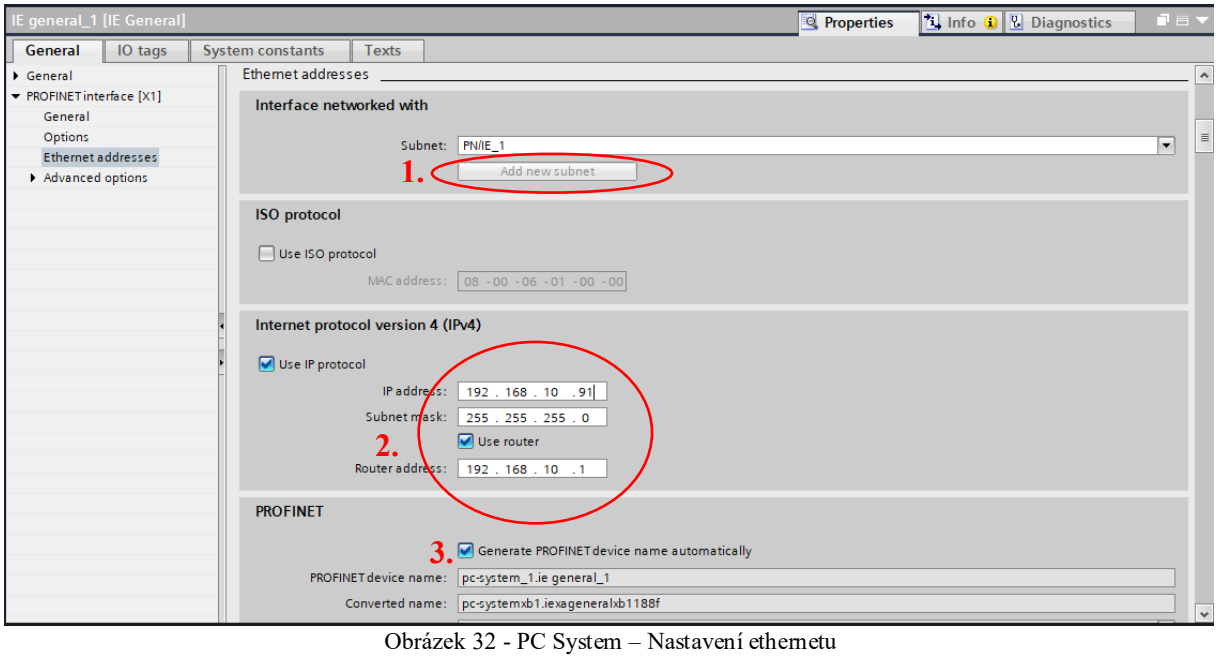
Obrázek 30 - Okno pro vložení nového zařízení – PC System

Po vložení PC Systému si opět otevřeme jeho „Device Configuration“, obdobně jako u PLC (Obrázek 15). Na pravé straně (Obrázek 31) si otevřeme pole „Hardware catalog“ a v záložce „communication modules“ zvolíme „PROFINETInternet“. Podle obrázku vložíme, pomocí „drag and drop“ do modrého pole nebo dvojklikem, síťové rozhraní „IE General“, které slouží k připojení průmyslového počítače (IPC) do sítě pro vizualizaci a komunikaci s PLC. Toto rozhraní je klíčové pro správné fungování SCADA systému, jako je WinCC Unified, protože zajišťuje výměnu dat mezi vizualizací a řídicím systémem.



Obrázek 31 - PC System - Device Configuration

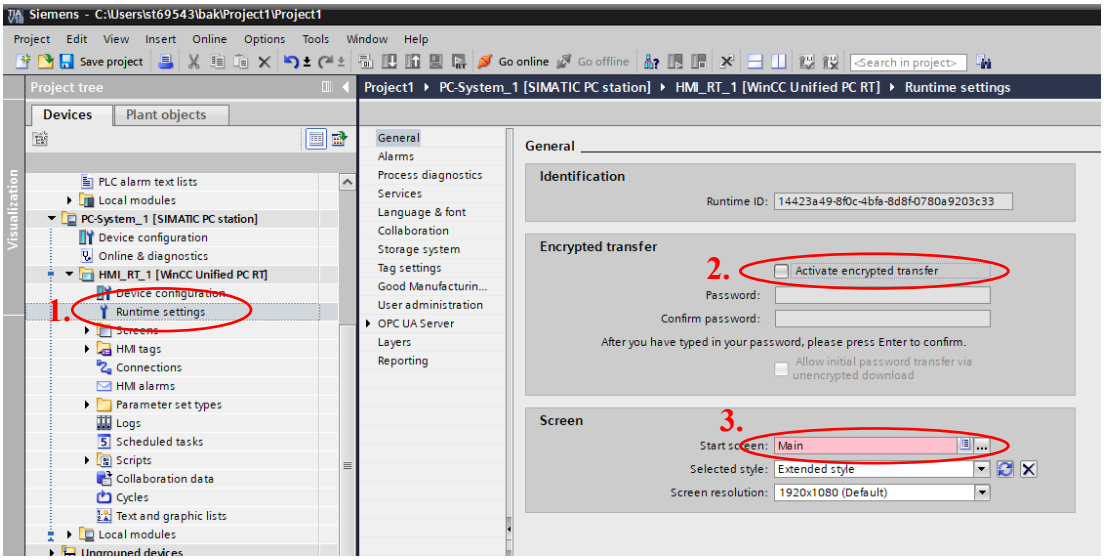
Dále nakonfigurujeme ethernetové rozhraní „IE General“. V „Device Configuration“ klikneme pravým tlačítkem myši na „IE General“ a zvolíme možnost „Properties“. Podle obrázku postupujeme dál (Obrázek 32). Zařízení přiřadíme do sítě PN/IE_1 kliknutím na pole „Add new subnet“ a nastavíme mu statickou IP adresu 192.168.10.xx, kde dosadíme požadované číslo, podle čísla pracoviště v učebně PL404 (Obrázek 18), za „xx“, s výchozí bránou 192.168.10.1.



Obrázek 32 - PC System – Nastavení ethernetu

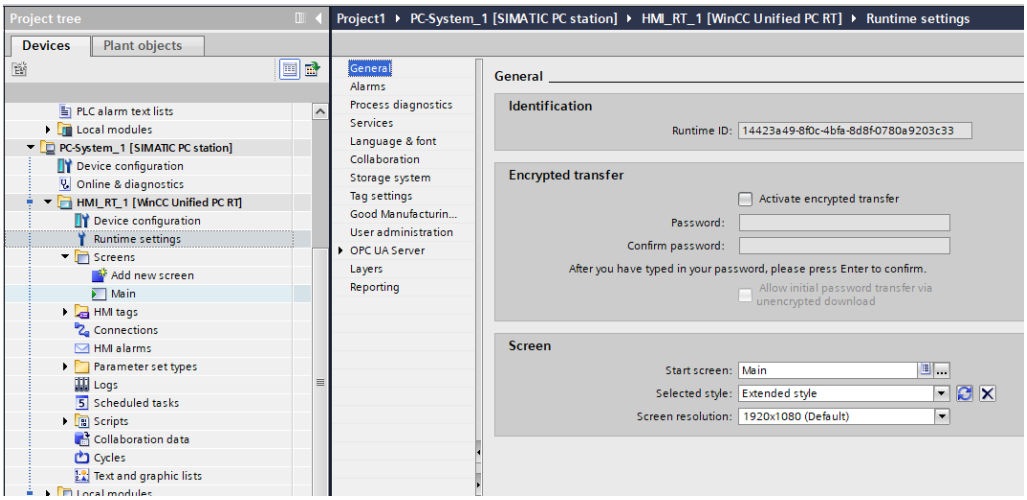
Jako další musíme nastavit parametry běhového prostředí „runtime“. Ten nastavíme kliknutím na „Runtime settings“ v projektovém stromě (Obrázek 33). Pro usnadnění nahrávání programu

do PC Systému vypneme šifrovaný přenos „Encrypted transfer“. Jako spouštěcí obrazovku si nastavíme obrazovku „Main“, na obrázku níže svítí červeně, protože zatím neexistuje.



Obrázek 33 - PC System – Runtime settings

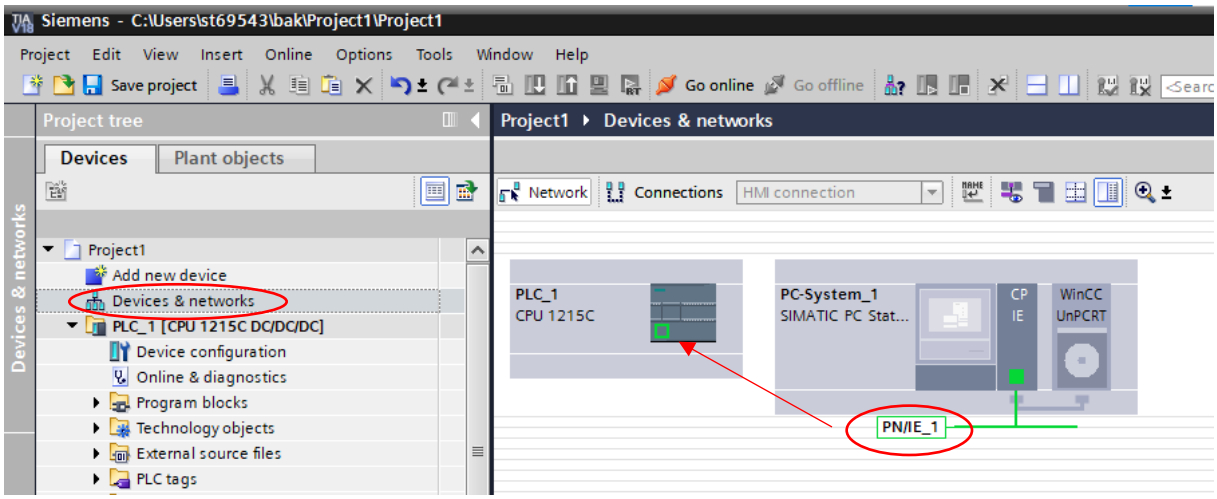
To snadno napravíme tím, že ji vytvoříme. V projektovém stromě, klikneme na záložku „Screens“, poté na „Add new screen“ (Obrázek 34), automaticky se vytvoří obrazovka se jménem „Screen_1“. Pravým kliknutím a zvolením možnosti „Rename“ ji přejmenujeme na „Main“.



Obrázek 34 - PC System - Vytvoření obrazovky

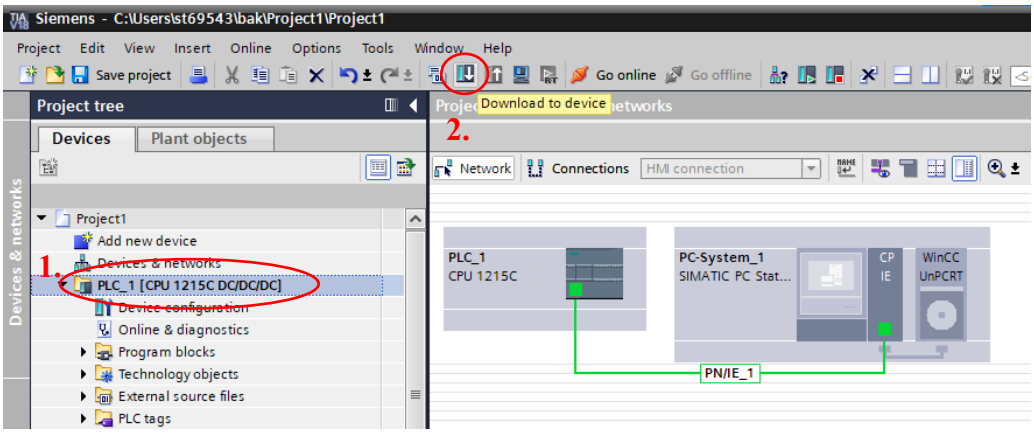
2.1.4 Propojení PLC s PC Systémem (IPC), Zkouška správného nastavení

Dvojklikem na položku „Devices & networks“ se otevře vyobrazení topologie zařízení v projektu (Obrázek 35). Vidíme zde námi vytvořenou podsíť PN/IE_1. Pomocí „Drag and drop“ spojíme obě zařízení. Hotové spojení můžeme vidět na: (Obrázek 36)



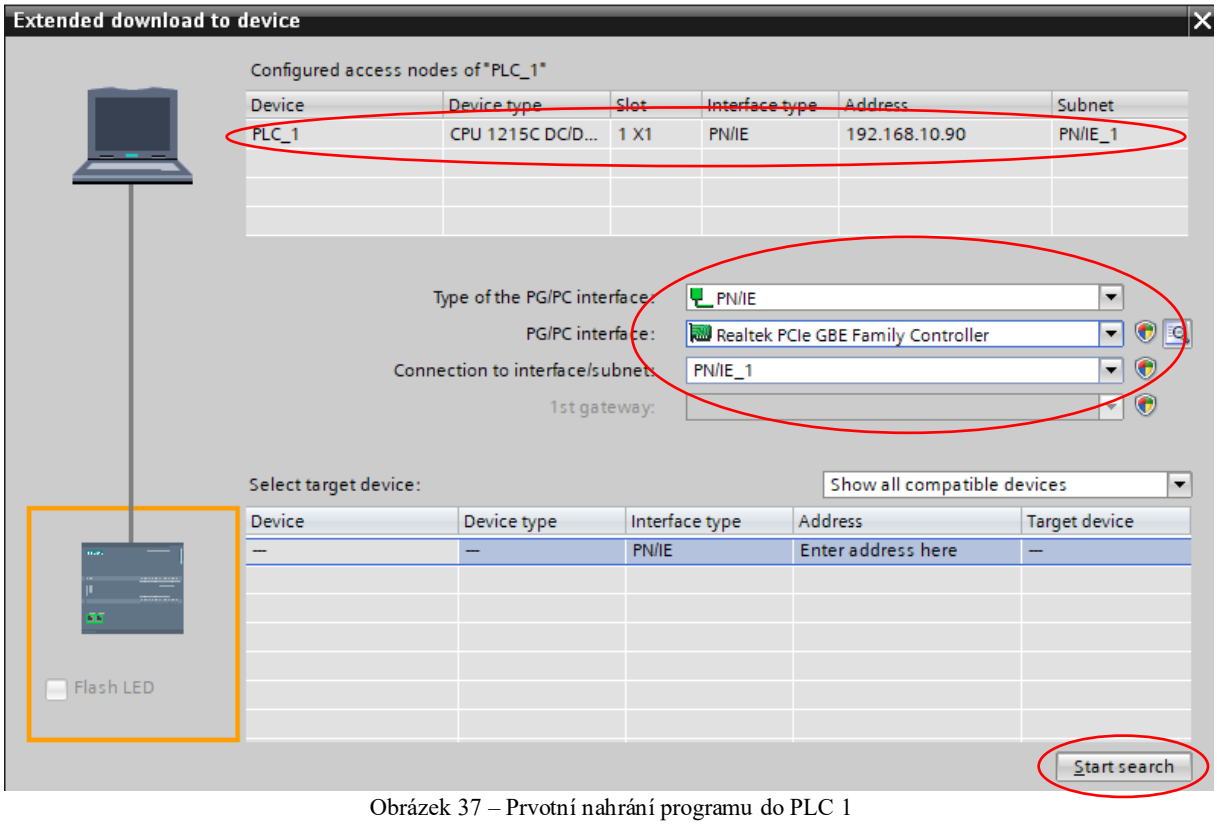
Obrázek 35 - Devices & networks

Jako zkoušku, jestli jsme vše nastavili správně, se zkusíme připojit k reálnému PLC automatu a k PC Systému. Jednou klikneme na složku našeho PLC v projektovém stromě (Obrázek 36), poté klikneme ikonku pro nahrávání programu do PLC.



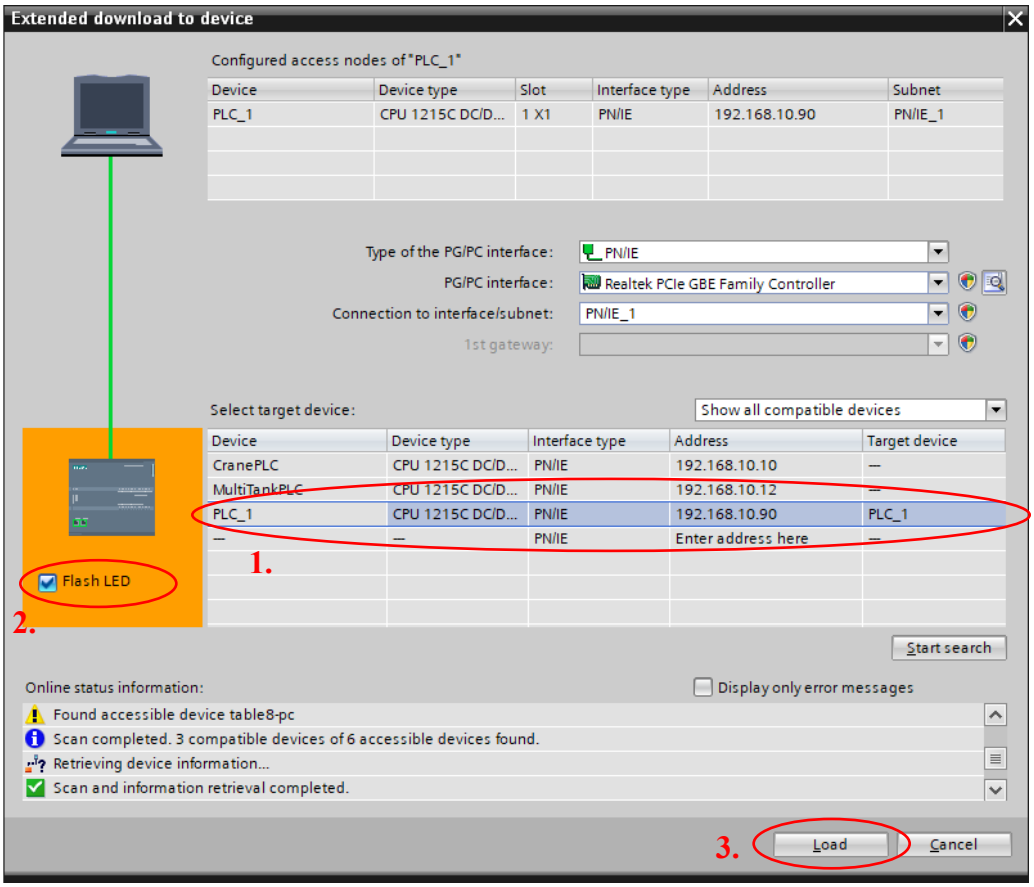
Obrázek 36 - PLC – Nahrávání programu

Zobrazí se nám následující okno (Obrázek 37), kde v horní části můžeme vidět parametry námi nakonfigurovaného PLC. Níže vidíme volbu, kde musíme vybrat pomocí jakého rozhraní bude mezi počítačem a PLC automatem, pomocí jaké síťové karty a jaké podsítě bude probíhat komunikace. Po nastavení rozhraní klikneme na „Start search“.



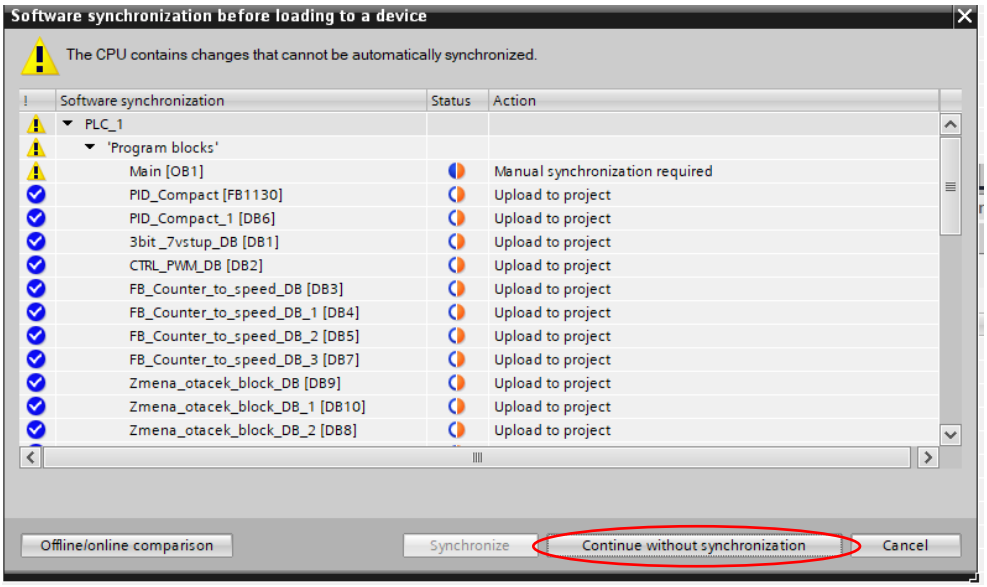
Obrázek 37 – Prvotní nahrání programu do PLC 1

Při správném nastavení rozhraní se nám ve spodní části (Obrázek 38) zobrazí všechna dostupná zařízení v podsíti. Jedním kliknutím zvolíme PLC s naší adresou a v levé části zaškrtneme pole „Flash LED“, pokud jsme zvolili správné zařízení, můžeme pozorovat blikající LED na PLC na našem pracovišti. Když je tato podmínka splněna, klikneme na tlačítko „Load“.



Obrázek 38 - Prvotní nahrání programu do PLC 2

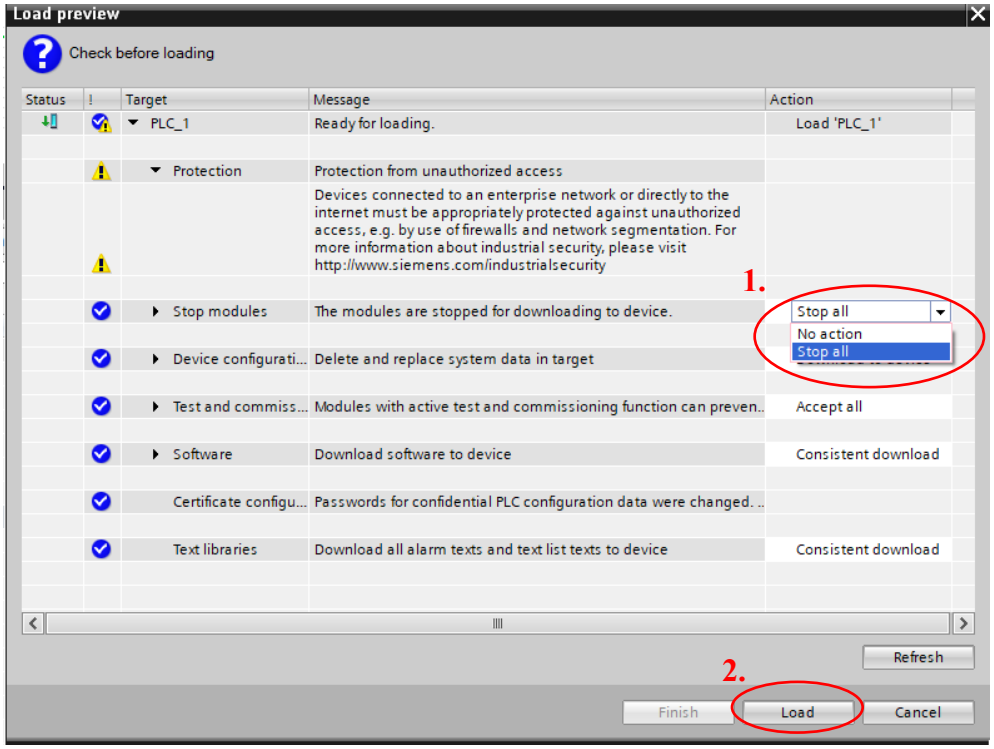
Než se program začne nahrávat, zobrazí se výstražná okna. Na obrázku níže (Obrázek 39) PLC informuje, že má v sobě nahrané jiné programové bloky, než se nyní snažíme nahrát, nyní nestojíme o synchronizaci, proto budeme pokračovat bez ní.



Obrázek 39 - Prvotní nahrání programu do PLC 3

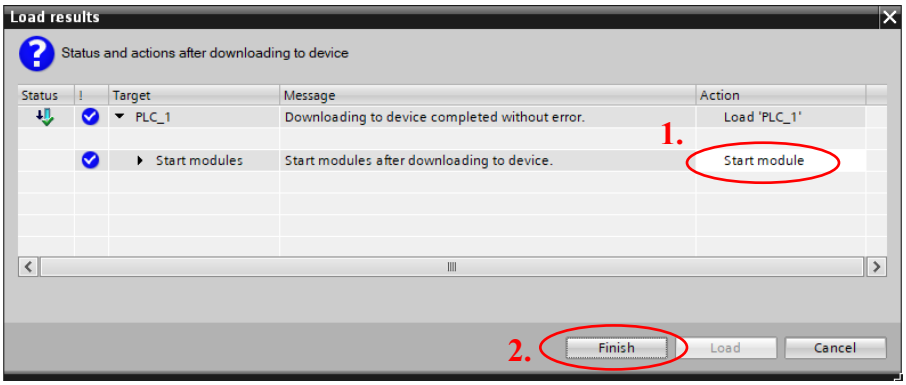
Na dalším snímku (Obrázek 40) přehled před nahráním programu. Zvolili jsem možnost „Stop all“, čímž se během nahrávání zastaví běh PLC, aby nedošlo k chybám během přepisu dat.

Tento postup zajišťuje bezpečné a bezproblémové nahrání projektu do řídicí jednotky. Následně stiskneme tlačítko „Load“.



Obrázek 40 - Prvotní nahrání programu do PLC 4

Po úspěšném nahrání programu opět spustíme chod PLC (Obrázek 41), které bylo po dobu nahrávání ve „Stop“ módu a dokončíme nahrávání.

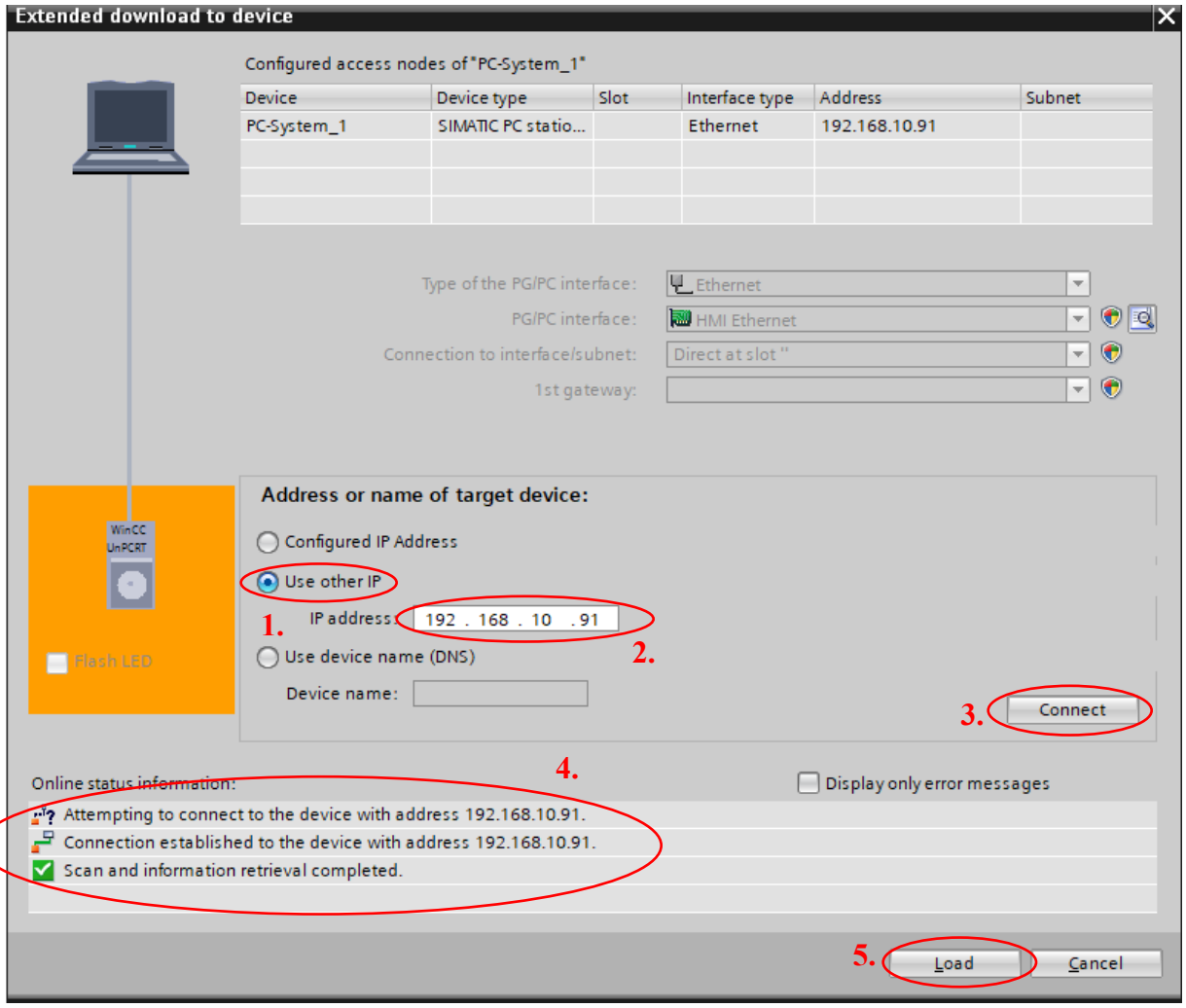


Obrázek 41 - Prvotní nahrání programu do PLC 5

Nahráním prázdného programu jsme dokončili kontrolu správnosti nastavení PLC automatu, nyní otestujeme nastavení PC Systému.

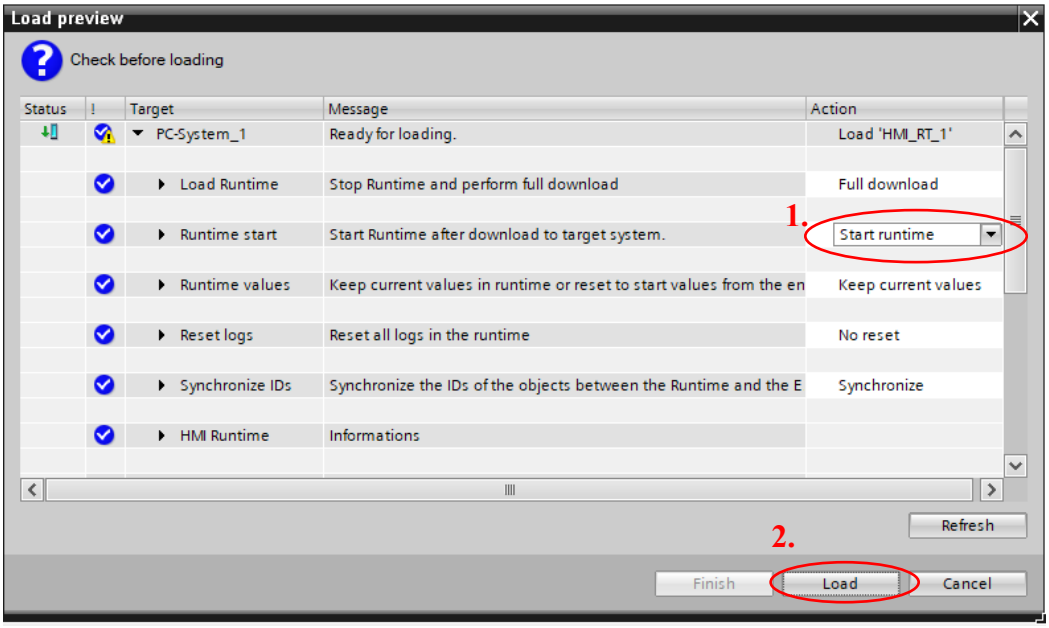
Obdobě jako u PLC (Obrázek 36) jednou klikneme na PC Systém v projektovém stromě a následně na ikonu nahrávání programu do zařízení „Download to device“. Zobrazí se nám obdobné okno jako při nahrávání do PLC (Obrázek 42), v horní části opět vidíme část konfigurace našeho systému. Typ rozhraní tentokrát nevolíme. Ve střední části okna

zaškrtneme možnost „Use other IP“ a ručně vyplníme IP adresu našeho PC Systému (IE General). Následně klikneme na tlačítko „Connect“. Pokud jsme vše zadali správně, ve spodní části okna se objeví potvrzovací zpráva o úspěšném spojení, poté stiskneme „Load“ pro načtení programu.



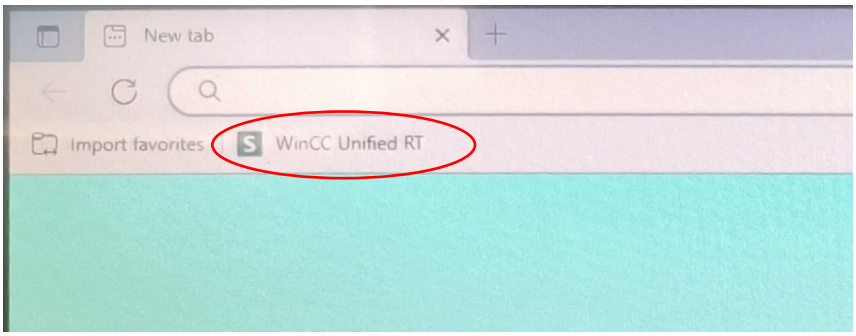
Obrázek 42 - Nahrávání programu do PC Systému 1

Na dalším snímku (Obrázek 43) je před nahrávací obrazovka pro PC Systém. Před nahráním vizualizace do runtime prostředí WinCC Unified jsme zvolili kompletní přenos projektu spolu s automatickým spuštěním vizualizace „Start runtime“ po dokončení nahrávání.



Obrázek 43 - Nahrávání programu do PC Systému 2

Pro zobrazení naši startovací pozice na HMI Panelu na něm musíme spustit internetový prohlížeč a zadat do vyhledávání naši adresu IE General: 192.168.10.x1 nebo kliknout na uložený odkaz (Obrázek 44).

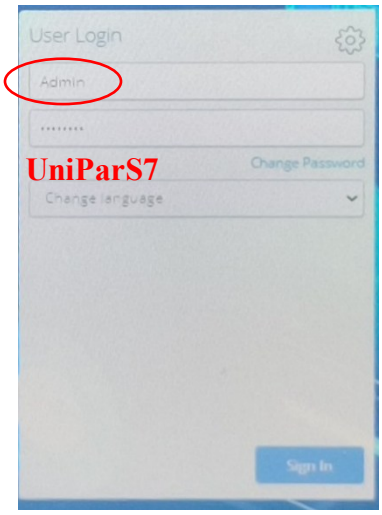


Obrázek 44 - PC System – Spuštění 1

Otevře se nám stránka, kde musíme vyplnit přihlašovací údaje (Obrázek 45).

LOGIN: Admin

HESLO: UniParS7



Obrázek 45 - PC System – Spuštění 2

Po přihlášení se vám zobrazí vaše Startovací obrazovka („Start screen“), kterou jsme dříve konfigurovali (Obrázek 34). Pokud se nezobrazila stačí zopakovat nahrávání do PC Systému.

2.2 Příprava úlohy Manual Setup

Budeme vycházet z úlohy v aplikaci Matlab.

Zobrazené okno „Servo Manual Setup“ (Obrázek 46) slouží k ručnímu testování a monitorování stavu Modular Servo Systemu připojeného přes RT-DAC/USB2 rozhraní. Toto rozhraní je součástí softwarového nástroje od společnosti Inteco a je úzce propojeno s MATLAB/Simulink prostředím.

Okno je rozděleno do několika logických sekcí.

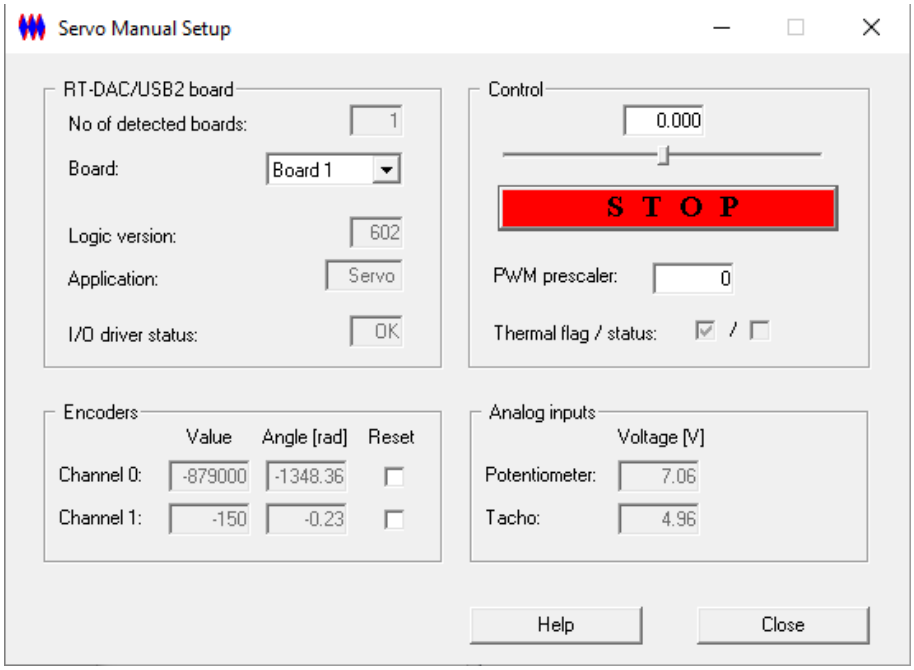
V levé horní části se nachází blok RT-DAC/USB2 board, v němž jsou zobrazovány základní informace o připojené hardwarové desce. Uživatel má k dispozici identifikaci desky, verzi nahrané logiky (FPGA), označení aktivní aplikační konfigurace (v tomto případě „Servo“) a stav ovladače vstupně-výstupního rozhraní, který by měl být ve stavu „OK“, aby byla zajištěna plná funkčnost systému.

V sekci Encoders, umístěné vlevo dole, jsou zobrazovány údaje z inkrementálních snímačů polohy (enkodérů). Kromě aktuální hodnoty čítače impulsů systém rovněž zobrazuje přepočítanou hodnotu úhlu v radiánech. K dispozici je rovněž funkce resetu, která umožňuje vynulovat čítač a tím nastavit výchozí referenční pozici pro měření.

V pravé horní části se nachází ovládací panel Control, který umožňuje nastavování řídicího signálu v rozsahu od −1.0 do 1.0. Tento signál je interně škálován na výstupní napětí pro motor. Dominantním prvkem je tlačítko STOP, jehož aktivací dojde k okamžitému zastavení řízení. Součástí bloku je také pole pro nastavení děliče PWM (tzv. prescaler) a indikátory teplotního

stavu výkonového zesilovače. V případě detekce přehřátí dojde k automatické deaktivaci výstupního PWM signálu.

Sekce Analog Inputs, umístěná v pravé dolní části, zobrazuje hodnoty analogových napětí měřených na vstupu z referenčního potenciometru a z tachogenerátoru. Tyto hodnoty slouží k odhadu aktuální polohy a rychlosti otáčení motoru a představují doplňkový způsob měření vedle enkodéru

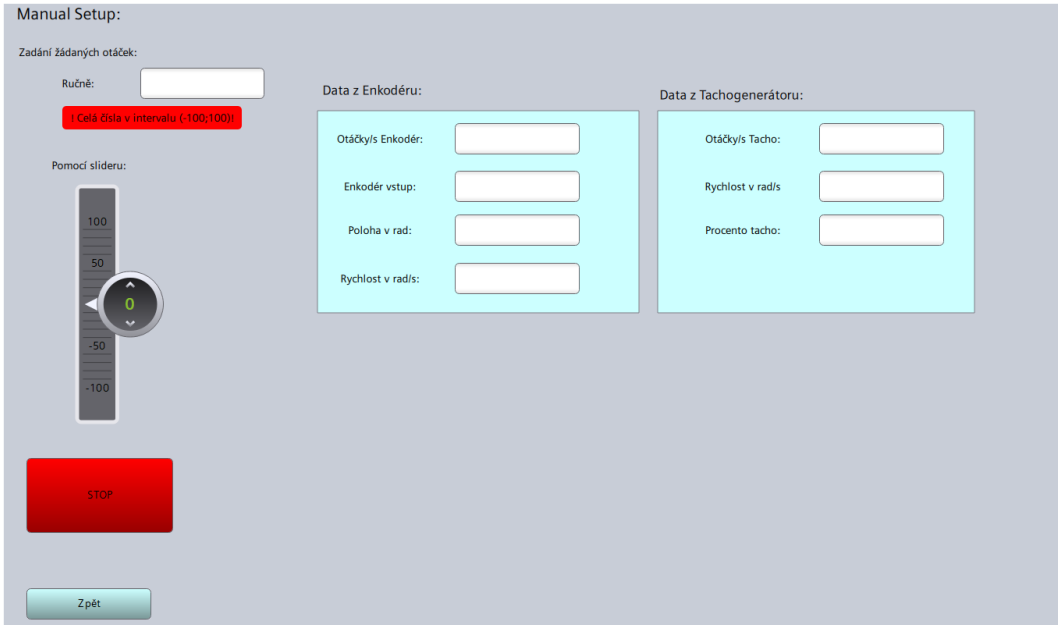


Obrázek 46 - Matlab – Okno Servo Manual Setup

Na základě předchozího rozhraní „Servo Manual Setup“ z prostředí MATLAB/Simulink, které slouží k testování a sledování servomechanismu pomocí RT-DAC/USB2 rozhraní, navrhneme vlastní vizualizaci v prostředí TIA Portal. Cílem je přenést klíčové funkce z akademického prostředí do podoby vhodné pro průmyslové PLC řízení.

Vytvoříme jednoduché uživatelské rozhraní pro zadávání vstupních otáček (ručně i pomocí posuvníku), zobrazování dat z enkodéru a tachogenerátoru a možnost nouzového zastavení systému. Struktura bude vycházet z původní aplikace, ale přizpůsobíme ji možnostem PLC a HMI panelu (Obrázek 47).

Oproti původní aplikaci nebudeme zobrazovat napětí z analogových vstupů ani nastavovat PWM prescaler. Zaměříme se na zobrazování již přepočtených fyzikálních veličin, jako jsou otáčky, poloha a rychlost.



Obrázek 47 - Manual Setup – HMI obrazovka

2.2.1 Vytvoření programové části

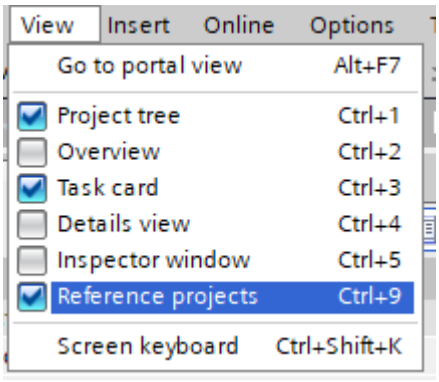
Budeme vycházet z našeho prvotního projektu, který jsme si v minulé úloze založili, vložili do něj zařízení podle reálného HW v učebně PL404 a otestovali správnost našeho nastavení.

Začneme tím, že si otevřeme referenční projekt ***UniverzitaPardubiceV18*** vytvořený firmou, která instalovala HW vybavení do učebny. Tento referenční projekt vám poskytne vyučující.

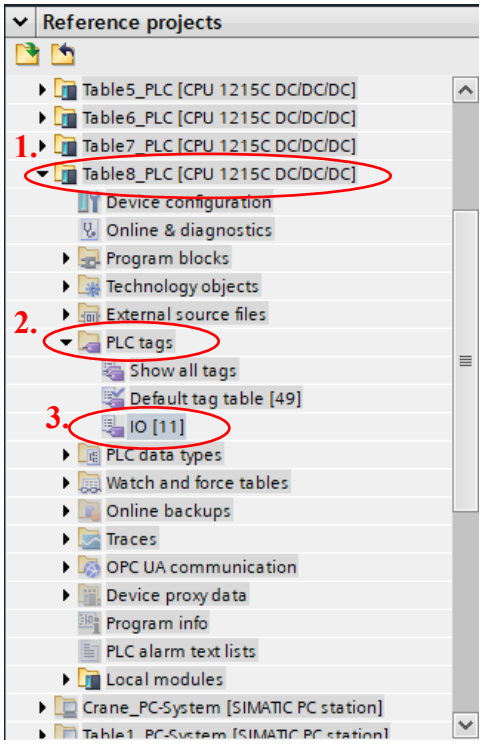
Reference project v TIA Portalu je šablonový projekt určený jako základ pro další projekty. Obsahuje připravené prvky, jako jsou programové bloky, HMI šablony nebo konfigurovaná zařízení, které lze kopírovat do jiných projektů. Otevírá se pouze pro čtení, což zajišťuje jeho neměnnost a usnadňuje standardizaci a opakované použití.

V projektovém stromě na levé straně obrazovky v TIA Portalu najdeme ve spodní části záložku „Reference projects“, pokud záložku nevidíme, v horním menu zvolíme možnost „View“ a zaškrtneme kolonku „Reference projects“ (Obrázek 48). Otevřeme záložku „Reference project“ a klikneme na ikonu „Open reference project“, v adresáři najdeme místo uložení projektu, který chceme otevřít jako referenční a otevřeme ho. Následně ho v záložce „Reference projects“ otevřeme, nyní z něj můžeme volně kopírovat obsah do našeho projektu.

Najdeme si PLC podle stolu, u kterého sedíme (Obrázek 18) a rozklikneme ho. Poté otevřeme záložku PLC tags (záložku s proměnnými), zde vidíme jednotlivé seznamy tagů, nás zajímá IO seznam (Obrázek 49).



Obrázek 48 - Zobrazení – Reference projects



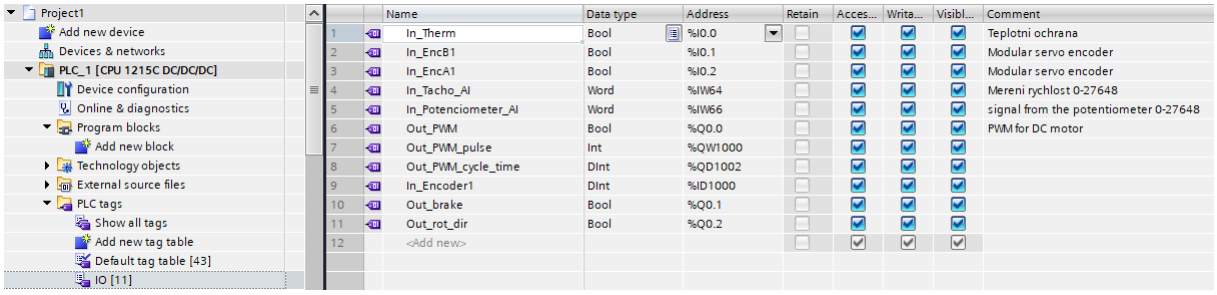
Obrázek 49 - Referenční projekt – IO tags

Odtud si zkopírujeme celý seznam do našeho projektu, buď pomocí příkazu Ctrl+C – Ctrl+V, nebo pomocí pouhého přetažení myší do složky „PLC tags“ v našem projektu.

V seznamu PLC tagů jsou definovány vstupy a výstupy (Obrázek 50), které zajišťují komunikaci mezi PLC a jednotlivými částmi servomechanismu. Mezi vstupy patří například In_Therm (%I0.0), což je digitální signál pro teplotní ochranu zařízení. Dvojice signálů In_EncB1 (%I0.1) a In_EncA1 (%I0.2) představuje vstupy z inkrementálního enkodéru, které slouží ke snímání polohy hřídele motoru. Analogové vstupy In_Tacho_AI (%IW64) a In_Potenciometer_AI (%IW66) přivádějí signály z tachogenerátoru a potenciometru; první měří skutečnou rychlost motoru, druhý slouží například k zadání hodnoty otáček.

Mezi výstupy nalezneme Out_PWM (%Q0.0), který aktivuje PWM výstup pro řízení motoru, a dále Out_PWM_pulse (%QW1000) a Out_PWM_cycle_time (%QD1002), které určují šířku

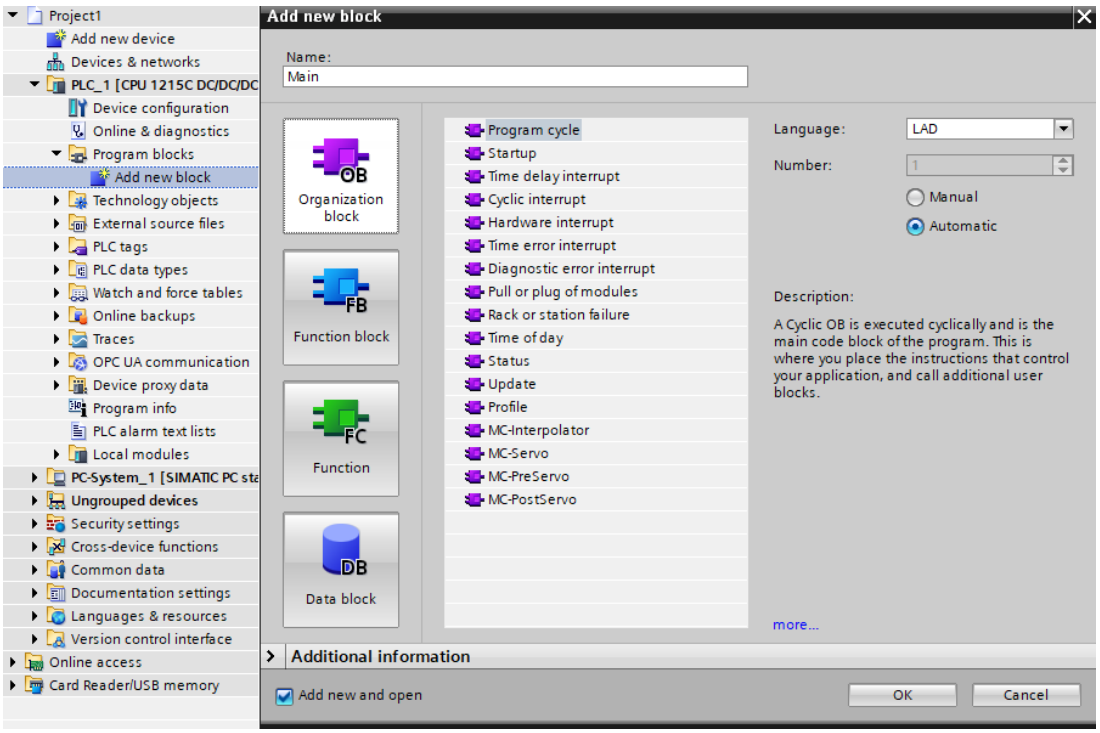
a délku PWM signálu. Datový výstup Out_Encoder1 (%ID1000) slouží pro přenos hodnoty z enkodéru. Dále zde nalezneme signál Out_brake (%Q0.1) pro aktivaci elektrické brzdy a Out_rot_dir (%Q0.2), kterým se určuje směr otáčení motoru.



Obrázek 50 - Seznam IO tagů v projektu

Přistoupíme k samotnému vytváření programu. Nejdříve si v záložce „Program Blocks“ vytvoříme základní organizační blok, jméno ponecháme „Main“ a zvolíme programovací jazyk LAD (Ladder diagram) pro svou přehlednost a čitelnost zejména v jednoduchých logických strukturách (Obrázek 51). Tento blok je základní součástí každého PLC projektu a je automaticky spouštěn cyklicky.

Volba bloku „Program cycle“ je nezbytná v případech, kdy chceme průběžně vyhodnocovat vstupní signály, provádět logické operace a aktualizovat výstupy, což je stěžejní pro řízení motoru v reálném čase. Do tohoto bloku zapisujeme hlavní logiku programu a v případě potřeby zde také voláme funkční bloky.



Obrázek 51 - Vkládání OB Main

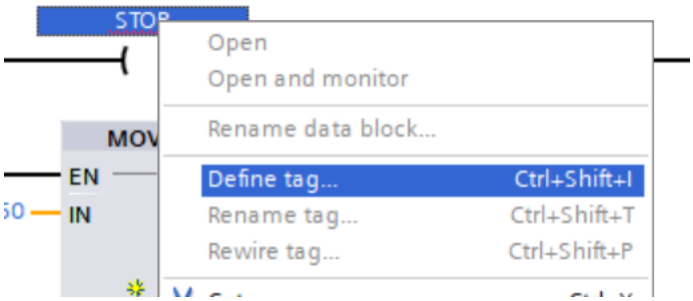
Vkládání jednotlivých prvků do programu v jazyce LAD je možné několika způsoby.

Prvním z nich je kliknutím na jejich ikonu v horní části obrazovky (Obrázek 56). Na obrázku (Obrázek 52) je zobrazeno šest základních prvků programovacího jazyka LAD v prostředí TIA Portal. Prvním symbolem zleva je „normally open contact“, který reprezentuje podmínku splněnou při logické jedničce. Druhý symbol je „normally closed contact“, tedy opačná podmínka, splněná při logické nule. Třetí prvek představuje „cívku (output coil)“, tedy klasický výstup, který je aktivován, pokud jsou splněny předchozí podmínky v řádku. Čtvrtý symbol označuje „funkční blok (box)“, pomocí kterého voláme funkce nebo funkční bloky. Pátý prvek slouží k vytvoření větvení programu, tedy paralelní logické větve v rámci jednoho síťového řádku, zatímco šestý symbol značí ukončení větvení a sloučení větví zpět do jedné logické linie.

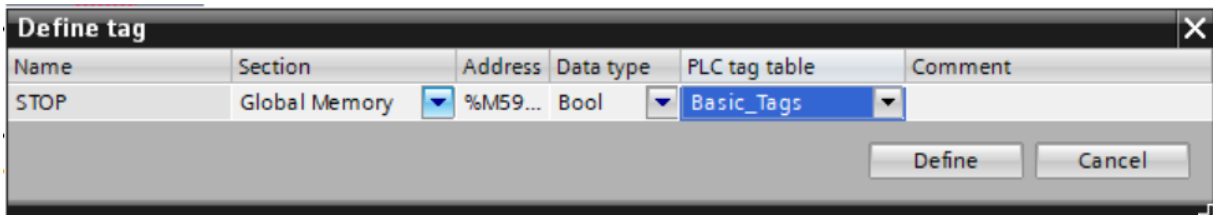
Obrázek 52 - Ikony pro tvorbu logiky v LAD editoru

Další způsob se týká zejména vkládání funkčních bloků, například matematických nebo časovačů, ty můžeme vložit přetažením ze záložky „Instructions“ na pravé straně obrazovky (Obrázek 56).

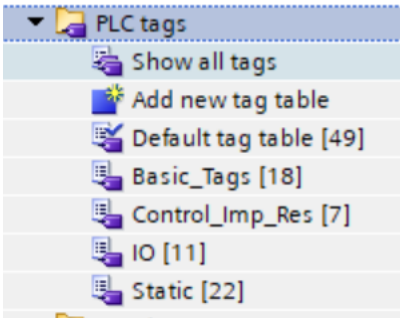
Je taky vhodné zmínit, jak vytvářet proměnné (tagy). Nejintuitivnější tvorba tagů je jejich tvorba přímo v programu. Do místa pro proměnnou napíšeme jméno tagu, který chceme vytvořit, poté na něj klikneme pravým tlačítkem na myši zvolíme možnost „Define tag…“ (Obrázek 53). Otevře se nám okno pro definici tagů (Obrázek 54). Zde zvolíme, jestli chceme, aby byl tag lokální nebo globální, lokální tag je dostupný pouze v rámci konkrétního programového bloku, zatímco globální tag je uložený v globální paměti PLC a je přístupný napříč celým projektem, má přiřazenou pevnou adresu, například M0.0 v oblasti bitové paměti (Merker). Dále si zvolíme, do jakého seznamu tagů chceme tag uložit, je doporučováno si třídit tagy do odlišných seznamů, například podle části, kde se v programu nachází. Nový seznam tagů vytvoříme kliknutím na pole „Add new tag table“ viz. Obrázek 55.



Obrázek 53 – Nedefinovaný tag



Obrázek 54 - Okno pro definici tagu



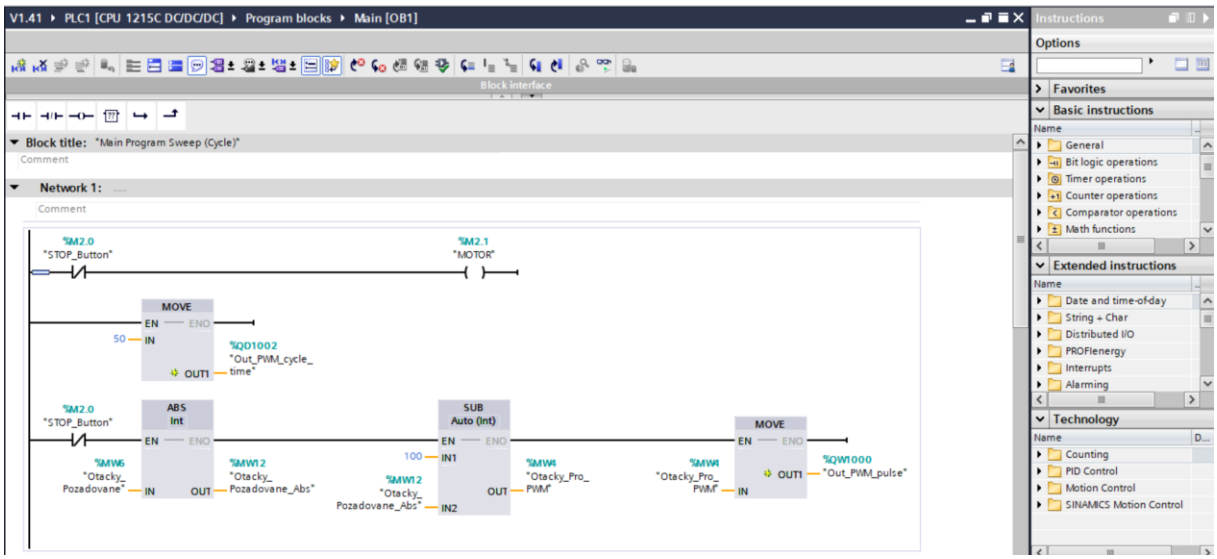
Obrázek 55 - Více seznamů tagů

Začneme implementováním logiky pro řízení otáček motoru pomocí PWM signálu na základě nastavené hodnoty v proměnné „Otacky_Pozadovane“ viz. Obrázek 56. Na obrázku vidíme, realizaci výpočtu šířky PWM signálu pro řízení otáček motoru. Výpočty se provádí pouze tehdy, pokud není aktivní signál STOP_Button.

V první větvi probíhá aktivace logické proměnné „MOTOR“, kterou využíváme v další části programu.

Ve druhé větvi pomocí bloku „MOVE“ přiřadíme pevnou hodnotu 50 proměnné „Out_PWM_cycle_time“, čímž nastavíme délku jednoho PWM cyklu.

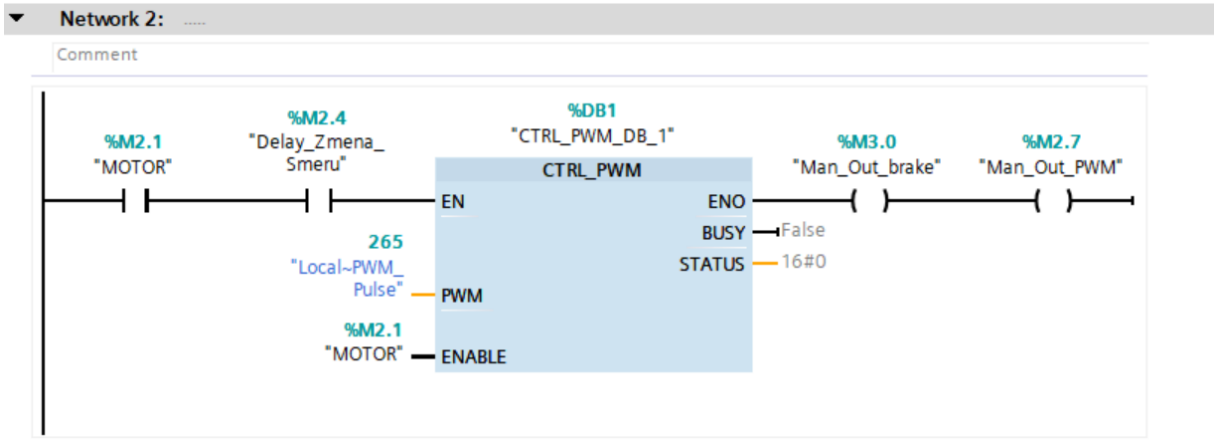
Ve druhé větvi převedeme vstupní hodnotu „Otacky_Pozadovane“ na absolutní hodnotu pomocí bloku „ABS“ a výsledek uložíme do proměnné „Otacky_Pozadovane_Abs“. Zadané otáčky převádíme do absolutní hodnoty proto, protože budeme provádět řízení motoru v obou směrech, tedy hodnota zadaných otáček bude nabývat i záporných hodnot. Samotný směr otáčení motoru má na starosti proměnná „OUT_rot_dir“. Tuto hodnotu následně odečteme od konstanty 100 v bloku „SUB“ a výsledek uložíme do proměnné „Otacky_Pro_PWM“. Zadanou hodnotu odečítáme od 100, protože střída PWM signálu v dané větvi programu odpovídá inverznímu směru řízení, tedy čím vyšší požadavek, tím nižší hodnota střídy. Tímto způsobem připravujeme PWM výstup pro řízení motoru v opačném směru, přičemž samotný směr je nastaven nezávisle přes proměnnou určující směr otáčení „OUT_rot_dir“. Pomocí dalšího bloku „MOVE“ pak tuto hodnotu přiřadíme proměnné „Out_PWM_pulse“, která určuje výslednou šířku PWM pulzu.



Obrázek 56 - Konfigurace PWM v programu

V další části programu vložíme funkční blok „CTRL_PWM“.

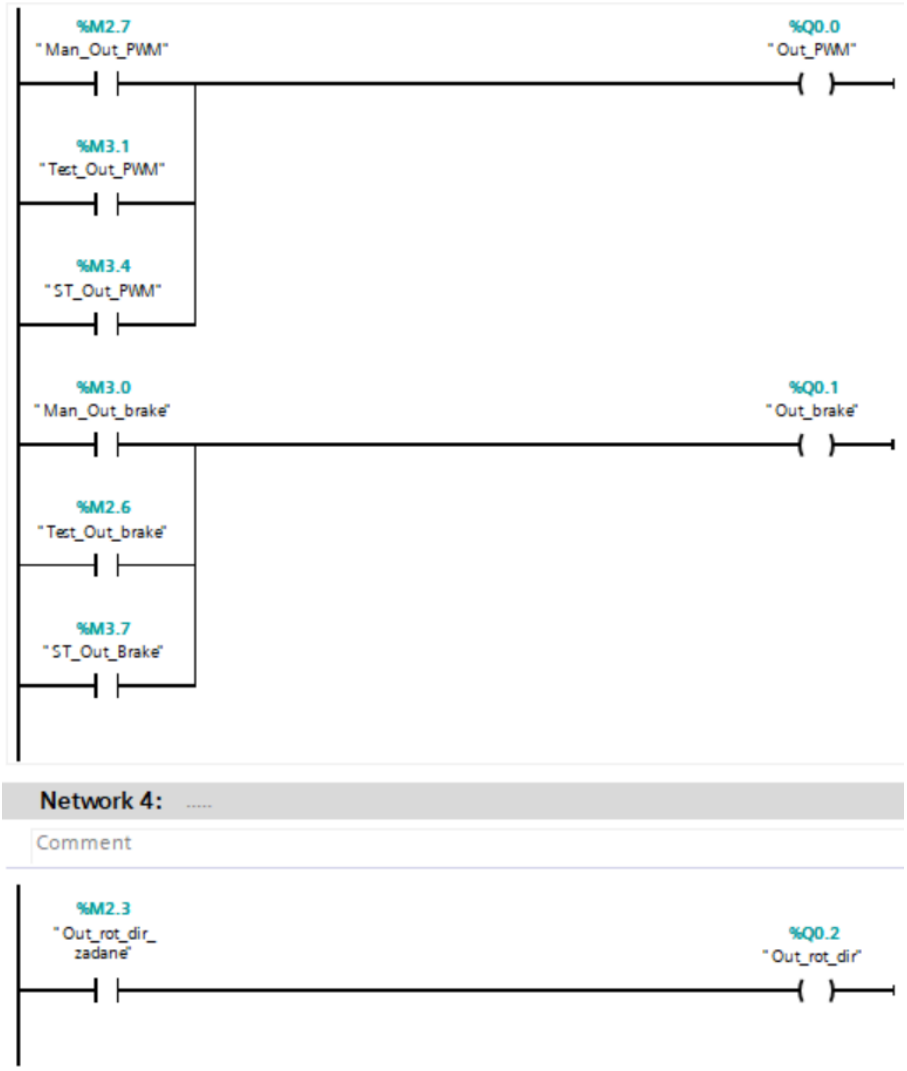
Funkční blok „CTRL_PWM“ slouží k řízení PWM signálu. Aktivuje se, pokud je na vstupu EN logická jednička (z proměnné „MOTOR“) a zároveň je aktivní zpoždění změny směru („Delay_Zmena_Smeru“), bude vysvětleno později. Do vstupu PWM zapisujeme hodnotu šířky pulzu („Local_PWM_Pulse“), která určuje intenzitu řízení. Vstup ENABLE je rovněž řízen proměnnou MOTOR, čímž se povoluje činnost bloku. Výstup nám aktivuje proměnné „Man_Out_PWM“ a „Man_Out_Brake“ (Obrázek 57).



Obrázek 57 - FB CTRL_PWM

Jelikož budeme postupně vytvářet ještě další dvě úlohy ve stejném projektu, ošetříme si jednotlivé následující výstupní proměnné následující logikou viz. Obrázek 58. Výstupní proměnná „Out_PWM“ slouží k aktivaci PWM signálu, který umožňuje řízení výkonu a otáček motoru podle nastavené šířky pulzu. Proměnná „Out_brake“ aktivuje elektrickou brzdu motoru, která slouží jako bezpečnostní prvek pro okamžité zastavení motoru po přechodu do logické 0. Ostatní proměnné na obrázku máme příhodně pojmenované pro usnadnění orientace v programu.

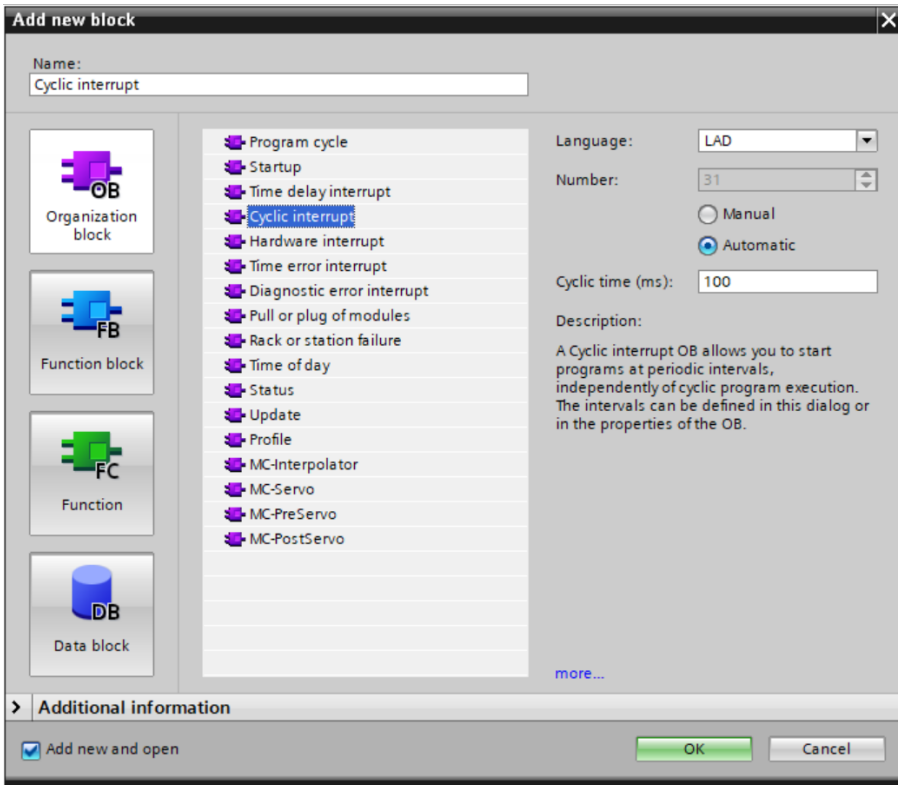
„Out_rot_dir“ je výstupní proměnná sloužící k určení směru otáčení motoru, v tento okamžik pouze přepisujeme hodnotu pomocné globální proměnné „Out_rot_dir_zadane“ (budeme nastavovat v dalších částech programu) na reálný výstup.



Obrázek 58 - Logika pro řízení výstupů PWM, brzdy a směru otáčení motoru

V následujícím kroku si vytvoříme organizační blok, ve kterém bude prováděn výpočet hodnot jako je například rychlost otáčení motoru nebo kontrola změny směru otáčení. Blok vložíme obdobně jako OB „Main“ viz. Obrázek 51.

V rámci řízení a sběru dat jsme vytvořili organizační blok typu „Cyclic interrupt“ (Obrázek 59), který zajišťuje spouštění části programu v přesně definovaných časových intervalech, nezávisle na hlavním programovém cyklu. Tento typ bloku je vhodný zejména pro periodické výpočty a časově citlivé úlohy. V našem případě jsme nastavili cyklický čas na 100 ms, což znamená, že obsažená logika se vykonává každých 100 milisekund. Takto lze zajistit pravidelný výpočet a aktualizaci dat bez ohledu na délku vykonávání ostatních částí programu.



Obrázek 59 - Vkládání OB Cyclic interrupt

Pokračujeme tvorbou samotných funkčních bloků. Funkční blok (FB) v TIA Portalu slouží k vytvoření opakovaně použitelné části programu, která může uchovávat vnitřní stav mezi jednotlivými voláními. Na rozdíl od běžné funkce (FC) má vlastní paměť (instanční datový blok). Jako první se budeme zabývat výpočtem rychlostí ze vstupu inkrementálního enkodéru. Obdobně jako u vkládání OB klikneme v záložce „Program bloks“ na ikonu „Add new block“ (Obrázek 51), zde zvolíme možnost „Function block“, zvolíme programovací jazyk a příhodně si ho pojmenujeme.

Ještě, než se pustíme do samotného programování, vysvětlíme si význam a účel jednotlivých typů proměnných obsažených ve funkčních blocích.

Viz. Obrázek 60. Ve funkčním bloku „FB_Counter_To_Speed“ využíváme různé typy proměnných, které jsou rozděleny podle svého účelu a způsobu použití. Vstupní proměnná (Input) „CycleTime“ slouží k předání časového intervalu cyklického volání bloku, který je nutný pro správný výpočet rychlosti, stejný jako čas spouštění našeho OB „Cyclic Interrupt“. Výstupní proměnné (Output) „Rad_za_sekundu“, „Rad_Poloha“ a „Otacky_za_Sekundu“ obsahují výsledky výpočtů, které předáváme dále do hlavního programu.

Pro uchování hodnot mezi jednotlivými cykly používáme statické proměnné (Static) jako například „Akt_Rad“, „Soucet_Rad“ nebo „Record_Counter“. Tyto proměnné jsou vhodné například pro sledování změn v čase nebo pro uchování předchozí hodnoty čítače enkodéru.

Dočasné proměnné (Temp), jako jsou „Current_Counter“, „Delta_Counter“ nebo „Pomocny_Real“, slouží k mezivýpočtům v rámci jednoho cyklu vykonání bloku a po jeho ukončení se jejich hodnoty neuchovávají. Nakonec je zde i konstantní proměnná „2*pi“, která reprezentuje hodnotu matematické konstanty a může být využita při převodu otáček nebo polohy do radiánové formy.

Využití jednotlivých proměnných bude vysvětleno dále v úloze.

FB_Counter_To_Speed

	Name	Data type	Default value	Retain	Accessible f...	Writa...	Visible in ...	Setpoint
1	Input				☐	☐	☐	☐
2	CycleTime	Int	0	Non-ret...	☑	☑	☑	☐
3	Output				☐	☐	☐	☐
4	Rad_za_sekundu	Real	0.0	Non-retain	☑	☑	☑	☐
5	Rad_Poloha	Real	0.0	Non-retain	☑	☑	☑	☐
6	Otacky_za_Sekundu	Real	0.0	Non-retain	☐	☐	☐	☐
7	InOut				☐	☐	☐	☐
8	<Add new>				☐	☐	☐	☐
9	Static				☐	☐	☐	☐
10	Akt_Rad	Real	0.0	Non-retain	☐	☐	☐	☐
11	Soucet_Rad	Real	0.0	Non-retain	☐	☐	☐	☐
12	Record_Counter	DInt	0	Non-retain	☐	☐	☐	☐
13	Temp				☐	☐	☐	☐
14	Current_Counter	DInt			☐	☐	☐	☐
15	Delta_Counter	DInt			☐	☐	☐	☐
16	Pomocny_Dint	DInt			☐	☐	☐	☐
17	Pom_Rad	Real			☐	☐	☐	☐
18	Pomocny_Real	Real			☐	☐	☐	☐
19	Constant				☐	☐	☐	☐
20	2*pi	Real	6.283185		☐	☐	☐	☐

Obrázek 60 - Proměnné ve funkčním bloku pro výpočet rychlosti z enkodéru

Na obrázku (Obrázek 61) je zobrazena první část výpočtu otáček motoru z hodnot enkodéru ve funkčním bloku. V první síti nejprve přiřadíme aktuální hodnotu z čítače enkodéru „In_Encoder1“ do proměnné „Current_Counter“. Následně odečteme od této hodnoty předchozí zaznamenanou hodnotu „Record_Counter“ a výsledek uložíme do proměnné „Delta_Counter“, čímž získáme počet impulsů zaregistrovaných během jednoho cyklu. Tato operace odpovídá výrazu (1):

$$Delta\_Counter = Current\_Counter - Record\_Counter \quad (1)$$

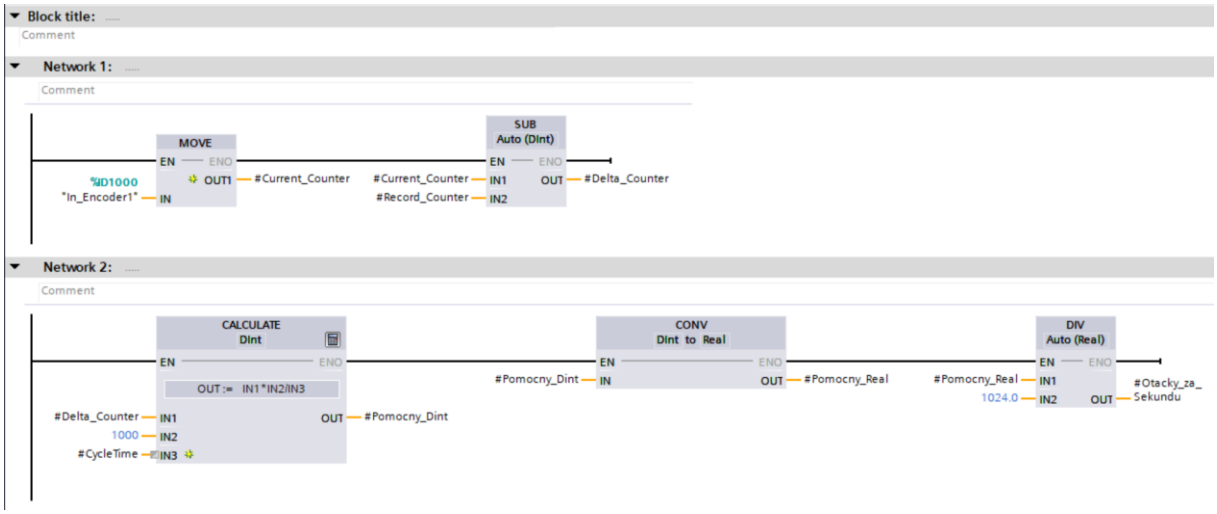
Ve druhé síti pokračujeme výpočtem otáček za sekundu. Pomocí bloku „CALCULATE“ vynásobíme hodnotu „Delta_Counter“ konstantou 1000 a výsledek vydělíme délkou cyklu „CycleTime“ v milisekundách. Tím získáme počet impulsů za sekundu, ten uložíme do proměnné „Pomocny_Dint“. Výpočet odpovídá rovnici(2):

$$Pomocny\_Dint = \frac{Delta\_Counter \times 1000}{CycleTime} \quad (2)$$

Následně provedeme konverzi datového typu z DInt na Real a výsledek uložíme do proměnné „Pomocny_Real“. V posledním kroku vydělíme hodnotu „Pomocny_Real“ konstantou 1024 (počet impulsů na jednu otáčku), čímž získáme výstupní hodnotu otáček za sekundu ve formě (3):

$$Otacky\_za\_Sekundu = \frac{Pomocny\_Real}{1024} \quad (3)$$

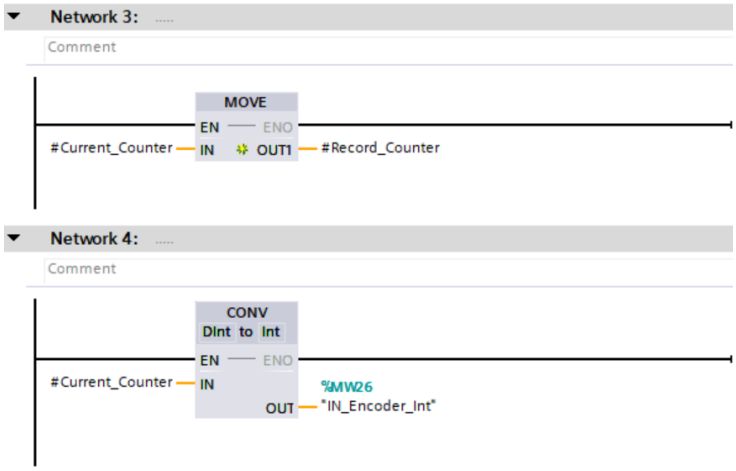
Tímto způsobem získáváme aktuální rychlost otáčení motoru na základě změny stavu čítače enkodéru v přesně definovaném časovém intervalu.



Obrázek 61 - FB výpočet rychlosti z enkodéru 1

Ve třetí síti (Obrázek 62) provádíme aktualizaci hodnoty proměnné „Record_Counter“ pro další cyklus výpočtu. Pomocí instrukce „MOVE“ přiřadíme aktuální hodnotu z proměnné „Current_Counter“ do proměnné „Record_Counter“. Tím si uložíme stav čítače enkodéru pro porovnání při příštím volání funkčního bloku.

Ve čtvrté síti (Obrázek 62) pak převádíme hodnotu „Current_Counter“ z datového typu DInt (double integer) na typ Int. Výsledek převodu uložíme do paměťové adresy „IN_Encoder_Int“, kterou lze následně použít například pro vizualizaci v HMI rozhraní. Tato hodnota představuje aktuální stav čítače enkodéru ve zjednodušené podobě, která je lépe interpretovatelná pro uživatele.



Obrázek 62 - FB výpočet rychlosti z enkodéru 2

V síti 5 (Obrázek 63) provádíme výpočet aktuální úhlové rychlosti a úhlové polohy v radiánech. Pomocí bloku „MUL“ nejprve vynásobíme hodnotu „Otacky_za_Sekundu“ konstantou „2*pi“, čímž získáme úhlovou rychlost v radiánech za sekundu. Tento výpočet odpovídá rovnici (4):

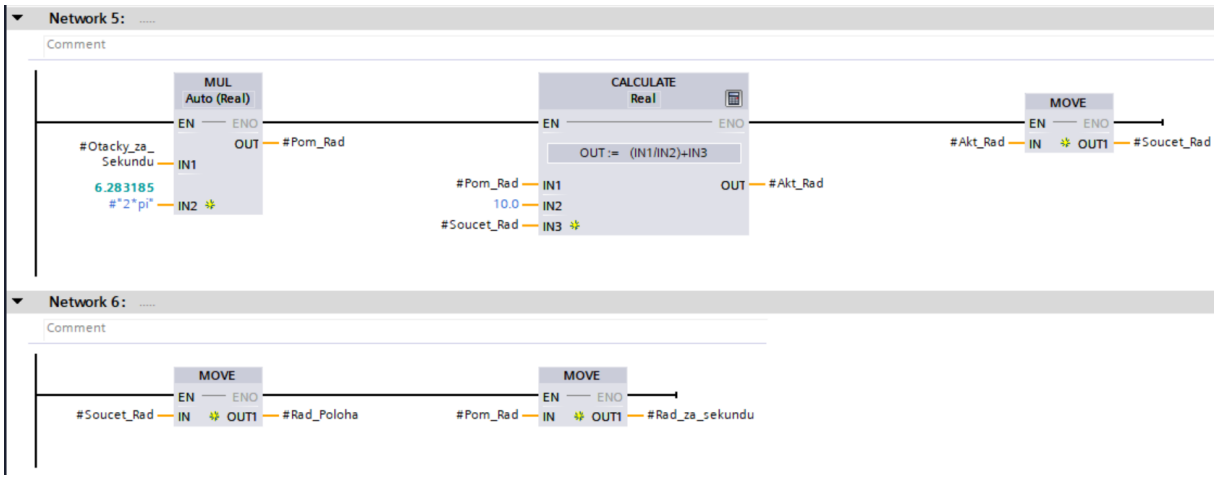
$$Pom\_Rad = Otacky\_za\_Sekundu \times 2\pi \tag{4}$$

V následujícím kroku počítáme aktuální úhlovou polohu „Akt_Rad“. Dělíme hodnotu „Pom_Rad“ konstantou 10, která vyjadřuje počet volání funkčního bloku za sekundu (vzhledem k tomu, že cyclic interrupt je nastaven na 100 ms), a výsledek přičítáme k předchozí hodnotě polohy „Soucet_Rad“. Tento výpočet odpovídá výrazu (5):

$$Akt\_Rad = \left(\frac{Pom\_Rad}{10}\right) + Soucet\_Rad \tag{5}$$

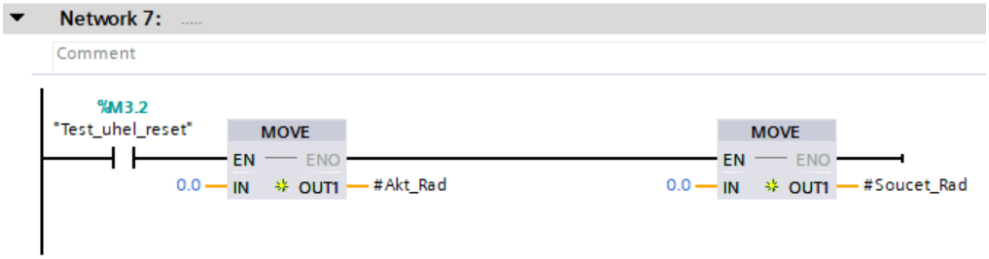
Na konci sítě pomocí bloku „MOVE“ přiřazujeme nově vypočtenou hodnotu „Akt_Rad“ zpět do proměnné „Soucet_Rad“, která bude použita jako základ pro výpočet v následujícím cyklu.

V síti 6 (Obrázek 63) aktualizujeme výstupní proměnné. Hodnota „Soucet_Rad“ je zkopírována do proměnné „Rad_Poloha“ a aktuální úhlová rychlost „Pom_Rad“ do proměnné „Rad_za_sekundu“. Tím je výpočet úhlové rychlosti a polohy motoru pomocí vstupních dat z enkodéru uzavřen pro daný cyklus.



Obrázek 63 - FB výpočet rychlosti z enkodéru 3

V síti 7 (Obrázek 64) nulujeme hodnoty s aktuálními parametry úhlové rychlosti a polohy uložené v paměti FB pomocí patřičné proměnné, kterou budeme aktivovat přes HMI rozhraní.



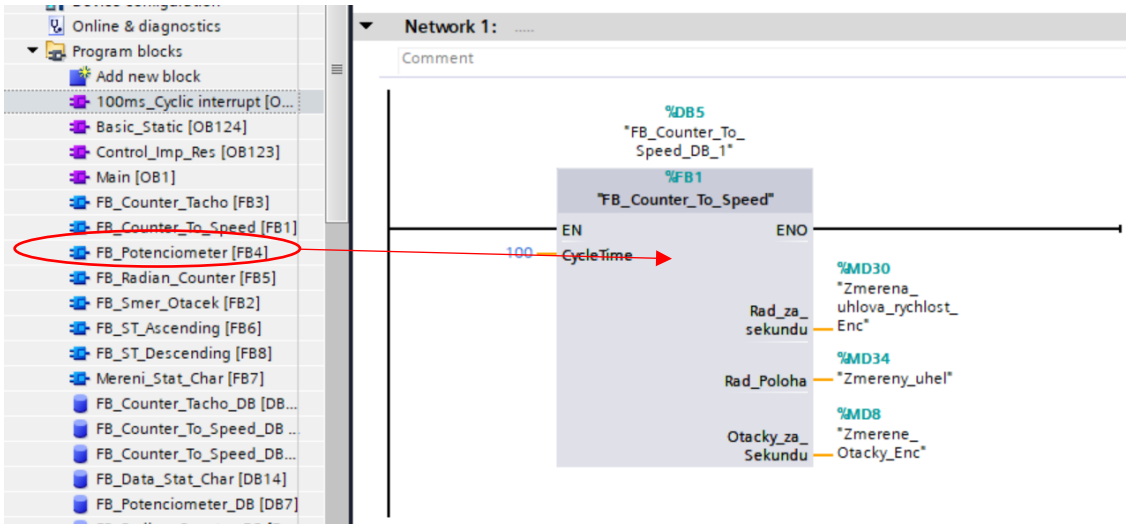
Obrázek 64 - FB výpočet rychlosti z enkodéru 4

Teď už jen zbývá vložit funkční blok do organizačního bloku OB „Cyclic interrupt“ viz. Obrázek 65. Kliknutím a přetáhnutím ze seznamu „Program blocks“ v projektovém stromě do sítě v OB „Cyclic Interrupt“. Jak je z obrázku patrné, funkční blok má svoje vstupy a výstupy, které jsme nadefinovali ve vnitřní struktuře FB viz. Obrázek 60.

Do vstupu „CycleTime“ je pevně přiřazena hodnota 100, která odpovídá době mezi jednotlivými voláními bloku (100 ms).

Výstupní proměnné „Rad_za_sekundu“, „Rad_Poloha“ a „Otacky_za_Sekundu“ jsou přiřazeny k paměťovým místům „Zmerena_uhlova_rychlost_Enc“, „Zmerený_uhel“ a „Zmerene_Otacky_Enc“, které jsme příhodně vytvořili, pojmenovali a uložili do příslušného seznamu tagů, kde jsou dále dostupné pro další zpracování nebo vizualizaci.

Volání bloku je propojeno s jeho instančním datovým blokem „FB_Counter_To_Speed_DB_1“, který uchovává interní stav funkčního bloku mezi jednotlivými cykly.



Obrázek 65 - Vložení FB pro výpočet rychlostí z dat z enkodéru do OB Cyclic Interrupt

Jako další FB pro výpočet rychlosti a polohy si připravíme blok, který bude čerpat data z tachogenerátoru.

Tachogenerátor je snímač, který poskytuje analogový napěťový signál úměrný okamžité rychlosti otáčení hřídele. Tento signál neobsahuje žádné informace o směru ani o celkové ujeté dráze – slouží výhradně k měření aktuálních otáček (rychlosti) v jednom směru. V PLC je tento napěťový výstup převáděn pomocí A/D převodníku na číselnou hodnotu, se kterou dále pracujeme v programu.

Nový funkční blok vytvoříme stejným postupem jako předchozí funkční blok, proto přejdeme hned na programování. Na obrázku níže (Obrázek 66) můžeme vidět seznam proměnných v následujícím FB, jejich význam jsme si vysvětlili u minulého FB. Jejich konkrétní využití je uvedeno dále v úloze.

FB_Counter_Tacho

	Name	Data type	Default value	Retain	Accessible f...	Writa...	Visible in ...	Setpoint
1	Input				☐	☐	☐	☐
2	In Tacho	DWord	16#0	Non-ret...	☑	☑	☑	☐
3	Output				☐	☐	☐	☐
4	Otacky_Tacho	Real	0.0	Non-retain	☑	☑	☑	☐
5	Procento	Real	0.0	Non-retain	☑	☑	☑	☐
6	Rad_za_sekundu	Real	0.0	Non-retain	☑	☑	☑	☐
7	InOut				☐	☐	☐	☐
8	<Add new>				☐	☐	☐	☐
9	Static				☐	☐	☐	☐
10	Soucet_Rad	Real	0.0	Non-retain	☐	☐	☐	☐
11	Akt_Rad	Real	0.0	Non-retain	☐	☐	☐	☐
12	Temp				☐	☐	☐	☐
13	Pomocna_prom	Int			☐	☐	☐	☐
14	Pom_Rad	Real			☐	☐	☐	☐
15	Constant				☐	☐	☐	☐
16	2*pi	Real	6.283185		☐	☐	☐	☐
17	Max_Otacky	Real	27.3		☐	☐	☐	☐

Obrázek 66 - Proměnné ve funkčním bloku pro výpočet rychlosti z tachogenerátoru

Na obrázku (Obrázek 67) je zachycena první část výpočtu aktuální rychlosti motoru na základě hodnoty z tachogenerátoru. Tachogenerátor poskytuje analogový signál, jehož napěťová úroveň odpovídá aktuální rychlosti otáčení hřídele. Tento signál je v PLC převeden A/D převodníkem na číselnou hodnotu, která je dále zpracovávána.

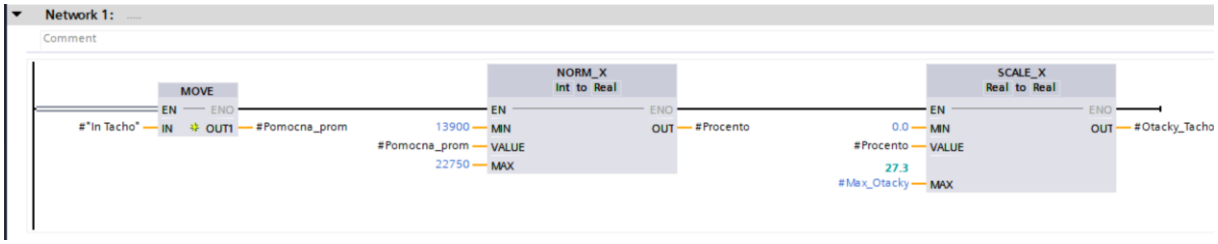
V síti nejprve přiřadíme hodnotu vstupu „In_Tacho“ do pomocné proměnné „Pomocna_prom“. Tato hodnota je následně normalizována pomocí bloku „NORM_X“ na rozsah 0 až 1. Při normalizaci zadáváme minimální a maximální hodnoty napěťového rozsahu (v tomto případě 13900 a 22750), které byly získány experimentálně při měření nulové a maximální rychlosti. Normalizační výpočet odpovídá rovnici (6):

$$Procento = \frac{Pomocna\_prom - 13900}{22750 - 13900} \tag{6}$$

Takto získaná hodnota „Procento“ je následně převedena na skutečnou rychlost motoru v otáčkách za sekundu pomocí bloku „SCALE_X“. Zde je výstup škálován z rozsahu 0–1 na rozsah 0–27,3, přičemž 27,3 odpovídá maximální rychlosti zjištěné experimentálně (Obrázek 91). Jako výchozí hodnotu lze použít například hodnota 30. Výpočet odpovídá rovnici (7):

$$Otacky\_Tacho = Procento \times 27{,}3 \tag{7}$$

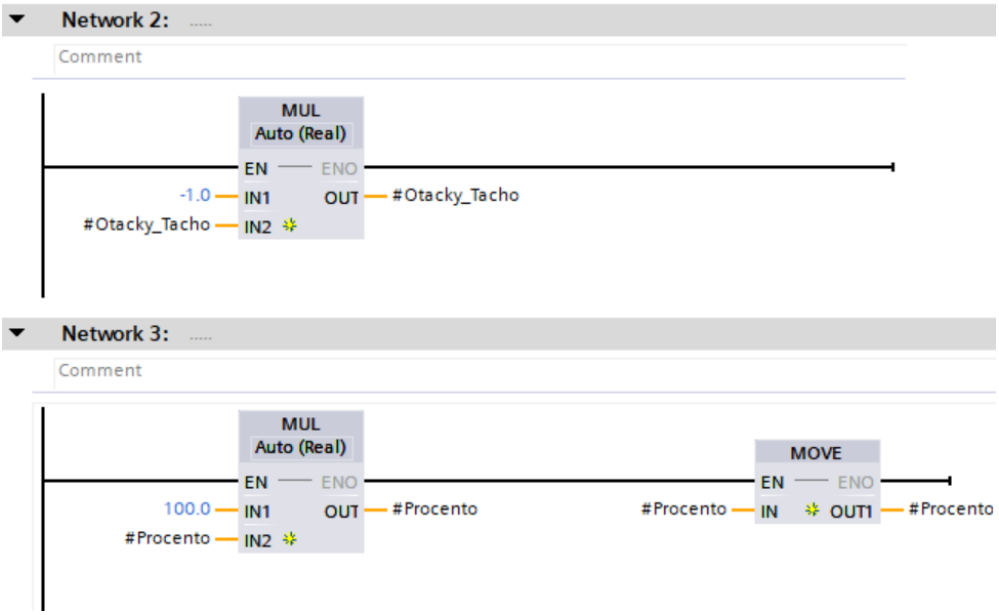
Výsledná hodnota „Otacky_Tacho“ pak představuje aktuální rychlost otáčení motoru získanou ze signálu tachogenerátoru.



Obrázek 67 - FB výpočet rychlosti z tachogenerátoru 1

V síti 2 (Obrázek 68) násobíme hodnotu „Otacky_Tacho“ konstantou −1, čímž přepočítáváme směr otáčení, důvodem je, že při porovnávání rychlosti z enkodéru byla rychlost z tachogenerátoru převrácená, proto se zde převrací její hodnota pro získání stejného výsledku.

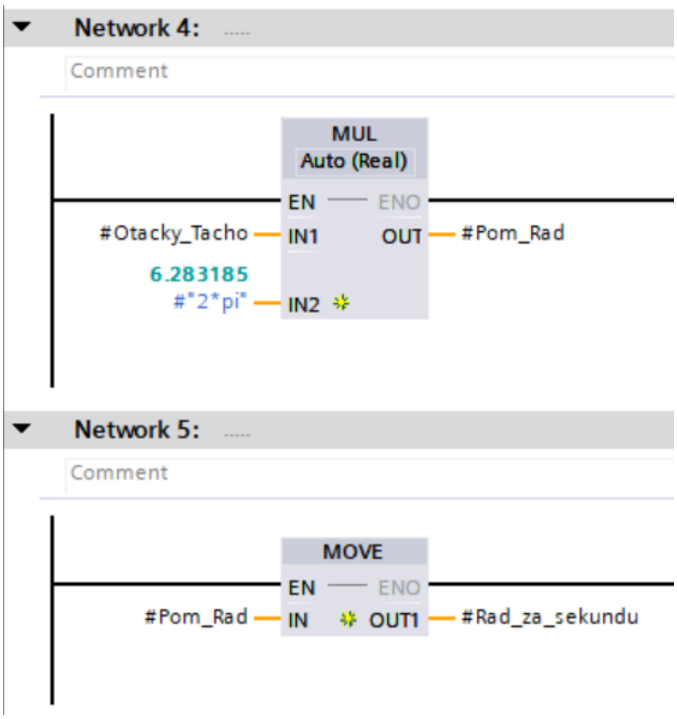
V síti 3 (Obrázek 68) převádíme hodnotu „Procento“ (normalizovanou mezi 0 a 1) na procentuální vyjádření v rozsahu 0–100 %. To provádíme násobením konstantou 100 a výsledek uložíme zpět do proměnné „Procento“, kterou následně použijeme pro vizualizaci v HMI.



Obrázek 68 - FB výpočet rychlosti z tachogenerátoru 2

V síti 4 (Obrázek 69) násobíme aktuální hodnotu otáček „Otacky_Tacho“ konstantou „2*pi“, čímž získáváme úhlovou rychlost v radiánech za sekundu. Výsledek ukládáme do proměnné „Pom_Rad“.

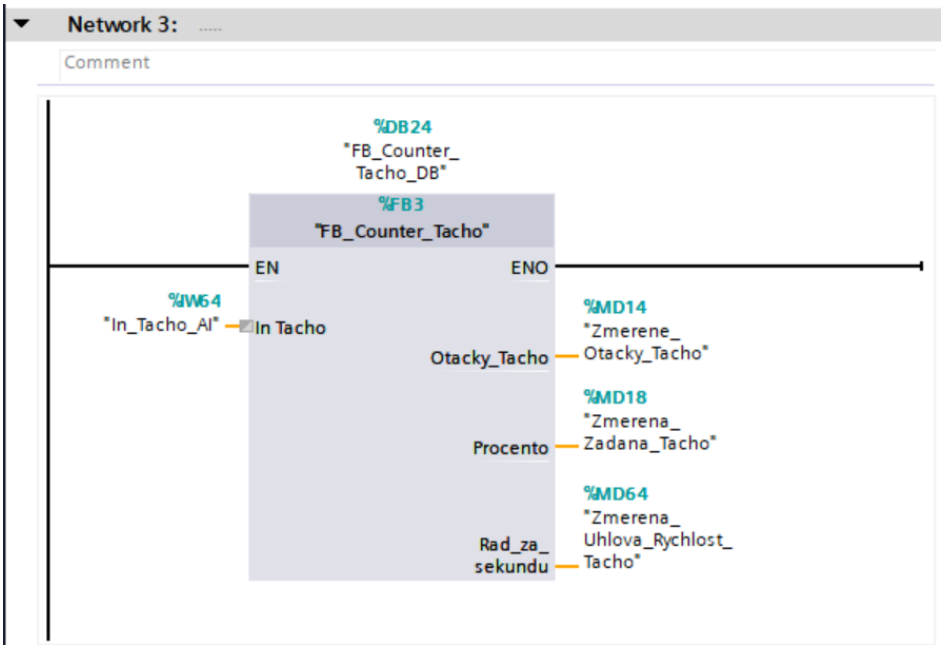
V síti 5 (Obrázek 69) pak tuto hodnotu jednoduše přiřazujeme do výstupní proměnné „Rad_za_sekundu“, kterou použijeme například pro vizualizaci v HMI.



Obrázek 69 - FB výpočet rychlosti z tachogenerátoru 3

Tímto jsme dokončili FB, který nám přepočítává vstupní hodnotu z tachogenerátoru, nyní ho vložíme do OB „Cyclic Interrupt“ stejně jako předchozí FB, viz. Obrázek 65. Opět si vytvoříme globální proměnné pro uložení vypočítaných hodnot, ty nám poslouží pro další využití vypočítaných dat.

Na obrázku (Obrázek 70) vidíme, že do vstupu „In_Tacho“ je přivedena analogová hodnota ze vstupu „In_Tacho_AI“, která představuje výstup z tachogenerátoru odpovídající aktuální rychlosti motoru.



Obrázek 70 - FB_Counter_Tacho v OB Cyclic Interrupt

Jako další si připravíme funkční blok, který nám bude ošetřovat změnu směru otáčení motoru. Při řízení motoru je nutné ošetřit změnu směru otáčení krátkým zpožděním. V našem hardwarovém zapojení by okamžité přepnutí směru mohlo způsobit proudový ráz nebo mechanické přetížení vlivem setrvačnosti. Zavedením prodlevy mezi vypnutím jednoho směru a zapnutím druhého zajistíme plynulejší a bezpečnější chod motoru i celého řídicího systému.

Nový funkční blok vytvoříme stejným postupem jako předchozí funkční bloky, proto přejdeme hned na programování.

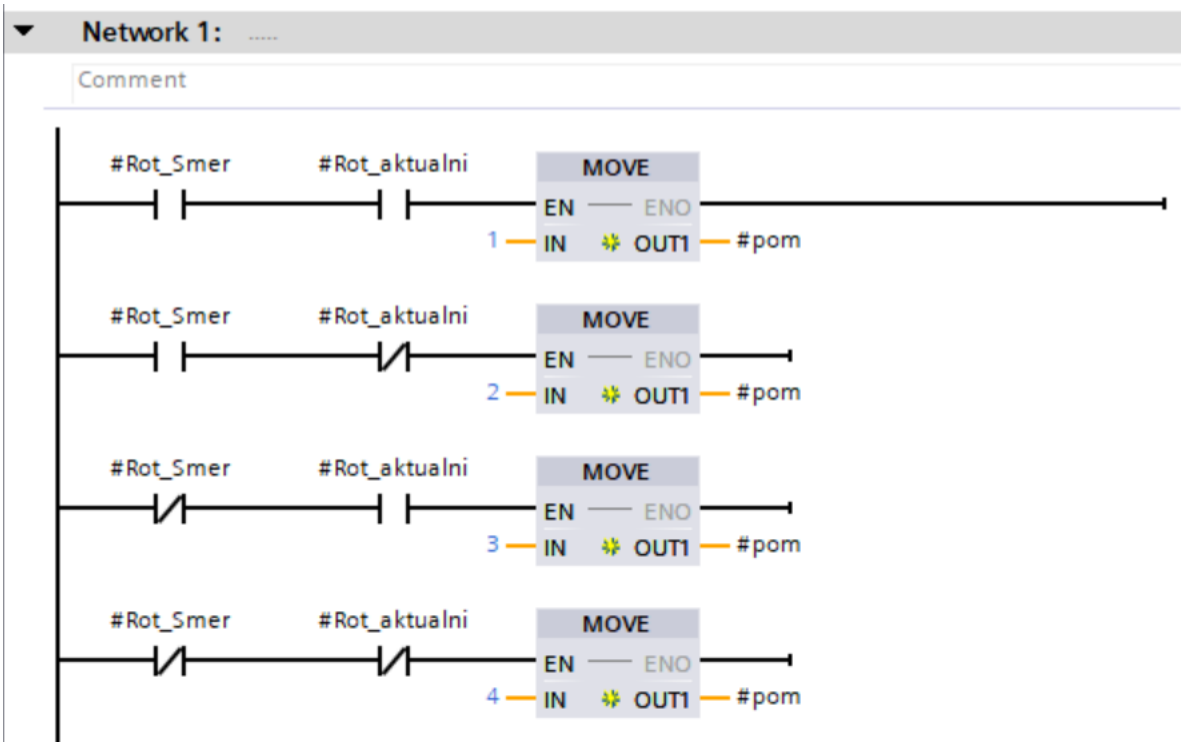
Na obrázku níže (Obrázek 71) můžeme vidět seznam proměnných v následujícím FB, jejich význam jsme si vysvětlili u minulého FB. Jejich konkrétní využití je uvedeno dále v úloze.

FB_Smer_Otacek

	Name	Data type	Default value	Retain	Accessible f...	Writa...	Visible in ...	Setpoint
1	Input				☐	☐	☐	☐
2	Zadane_otacky	Int	0	Non-ret...	☑	☑	☑	☐
3	Output				☐	☐	☐	☐
4	Smer_toceni	Bool	false	Non-retain	☑	☑	☑	☐
5	InOut				☐	☐	☐	☐
6	<Add new>				☐	☐	☐	☐
7	Static				☐	☐	☐	☐
8	Rot_aktualni	Bool	false	Non-retain	☐	☐	☐	☐
9	Rot_Smer	Bool	false	Non-retain	☑	☑	☑	☐
10	Temp				☐	☐	☐	☐
11	pom	Int			☐	☐	☐	☐
12	Constant				☐	☐	☐	☐
13	delay	Time	T#5s		☐	☐	☐	☐

Obrázek 71 - Proměnné ve funkčním bloku pro ošetření směny směru otáčení

V síti 1 (Obrázek 72) je na základě porovnání proměnných „Rot_Smer“ a „Rot_aktualni“ určeno, zda došlo ke změně směru otáčení motoru. Pokud jsou hodnoty různé, zapíše se do pomocné proměnné „pom“ hodnota 2 nebo 3 podle směru přechodu. Pokud jsou stejné, zapíše se hodnota 1 nebo 4. Tento mechanismus funguje podobně jako konstrukce „switch–case“. Díky tomu je možné přesně reagovat na různé typy změn stavu.



Obrázek 72 - FB ošetření změny směru 1

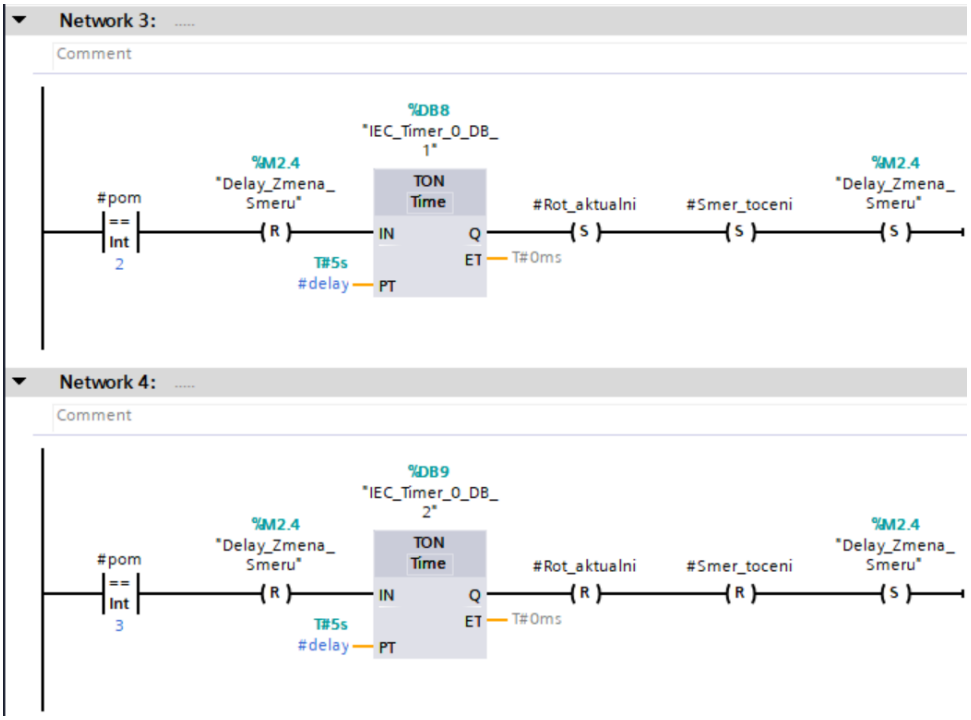
Na obrázku (Obrázek 73) jsou znázorněny sítě 3 a 4 funkčního bloku, které slouží k realizaci časového zpoždění při změně směru otáčení motoru.

Logika vychází z hodnoty pomocné proměnné „pom“, která nese číselnou informaci o aktuálním přechodu. Pokud má tato proměnná hodnotu 2 nebo 3, znamená to, že došlo ke změně

směru otáčení. V takovém případě se aktivuje příslušný časovač typu TON, který odpočítá předem definované zpoždění uložené v konstantě „delay“ (např. 5 sekund).

Po doběhnutí časovače (stav Q = TRUE) se prostřednictvím výstupních cívkových instrukcí přenastavují stavové proměnné. V programu jsou použity standardní cívky s označením „S“ a „R“, které slouží k řízení logických hodnot. Instrukce „S“ (Set) nastavuje danou proměnnou do logické hodnoty 1 (TRUE), zatímco instrukce „R“ (Reset) ji přepíná do stavu 0 (FALSE).

Pomocí těchto cívkových operací se po uplynutí zpoždění aktivuje nový směr otáčení („Rot_aktualni“), povolí se výstupní signál směru („Smer_toceni“) a zároveň se nastaví signál „Delay_Zmena_Smeru“ jako potvrzení úspěšného přepnutí, tuto proměnnou následně používáme v hlavním programu „Main“ (Obrázek 57). Současně se předchozí stavové proměnné resetují.

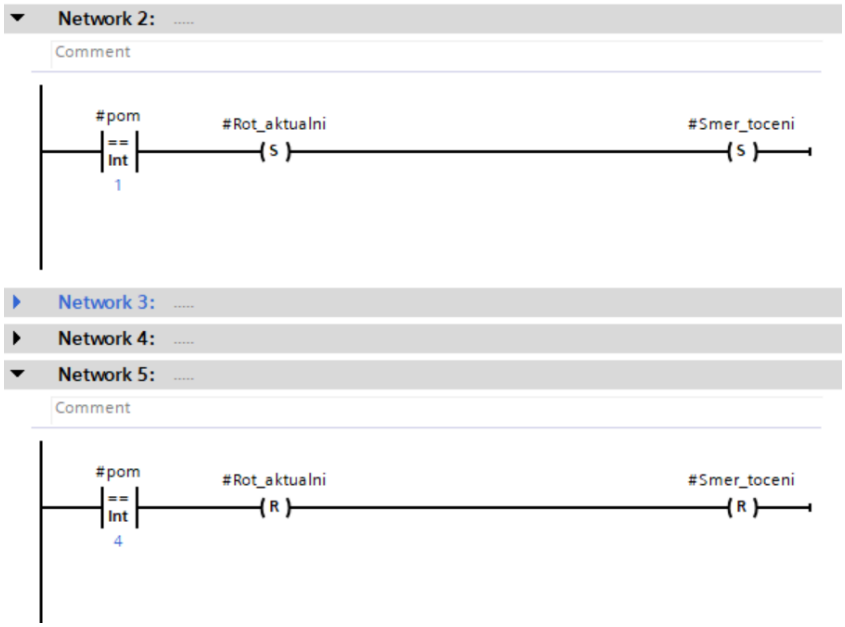


Obrázek 73 - FB ošetření změny směru 2

Síť 2 (Obrázek 74) udržuje výstupní proměnné „Rot_aktualni“ a „Smer_toceni“ ve stavu logická 1 pomocí instrukcí Set (S), pokud je hodnota pomocné proměnné „pom“ rovna 1.

Síť 5 (Obrázek 74) naopak zachovává tyto výstupy v logické 0 pomocí instrukcí Reset (R), pokud je „pom“ rovno 4.

Tímto způsobem se pouze potvrzuje stávající stav bez vyvolání změny, tedy není spuštěn časovač ani další logika – systém pokračuje ve stávajícím směru.

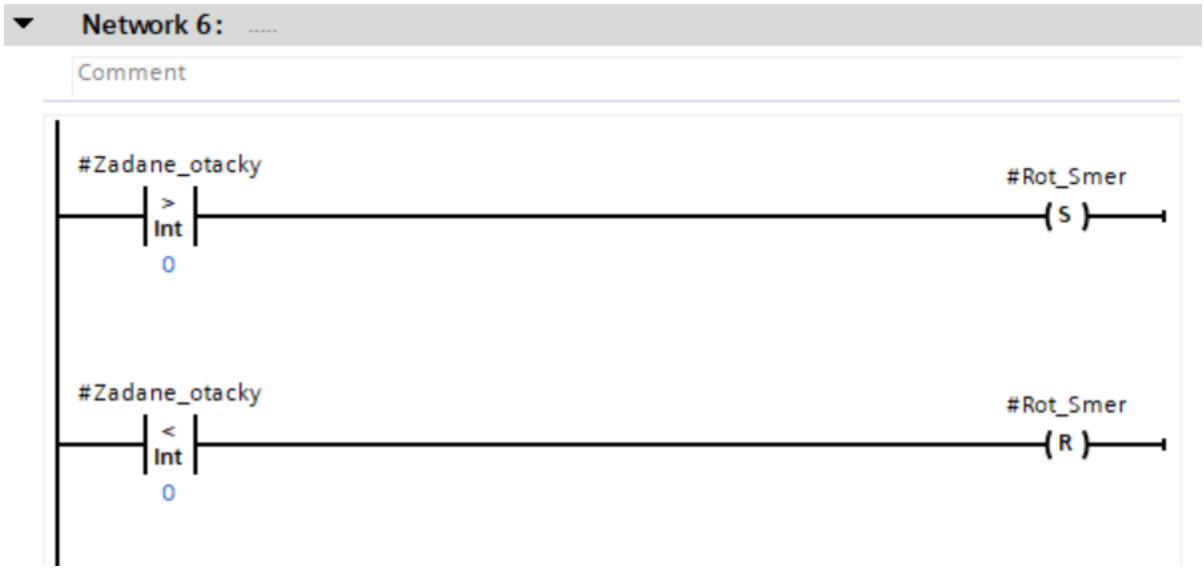


Obrázek 74 - FB ošetření změny směru 3

Na obrázku (Obrázek 75) je síť, která nastavuje hodnotu proměnné „Rot_Smer“ na základě hodnoty vstupní proměnné „Zadane_otacky“.

Pokud je hodnota „Zadane_otacky“ větší než 0, nastaví se pomocí instrukce Set (S) proměnná „Rot_Smer“ na logickou 1. Pokud je naopak menší než 0, provede se pomocí Reset (R) nastavení na logickou 0.

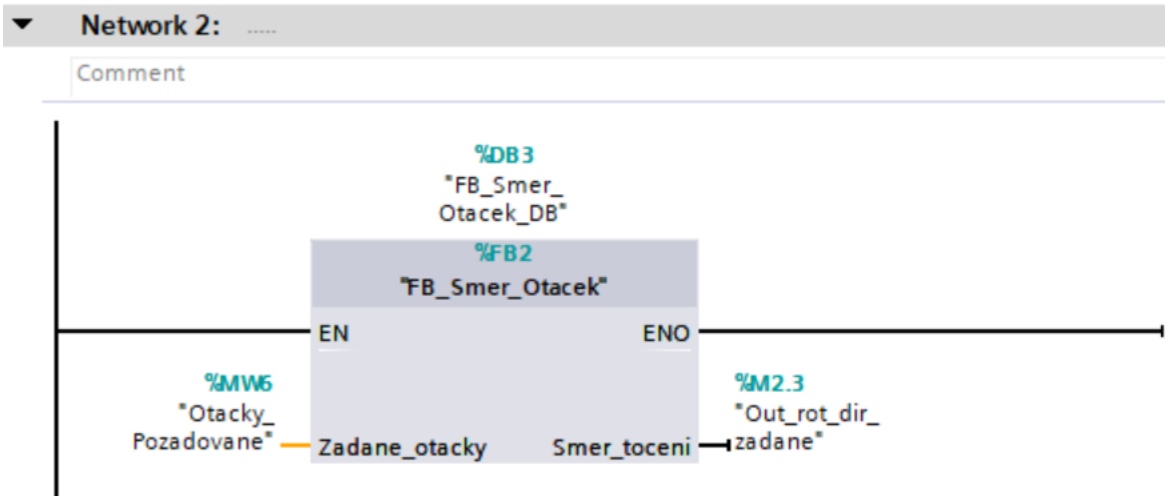
Následný stav proměnné „Rot_Smer“ je porovnáván s aktuálním směrem otáčení viz. Obrázek 72



Obrázek 75 - FB ošetření změny směru 4

Tímto krokem jsme dokončili FB, který nám ošetřuje změnu směru otáčení motoru, nyní ho obdobě jako předchozí funkční bloky vložíme do OB „Cycle Interrupt“.

Do vstupu „Zadane_otacky“ (Obrázek 76) je přivedena hodnota z proměnné „Otacky_Pozadovane“. Blok vyhodnocuje znaménko, a jeho změnu, této hodnoty a podle toho nastavuje výstupní proměnnou „Smer_toceni“, která je následně přiřazena do globální proměnné „Out_rot_dir_zadane“. Tou následně v hlavní části programu „Main“ řídíme reálnou výstupní proměnnou „Out_rot_dir“ (Obrázek 58).

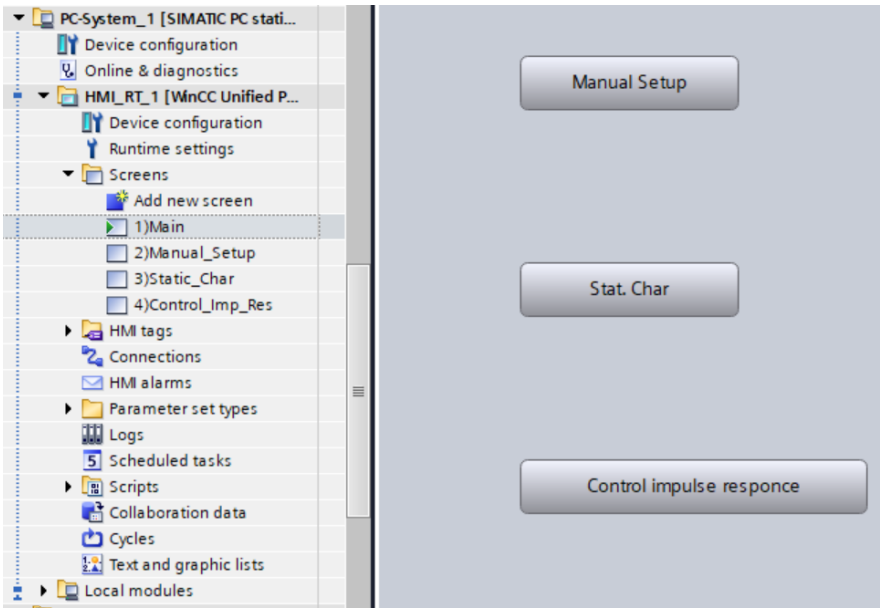


Obrázek 76 - FB_Smer_Otacek v OB Cyclic Interrupt

2.2.2 Tvorba HMI prostředí

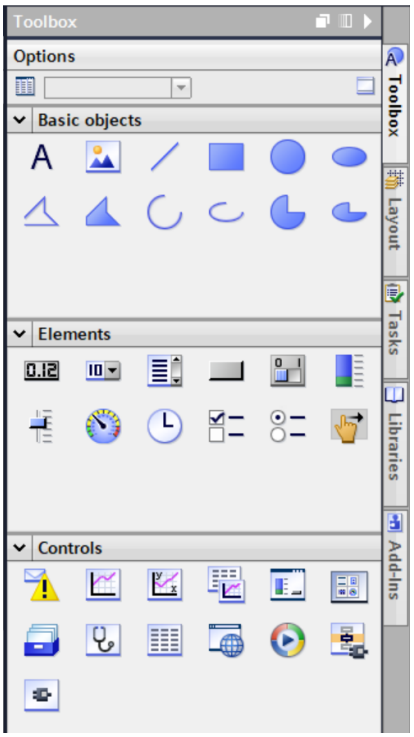
Nyní máme hotový samotný program, který nám bude řídit chod motoru, nyní musíme vytvořit uživatelské rozhraní, pomocí kterého budeme zadávat „vstupní otáčky“ a na kterém budeme zobrazovat změřené hodnoty.

Jako první si rozklikneme PC Systém v projektovém stromě a ve záložce „HMI_RT“ najdeme záložku „Screens“ (Obrázek 77). Protože budeme implementovat více úloh v jednom projektu, rozdělíme si jednotlivé úlohy na více obrazovek, mezi kterými budeme moct uživatel libovolně přepínat. Novou obrazovku přidáme kliknutím na položku „Add new screen“, po vytvoření si překontrolujeme v nastavení „Runtime settings“, že se název počáteční obrazovky shoduje se jménem obrazovky, kterou chceme mít jako startovní (Obrázek 34).



Obrázek 77 - Náhled obrazovky Main

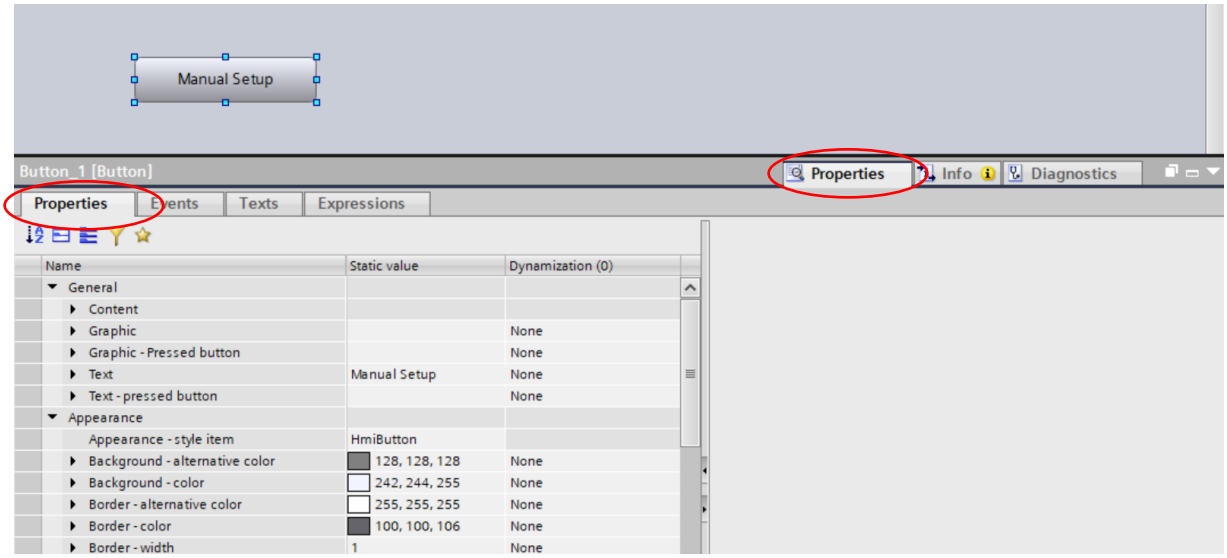
Jednotlivé prvky, jako jsou tlačítka, vstupně výstupní pole, posuvníky, text nebo pouze objekty pro stylizaci pozadí nalezneme v menu „Toolbox“ na pravé straně obrazovky (Obrázek 78). Odtud pak přesouváme jednotlivé položky myší přímo na právě upravovanou obrazovku.



Obrázek 78 – Záložka Toolbox v HMI

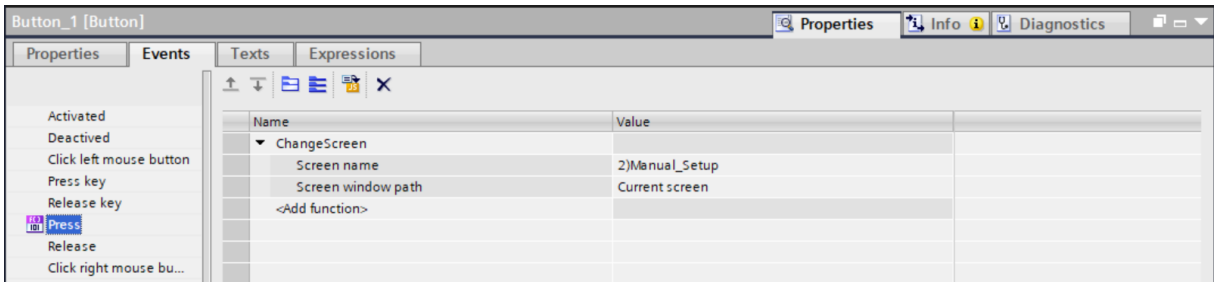
Pro přepínání obrazovek budeme používat tlačítka (Obrázek 77). Po vložení tlačítka na obrazovku „Main“ na něj klineme a ve spodní části obrazovky otevřeme položku „Properties“. Zde opět otevřeme záložku „Properties“, kde můžeme nastavit obecný vzhled tlačítka, barvu,

jeho barvu po stisknutí, text na tlačítku nebo třeba rozměry (Obrázek 79). Zde si tedy pojmenujeme tlačítko podle našeho uvážení.



Obrázek 79 - Základní konfigurace tlačítka v HMI

Když máme vzhledové parametry tlačítka nastavené přepneme do záložky „Events“ (Obrázek 80), zde nastavujeme, jaké události se budou dít například po stisku nebo po puštění tlačítka. V našem případě chceme, aby se po stisku tlačítka přepnula obrazovka. Přejdeme tedy do záložky „Press“, všechny události uvedené v této záložce se stanou, při stisku tlačítka (reakce na náběžnou hranu). Klikneme na „Add function“ a napíšeme „ChangeScreen“, poté níže napíšeme název obrazovky, na kterou chceme přepnout. Po stisku tlačítka se nám na HMI panelu změní zobrazovaná obrazovka z „Main“ na „Manual_Setup“.



Obrázek 80 - Nastavení událostí tlačítka

Budeme pokračovat na obrazovce „Manual_Setup“. Našim cílem bude vytvořit funkčně obdobné okno jako je použito v aplikaci Matlab/Simulink (Obrázek 46). Náš cíl si můžete prohlédnout na obrázku (Obrázek 47), kde jsou zachované všechny důležité fuknce.

Na obrazovce „Manual Setup“ použijeme různé vizualizační prvky pro intuitivní ovládání a sledování aktuálních hodnot motoru. Pro zadávání otáček využijeme vstupní pole, do kterého

zadáme hodnotu ručně. Alternativně můžeme otáčky nastavovat pomocí posuvníku (slideru), který umožňuje plynulé zadání pomocí myši.

Pro rychlé zastavení motoru použijeme tlačítko „STOP“, které realizujeme jako klasický prvek typu Button.

Výsledné hodnoty z enkodéru a tachogenerátoru zobrazíme ve výstupních polích. Data rozčleníme do dvou částí – v sekci „Data z Enkodéru“ zobrazíme otáčky, vstupní hodnotu enkodéru, aktuální polohu v radiánech a úhlovou rychlost. V části „Data z Tachogenerátoru“ pak zobrazíme otáčky, rychlost v radiánech za sekundu a procentuální využití.

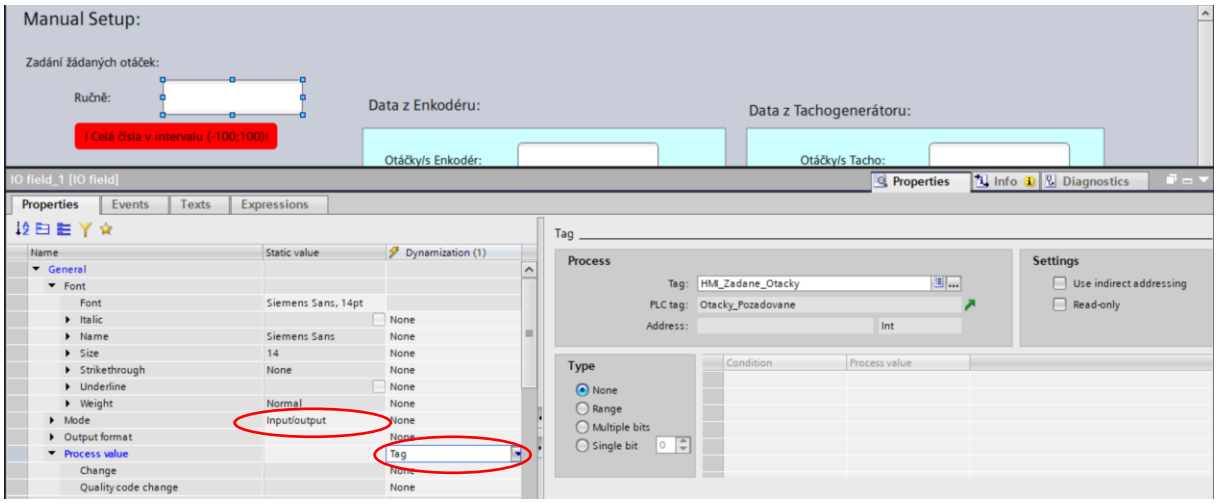
Rozložení obrazovky navrhneme tak, aby bylo přehledné a snadno pochopitelné. Umožní nám pohodlně zadávat vstupní hodnoty a zároveň sledovat odezvu systému v reálném čase.

Začneme vložením tzv. vstupně/výstupní IO pole (IO Field) pro zadávání otáček. Po vložení IO pole na obrazovku na něj klikneme a otevřeme záložku „Properties“ (Obrázek 81), podobně jako u tlačítka (Obrázek 79). Zde nastavíme mode IO pole na Input/Output, což nám umožní zadávat do pole hodnoty pomocí klávesnice.

Dále je potřeba zvolit „Process value“ (Obrázek 81), ta nám určuje, s jakou proměnnou bude daný vizualizační prvek na obrazovce pracovat – tedy jakou hodnotu bude zobrazovat nebo do ní zapisovat. Vybereme zde možnost „Tag“, po zvolení této možnosti se nám v pravé části obrazovky zobrazí konfigurační pole, kde již zadáme konkrétní proměnnou. Zadáváme však HMI tag, nikoliv PLC tag.

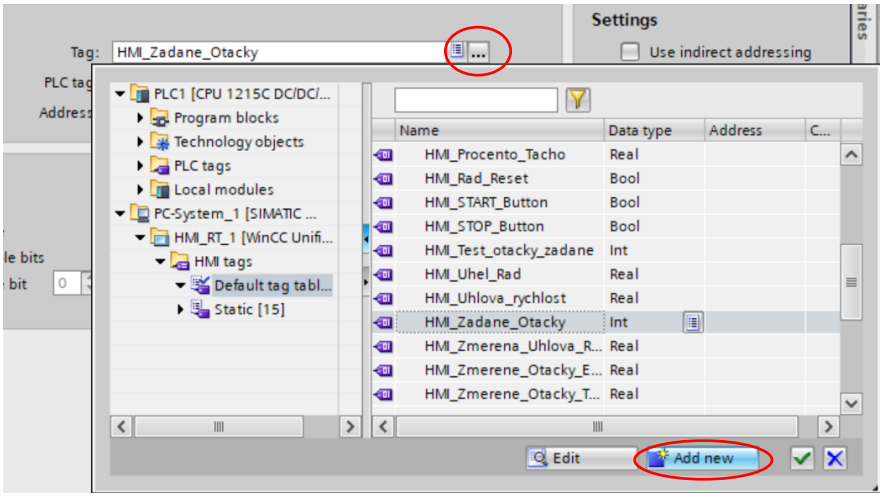
HMI tag slouží k propojení vizualizačního prvku s tagem v PLC – zajišťuje komunikaci mezi HMI panelem a řídicím systémem.

PLC tag je samotná proměnná uložená v PLC, se kterou pracuje řídicí logika. HMI tag na ni pouze odkazuje, aby mohla zobrazovat nebo měnit její hodnotu.



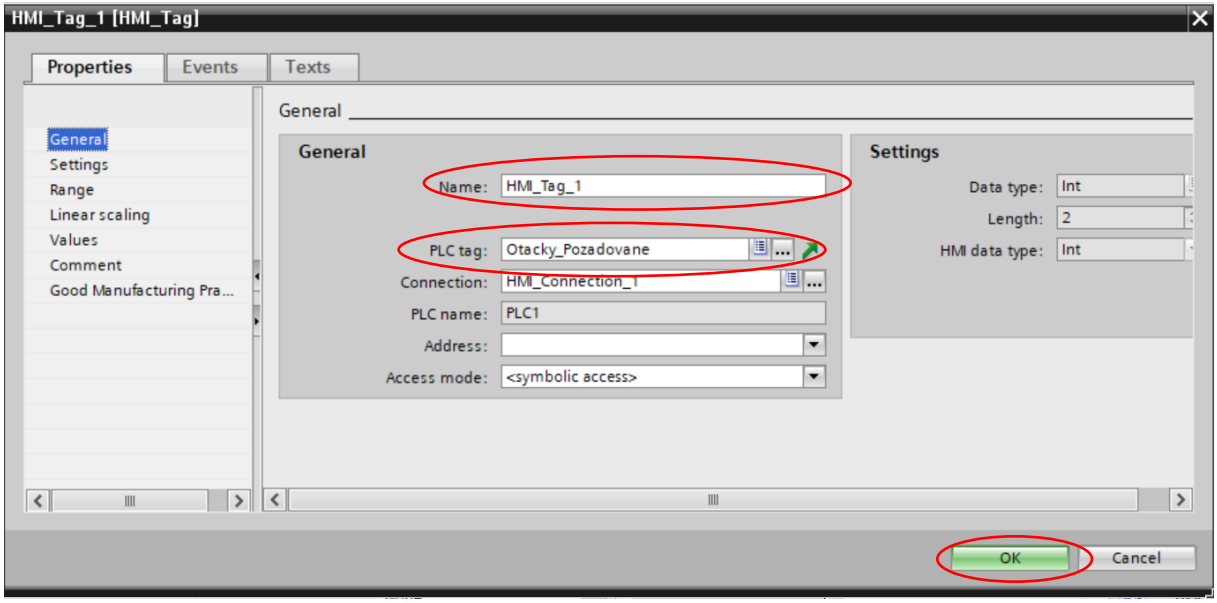
Obrázek 81 - Konfigurace IO pole pro zadávání požadovaných otáček

Vytvořit HMI tag můžeme v HMI tag tablu nebo přímo v konfiguraci (Obrázek 82), zde klikneme na tři tečky na kraji pole pro vkládání názvu HMI tagu, následně můžeme vybrat z menu existující tagy, my však zvolíme možnost „Add new“ ve spodní části.



Obrázek 82 - Vytváření HMI tagu 1

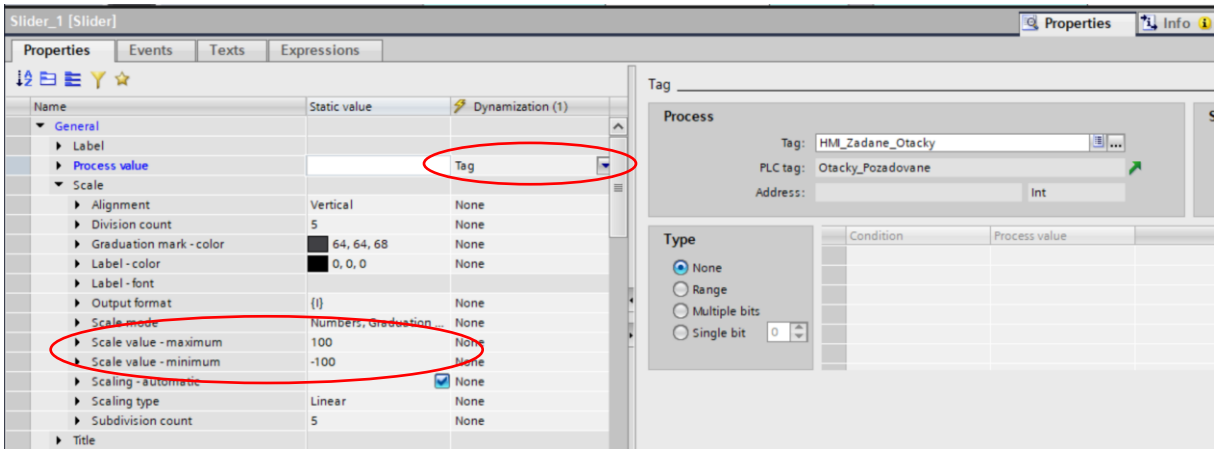
Otevře se nám okno pro definici nového HMI tagu (Obrázek 83), zde si příhodně zvolíme jeho jméno, dále zvolíme PLC tag, se kterým má být náš HMI tag propojen, zbytek okna vyplní TIA Portal automaticky. Volíme zde PLC tag „Otacky_Pozadovane“, tedy když uživatel zadá do IO pole hodnotu, vloží ji tím do HMI tagu a ten následně do PLC tagu.



Obrázek 83 - Vytváření HMI tagu 2

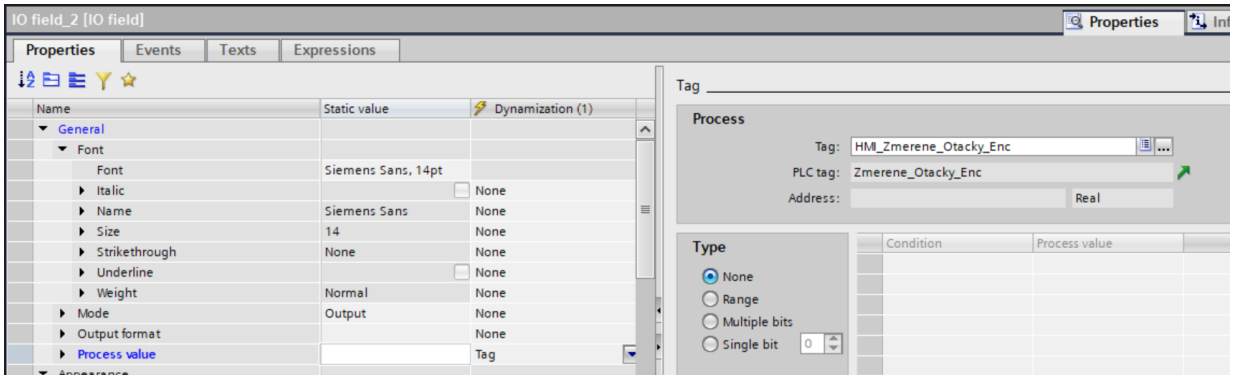
Jako další možnost zadávání otáček použijeme „Slider“, kde bude uživatel nastavovat hodnotu posouváním „jezdce“. Po jeho vložení se dostaneme stejným způsobem, jako u předchozího IO pole, do záložky „Properties“ (Obrázek 84). Zde opět jako u tlačítka zvolíme v „Process value“ možnost „Tag“ a použijeme HMI tag propojený s PLC tagem s hodnotou vstupních otáček.

Dále musíme upravit rozsah hodnot na Slideru, a to v záložce „Scale“.



Obrázek 84 - Konfigurace Slideru pro zadávání žádaných otáček

Nyní si uvedeme příklad konfigurace IO pole, které je určeno pouze pro zobrazování hodnoty tagu. Konfigurace bude velmi podobná jako u pole, kde jsme zadávali hodnotu žádaných otáček. Rozdíl bude v tom, že „Mode“ nastavíme na „Output“ (Obrázek 85). Ve vzorovém obrázku je vidět, že jsme jako „Process value“ nastavili opět „Tag“. Podle předchozího postupu (Obrázek 82,Obrázek 83) jsme si vytvořili HMI tag odpovídající našim potřebám.



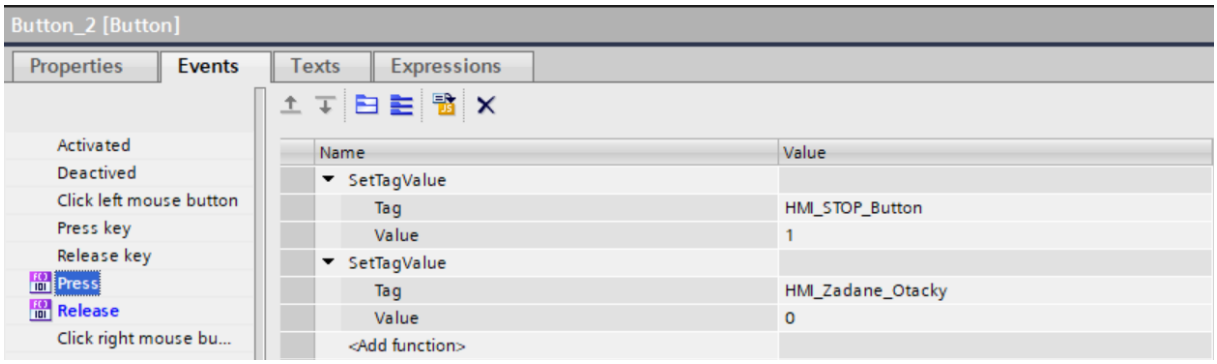
Obrázek 85 - Konfigurace IO pole pro zobrazování hodnoty

Stejně budeme postupovat i u všech ostatních IO polí, která jsou určena pro zobrazování aktuálních sledovaných veličin.

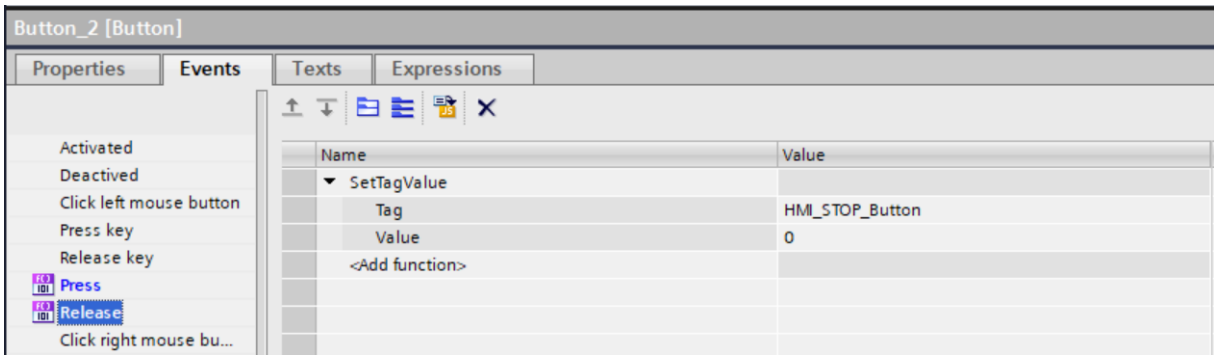
Jako další si nakonfigurujeme tlačítko STOP, to nám bude sloužit k okamžitému zastavení motoru. Po vložení tlačítka na obrazovku si můžeme změnit jeho barvu, výstižně na červenou (Obrázek 86).

Obrázek 86 - Konfigurace STOP tlačítka 1

Dále přejdeme do záložky „Events“. Zde v záložce „Press“ (Obrázek 87) přidáme událost jménem „SetTagValue“, tento příkaz nám zajistí, že při stisku tlačítka dojde k naplnění určitých proměnných požadovanými hodnotami. V našem aktivujeme proměnnou „STOP_Button“, tím deaktivujeme celý OB „Main“ (Obrázek 56), zároveň nastavíme vstupní otáčky na hodnotu „0“. V záložce „Release“ (Obrázek 88) zvolíme události, které se stanou po uvolnění tlačítka. Zde deaktivujeme „STOP_Button“, přesto se motor znovu nerozjede, protože při stisku jsme nastavili žádané otáčky na „0“.



Obrázek 87 - Konfigurace STOP tlačítka 2

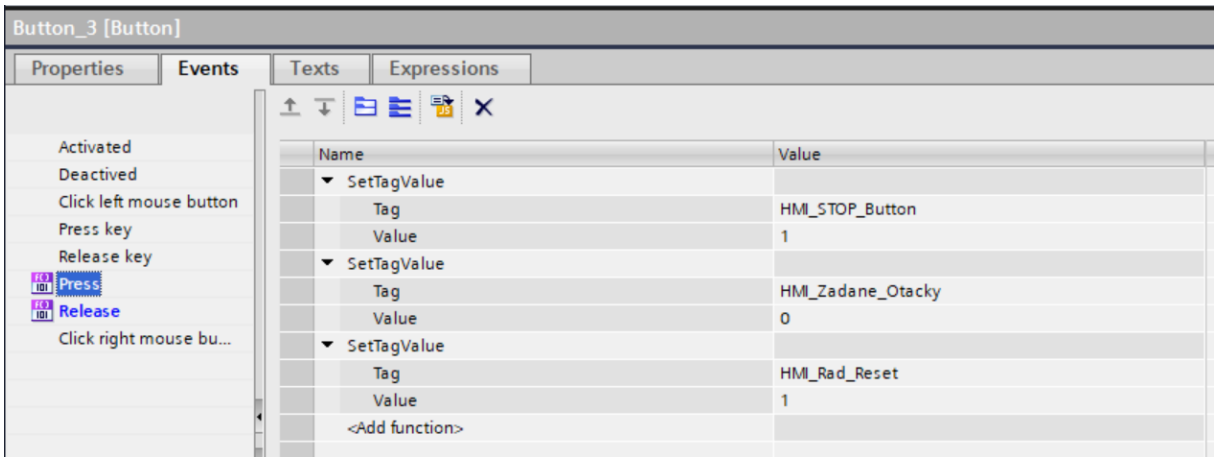


Obrázek 88 - Konfigurace STOP tlačítka 3

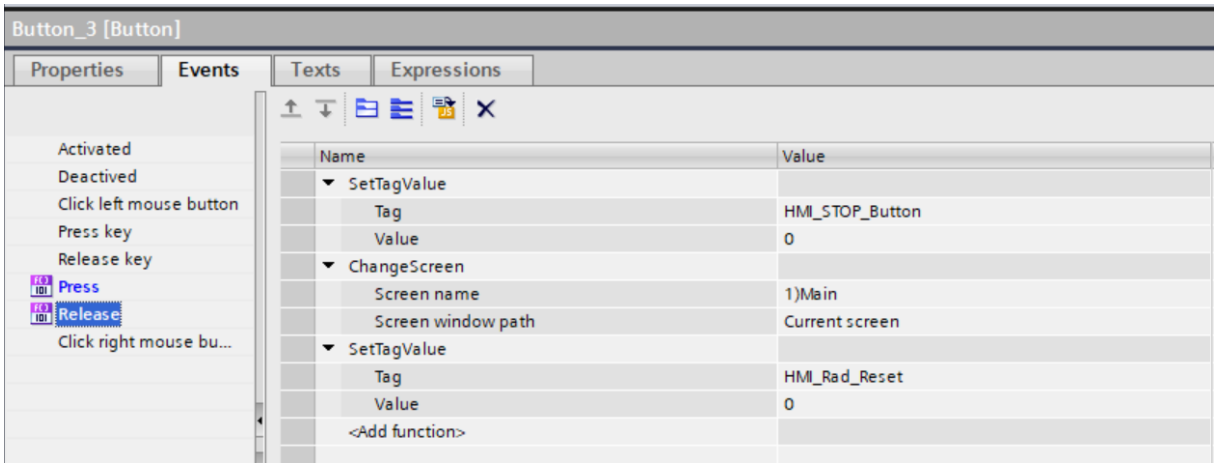
Jako poslední si vytvoříme tlačítko „Zpět“. Toto tlačítko bude uživatele vracet na výchozí obrazovku, zároveň bude zastavovat běžící motor a nulovat aktuální hodnoty proměnných, které by po opětovném návratu byly nenulové, jako je například poloha v radiánech.

Při stisku tlačítka se aktivuje STOP tlačítko, vynuluje žádaná hodnota a aktivuje proměnná pro nulování úhlů (Obrázek 89).

Po uvolnění tlačítka se hodnoty předem zmíněných proměnných vrátí do nuly, a pomocí „ChangeScreen“ se vrátíme zpět na obrazovku „Main“ (Obrázek 90).



Obrázek 89 - Konfigurace Zpět tlačítka 1

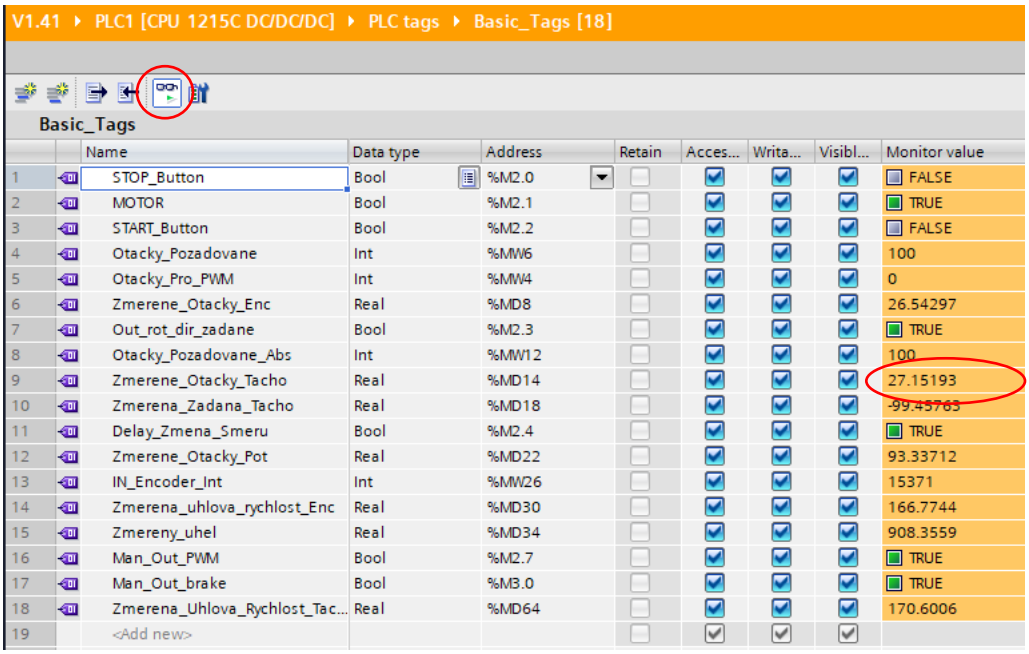


Obrázek 90 - Konfigurace Zpět tlačítka 2

Nyní si můžeme vyzkoušet funkčnost našeho řešení.

Jak bylo zmíněno v předchozích částech úlohy, nyní si experimentálně ověříme, že jsme maximální hodnotu pro škálování pro výpočet procenta z rychlosti tachogenerátoru určili správně. Rozklikneme si, proto seznam tagů, kde máme uložený tag s hodnotou rychlosti. V horní části obrazovky se přepneme do online módu, kliknutím na ikonu „Go online“. Online mód v TIA Portalu slouží k připojení k PLC v reálném čase. Umožňuje nám sledovat aktuální hodnoty proměnných, diagnostikovat chyby, testovat program a ověřovat funkčnost bez nutnosti opakovaného stahování do PLC.

Poté klikneme na ikonu „Monitor all“ (Obrázek 91), žlutě se nám zobrazí aktuální hodnota každého tagu. Pomocí HMI panelu nastavíme maximální hodnotu otáček. Následně hodnotu odečteme a patřičně upravíme program (Obrázek 67).



	Name	Data type	Address	Retain	Acces...	Writa...	Visibl...	Monitor value
1	STOP_Button	Bool	%M2.0		☑	☑	☑	FALSE
2	MOTOR	Bool	%M2.1		☑	☑	☑	TRUE
3	START_Button	Bool	%M2.2		☑	☑	☑	FALSE
4	Otacky_Pozadovane	Int	%MW6		☑	☑	☑	100
5	Otacky_Pro_PWM	Int	%MW4		☑	☑	☑	0
6	Zmerene_Otacky_Enc	Real	%MD8		☑	☑	☑	26.54297
7	Out_rot_dir_zadane	Bool	%M2.3		☑	☑	☑	TRUE
8	Otacky_Pozadovane_Abs	Int	%MW12		☑	☑	☑	100
9	Zmerene_Otacky_Tacho	Real	%MD14		☑	☑	☑	27.15193
10	Zmerena_Zadana_Tacho	Real	%MD18		☑	☑	☑	-99.45763
11	Delay_Zmena_Smeru	Bool	%M2.4		☑	☑	☑	TRUE
12	Zmerene_Otacky_Pot	Real	%MD22		☑	☑	☑	93.33712
13	IN_Encoder_Int	Int	%MW26		☑	☑	☑	15371
14	Zmerena_uhlova_rychlost_Enc	Real	%MD30		☑	☑	☑	166.7744
15	Zmereny_uhel	Real	%MD34		☑	☑	☑	908.3559
16	Man_Out_PWM	Bool	%M2.7		☑	☑	☑	TRUE
17	Man_Out_brake	Bool	%M3.0		☑	☑	☑	TRUE
18	Zmerena_Uhlova_Rychlost_Tac...	Real	%MD64		☑	☑	☑	170.6006
19	<Add new>				☑	☑	☑	

Obrázek 91 – Funkce monitoring

2.3 Příprava úlohy Control impulse responce

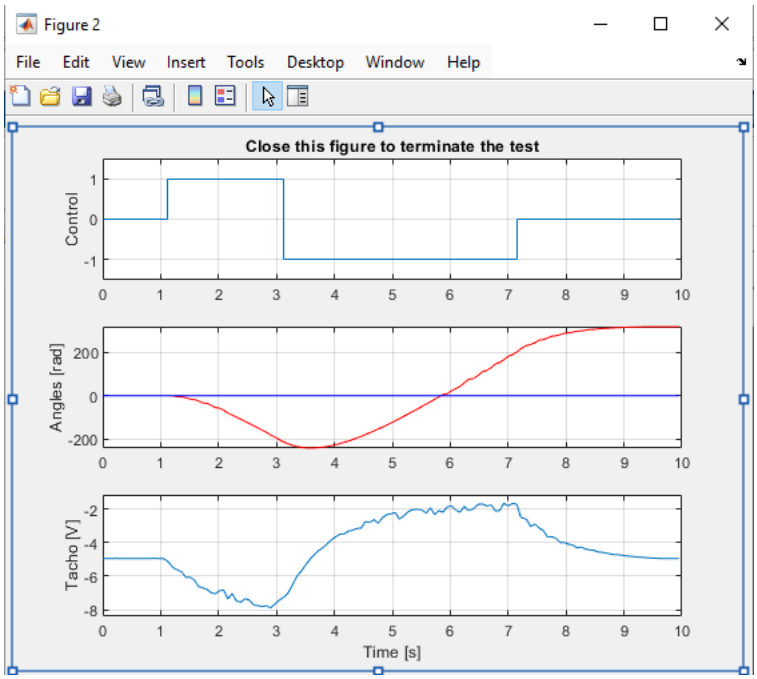
Úloha „Control Impulse Responce“, dostupná v prostředí MATLAB pod záložkou Basic Tests, slouží k analýze dynamické odezvy servo systému na skokovou změnu řídicího signálu. Během testu program aplikuje do systému dvě úrovně řídicího signálu, a to −1 a +1 a sledujeme chování motoru.

Výsledkem testu jsou tři grafy (Obrázek 92). První graf („Control“) zobrazuje průběh aplikovaného řídicího signálu v čase. Druhý graf („Angles [rad]“) ukazuje změnu úhlu natočení motoru v radiánech jako reakci na řízení. Třetí graf („Tacho [V]“) znázorňuje napětí z tachogenerátoru, které odpovídá aktuální rychlosti otáčení motoru.

Tento test nám umožňuje posoudit základní dynamické vlastnosti systému, jako je odezva na změnu směru, rychlost náběhu či stabilita motoru při změnách řídicího signálu. Úloha slouží jako základní diagnostický nástroj před prováděním složitějších experimentů.

V námi připravované úloze v TIA Portalu však nebudeme měřit napětí z tachogenerátoru přímo, ale zaměříme se pouze na zpracování hodnot otáček a úhlové rychlosti.

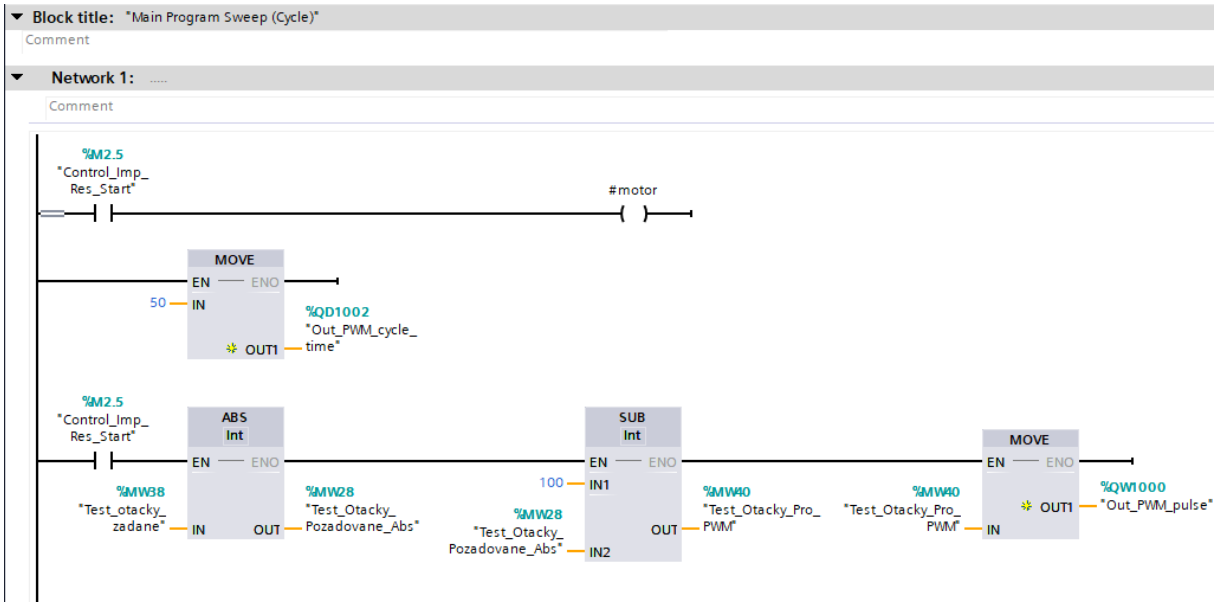
Jak bylo popsáno v úvodu pro tvorbu úloh, tato úloha se bude zabývat tvorbou programu, který poběží automaticky, bez zásahu uživatele. Hlavní cíle úlohy jsou naučit se používat a konfigurovat položku „Trend Control“ v HMI prostředí a používat funkci „Trace“ pro následný zisk naměřených dat v TIA Portalu.



Obrázek 92 - Graf změřených hodnot reakce na jednotkový skok v Matlabu

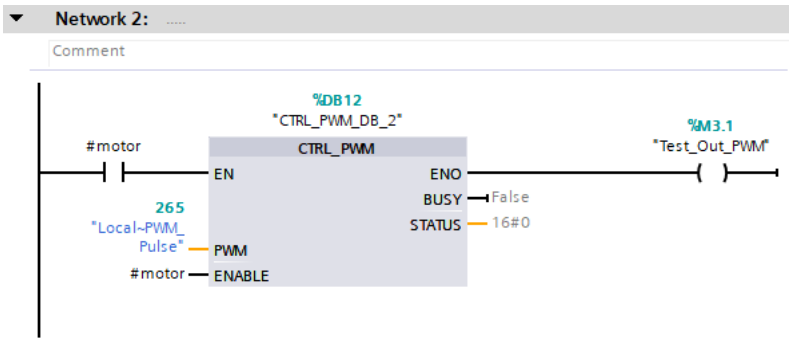
Začneme vytvořením organizačního bloku (OB), postupujeme obdobě jako u vytváření OB „Main“ (Obrázek 51), tedy v projektovém stromě v záložce „Program blocks“, zvolíme „Add new blok“. V nově otevřeném okně vybereme možnost „Organization block“, typu „Program cycle“. Programovací jazyk ponecháme LAD.

Vytvořený programový cyklus si otevřeme a vytvoříme první část programu (Obrázek 93). Je stejný jako u první úlohy (Obrázek 56), jen je spouštěný jinou proměnnou, využíváme zde i nové proměnné pro zpracování vstupních otáček. Tyto proměnné si pro větší přehlednost vložíme do odděleného seznamu tagů.



Obrázek 93 - Control Impulse Responce – Vytváření programu 1

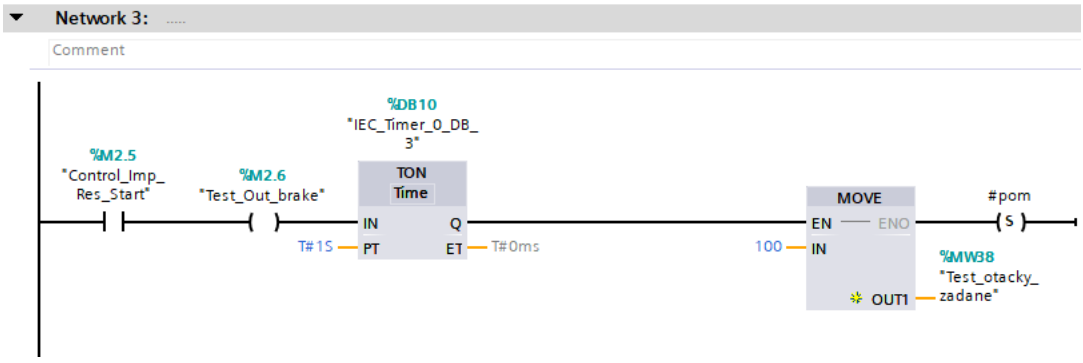
Dále opět vložíme funkční blok „CTRL_PWM“ (Obrázek 94), oproti první úloze (Obrázek 57) se v této nebudeme zabývat ošetřením při změně směru otáček, zde je našim cílem zaznamenat okamžitou reakci. Také výstupní brzdu budeme ovládat až dále v programu. Proměnnou „Test_Out_PWM“ používáme v programu „Main“ (Obrázek 58), kde nám povoluje samotný výstup.



Obrázek 94 - Control Impulse Responce – Vytváření programu 2

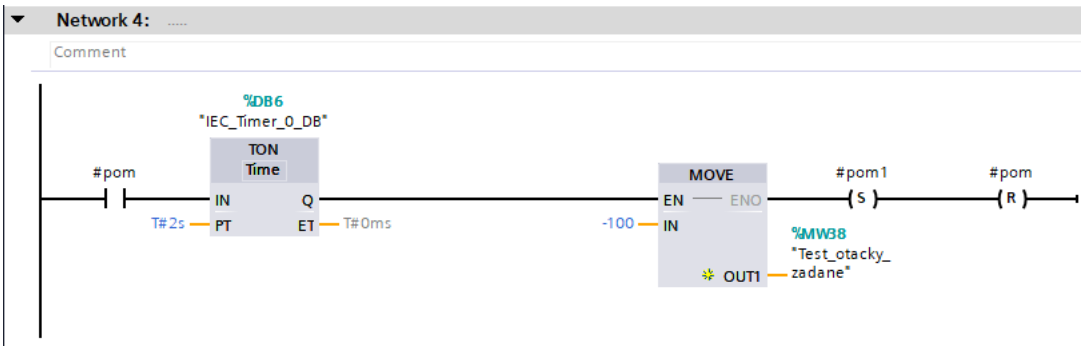
V síti 3 (Obrázek 95) používáme stejnou proměnnou jako v síti 1 (Obrázek 93) pro spuštění celého programu. Po přechodu proměnné „Control_Imp_Res_Start“ do logické 1 povolíme uvolnit elektromagnetickou brzdu motoru nastavením proměnné „Test_Out_Brake“, tu opět následně použijeme v OB „Main“ (Obrázek 58).

Následně spustíme časovač typu TON, reaguje na náběžnou hranu vstupního signálu, je nastavený na čas 1 sekunda, vycházíme zde z grafu získaného z Matlabu (Obrázek 92), po tuto dobu jsou vstupní otáčky rovny 0. Po uplynutí času se vstupní otáčky změní na 100 % neboli +1. Zároveň nastavujeme lokální proměnnou „pom“ do logické 1, tím řídíme pokračování chodu programu do další sítě (Obrázek 96).



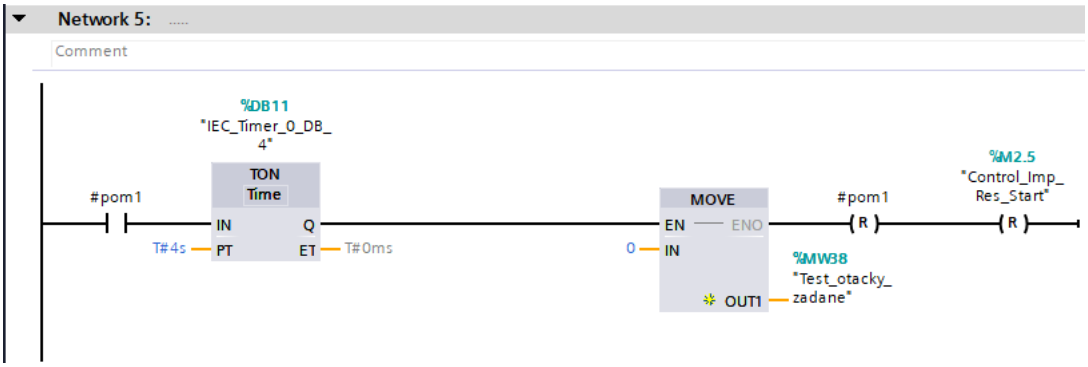
Obrázek 95 - Control Impulse Responce – Vytváření programu 3

Zde (Obrázek 96) opět pomocí časovače typu TON vytvoříme zpoždění, po jeho dobu zůstává vstupní hodnota „+1“. Po uplynutí času 2 sekundy, opět jsme vycházeli z grafu viz. Obrázek 92, změníme vstupní otáčky na -100 %, tedy „-1“. zároveň řídíme běh programu pomocí lokálních proměnných.



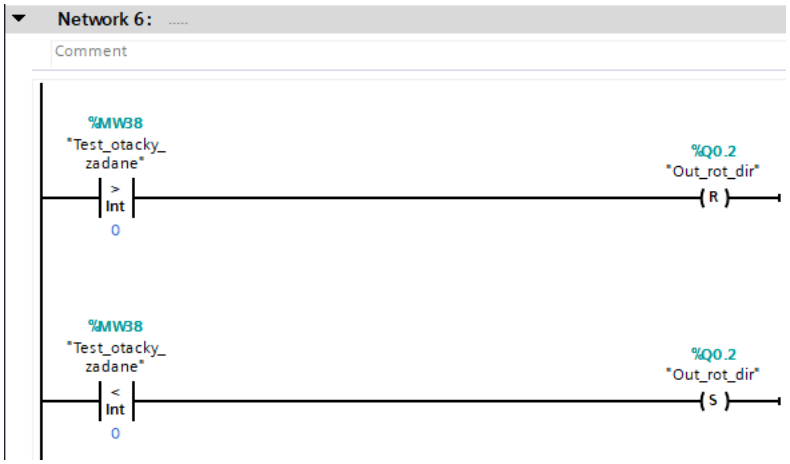
Obrázek 96 - Control Impulse Responce – Vytváření programu 4

V následující části (Obrázek 97) opět ponecháme vstupní hodnotu na -100 % po dobu vyčtenou z grafu naměřených hodnot z Matlabu (Obrázek 92). Poté nastavíme žádanou hodnotu otáček na 0.



Obrázek 97 - Control Impulse Responce – Vytváření programu 5

Jako poslední si v síti 6 (Obrázek 98) vytvoříme podmínky, které nám bude ovládat směr točení motoru na základě znaménka vstupních otáček.

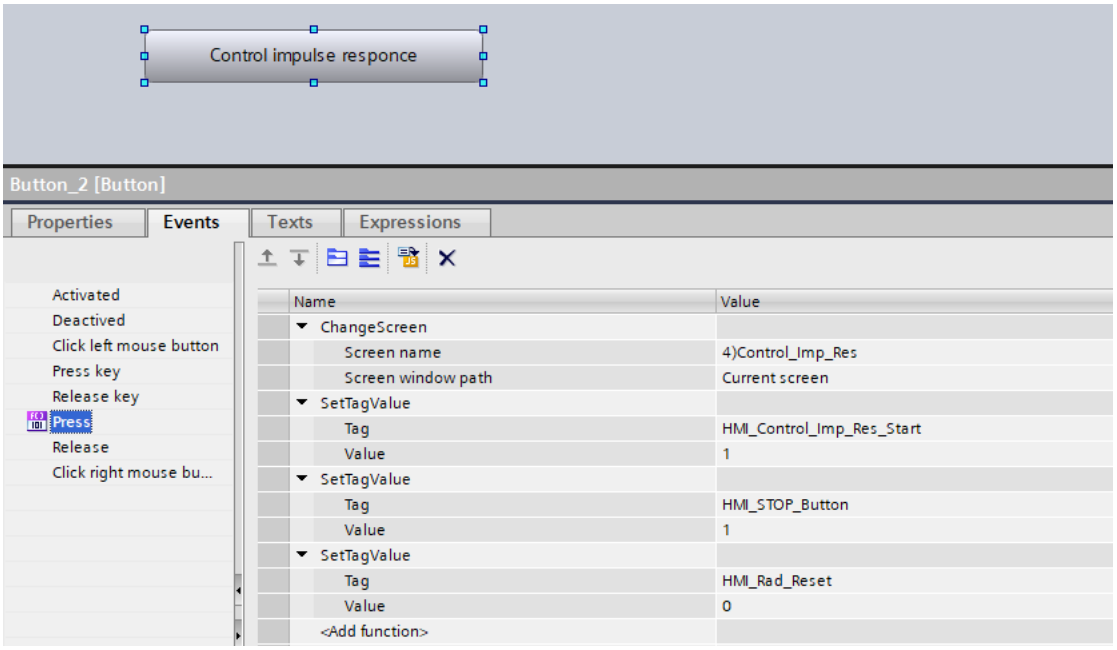


Obrázek 98 - Control Impulse Responce – Vytváření programu 6

Nyní, když máme programovou část vytvořenou se přesuneme do HMI prostředí, kde budeme samotný program spouštět a živě zobrazovat hodnoty do grafu.

V projektovém stromě si otevřeme PC Systém, v něm „HMI_RT“ prostředí, zde si otevřeme záložku „Screens“ (Obrázek 77) a vytvoříme si novou obrazovku, kterou si příhodně pojmenujeme.

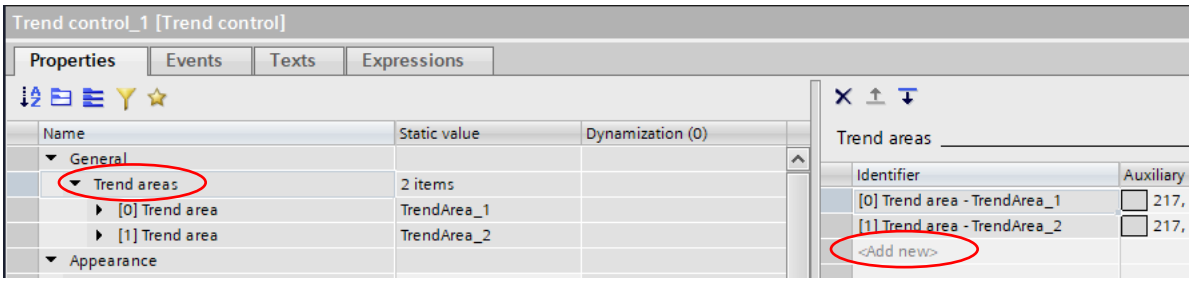
Na „Main“ obrazovku si vložíme další tlačítko, kterým budeme spouštět program (Obrázek 99), přepínat zobrazovanou obrazovku a nastavovat další potřebné proměnné, jako je STOP tlačítko, které nám zabrání spuštění nežádoucí části programu.



Obrázek 99 - Konfigurace tlačítka Control impulse respoce na obrazovce Main v HMI

Přejdeme na obrazovku, kterou jsme si v minulosti vytvořili pro zobrazení „trendu“ proměnných „Control_Imp_Res“. Zde vložíme prvek „Trend Control“, otevřeme si jeho „Properties“.

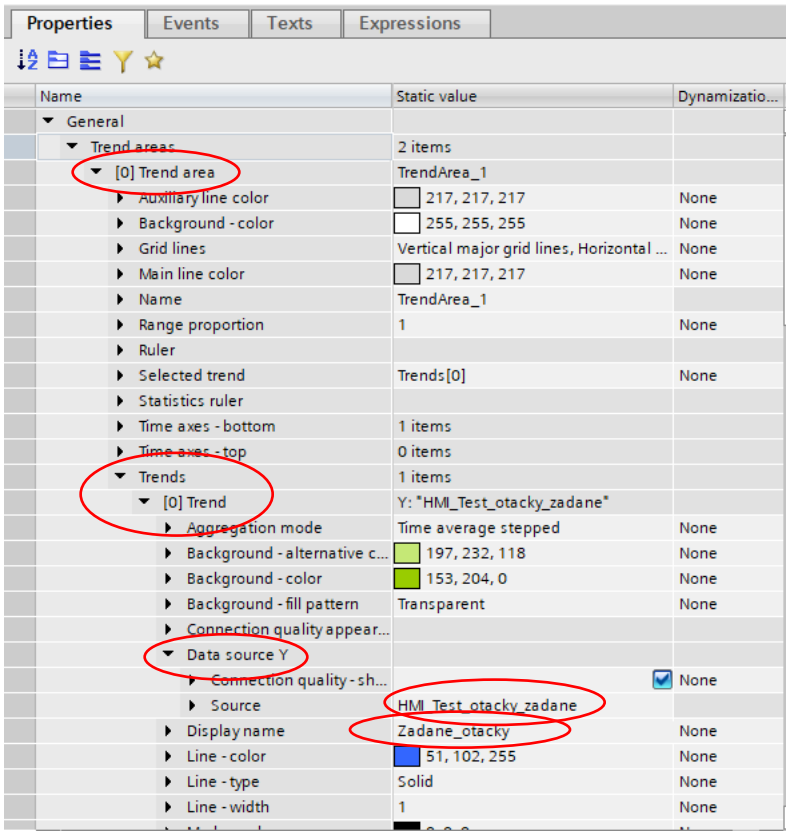
Budeme chtít zobrazovat dva průběhy najednou, proto jako první v záložce „General“ klikneme na „Trend areas“, v pravé části okna se nám otevře seznam trendů, zde klikneme na pole „Add new“ pro vložení nového trendu (Obrázek 100).



Obrázek 100 - HMI – vkládání nového trendu do Trend control

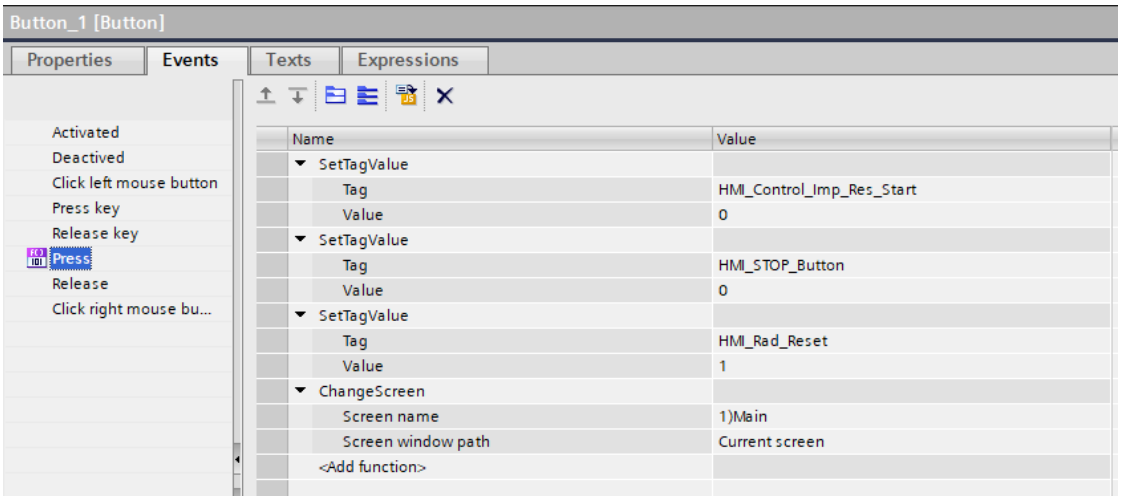
Dále rozklikneme oblast trendu (Obrázek 101), otevřeme záložku „Trends“, zde můžeme zvolit barvu zobrazovaného trendu, jeho název, ale hlavně zdroj dat pro vykreslení, a to v záložce

„Data source Y“, kde, jakou zdroj (Source) zvolíme HMI tag, který chceme vykreslit. Obdobně postupujeme i u druhého trendu. Jako první trend si zvolíme vstupní hodnotu otáček a jako druhý trend si zvolíme polohu v radiánech.



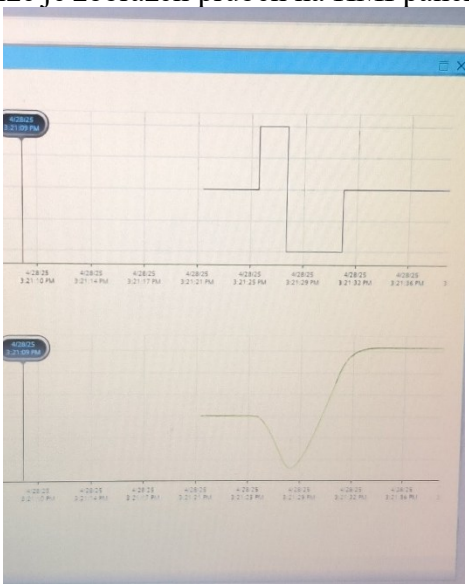
Obrázek 101 - HMI – Nastavení trendu v Trend control

Jako poslední si na této obrazovce vložíme a nakonfigurujeme tlačítko „Zpět“ (Obrázek 102), které bude uživatele vracet zpět na výchozí obrazovku, zároveň upraví stavy hodnot kontrolující běh programu a v poslední řadě aktivuje proměnnou, která nám resetuje stav napočítané polohy v Radiánech. Tuto proměnnou požíváme ve funkčním bloku pro výpočet rychlosti z dat získaných v enkodéru (Obrázek 64).



Obrázek 102 - HMI – Konfugurace Zpět tlačítka pro Control impulse responce

Na obrázku (Obrázek 103) níže je zobrazen průběh na HMI panelu pomocí „Trend control“



Obrázek 103 - Výsledek Control impulse responce na HMI panelu

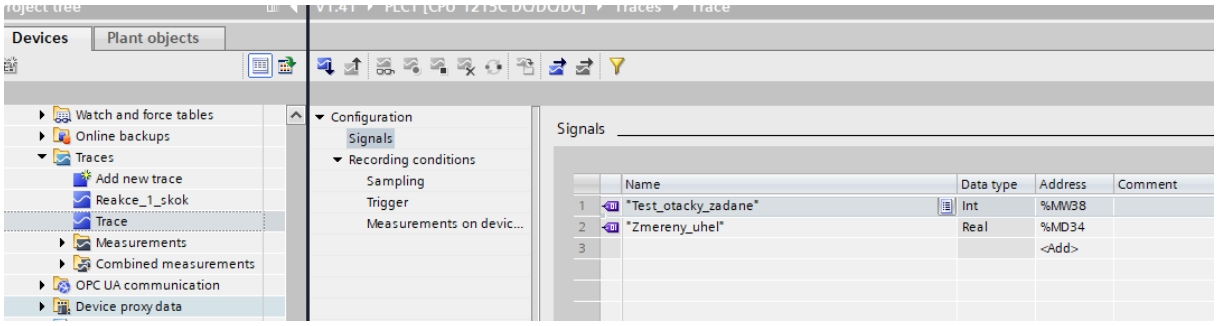
Pro zisk dat pro následné zpracování dat pomocí funkce „Traces“

Funkce Traces v prostředí TIA Portal slouží k záznamu, analýze hodnot a následnému uložení vybraných signálů během běhu programu v PLC. Umožňuje nám sledovat průběhy proměnných v čase podobně jako na osciloskopu.

Pomocí nástroje Trace si vybereme konkrétní signály, které chceme monitorovat. Definujeme podmínky vzorkování, spouštěcí podmínky (trigger) a následně během testu zaznamenáme změny hodnot těchto proměnných.

V projektovém stromě rozklikneme záložku „Traces“ (Obrázek 104), zde vložíme nový průběh kliknutím na „Add new trace“, ten si pro přehlednost příhodně pojmenujeme.

V záložce „Configuration“ (Obrázek 104), vybereme z proměnných signály, které chceme sledovat a vložíme je do seznamu.



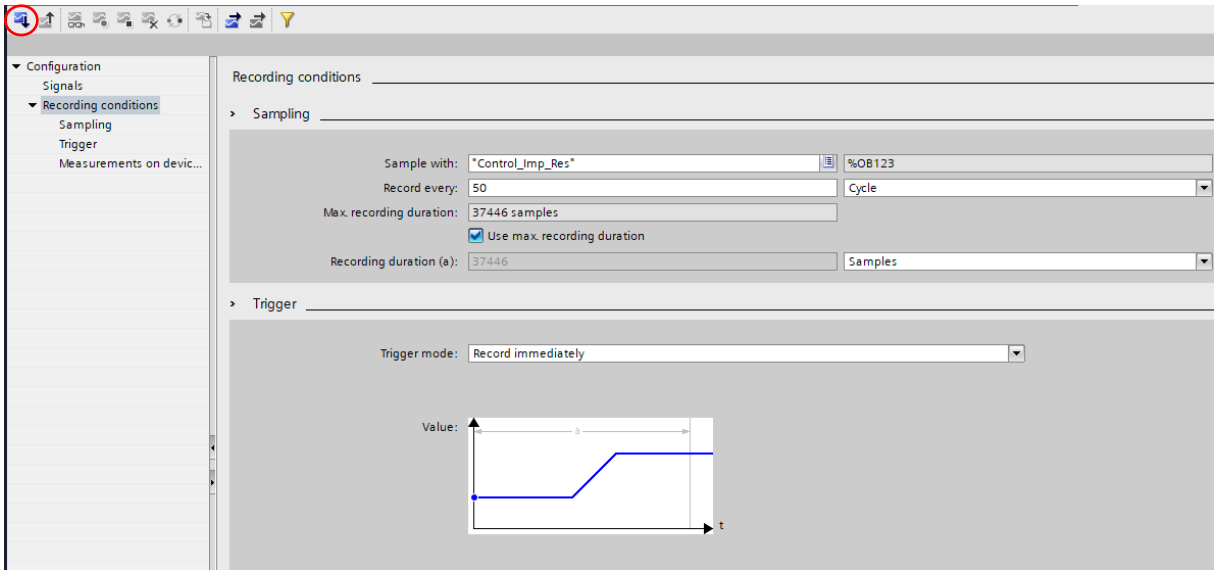
Obrázek 104 - Vkládání a konfigurace Trace

Dále nastavíme podmínky pro nahrávání průběhu (Obrázek 105).

V části „Sampling“ nastavujeme parametry vzorkování. V poli „Sample with“ vybíráme blok, se kterým bude záznam synchronizován, v tomto případě „Control_Imp_Res“, OB, kde je uložen program. V poli „Record every“ nastavujeme, že hodnoty budou zaznamenávány každých 50 cyklů (průchodů hlavního programového cyklu CPU). Čím nižší číslo nastavíme, tím častěji se budou vzorky ukládat a záznam bude podrobnější. Čím vyšší číslo, tím řidší bude vzorkování.

V části „Trigger“ nastavujeme způsob spuštění záznamu. V tomto případě je zvolen režim „Record immediately“, což znamená, že záznam začne ihned po spuštění měření, bez čekání na specifickou událost.

Takto nastavený Trace nahrajeme do PLC pomocí ikony v levém horním rohu (Obrázek 105) „Transfer trace configuration to device“.

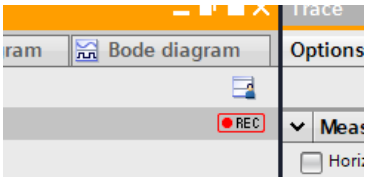


Obrázek 105 - Trace – Konfigurace

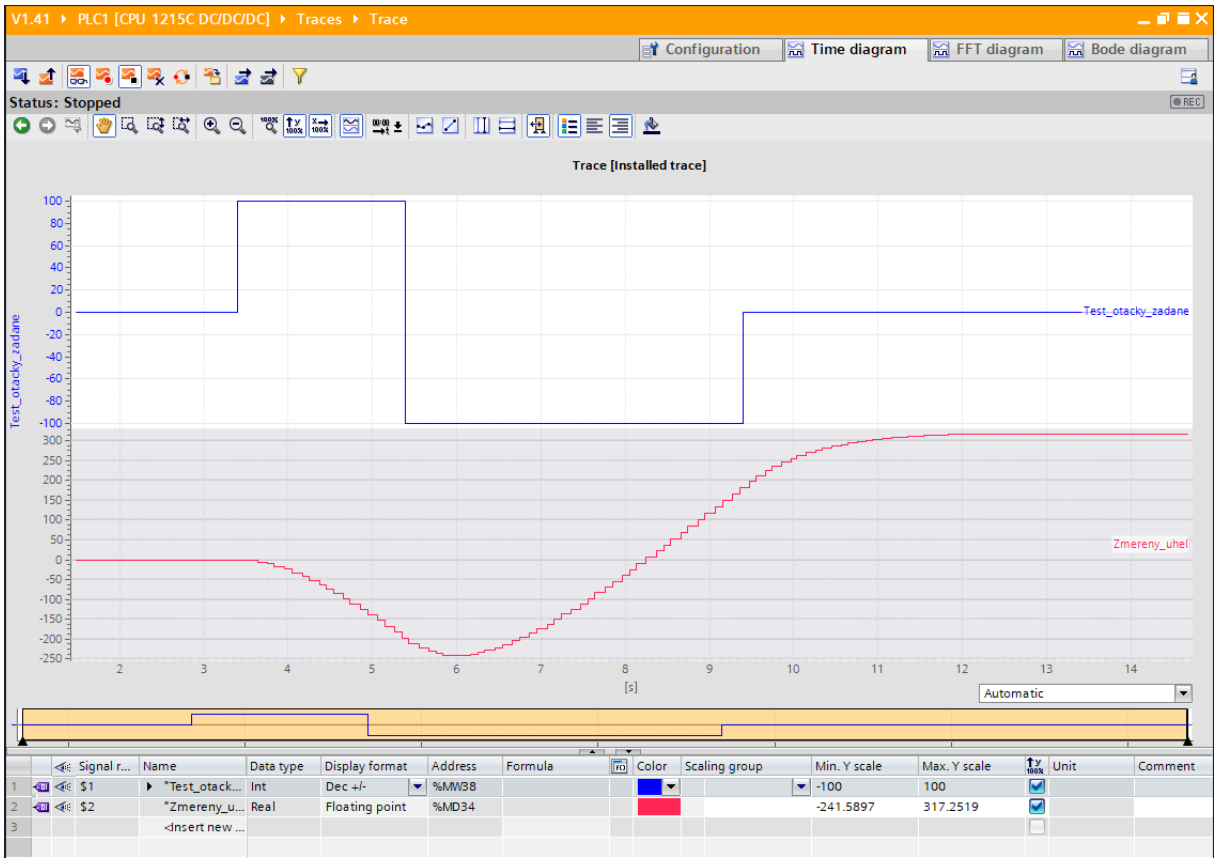
Po nahrání našeho Trace do PLC si otevřeme záložku „Time diagram“ (Obrázek 108). Zde vidíme prostor, pro vykreslení našeho grafu, zároveň nás bude zajímat ikonové menu v levém horním rohu (Obrázek 106). Jako první zde aktivujeme funkci „observe“. Dále zapneme

nahrávání pomocí ikony „Activate recording“. Po spuštění nahrávání se nám zobrazí „REC“ (Obrázek 107) v pravém horním rohu obrazovky, v tuto chvíli spustíme program, který chceme měřit, v našem případě kliknutím na HMI tlačítko „Control imulse responce“ na obrazovce „Main“. Počkáme na ustálení hodnot a ukončíme nahrávání kliknutím na „Deactivate recording“. Nyní máme změřeno, zbývá jen exportovat data kliknutím na „Export measurement with the settings from the current view“. Data si uložíme do příslušného adresáře ve formátu .csv. Po uložení dat vymažeme aktuální průběh z úložiště zařízení kliknutím na „Delete trace from the device“.

Obrázek 106 – Trace – menu kontroly nahrávání



Obrázek 107 – Trace – Recording



Obrázek 108 – Trace – Výsledek

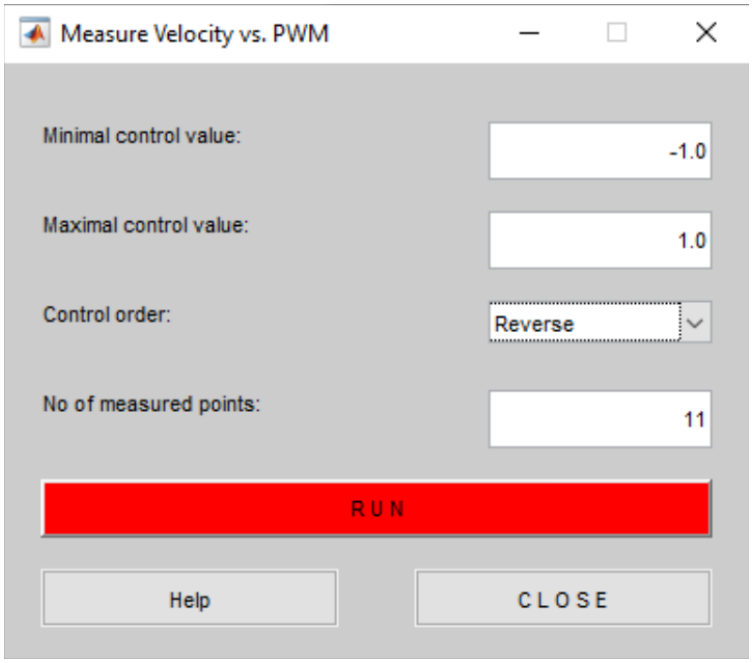
2.4 Příprava úlohy pro měření statické charakteristiky

V této úloze se zaměříme na měření statické charakteristiky DC motoru, konkrétně na vztah mezi velikostí řídicího PWM signálu a výslednou rychlostí otáčení. Stejně jako u předchozích úlohy vycházíme z předpřipravených úloh v aplikaci Matlab/Simulink (Obrázek 5).

V dialogovém okně (Obrázek 109) zadáváme minimální a maximální hodnotu vstupního signálu (typicky −1.0 až 1.0), počet měřených bodů a režim průběhu testu. V nabídce Control order volíme směr průběhu testu:

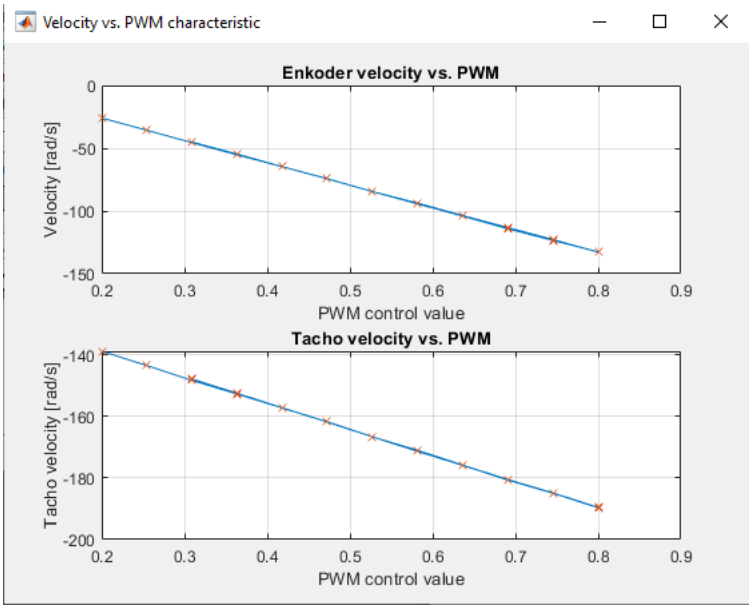
- Ascending – PWM signál roste od minimální po maximální hodnotu,
- Descending – signál klesá od maximální po minimální,
- Reverse – signál prochází z minima do maxima a následně zpět do minima (symetricky).

Po spuštění měření tlačítkem RUN je každá hodnota PWM aplikována na motor, vyčkává se na ustálení rychlosti a následně se zaznamená výstupní hodnota otáček.



Obrázek 109 - Matlab – Okno pro zadání parametrů měření stat. char.

Výsledkem měření je grafická závislost, kterou vidíme na obrázku (Obrázek 110). První graf (Enkoder velocity vs. PWM) zobrazuje rychlost motoru v radiánech za sekundu snímanou enkodérem. Druhý graf (Tacho velocity vs. PWM) ukazuje rychlost vypočtenou z výstupního napětí tachogenerátoru. Oba grafy potvrzují téměř lineární závislost mezi PWM signálem a výstupní rychlostí.

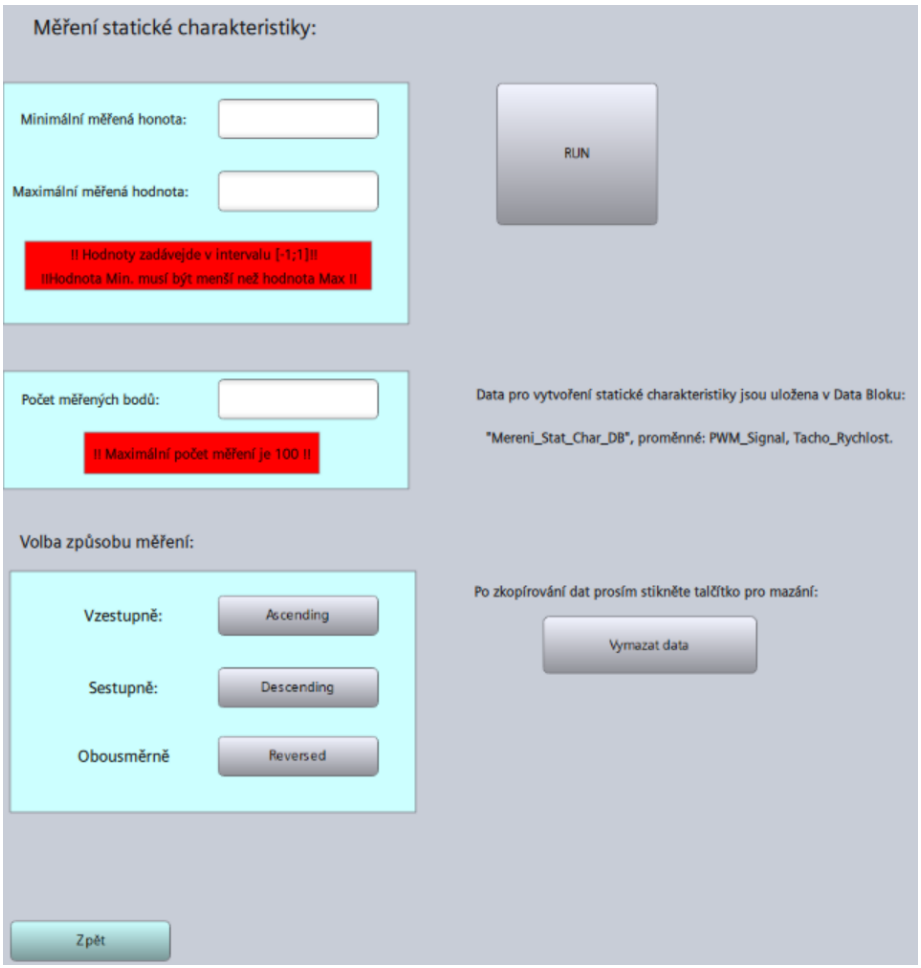


Obrázek 110 - Matlab – průběh měření stat. char. (0,2;0,8;10)

Pro účely porovnání budeme dále pracovat pouze s hodnotami rychlosti z tachogenerátoru. Tyto hodnoty vykazovaly stabilní průběh a byly v souladu s výsledky měření v prostředí TIA Portalu. Díky své plynulosti a spolehlivosti poskytl tachogenerátor dostatečně přesný obraz statické charakteristiky motoru.

Cíl úlohy je vytvořit program, kde bude ošetřeno zadávání vstupních hodnot, programový cyklus bude ošetřen na základě počtu měření a bude prováděn zápis naměřených hodnot v určitých fázích chodu programu.

Dále vidíme cíl vizualizace HMI prostředí (Obrázek 111), obrazovka umožňuje veškeré nastavení, stejné jako v aplikaci Matlab/Simulink. Jednotlivá omezení budou vysvětlena v dalších krocích úlohy.

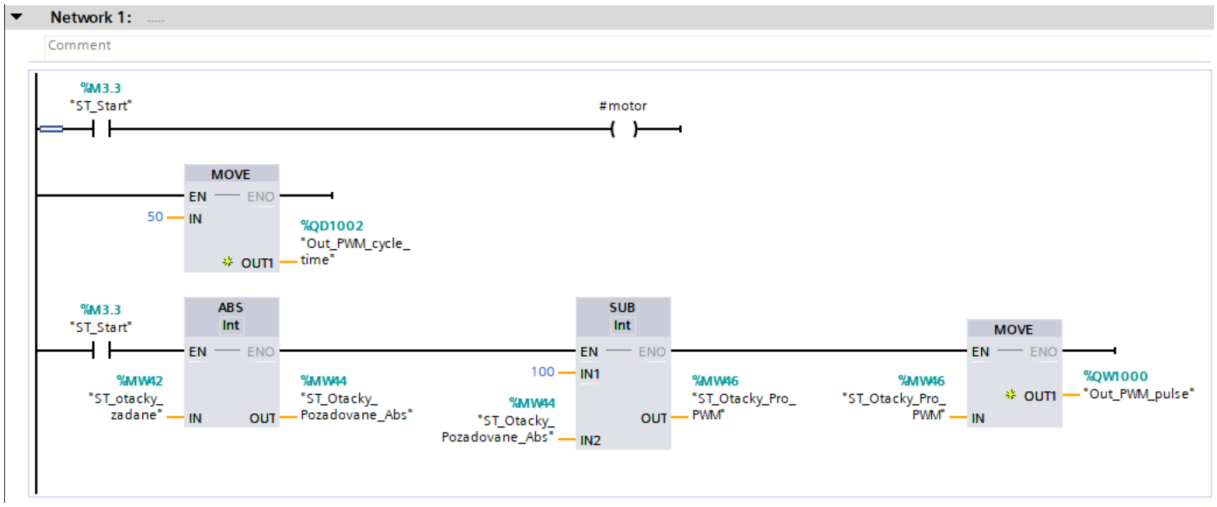


Obrázek 111 - HMI – Měření statické charakteristiky

Obdobně jako u předchozích úloh si vytvoříme organizační blok (OB) (Obrázek 51) pro vložení programu, který nám bude řídit měření statické charakteristiky.

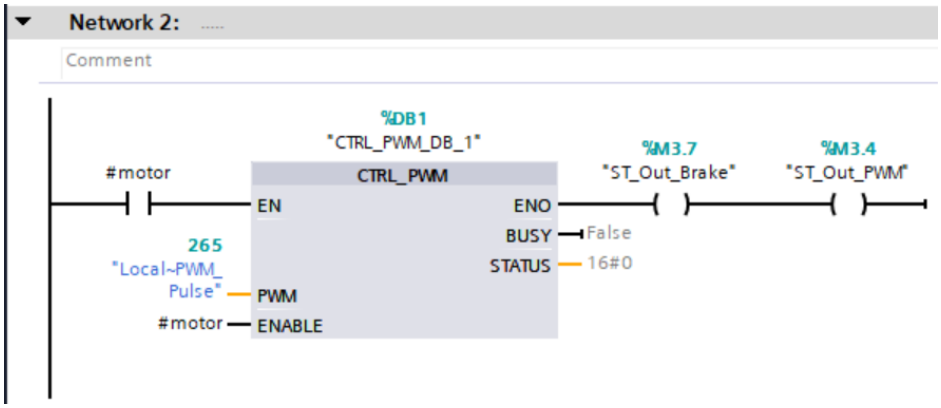
V síti 1 (Obrázek 112) programu vypočítáme střídu PWM signálu pro řízení otáček motoru. Princip výpočtu je shodný s předchozími úlohami, pouze pracujeme s jinými proměnnými, které jsme si vytvořili a příhodně pojmenovali pro tento program, uložíme je do odděleného

seznamu proměnných, pro větší přehlednost v projektu. Na základě zadaných otáček vypočítáme hodnotu „Out_PWM_pulse“, která určuje šířku výstupního signálu.



Obrázek 112 – Static Characteristic – Vytváření programu 1

V síti 2 (Obrázek 113) aktivujeme výstup PWM pomocí funkčního bloku CTRL_PWM, pokud je aktivní proměnná „motor“. Výstupní proměnné „ST_Out_Brake“ a „ST_Out_PWM“. Následně porovnáváme v OB „Main“ (Obrázek 58), kde podle nich aktivujeme konkrétní výstupy.



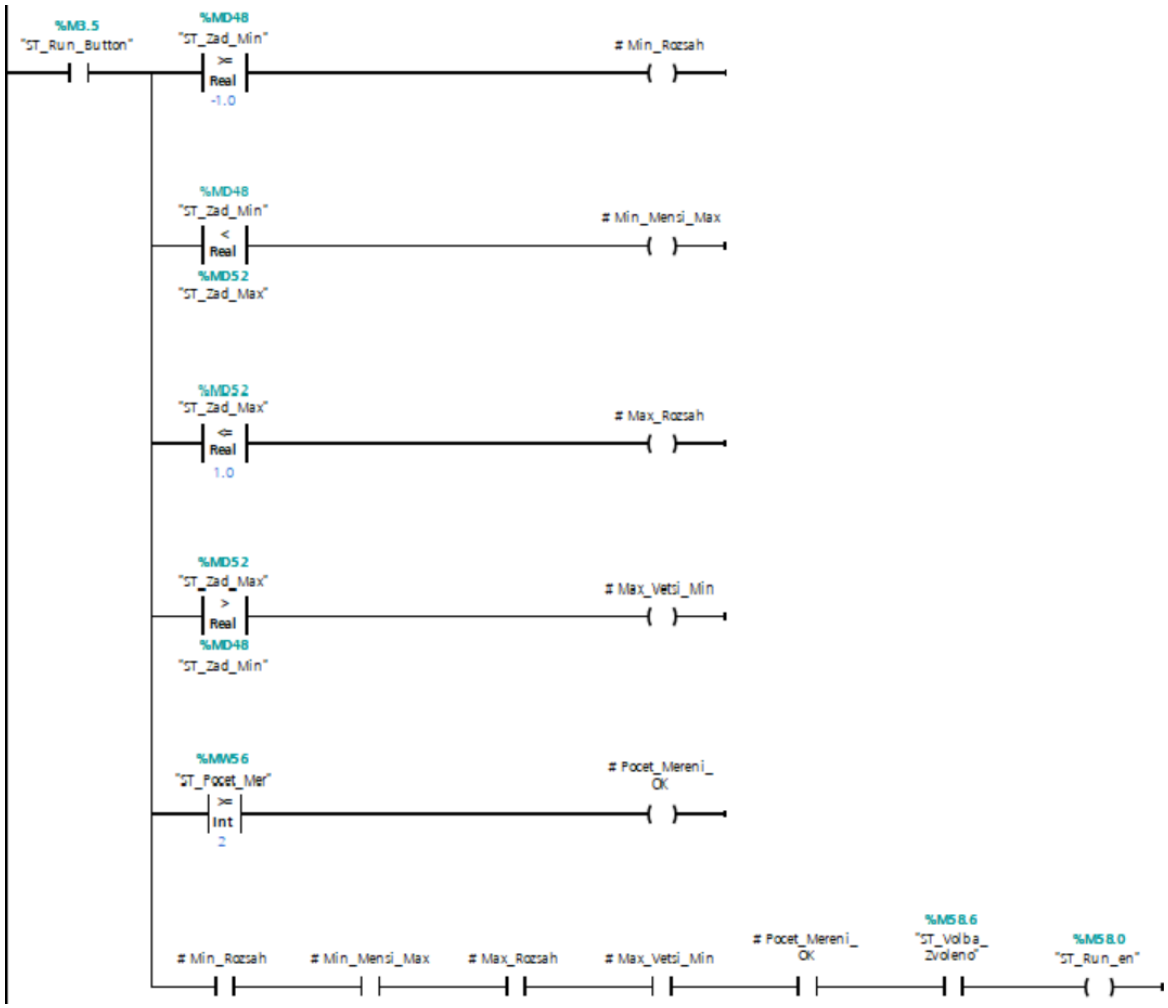
Obrázek 113 - Static Characteristic – Vytváření programu 2

V síti 3 (Obrázek 114) ošetřujeme vstup volby režimu měření statické charakteristiky. Tyto proměnné budeme aktivovat pomocí tlačítek v HMI panelu (Obrázek 111). Proměnnou „ST_Volba_Zvoleno“ použijeme v dalším kroku programu jako podmínku pro spuštění měření.

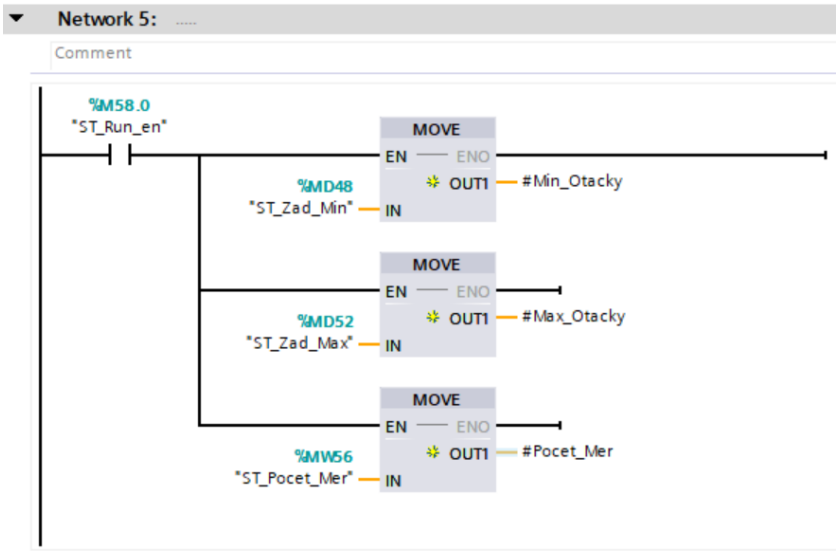
Obrázek 114 - Static Characteristic – Vytváření programu 3

V síti 4 (Obrázek 115) kontrolujeme správnost vstupních parametrů před spuštěním měření. Ověřujeme, zda minimální hodnota rozsahu není menší než −1.0 a maximální není větší než 1.0. Dále kontrolujeme, že minimální hodnota je skutečně menší než maximální, a že počet měřených bodů je alespoň 2. Pokud jsou všechny tyto podmínky splněny, nastaví se proměnná „ST_Run_en“, která umožní spuštění měření, za předpokladu, že je aktivní tlačítko „ST_Run_Button“ a byla provedena volba režimu měření („ST_Volba_Zadano“).

V síti 5 (Obrázek 116) probíhá přenos zadaných hodnot do pracovních proměnných, se kterými budeme dále pracovat během měření.



Obrázek 115 - Static Characteristic – Vytváření programu 4



Obrázek 116 - Static Characteristic – Vytváření programu 5

V této síti připravujeme hodnoty pro výpočet velikosti jednoho kroku mezi jednotlivými měřenými body.

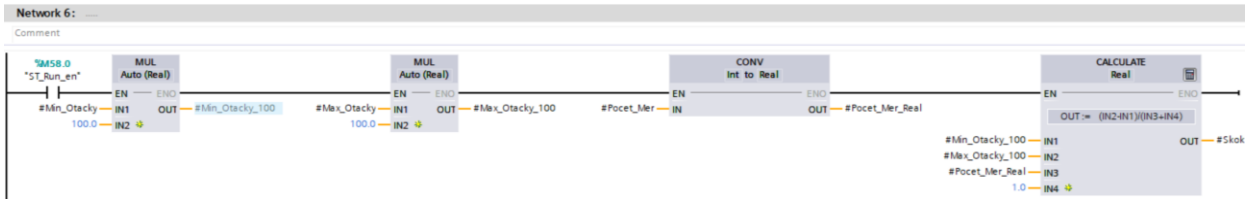
Nejprve převedeme minimální a maximální hodnotu otáček ze zadaného rozsahu na jednotky odpovídající PWM střídě, tedy vynásobením hodnotou 100 (např. z rozsahu −1.0 až 1.0

vytvoříme −100 až 100). Výsledky ukládáme do proměnných „Min_Otacky_100“ a „Max_Otacky_100“.

Následně převedeme počet měření z typu Int na Real (proměnná Pocet_Mer_Real), abychom s ním mohli počítat v reálném formátu.

V posledním kroku pomocí bloku CALCULATE spočítáme velikost jednoho kroku mezi měřenými hodnotami podle vztahu (8):

$$Skok = \frac{Max\_Otacky\ 100 - Min\_Otacky\ 100}{Pocet\_Mer\_Real - 1} \tag{8}$$



Obrázek 117 - Static Characteristic – Vytváření programu 6

Nyní máme připraveny všechny potřebné proměnné pro samotné měření. To budeme realizovat pomocí funkčních bloků, které si nyní vytvoříme. Začneme funkčním blokem pro vzestupné měření (Ascending).

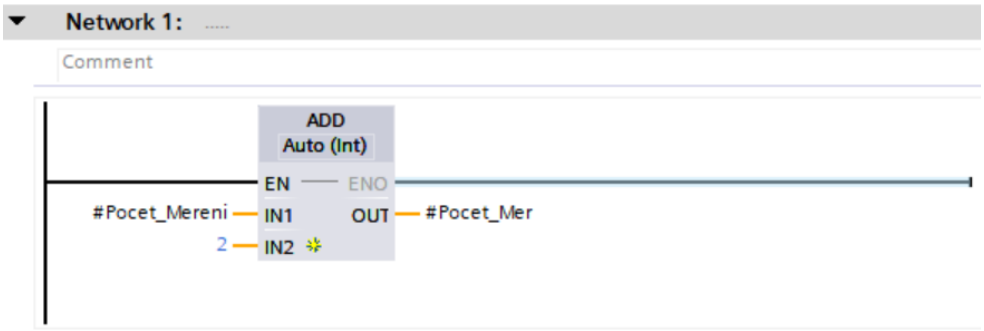
Na obrázku (Obrázek 118) můžeme vidět jednotlivé vstupy, výstupy a proměnné právě vytvořeného funkčního bloku, které budou následně použity v programu FB.

FB_ST_Ascending

	Name	Data type	Default value	Retain	Accessible f...	Writa...	Visible in ...	Setpoint
1	Input				☐	☐	☐	☐
2	Min	Real	0.0	Non-retain	☑	☑	☑	☐
3	Max	Real	0.0	Non-retain	☑	☑	☑	☐
4	Skok	Real	0.0	Non-retain	☑	☑	☑	☐
5	Pocet_Mereni	Int	0	Non-retain	☑	☑	☑	☐
6	Output				☐	☐	☐	☐
7	Otacky_Pozadovane	Real	0.0	Non-retain	☑	☑	☑	☐
8	Konec	Bool	false	Non-retain	☑	☑	☑	☐
9	InOut				☐	☐	☐	☐
10	<Add new>				☐	☐	☐	☐
11	Static				☐	☐	☐	☐
12	Akt_Pocet	Int	0	Non-retain	☐	☐	☐	☐
13	Otacky_Akt	Real	0.0	Non-retain	☐	☐	☐	☐
14	Pocet_Mer	Int	0	Non-retain	☑	☑	☑	☐
15	Temp				☐	☐	☐	☐
16	Spust_Mereni	Bool			☐	☐	☐	☐
17	Constant				☐	☐	☐	☐
18	cas	Time	T#6500ms		☐	☐	☐	☐
19	cas1	Time	T#500ms		☐	☐	☐	☐

Obrázek 118 - FB Ascedning – proměnné

V síti 1 (Obrázek 119) zvyšujeme hodnotu proměnné „Pocet_Mer“ o dvě jednotky oproti vstupní hodnotě „Pocet_Mereni“. Tímto způsobem si zajišťujeme rezervu na začátku a na konci měřicího rozsahu, abychom pokryli celý interval včetně krajních hodnot.



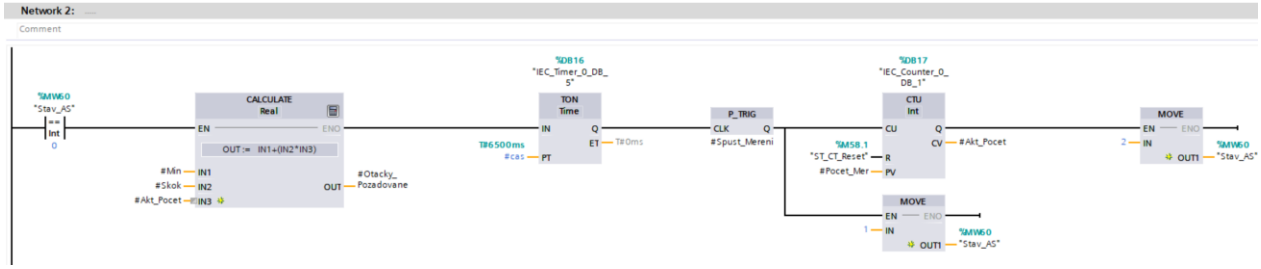
Obrázek 119 - FB Ascending – tvorba programu 1

V této síti (Obrázek 120) probíhá samotné měření jednoho bodu statické charakteristiky. Pokud je proměnná „Stav_AS“, řídíme chod programu pomoví principu „Switch – Case“, rovna nule, spočítáme požadovanou hodnotu otáček jako (9):

$$Otacky\_Pozadovane = Min + (Skok \times Akt\_Pocet) \quad (9)$$

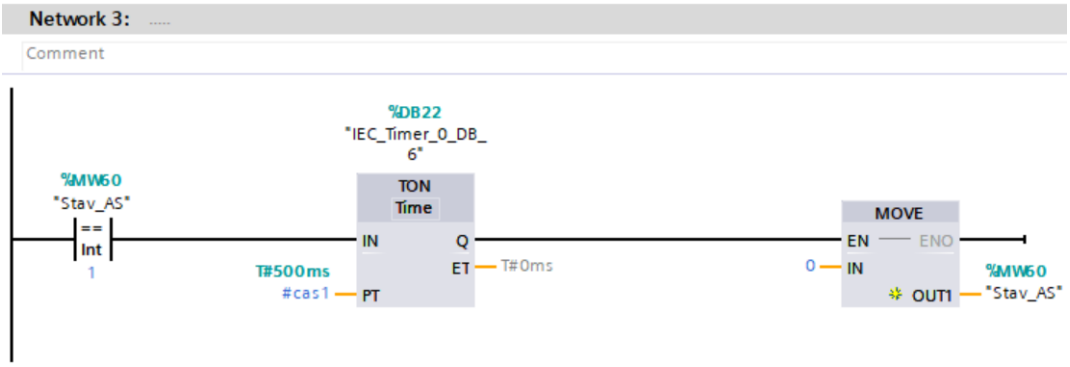
Jelikož je hodnota proměnné „Akt_Pocet“ po volání programu rovna 0, bude první měření hodnota rovna „Min“.

Po zadání této hodnoty spustíme časovač „cas“, který vytvoří prodlevu (např. 6,5 sekundy) pro ustálení systému. Po jeho doběhnutí se pomocí impulsního bloku „P_TRIG“ spustí měření, později si vytvoříme funkční blok volaný touto proměnnou, a současně se o 1 zvýší hodnota čítače „Akt_Pocet“. Jakmile se odečte požadovaný počet bodů, nastavíme proměnnou „Stav_AS“ na hodnotu 2, čímž přecházíme do dalšího stavu sekvence.



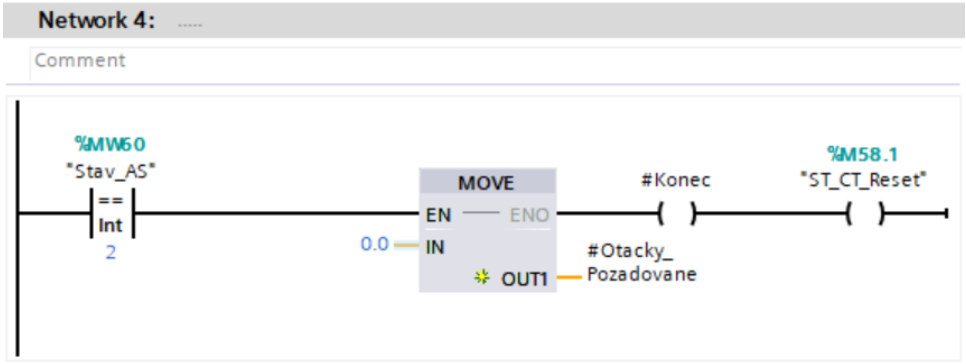
Obrázek 120 - FB Ascending – tvorba programu 2

V síti 3 (Obrázek 121) zajišťujeme návrat systému do výchozího stavu. Pokud je proměnná „Stav_AS“ rovna 1, spustí se časovač „cas1“ s krátkou prodlevou (500 ms). Po jejím uplynutí se pomocí bloku MOVE nastaví hodnota „Stav_AS“ zpět na 0, čímž se připraví nový měřicí cyklus. Tento krátký mezistav slouží k oddělení jednotlivých fází sekvence, zajišťuje bezpečné vynulování předchozího časovače.



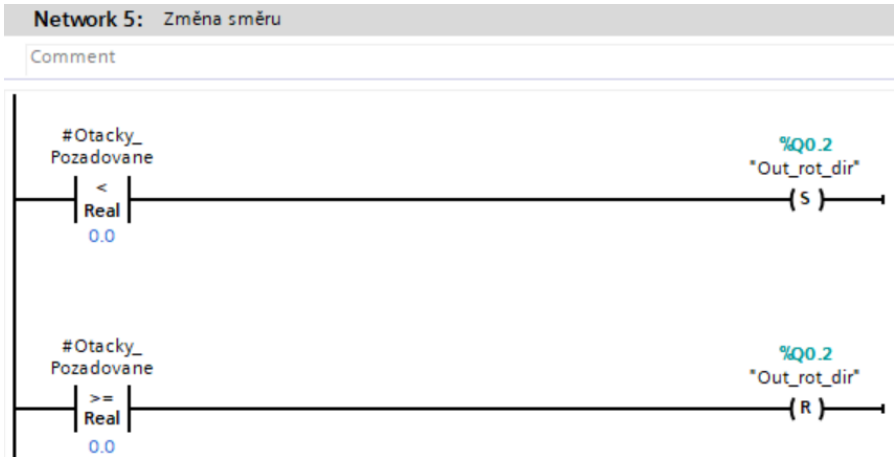
Obrázek 121 - FB Ascending – tvorba programu 3

V síti 4 (Obrázek 122) zajišťujeme ukončení měřicí sekvence. Pokud je proměnná „Stav_AS“ rovna 2, nastavíme pomocí bloku MOVE hodnotu „Otacky_Pozadovane“ na 0.0, čímž zastavíme motor. Současně aktivujeme signály „Konec“, ten budeme používat v dalších částech úlohy, a „ST_CT_Reset“, které slouží k připravení systému na nový měřicí cyklus, tím že vyresestuje čítač měření.



Obrázek 122 - FB Ascending – tvorba programu 4

V síti 5 (Obrázek 123) zajišťujeme změnu točení motoru podle hodnoty požadovaných otáček, a to přímým nastavením výstupní proměnné „Out_rot_dir“.



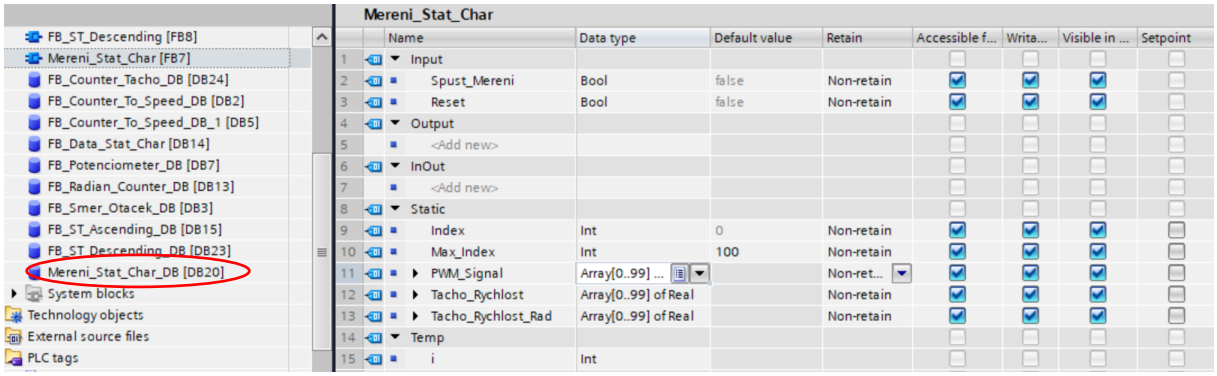
Obrázek 123 - FB Ascending – tvorba programu 5

Jako další si připravíme funkční blok, který se nám bude starat o ukládání naměřených hodnot do svého Data bloku.

Začneme tím, že si samotný blok vytvoříme, postupujeme stejně jako u předchozích FB, ale tentokrát zvolíme programovací jazyk SCL (Structured Control Language), v TIA Portalu ho využíváme tam, kde je programování v LAD nebo FBD nepřehledné nebo omezené.

Na obrázku (Obrázek 124) vidíme jednotlivé proměnné FB pro měření charakteristiky. Ve statických proměnných jsou mimo jiné definována pole („PWM_Signal“, „Tacho_Rychlost“, „Tacho_Rychlost_Rad“), která jsou typu Array[0..99] of Real. To znamená, že každé z těchto polí má 100 prvků, do kterých můžeme postupně ukládat změřené hodnoty během běhu programu. Statické proměnné uchovávají svou hodnotu po celou dobu aktivní instance bloku a jsou vhodné pro záznam dat v opakovaných cyklech měření.

Výsledné hodnoty budeme schopni číst přímo z Data Bloku DB (Obrázek 124), pomocí funkce monitoring.



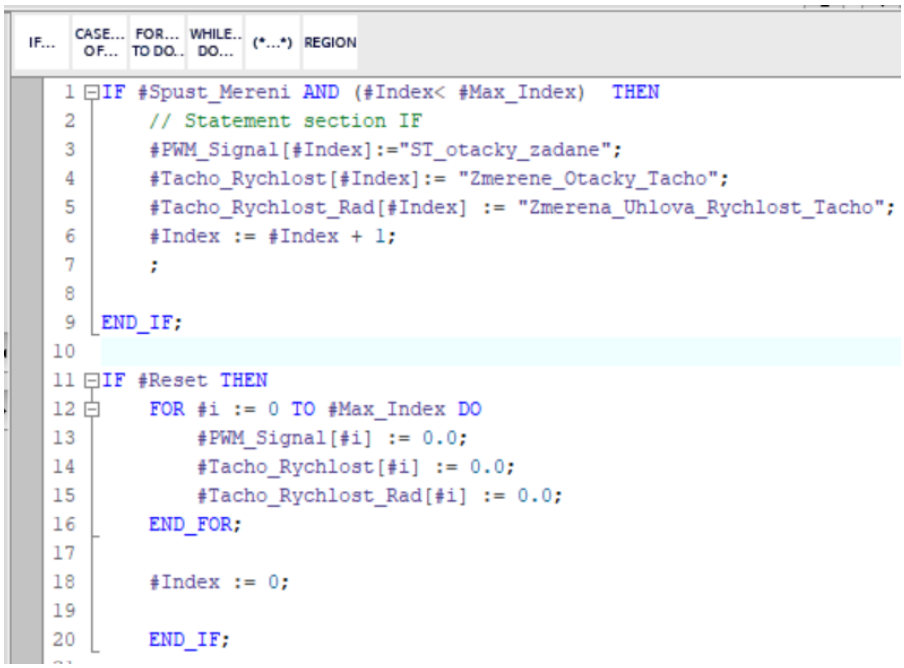
Obrázek 124 - Seznam proměnných FB pro ukládání dat statické charakteristiky

Na obrázku (Obrázek 125) je samotná logika funkčního bloku. V první části programu ověřujeme, zda je aktivní vstupní signál „Spust_Mereni“ a zároveň zda hodnota proměnné „Index“ nedosáhla maximálního povoleného počtu měření („Max_Index“). Pokud jsou podmínky splněny, uložíme do aktuálního indexu pole změřené hodnoty: hodnotu PWM signálu („ST_otacky_zadane“), otáčky z tachogenerátoru („Zmerene_Otacky_Tacho“) a úhlovou rychlost („Zmerena_Uhlova_Rychlost_Tacho“). Poté zvýšíme hodnotu „Index“ o 1.

V druhé části programu probíhá mazání dat. Pokud je aktivní signál „Reset“, spustí se cyklus FOR, ve kterém jsou všechna pole vynulována. Po dokončení se proměnná „Index“ nastaví zpět na 0, čímž připravíme blok pro nové měření.

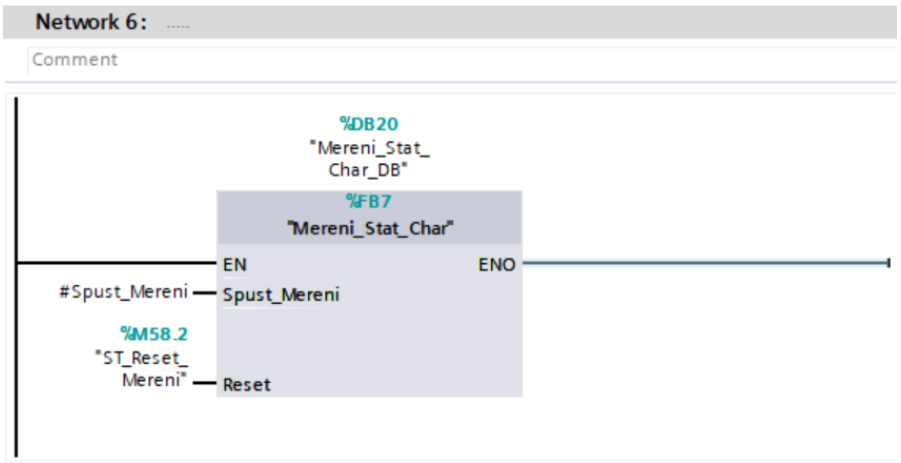
V jazyce SCL používáme:

- znak # pro označení lokálních proměnných v rámci funkčního bloku (např. „#Index“),
- proměnné bez # označují globální proměnné
- přiřazovací operátor **:=** používáme pro přiřazení hodnoty, stejně jako = v jiných jazycích.



Obrázek 125 – FB Mereni_Stat_Char – Program

Po dokončení funkčního bloku pro zápis naměřených hodnot ho vložíme do funkčního bloku „FB Ascending“ (Obrázek 126). Zde ho voláme pomocí proměnné „Spust_Mereni“, kterou spouštíme výše v programu (Obrázek 120).

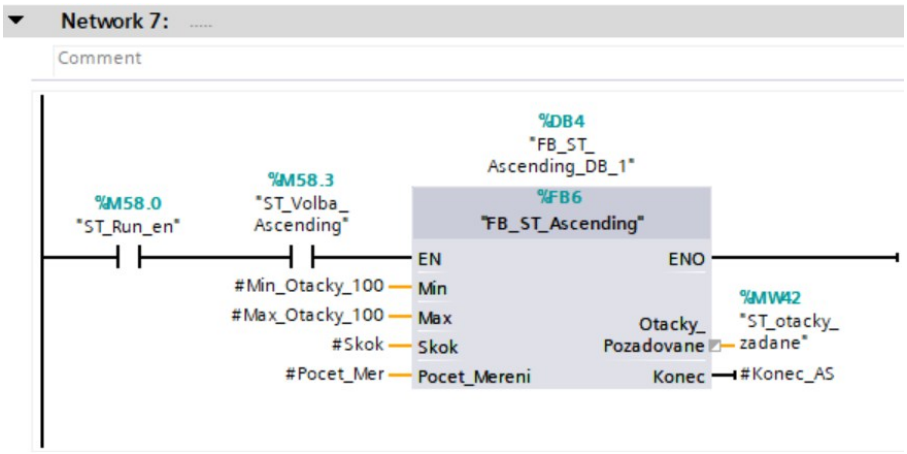


Obrázek 126 - FB Ascending – tvorba programu 6

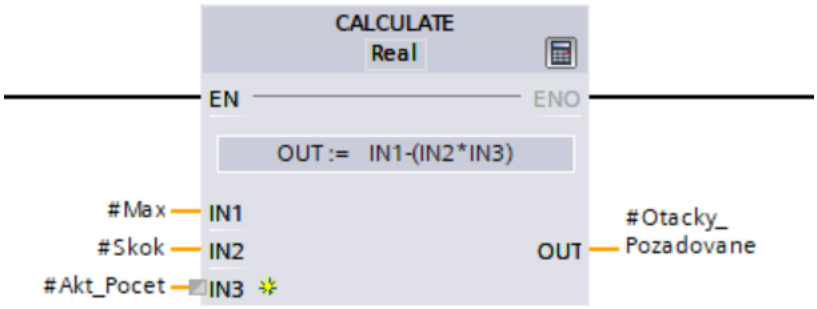
Celý vytvořený funkční blok vložíme do organizačního bloku pro měření statické charakteristiky (Obrázek 127).

Stejným způsobem vytvoříme i druhý funkční blok, pro sestupné měření (Descending). FB bude naprosto stejný s jedinou změnou v počítání požadované hodnoty otáček (Obrázek 128). Na rozdíl od předchozího zde nepřičítáme k minimální hodnotě otáček, ale odečítáme od maximální hodnoty otáček.

Celý funkční blok pak vložíme do programu stejně jako „FB_ST_Ascending“ (Obrázek 127). Jako volací proměnné použijeme „ST_Run_en“ a „ST_Volba_Descending“.

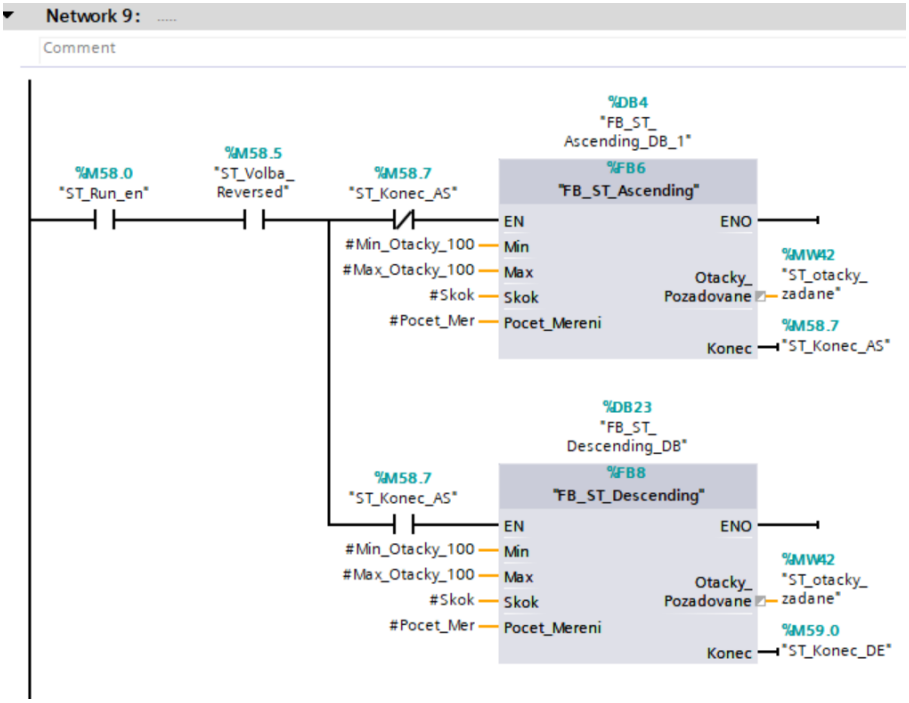


Obrázek 127 - Static Characteristic – Vytváření programu 7



Obrázek 128 - FB Descending – Blok Calculate

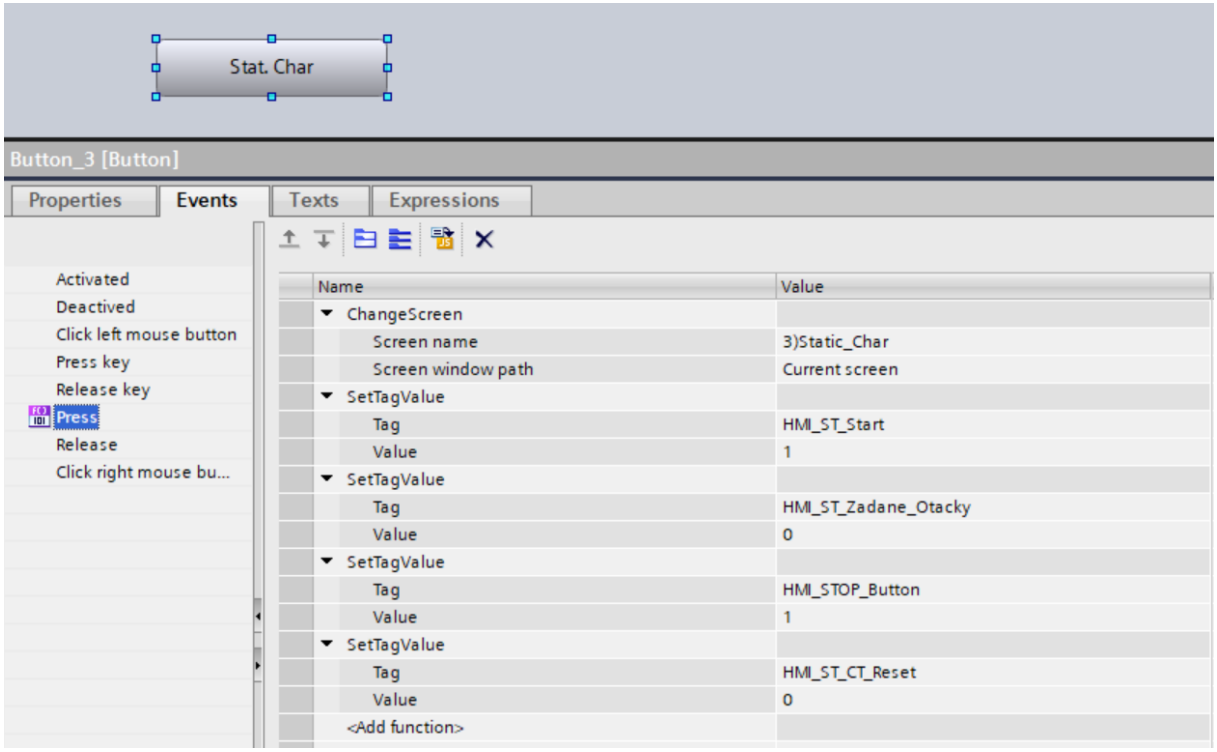
Realizaci měření v obou směrech provedeme tak, že pustíme jednotlivé bloky, které už máme vytvořené po sobě (Obrázek 129). Jakmile skončí měření jednoho směru, začne druhé.



Obrázek 129 - Static Characteristic – Vytváření programu 8

Nyní máme programovou část hotovou, přejdeme ke tvorbě HMI prostředí. Jako u předchozích úloh si nejdříve na obrazovce „Main“ vytvoříme tlačítko, pomocí kterého se dostaneme na obrazovku s nastavením měření statické charakteristiky. V záložce „Properties“ ve složce „Events“ (Obrázek 130) nastavíme potřebné události, které se budou dít po stisku tlačítka.

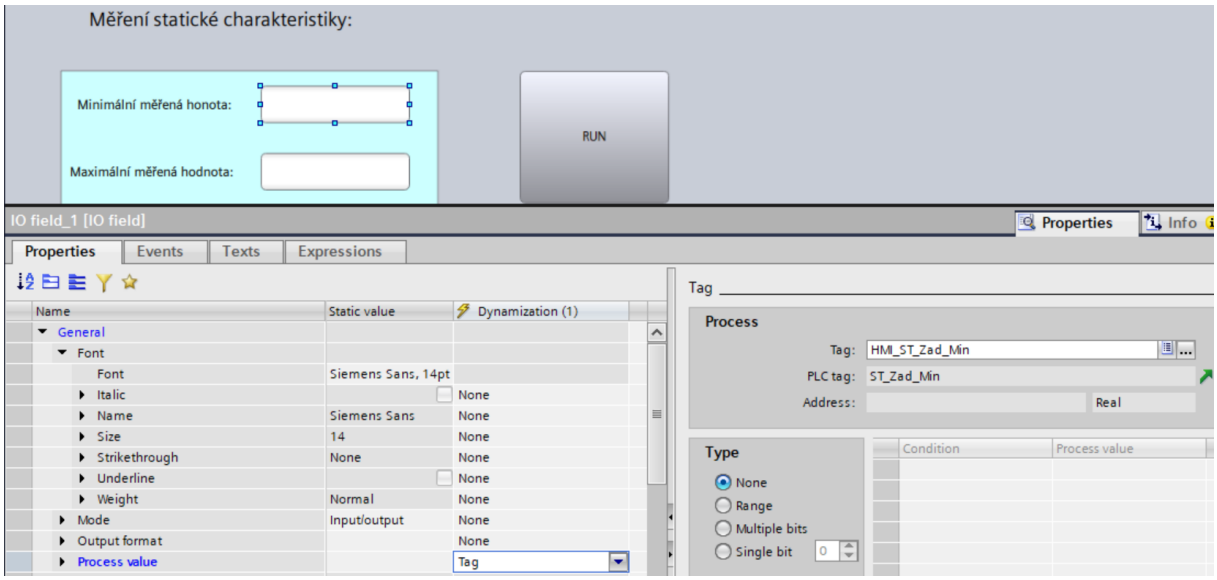
Jako první pomocí „ChangeScreen“ nastavíme změnu obrazovky při stisku. Jako další nastavíme tagy „HMI_ST_Start“ tím povolíme výpočet PWM signálu v požadovaném OB. Dále nastavíme z bezpečnostního hlediska požadované otáčky na 0. Dále aktivujeme Stop tlačítko v OB „Main“, to nám zamezí chodu programu „Main“. A jako poslední deaktivujeme proměnnou pro reset uložených hodnot, kterou aktivujeme stiskem tlačítka „Zpět“ na řídící obrazovce.



Obrázek 130 - HMI – Static characteristic 1

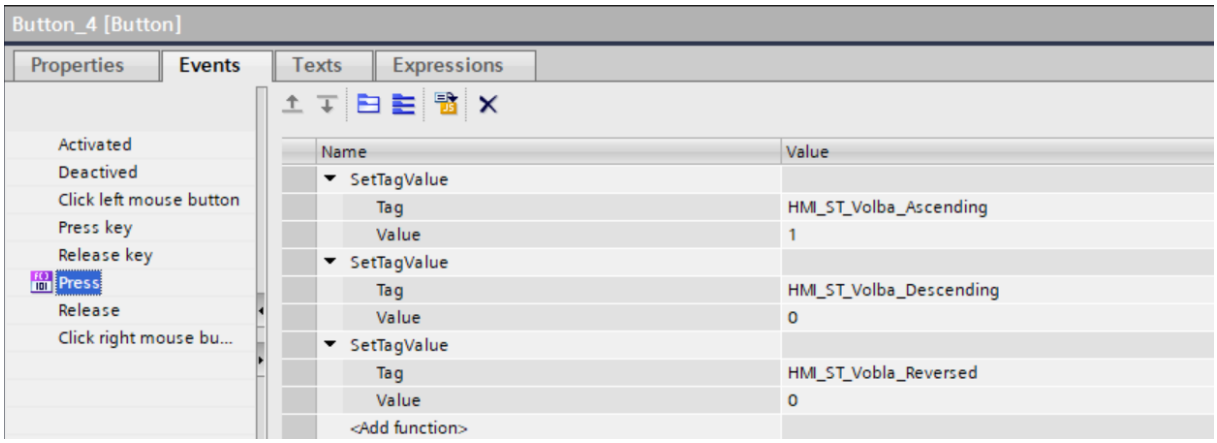
Přesuneme se na obrazovku pro ovládání měření statické charakteristiky, zde si vložíme vstupně výstupní pole (IO Field), pomocí kterých budeme zadávat rozsah a počet měření.

Pole nastavíme na mód „Input/Output“, a jako „Process value“ tag (Obrázek 131). Stejným způsobem nastavíme i IO Field pro maximální hodnotu a počet měření (Obrázek 111)

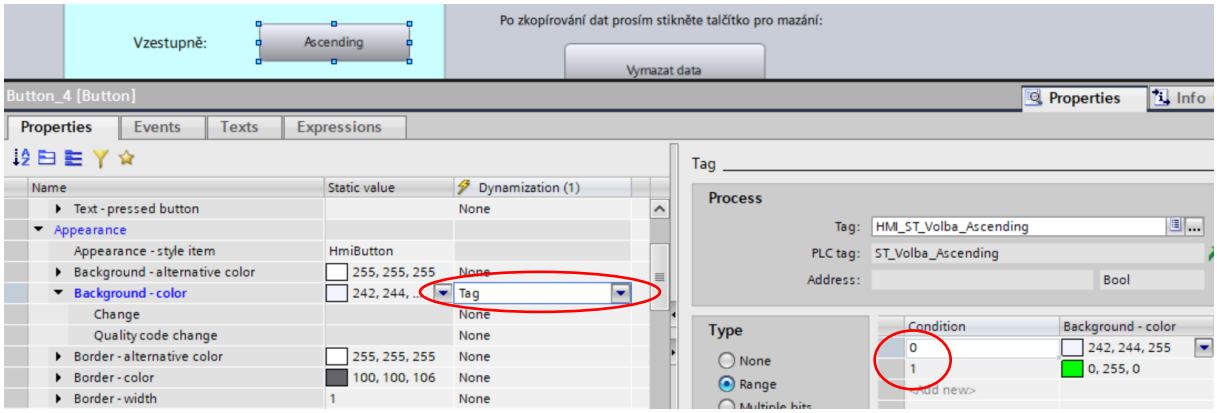


Obrázek 131 - HMI – Static characteristic 2

Poté si vložíme tlačítka, pomocí kterých budeme nastavovat způsob měření. Po vložení si otevřeme záložku „Properties“ složku „Events“ a nastavíme ho podle obrázku (Obrázek 132). Obdobně nastavíme i zbylá dvě tlačítka, zaručíme tím, že bude vždy zvolena pouze jedna volba. Jako další si nastavíme změnu barvy tlačítka, když bude zvoleno (Obrázek 133). Na základě změny stavu proměnné se bude měnit barva pozadí tlačítka.

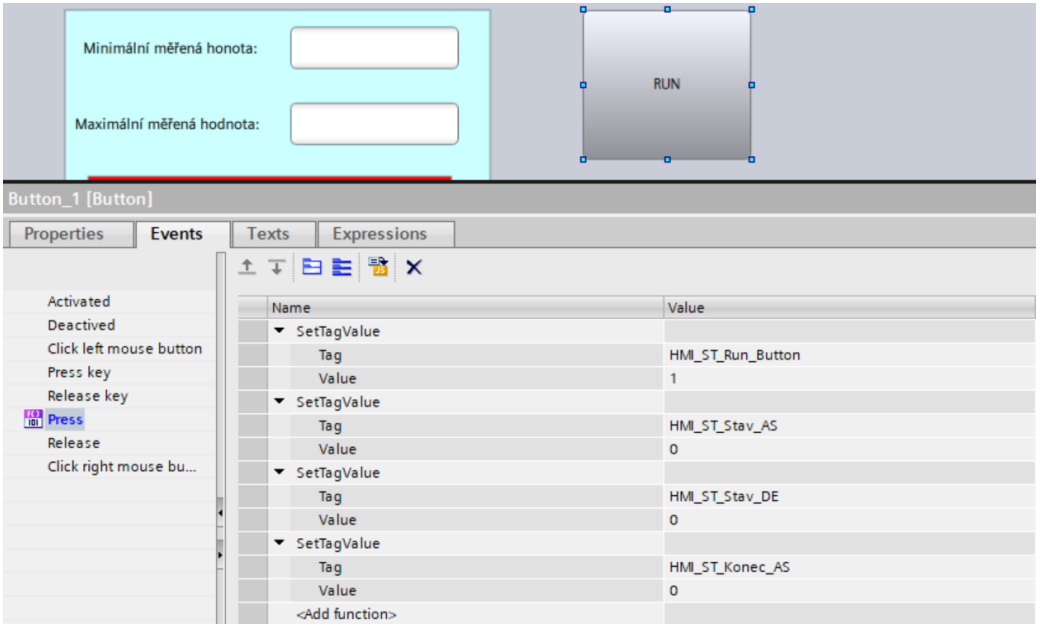


Obrázek 132 - HMI – Static characteristic 3



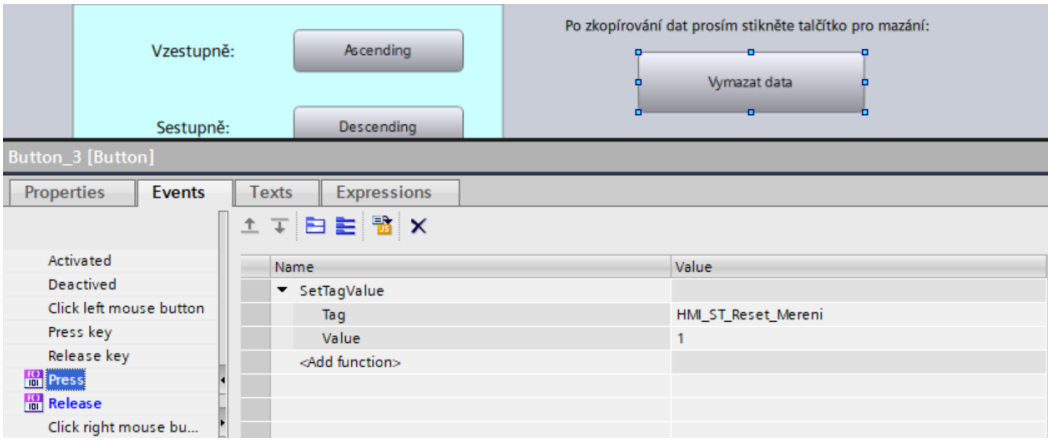
Obrázek 133 - HMI – Static characteristic 4

Další si vložíme tlačítko „RUN“, které nám bude spouštět měření a zároveň nulovat určité proměnné (Obrázek 134). Stavové proměnné nulujeme z toho důvodu, že je zde pravděpodobnost, že uživatel ukončil test ještě před jeho dokončením, tím se tedy tyto proměnné nevynulovali programově, ale musíme jejich nulování pojistit tímto způsobem.



Obrázek 134 - HMI – Static characteristic 5

Tlačítko pro mazání dat (Obrázek 135) aktivuje při stisku a deaktivuje při uvolnění proměnnou „ST_Reset_Mereni“, tu přivádíme na vstup FB pro ukládání dat z měření (Obrázek 126). Tím se spustí část programu, která nám naplní pole s hodnotami nulami.



Obrázek 135 - HMI – Static characteristic 6

Nakonec si nastavíme tlačítko „Zpět“ (Obrázek 136). Toto tlačítko nám mimo návratu na obrazovku „Main“ zajistí, že se nám všechny stavové proměnné použité v programu nastaví zpět do nuly, to stejné platí o zadaných veličinách pro měření, také kontrolně vynulujeme čítače aktivací proměnné „HMI_ST_CT_Reset“.

Name	Value
ChangeScreen	
Screen name	1)Main
Screen window path	Current screen
SetTagValue	
Tag	HMI_ST_Start
Value	0
SetTagValue	
Tag	HMI_ST_Run_en
Value	0
SetTagValue	
Tag	HMI_ST_Zad_Min
Value	0
SetTagValue	
Tag	HMI_ST_Zad_Max
Value	0
SetTagValue	
Tag	HMI_ST_Pocet_Mer
Value	0
SetTagValue	
Tag	HMI_ST_Run_Button
Value	0
SetTagValue	
Tag	HMI_STOP_Button
Value	0
SetTagValue	
Tag	HMI_ST_CT_Reset
Value	1
<Add function>	

Obrázek 136 - HMI – Static characteristic 7

Když máme HMI prostředí hotové, otevřeme si v projektovém stromě Data Blok našeho funkčního bloku pro zápis změřených hodnot (Obrázek 137). Přepneme PLC do online módu a klikneme na „Monitor all“, zobrazí se nám aktuální hodnoty jednotlivých polí v polích. Na obrázku můžeme vidět, že jsme v minulém měření měřili sestupně (Descending), a v aktuálním měření jsme měřili obousměrně. Když v horní části obrazovky klikneme na „Snapschot of the actual values“ převedou se nám aktuálně uvedené hodnoty do sloupečku „Snapshot“. Když poté v HMI panelu klikneme na vymazat data, vymažou se pouze aktuální data a v sloupci „Snapshot“ nám zůstanou. Odtud si data můžeme nakopírovat pro další použití.

V1.41 ▸ PLC1 [CPU 1215C DC/DC/DC] ▸ Program blocks ▸ Mereni_Stat_Char_DB [DB20]

Keep actual values | Snapshot | Copy snapshots to start values | Load start values as actual values

Mereni_Stat_Char_DB (snapshot created: 4/28/2025 3:42:15 PM)

	Name	Data type	Start value	Snapshot	Monitor value	Retain	Accessible f...	Writa...	Visible in ...	Setpoint
1	Input									
2	Spust_Mereni	Bool	false	FALSE	FALSE					
3	Reset	Bool	false	FALSE	FALSE					
4	Output									
5	InOut									
6	Static									
7	Index	Int	0	12	25					
8	Max_Index	Int	100	100	100					
9	PWM_Signal	Array[0..99] of ...								
10	PWM_Signal[0]	Real	0.0	100.0	-100.0					
11	PWM_Signal[1]	Real	0.0	82.0	-82.0					
12	PWM_Signal[2]	Real	0.0	64.0	-64.0					
13	PWM_Signal[3]	Real	0.0	45.0	-45.0					
14	PWM_Signal[4]	Real	0.0	27.0	-27.0					
15	PWM_Signal[5]	Real	0.0	9.0	-9.0					
16	PWM_Signal[6]	Real	0.0	-9.0	9.0					
17	PWM_Signal[7]	Real	0.0	-27.0	27.0					
18	PWM_Signal[8]	Real	0.0	-45.0	45.0					
19	PWM_Signal[9]	Real	0.0	-64.0	64.0					
20	PWM_Signal[10]	Real	0.0	-82.0	82.0					
21	PWM_Signal[11]	Real	0.0	-100.0	100.0					
22	PWM_Signal[12]	Real	0.0	0.0	0.0					
23	PWM_Signal[13]	Real	0.0	0.0	100.0					
24	PWM_Signal[14]	Real	0.0	0.0	82.0					
25	PWM_Signal[15]	Real	0.0	0.0	64.0					
26	PWM_Signal[16]	Real	0.0	0.0	45.0					
27	PWM_Signal[17]	Real	0.0	0.0	27.0					
28	PWM_Signal[18]	Real	0.0	0.0	9.0					
29	PWM_Signal[19]	Real	0.0	0.0	-9.0					
30	PWM_Signal[20]	Real	0.0	0.0	-27.0					
31	PWM_Signal[21]	Real	0.0	0.0	-45.0					
32	PWM_Signal[22]	Real	0.0	0.0	-64.0					
33	PWM_Signal[23]	Real	0.0	0.0	-82.0					
34	PWM_Signal[24]	Real	0.0	0.0	-100.0					

Obrázek 137 - Zisk dat pro vytvoření statické charakteristiky

2.5 Porovnávání získaných výsledků z vypracovaných úloh v TIA Portalu a výsledků získaných měřením v aplikaci Matlab Simulink.

V této části práce jsem se zaměřil na porovnání výsledků měření získaných ve dvou rozdílných softwarových prostředích – v TIA Portalu a v MATLABu. Cílem bylo ověřit, zda se chování řízeného systému shodovalo při použití odlišných přístupů k řízení a akvizici dat. Nejprve jsem porovnal výsledky impulzní odezvy systému, které jsem získal pomocí připravených skriptů v MATLABu a nástroje Trace v TIA Portalu. Jelikož nebyly měřicí body časově sladěné, zvolil jsem metodu nepřímého porovnání pomocí identifikace přenosové funkce. K tomuto účelu jsem využil nástroj System Identification Toolbox, který je součástí prostředí MATLAB. V další části jsem se věnoval porovnání statických charakteristik, kde jsem zajistil shodný rozsah i počet vzorků. V tomto případě jsem mohl provést přímé srovnání dat, a to jak numericky, tak graficky.

2.5.1 Porovnávání získaných výsledků z úlohy Control Impulse Responce

Jak je popsáno v úlohách, data z TIA Portalu jsem získal pomocí funkce „Trace“. Zisk dat z aplikace v Matlabu nebyl možný přímo, proto jsem si po provedení měření uložil graf (Obrázek 92), zobrazující průběh žádané veličiny a výstupního úhlového poloha. Následně jsem si vytvořil pomocný skript v matlabu (Obrázek 138), který mi umožnil z uloženého souboru „.fig“ získat data, které byla potřebná pro vyobrazení grafu. Program otevře zadaný soubor, identifikuje všechny objekty typu osy (axes) a postupně prochází prostřední a horní podgraf. Z horního podgrafu jsou získávána data řídicího signálu, přičemž jsou vyhledávány objekty typu Stair, které odpovídají stupňovitému průběhu signálu. Z prostředního podgrafu jsou naopak získávána data úhlové polohy, a to prostřednictvím objektů typu line. V obou případech jsou data extrahována ve formě hodnot na osách X a Y a následně uložena do CSV souborů pro další zpracování pomocí nástroje System Identification ve MATLABu.

program_zisk_dat.m

```
fig = openfig('matlab_imp2.fig');  % Otevření uloženého .fig souboru
axesHandles = findall(fig, 'Type', 'axes');  % Nalezení všech os v grafu

for i = 2:3  % Procházení prostředního (2) a horního (3) podgrafu
    ax = axesHandles(i);  % Výběr aktuální osy

    if i == 3
        % Pokud se jedná o horní podgraf - hledám objekt typu Stair (stupňovitý průběh)
        plotObjs = findall(ax, 'Type', 'Stair');
    else
        % Jinak hledáme klasické křivky (line)
        plotObjs = findall(ax, 'Type', 'line');
    end

    for j = 1:length(plotObjs)  % Pro každý nalezený grafický objekt
        x = get(plotObjs(j), 'XData');  % Získání hodnot na ose X
        y = get(plotObjs(j), 'YData');  % Získání hodnot na ose Y

        if i == 3
            % Uložení dat z horního podgrafu (řídicí signál)
            filename = sprintf('data_control_line%d.csv', j);
        elseif i == 2
            % Uložení dat z prostředního podgrafu (úhly)
            filename = sprintf('data_angles_line%d.csv', j);
        end

        writematrix([x(:), y(:)], filename);  % Uložení dat do CSV
    end
end
```

Obrázek 138 – Skript v Matlabu pro zisk dat ze souboru .fig

System Identification Toolbox je nástroj, který slouží k modelování dynamických systémů na základě naměřených dat. Uživatel pomocí tohoto nástroje získává matematický model systému (například přenosovou funkci nebo stavový model) bez nutnosti znát vnitřní strukturu systému – postačí mu vstupní a výstupní signály.

Toolbox umožňuje import dat v časové nebo frekvenční oblasti, jejich předzpracování (např. filtrování, výběr oblasti zájmu), volbu vhodného typu modelu (např. ARX, přenosová funkce,

stavový prostor), odhad parametrů a následné vyhodnocení přesnosti modelu. Vizuální rozhraní tohoto nástroje, které lze spustit pomocí příkazu „ident“, výrazně usnadňuje celý proces.

Pro účely přesnější identifikace dynamiky systému jsem se rozhodl nahradit výstupní veličinu, kterou byla úhlová poloha hřídele motoru, její časovou derivací – tedy úhlovou rychlostí. Tento krok byl nezbytný zejména proto, že chování systému při impulzní odezvě je výrazně lépe patrné právě na rychlostním signálu. Zatímco poloha přirozeně integruje rychlost a její průběh je pozvolnější, rychlost okamžitě reaguje na změny řídicího signálu, což je pro analýzu odezvy systému a následnou identifikaci modelu výhodnější.

V tomto případě jsem pracoval s daty získanými z měření v prostředí MATLAB, konkrétně ze souboru obsahujícího časový průběh úhlové polohy. Jelikož se jednalo o diskrétní časovou řadu s rovnoměrným vzorkováním, bylo možné tuto veličinu převést na rychlost pomocí numerické derivace. Použil jsem dopředný diferenciální podíl (10), který odpovídá numerické derivaci prvního řádu:

$$\omega_i = \frac{\theta_i - \theta_{i-1}}{t_i - t_{i-1}} \tag{10}$$

Kde:

ω_i je úhlová rychlost v časovém bodě i,

θ_i je úhlová poloha v čase t_i

t_i je čas odpovídající vzorku i

Jako příklad lze uvést konkrétní výpočet (11) z dat z Matlabu (Tabulka 1):

$$\omega = \frac{0.157 - 0.117}{1.785 - 1.650} = \frac{0.04}{0.135} \approx 0.296\,\text{rad/s} \tag{11}$$

Tabulka 1 - Vzorek dat použitých pro výpočet získaných z Matlabu

t [s]	Poloha [rad]	Vstup	Rychlost [rad/s]	t [s]	Poloha [rad]	Vstup	Rychlost [rad/s]
0,00	0,00	0	0,00	0,99	0,00	0	0,00
0,29	0,00	0	0,00	1,07	0,00	0	-3,52
0,40	0,00	0	0,00	1,14	-0,26	1	-16,29
0,48	0,00	0	0,00	1,21	-1,45	1	-28,36
0,55	0,00	0	0,00	1,28	-3,49	1	-39,28
0,62	0,00	0	0,00	1,36	-6,40	1	-49,62
0,70	0,00	0	0,00	1,43	-9,97	1	-58,43
0,77	0,00	0	0,00	1,51	-14,53	1	-69,15
0,84	0,00	0	0,00	1,59	-19,92	1	-74,85
0,92	0,00	0	0,00	1,67	-26,06	1	-84,61

Obdobně jsem pokračoval i u všech dalších výpočtů s daty získaných z matlabu. Tímto způsobem vypočtené hodnoty úhlové rychlosti byly následně použity jako výstupní signál při identifikaci přenosové funkce v prostředí Matlab.

Při zpracování dat exportovaných z prostředí TIA Portalu bylo nutné zohlednit způsob, jakým byla tato data naměřena. V řízené části běžel blok pro výpočet výstupních hodnot v režimu OB cyclic interrupt s periodou 100 ms. Naproti tomu samotné vzorkování dat pomocí nástroje Trace probíhalo častěji – každých 50 interních cyklů CPU. Tento rozdíl mezi frekvencí výpočtu a frekvencí záznamu vedl k tomu, že v naměřeném souboru se často vyskytovaly vícekrát po sobě stejné hodnoty úhlové polohy, přestože měření probíhalo správně (Obrázek 141).

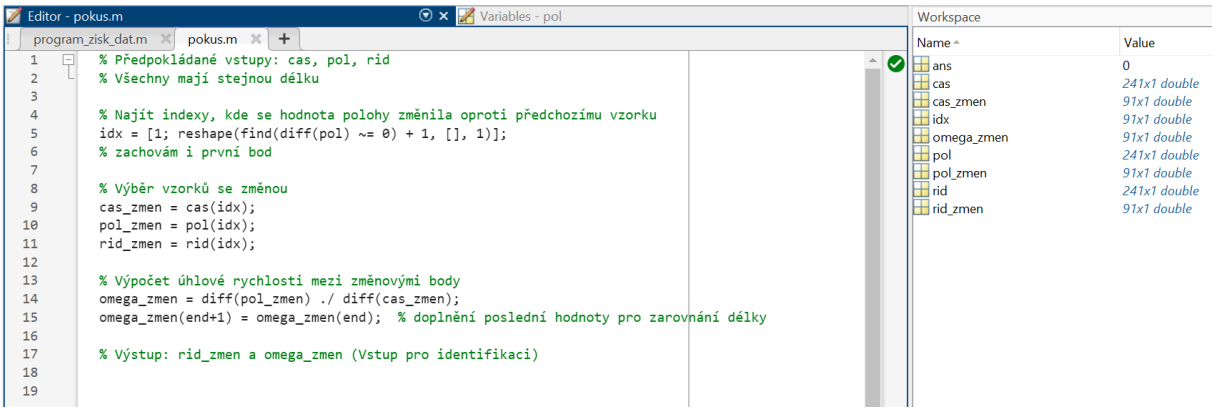
Tento jev vedl ke vzniku duplicitních vzorků, které způsobují problémy při výpočtu numerické derivace. Konkrétně při použití klasické dopředné diference vznikají úseky s nulovou rychlostí, které neodpovídají skutečnému pohybu motoru. Z tohoto důvodu byl vytvořen vlastní skript, který z dat odstraňuje všechny po sobě jdoucí vzorky s identickou hodnotou úhlové polohy (Obrázek 140). Derivace je pak počítána pouze mezi vzorky, kde ke skutečné změně došlo, což vede k věrnějšímu výpočtu rychlosti využitelné při identifikaci přenosové funkce.

Vytvořený skript v Matlabu (Obrázek 139) slouží k výběru relevantních vzorků a výpočtu úhlové rychlosti mezi nimi. Na vstupu pracuje s vektory časových značek (cas), úhlové polohy (pol) a řídicího signálu (rid). Pomocí funkce „diff“ a následného výběru indexů jsou určeny pouze ty vzorky, ve kterých došlo ke změně hodnoty úhlové polohy. Tyto vzorky jsou následně extrahovány do nových vektorů *_zmen.

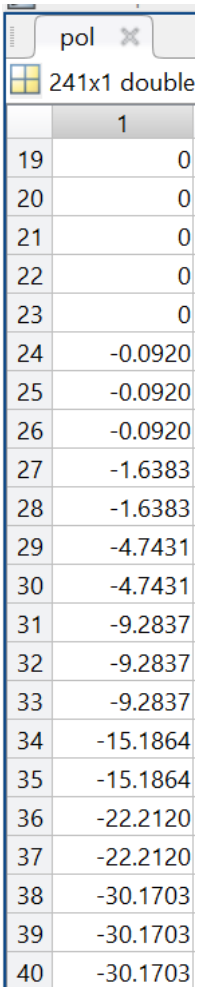
Výpočet úhlové rychlosti probíhá klasickou numerickou metodou pomocí dopředného diferenciálního podílu mezi dvěma sousedními body (10):

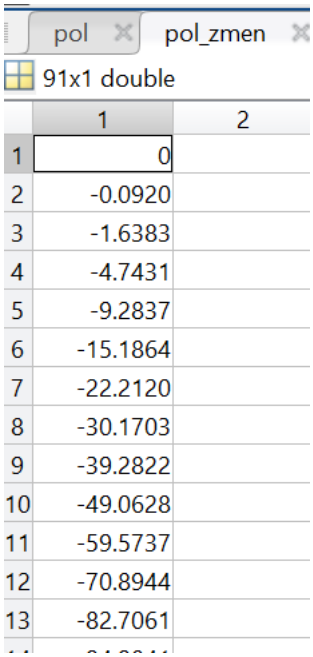
$$\omega_i = \frac{\theta_i - \theta_{i-1}}{t_i - t_{i-1}} \tag{10}$$

Pro zachování konzistentní délky výstupního signálu je na konec výsledného vektoru „omega_zmen“ doplněna poslední vypočtená hodnota. Výstupem skriptu jsou vektory „rid_zmen“ a „omega_zmen“, které tvoří vstupní a výstupní signál pro následnou identifikaci přenosové funkce pomocí nástroje „ident“.



Obrázek 139 - Skript v Matlabu pro úpravu dat získaných z TIA Portalu





Obrázek 140 - Vzorek dat z TIA Portalu po zpracování

Obrázek 141 - Vzorek dat z TIA Portalu před zpracováním

Po předzpracování dat jsem vektory rid_zmen (vstupní signál) a omega_zmen (vypočtenou úhlovou rychlost) importoval do nástroje System Identification Toolbox pomocí funkce ident. V dialogovém okně „Import Data“ (Obrázek 142) jsem zvolil možnost Time Domain Signals, protože jsem pracoval s časovými průběhy signálů.

Do pole Input jsem zadal proměnnou rid_zmen, do pole Output proměnnou omega_zmen. Tyto proměnné jsem měl připravené ve Workspace Matlabu a ověřil jsem, že mají stejnou délku – což je nezbytná podmínka pro úspěšné načtení dat, protože ke každému vstupnímu vzorku musí existovat odpovídající výstupní hodnota.

Před samotným importem jsem zadal i hodnotu vzorkovací doby (Sample time), kterou jsem spočítal z časového vektoru cas_zmen podle vztahu (12) :

$$T_s = \frac{t_n - t_1}{n-1} \tag{12}$$

kde:

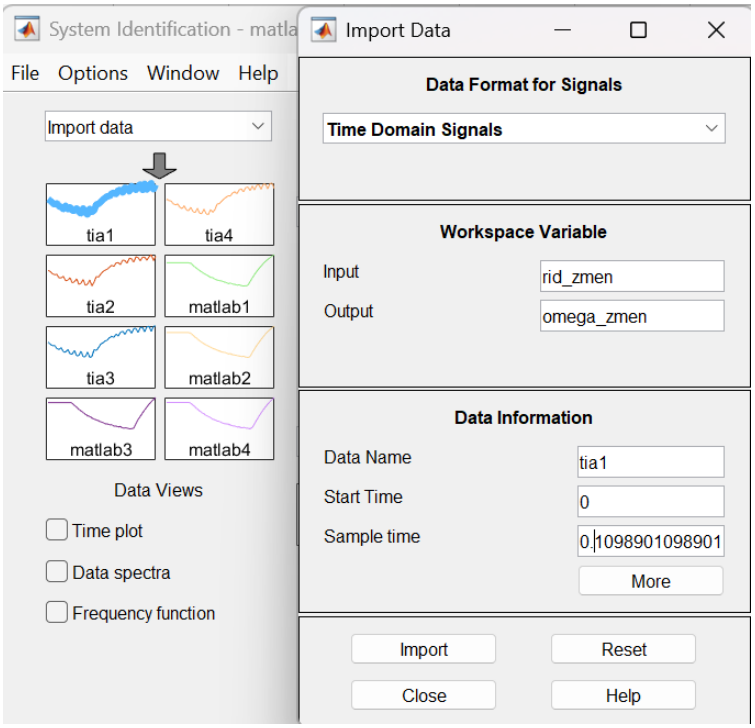
T_s je vzorkovací doba,

t_1 *a* t_n jsou časové značky prvního a posledního vzorku,

n je počet vzorků.

Například při délce vektoru cas_zmen rovné 91 a rozsahu od -0,01 do 9,91 s vychází vzorkovací doba jako (13):

$$T_s = \frac{9{,}93-(-0{,}01)}{91-1} = \frac{9{,}94}{90} \approx 0{,}1104\ \text{s} \tag{13}$$



Obrázek 142 - Vkládání dat do System Identification

Po úspěšném importu dat do System Identification Toolboxu jsem přistoupil k odhadu přenosové funkce systému. Využil jsem volbu Estimate Transfer Functions (Obrázek 143), kde jsem ručně nastavil strukturu modelu. Do pole Model name jsem zadal název datové sady, v mém případě tia1. V části Orders and Domain jsem zvolil 1 pól a 0 nul, což odpovídá očekávanému chování systému.

Tato volba vychází z fyzikálního modelu stejnosměrného motoru uvedeného v dokumentaci výrobce (Inteco), kde je systém popsán jako první řád bez nul. Matematický model systému je tvořen diferenciální rovnicí (14):

$$T_s \cdot \frac{d\omega(t)}{dt} + \omega(t) = K_s \cdot u(t) \tag{14}$$

kde:

T_s je časová konstanta motoru [s],

K_s je zesílení systému [rad/s],

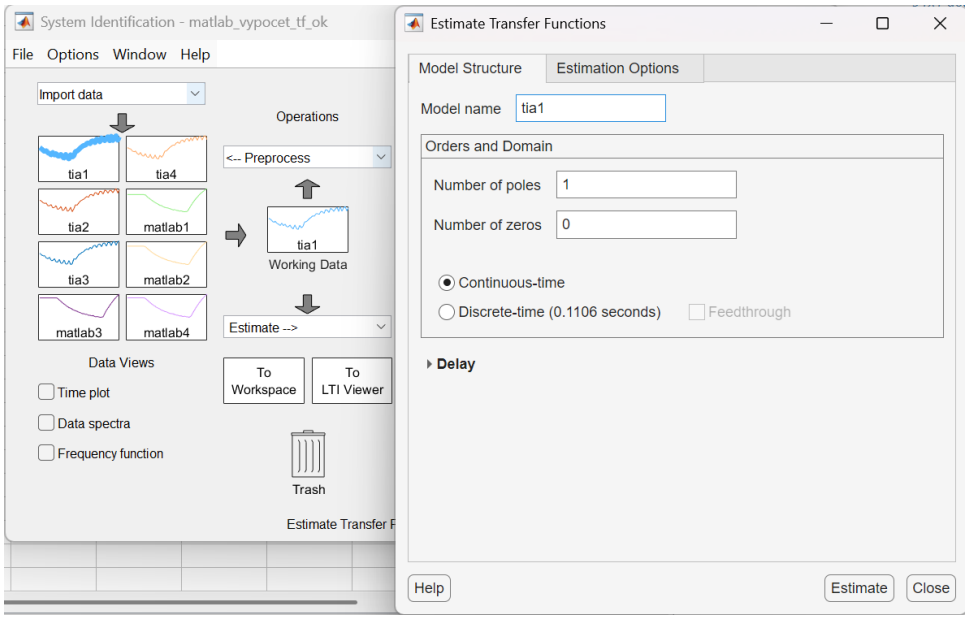
$u(t)$ je vstupní řídicí signál (normalizovaný napěťový signál),

$\omega(t)$ je výstupní úhlová rychlost [rad/s].

Odpovídající přenosová funkce ve frekvenční oblasti pak má tvar (15):

$$G(s) = \frac{K_s}{T_s s + 1} \tag{15}$$

což je přesně model s jedním pólem a žádnou nulou. Předpokládal jsem, že motor je modelován v lineárním rozsahu, bez saturací a nelinearit, takže jsem ponechal volbu Continuous-time aktivní. Parametry modelu byly následně odhadnuty na základě metody nejmenších čtverců, kterou nástroj ident používá pro odhad lineárních systémů.



Obrázek 143 - Struktura modelu pro výpočet přechodové funkce

Stejným způsobem jsem následně zpracoval i ostatní datové sady. Pro každou jsem provedl základní předzpracování, výpočet úhlové rychlosti a import do nástroje System Identification Toolbox, kde jsem provedl identifikaci přenosové funkce. Ve všech případech jsem zachoval jednotnou strukturu modelu s jedním pólem a žádnou nulou, která odpovídá dynamice daného systému. Výsledkem je přehledné okno se zobrazením nalezené přenosové funkce, včetně zesílení a polohy pólu.

```
From input "u1" to output "y1":
 -190.2
---------
s + 1.145

Name: tia1
Continuous-time identified transfer function.

Parameterization:
```

Obrázek 144 - Výstup výpočtu přenosové fukce v System Identification

V tabulkách (Tabulka 2, Tabulka 3) a v rovnicích: (16), (17), (18), (19), (20), (21), (22), (23), níže jsou uvedeny jednotlivé parametry každého měření.

Tabulka 2 - Výsledky výpočtů TIA Portal

TIA Portal			
Model	K	Pól	T [s]
tia1	−190,2	-1,145	0,873
tia2	−206,8	-1,247	0,802
tia3	−208,1	-1,246	0,802
tia4	−198,4	-1,188	0,841

$$G_{\text{tia1}}(s) = \frac{-190.2}{s+1.145} \tag{16}$$

$$G_{\text{tia2}}(s) = \frac{-206.8}{s+1.247} \tag{17}$$

$$G_{\text{tia3}}(s) = \frac{-208.1}{s+1.246} \tag{18}$$

$$G_{\text{tia4}}(s) = \frac{-198.4}{s+1.188} \tag{19}$$

Tabulka 3 - Výsledky výpočtů Matlab/Simulink

Matlab/Simulink			
Model	K	Pól	T [s]
matlab1	−207,6	-1,293	0,774
matlab2	−204	-1,273	0,785
matlab3	−206,3	-1,297	0,771
matlab4	−205,1	-1,284	0,779

$$G_{\text{matlab1}}(s) = \frac{-207.6}{s+1.293} \tag{20}$$

$$G_{\text{matlab2}}(s) = \frac{-204}{s+1.273} \tag{21}$$

$$G_{\text{matlab3}}(s) = \frac{-206.3}{s+1.297} \tag{22}$$

$$G_{\text{matlab4}}(s) = \frac{-205.1}{s+1.284} \tag{23}$$

Na základě vypočtených průměrných hodnot parametrů přenosových funkcí z dat získaných v prostředích Matlab a TIA Portal jsem porovnal jejich shodu pomocí absolutní a relativní chyby (Tabulka 4). Výsledky ukazují velmi dobrou konzistenci mezi oběma modely.

Zesílení systému K se mezi oběma sadami dat liší v průměru o 4,875, což odpovídá relativní chybě 2,369 %. Tato odchylka je velmi malá a potvrzuje, že modely popisují velikost odezvy systému téměř shodně.

Průměrná hodnota pólu u modelů z TIA činí -1,2065, zatímco u Matlab modelů -1,28675. Absolutní chyba tak dosahuje 0,08025, což představuje relativní chybu 6,237 %. Všechny póly se nacházejí v levé polorovině, což potvrzuje stabilitu všech identifikovaných modelů.

Největší rozdíl byl pozorován u časové konstanty T, kde průměrná hodnota z TIA je 0,8295 a z Matlabu 0,77725. Rozdíl 0,05225 odpovídá relativní chybě 6,722 %. Tato odchylka je rovněž nízká a neovlivňuje celkové dynamické chování systému zásadním způsobem.

Lze tedy uzavřít, že identifikace systémů v obou prostředích poskytla velmi podobné výsledky a že rozdíly v hodnotách jsou zanedbatelné z pohledu technické relevance.

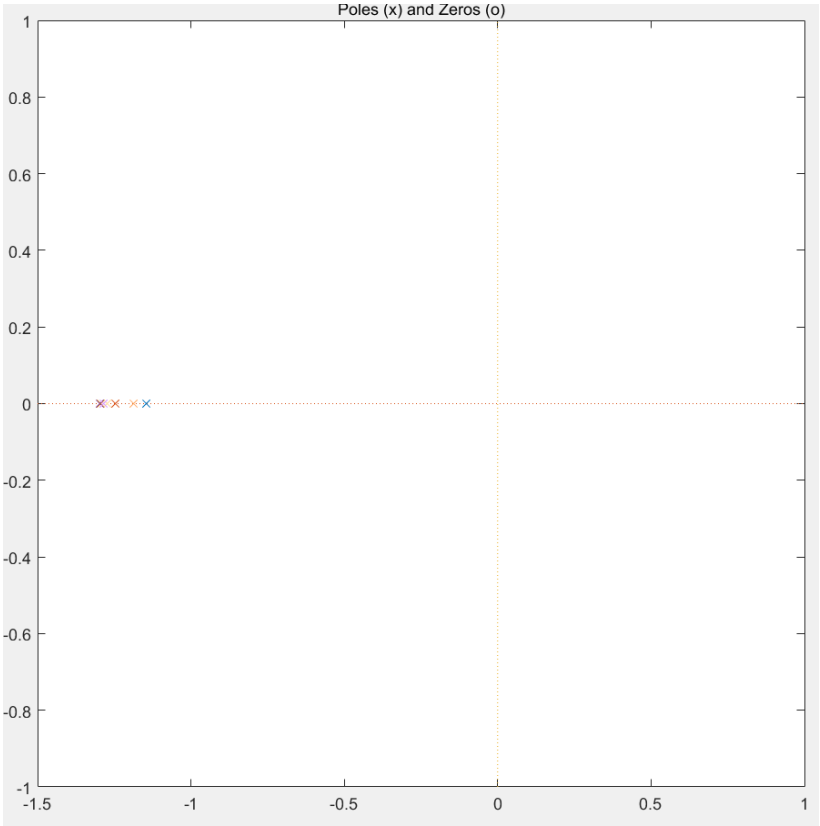
Tabulka 4 - Porovnání modelových parametrů

Porovnání modelových parametrů				
	Tia	Matlab	Abs. chyba	Rel. chyba [%]
Průměr K	-200,875	-205,75	4,875	2,369380316
Průměr Pólů	-1,2065	-1,28675	0,08025	6,236642704
Průměr T [s]	0,8295	0,77725	0,05225	6,722418784

Na základě zobrazení pólů v komplexní rovině lze jednoznačně potvrdit, že všechny identifikované modely jsou stabilní. V grafu (Obrázek 145) jsou viditelné pouze póly (označené křížky), protože struktura modelu neobsahuje žádné nuly a pracuje s jedním pólem.

Všechny póly se nacházejí v levé polorovině, přibližně v rozmezí od -1,15 do -1,30 na reálné ose. Žádný z nich neleží na imaginární ose ani v pravé části roviny, což potvrzuje asymptotickou stabilitu všech modelů. Výstupy těchto systémů tedy při konstantním vstupu po čase konvergují k ustálené hodnotě bez oscilací či nestability.

Tento výsledek odpovídá očekávanému chování reálného motoru, který je z principu stabilní systém s jedním dominantním časovým pólem.



Obrázek 145 – Póly přenosových funkcí z vypočítaných modelů

2.5.2 Porovnávání získaných výsledků z úlohy pro měření statické charakteristiky

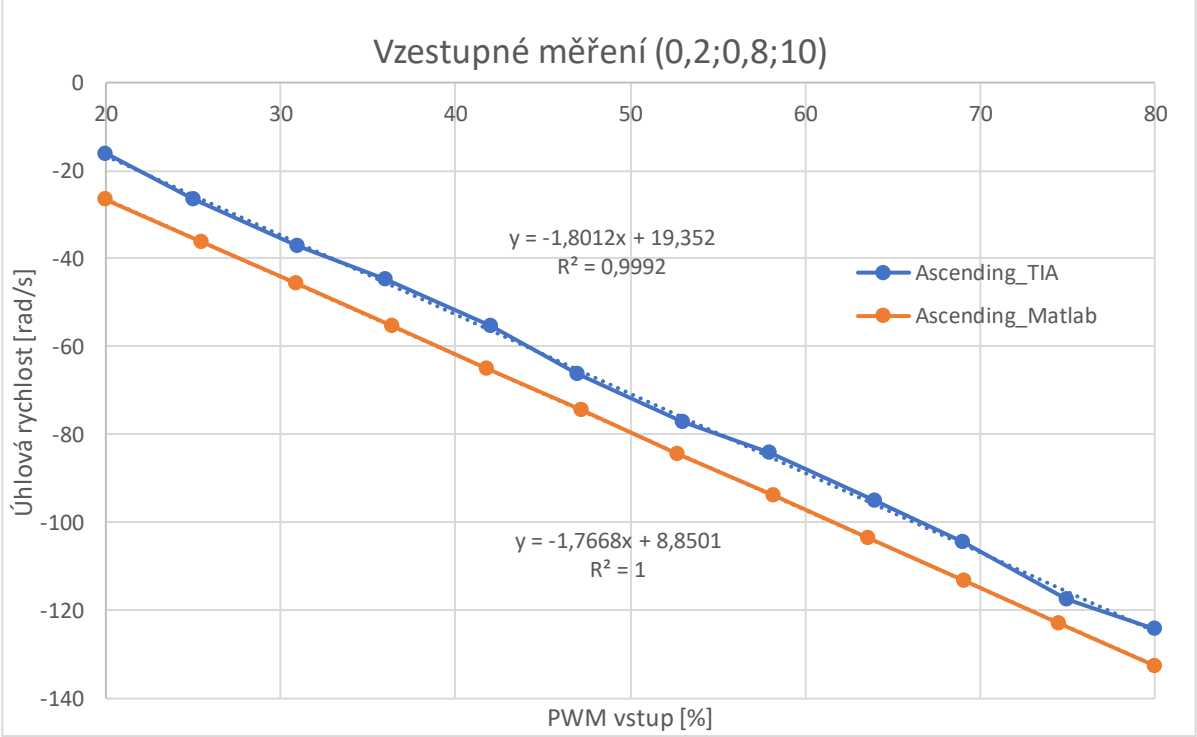
Porovnání statických charakteristik slouží k vyhodnocení vztahu mezi vstupním a výstupním signálem za ustáleného stavu systému. Cílem této části je zjistit, zda se výstupy systémů identifikovaných v různých prostředích (MATLAB/Simulink a TIA Portal) chovají shodně při stejných vstupních hodnotách. Statická charakteristika zároveň umožňuje odhadnout zesílení systému a odhalit případné nelinearity nebo odchylky způsobené například způsobem měření či přepočtem dat.

Jak jsem získal data z TIA portalu je uvedeno na konci poslední vypracované úlohy. V aplikaci v Matlabu nejdřív změřím samotnou charakteristiku. Po dokončení samotného měření (pomocí tlačítka **Run**, které postupně aktivuje motor pro různé hodnoty řídicí veličiny, dokud nedojde k ustálenému stavu), jsou všechna měření uložena do souboru ChStat.mat. Tento soubor obsahuje data získaná z tachogenerátoru i z inkrementálního snímače.

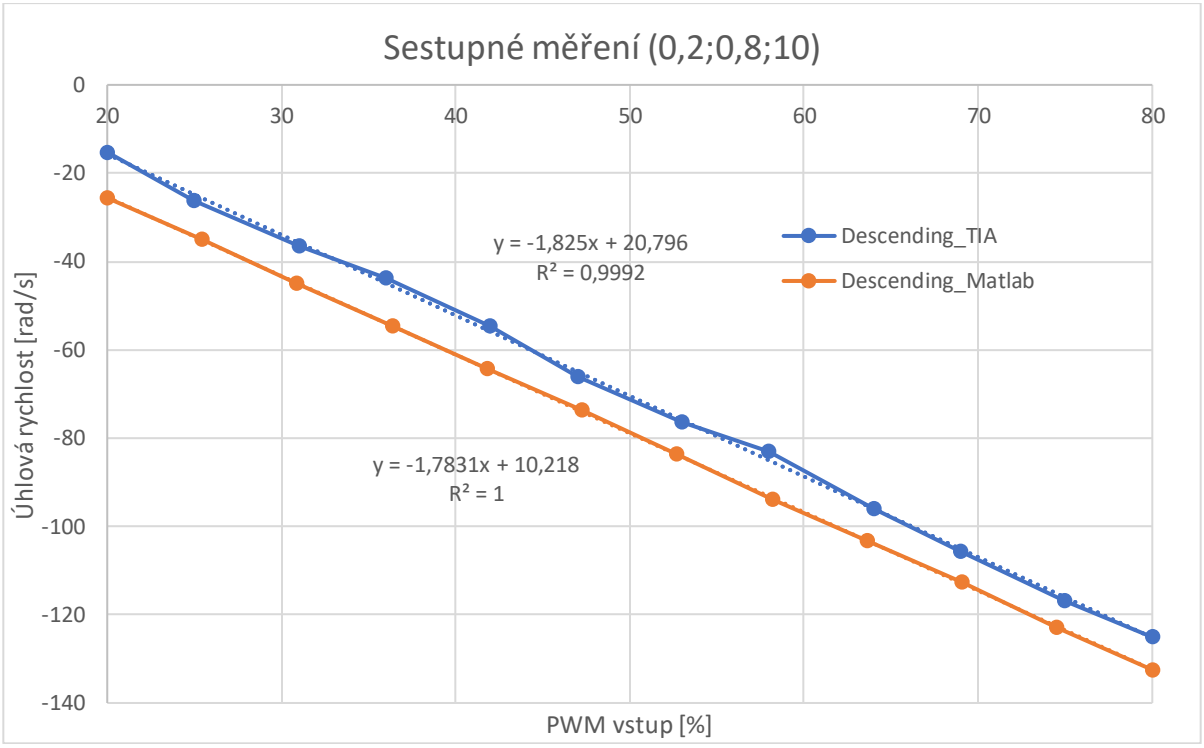
2.5.2.1 Porovnání lineárních oblastí statických charakteristik

V této části jsem se zaměřil na porovnání lineárních oblastí statických charakteristik motoru získaných pomocí TIA Portalu a prostředí MATLAB. Pro objektivní srovnání jsem zvolil úseky vstupního napětí v rozsahu od 0,8 do 0,2, ve kterých se odezva systému chová přibližně lineárně.

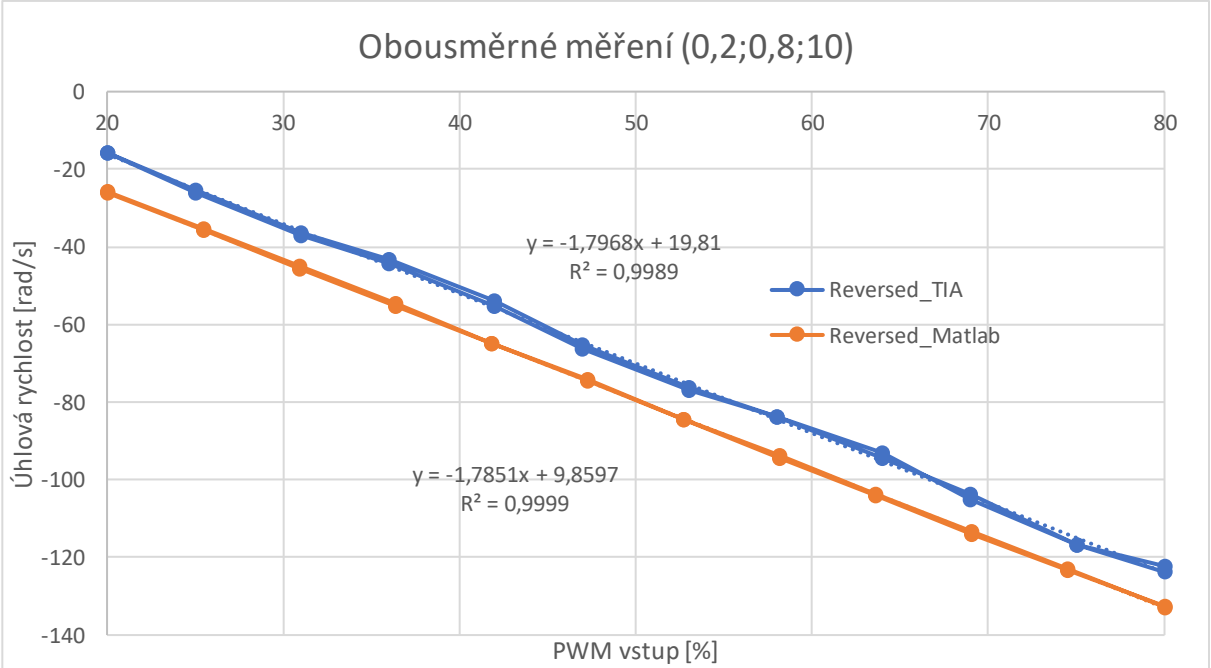
Pomocí lineární regrese jsem pro každý průběh určil rovnici přímky a vypočítal hodnotu statického zesílení. Zároveň jsem zaznamenal hodnotu koeficientu determinace R^2, který vyjadřuje míru shody mezi modelem a naměřenými daty.



Graf 1 – Vzestupné měření lineární oblasti statické charakteristiky



Graf 2 - Sestupné měření lineární oblasti statické charakteristiky



Graf 3 - Obousměrné měření lineární oblasti statické charakteristiky

V jednotlivých grafech je znázorněn průběh statických charakteristik získaných v prostředí MATLAB a TIA Portal. U všech tří případů – vzestupného (Graf 1), sestupného (Graf 2) i reverzního (Graf 3) měření – je patrná vysoká míra shody, což potvrzují také hodnoty koeficientu determinace R^2 (Tabulka 5), které se ve všech případech pohybují velmi blízko hodnotě 1. Ze sklonu spojnic trendu v grafech byly následně určeny hodnoty statického zesílení

K_S (Tabulka 5). Rozdíly mezi oběma sadami dat jsou minimální a lze je přičíst zejména odlišné metodice výpočtu úhlové rychlosti.

Tabulka 5 - Statické zesílení lineární části statické charakteristiky

Statické zesílení K_s (0,2;0,8;10)			
	Ascending	Descending	Reversed
TIA Portal	-1,8012	-1,825	-1,7968
Matlab	-1,7668	-1,7831	-1,7851
Abs. chyba	0,0344	0,0419	0,0117
Rel. chyba	1,9470	2,3498	0,6554

Pro vyhodnocení přesnosti výpočtu statického zesílení K_s jsem provedl výpočet absolutní a relativní chyby mezi hodnotami získanými v prostředí TIA Portal a MATLAB. Za referenční hodnotu byla zvolena hodnota zesílení vypočítaná v MATLABu, a to z důvodu vyšší numerické přesnosti a použití přesně definovaných měřicích bodů.
Absolutní chyba ε_{abs} byla vypočítána podle vztahu (24):

$$\varepsilon_{abs} = \left| K_{s,\mathrm{TIA}} - K_{s,\mathrm{Matlab}} \right| \tag{24}$$

Relativní chyba ε_{rel} pak udává velikost této odchylky vzhledem k hodnotě z MATLABu a je definována vztahem (25):

$$\varepsilon_{rel} = \left(\frac{\left| K_{s,\mathrm{TIA}} - K_{s,\mathrm{Matlab}} \right|}{\left| K_{s,\mathrm{Matlab}} \right|} \right) \cdot 100 \tag{25}$$

Z hodnot statického zesílení vyplývá (Tabulka 5), že rozdíly mezi oběma prostředími jsou minimální. Největší absolutní chyba činí přibližně 0,0419 (pro sestupné měření), zatímco nejnižší byla zaznamenána u reverzního měření (0,0117). Relativní chyba se pohybuje do 2,35 %, což naznačuje dobrou shodu mezi oběma metodami měření a výpočtu.

Tabulka 6 - Koeficient determinace R^2

Determinace R^2 (0,2;0,8;10)			
	Ascending	Descending	Reversed
TIA Portal	0,9992	0,9992	0,9989
Matlab	1	1	0,9999

Hodnoty determinace R^2 (Tabulka 6) potvrzují vysokou míru korelace mezi vstupem a výstupem. MATLAB dosahuje v rámci daného rozsahu téměř ideální hodnoty 1, zatímco

výsledky z TIA Portalu jsou jen nepatrně nižší. Tato skutečnost podporuje závěr, že linearita dat je velmi dobrá a případné rozdíly jsou způsobeny převážně metodikou zpracování dat, například výpočtem úhlové rychlosti.

Pro detailní porovnání jednotlivých hodnot úhlové rychlosti mezi měřením v TIA Portalu a MATLABu jsem musel přistoupit k interpolaci. Důvodem byla skutečnost, že odpovídající hodnoty vstupního signálu v obou prostředích nejsou shodné a přímé porovnání by tak nebylo objektivní (Tabulka 7). Abych zajistil co nejpřesnější vyhodnocení, dopočítal jsem hodnoty z TIA Portalu tak, aby odpovídaly konkrétním vstupním hodnotám z MATLABu.

Tabulka 7 - Změřená data pro tvorbu vzestupné statické charakteristiky

Data vzestupného měření statické charakteristiky			
Matlab		TIA Portal	
PWM [%]	Úhlová rychlost [rad/s]	PWM [%]	Úhlová rychlost [rad/s]
20	-26,635137	20	-16,0677
25,44567	-36,12543882	25	-26,53399
30,89133	-45,61852292	31	-37,13596
36,36142	-55,30971905	36	-44,73372
41,80708	-65,12988489	42	-55,37446
47,25275	-74,53877233	47	-66,18963
52,72283	-84,25882707	53	-77,218
58,1685	-93,92876672	58	-84,19553
63,61416	-103,4663918	64	-94,95255
69,08425	-113,2140527	69	-104,3335
74,52991	-122,9324713	75	-117,5908
80	-132,5258561	80	-124,1806

Pro výpočet jsem využil lineární interpolaci mezi dvěma nejbližšími body v TIA datech, které danou hodnotu obklopují. Tímto způsobem jsem získal odpovídající výstupní hodnotu úhlové rychlosti v TIA, která se vztahuje ke každé hodnotě PWM z MATLABu. Použitý vzorec lineární interpolace má obecnou podobu (26):

$$y = y_1 + \left(\frac{x - x_1}{x_2 - x_1}\right) \cdot (y_2 - y_1) \tag{26}$$

kde x je požadovaná hodnota PWM (z MATLABu), x_1 a x_2 jsou nejbližší hodnoty PWM z TIA, a y_1, y_2 odpovídající výstupní hodnoty úhlové rychlosti.

Například pro vstupní hodnoty (Tabulka 7):

$x = 30{,}89133, \quad x_1 = 25, \quad y_1 = -26{,}53399, \quad x_2 = 36, \quad y_2 = -44{,}73372$

Dosazením do vzorce (26) se získá rovnice (27):

$$y = -26{,}53399 + \left(\frac{30{,}89133-25}{36-25}\right) \cdot (-44{,}73372 + 26{,}53399) \quad (27)$$

$$y = -36{,}28132014$$

Stejným postupem byly v excelu dopočítány všechny interpolované hodnoty, výsledky viz. Tabulka 8, Tabulka 9, Tabulka 10

Tabulka 8 - Interpolace hodnot vzestupného měření

Interpolace hodnot vzestupného měření				
Interpolace y [rad/s]	Abs. Chyba	Rel. Chyba [%]	Prům. Abs. chyby	8,954699234
-16,0677	10,567437	39,67479875	Prům. Rel. Chyby [%]	14,77977711
-26,49776323	9,627675598	26,65068138		
-36,28132014	9,33720278	20,46800769		
-46,02543203	9,284287018	16,78599562		
-56,06064951	9,069235379	13,92484478		
-65,80524134	8,733530994	11,71676259		
-75,55733488	8,701492182	10,32709864		
-85,5508172	8,377949517	8,919471436		
-94,47351713	8,992874645	8,691590081		
-105,4160502	7,798002489	6,887839722		
-114,3110242	8,62144707	7,01315688		
-124,1806	8,345256136	6,297077702		

Tabulka 9 – Interpolace hodnot sestupného měření

Interpolace hodnot sestupného měření				
Interpolace y [rad/s]	Abs. Chyba	Rel. Chyba [%]	Prům. Abs. chyby	8,462892692
-125,0916	7,397670334	5,583599574	Prům. Rel. Chyby [%]	14,47649091
-115,3954757	5,886123132	4,792560387		
-105,6639079	7,138993983	6,332696454		
-94,55659022	7,229730897	7,004906266		
-85,59165276	10,73571772	11,44682814		
-74,86941111	7,210879779	8,625923777		
-65,09332916	7,796628739	10,56562153		
-55,4733757	9,656641876	14,98774503		
-45,43452131	10,96062688	20,04976943		
-35,56012586	8,294354756	18,49401685		
-25,78907362	8,874698387	25,32701871		
-15,23427	10,37264582	40,50720474		

Tabulka 10 - Interpolace hodnot obousměrného měření

Interpolace hodnot obousměrné měření				
Interpolace y [rad/s]	Abs. Chyba	Rel. Chyba [%]	Prům. Abs. chyby	9,334482776
-15,69944	10,27265267	39,55265677	Prům. Rel. Chyby [%]	15,19202669
-26,28303139	9,670615018	27,10275794		
-35,80990741	8,500153804	18,64969654		
-45,99562955	11,00015691	19,88909911		
-55,93097083	9,322536389	14,40953508		
-65,63864807	8,045239145	10,81777709		
-75,4409141	7,670425661	9,073179532		
-84,48249695	10,50694673	11,13571405		
-94,64888773	10,97488552	10,54796958		
-104,082436	9,148503882	8,013734697		
-113,6739587	6,536985292	5,296107973		
-123,7736	10,3646923	7,816091975		
-122,2424	8,971915326	6,758733283		
-113,7787096	6,164386978	5,021221253		
-104,6464545	9,529611604	8,418079051		
-93,96635855	9,278141272	8,951481027		
-84,66110401	9,845470685	10,50832302		
-74,83651534	8,494331662	10,04658873		
-64,52079829	9,176914901	12,36075073		
-54,68028453	10,71665149	16,56472967		
-44,91804853	11,58598492	21,1992348		
-34,90251477	8,750590502	19,42282457		
-25,94815457	9,807945797	27,78823641		
-15,79635	9,949445198	38,64493259		

Na základě uvedených tabulek lze konstatovat, že výsledky interpolace hodnot jednotlivých měření vykazují poměrně konzistentní odchylky mezi daty získanými v prostředí TIA Portal a MATLAB. Průměrné absolutní chyby se pohybují v rozmezí přibližně 8,5 až 9,3 rad/s, což značí, že velikost rozdílu v úhlové rychlosti je relativně stabilní napříč různými typy měření (vzestupné, sestupné i obousměrné). Tato stabilita může být způsobena systematickou odchylkou při výpočtu úhlové rychlosti v prostředí TIA Portal.

V případě relativních chyb je patrné, že průměrná relativní chyba ve vzestupném a sestupném měření dosahuje hodnot okolo 14,5 %, zatímco u obousměrného měření je mírně vyšší (15,2 %). To může být způsobeno složitějším chováním systému při změně směru otáčení, které je náročnější na přesnou synchronizaci mezi oběma softwarovými platformami.

Celkově lze říct, že ačkoliv rozdíly existují, interpolací hodnot a následným vyčíslením absolutních a relativních chyb bylo možné přesněji kvantifikovat míru odchylky mezi jednotlivými měřeními. Tyto výstupy potvrzují, že navržený způsob porovnání dat pomocí

lineární interpolace poskytuje užitečný a dobře interpretovatelný přehled o přesnosti mezi dvěma nezávislými měřicími prostředími.

Při měření statické charakteristiky jsem porovnal chování systému v prostředí MATLAB a TIA Portal v lineární oblasti vstupního signálu. Výsledky ukázaly velmi dobrou shodu mezi oběma měřicími platformami, kterou jsem ověřil pomocí výpočtu statického zesílení a hodnot koeficientu determinace R^2. Doplňkovou analýzou pomocí interpolace a výpočtu absolutní a relativní chyby jsem potvrdil konzistenci měření.

ZÁVĚR

Bakalářská práce se zaměřuje na návrh a realizaci výukových úloh souvisejících s řízením servopohonu v prostředí průmyslové automatizace. Hlavním cílem této práce je vytvořit ucelený soubor úloh využitelných pro výuku práce s programovatelnými automaty Siemens S7-1200 a vývojovým prostředím TIA Portal, přičemž zároveň umožňuje porovnání naměřených dat s referenčními výstupy z prostředí MATLAB/Simulink.

První navržená úloha je koncipována jako úvod do prostředí TIA Portal. Student se v ní seznámí se základní strukturou PLC programu, konfigurací hardwaru, vytvářením proměnných a nahráváním programu do automatu. Tato část nemá za cíl generovat měřitelná data, ale sloužit k osvojení základních principů práce s prostředím a reálným hardwarem.

Další úlohy jsou již zaměřeny na analýzu a měření konkrétních vlastností řízeného systému. V úloze zaměřené na impulsní odezvu systému je motor řízen pomocí pulzně šířkového signálu (PWM) a výstupní poloha je zaznamenána. Data jsou následně zpracována v prostředí Matlab, kde je proveden výpočet úhlové rychlosti a identifikace přenosové funkce systému prvního řádu. Časové konstanty identifikované z úloh realizovaných v TIA Portalu se pohybují mezi 0,26 s a 0,32 s, zatímco referenční přenosové funkce z Matlabu mají nižší hodnoty, což odpovídá rozdílům v přesnosti měření a vzorkovací frekvenci.

V další úloze je měřena statická charakteristika systému. Pro různé úrovně PWM signálu jsou zaznamenávány odpovídající hodnoty výstupní polohy, ze kterých jsou pomocí lineární regrese určují hodnoty statického zesílení. Naměřené hodnoty se pohybují od 2,11 do 2,39. Hodnocení kvality přiblížení pomocí koeficientu determinace R^2 potvrzuje velmi dobrou shodu, přičemž ve všech případech překračuje hodnota 0,985. Relativní chyba vůči referenčním datům z Matlabu je v rozsahu 1,88 % až 8,95 %.

Závěrem lze konstatovat, že vytvořené úlohy úspěšně splňují svůj vzdělávací účel. Umožňují studentům pochopit nejen strukturu PLC programu a způsob řízení reálného systému, ale také postup zpracování a vyhodnocení měřených dat.

Úvodní část bakalářské práce navíc může sloužit jako praktický návod pro studenty začínající s TIA Portal a programováním PLC, a tedy i jako metodický základ pro laboratorní výuku.

Díky provedenému porovnání s referenčními výsledky lze úlohy využít i pro demonstraci základních metod identifikace dynamických systémů. Do budoucna je možné výukový obsah dále rozšířit například o úlohy zpětnovazebního řízení nebo regulaci podle různých typů vstupních signálů.

POUŽITÁ LITERATURA

1. **AutomationDirect. 2015.** History of the PLC. [Online] 2015. [Citace: 9. květen 2025.] https://library.automationdirect.com/history-of-the-plc/.
2. **Cloud Studio IoT. 2025.** PLCs and their future in the industry in 2025. [Online] 2025. [Citace: 9. květen 2025.] https://www.cloud.studio/plcs-and-their-future-in-the-industry-in-2025/.
3. **CODESYS Group. 2023.** CODESYS Development System. [Online] 2023. [Citace: 9. květen 2025.] https://www.codesys.com/products/engineering/development-system/.
4. **Control Engineering. 2024.** Back to the future of the PLC. [Online] 2024. [Citace: 9. květen 2025.] https://www.controleng.com/back-to-the-future-of-the-plc/.
5. **Empowered Automation Solutions LLC. 2023.** What are the 3 types of PLC. [Online] 2023. [Citace: 9. květen 2025.] https://www.empoweredautomation.com/what-are-the-3-types-of-plc.
6. **GeeksforGeeks. 2023.** What are the different types of PLC? [Online] 2023. [Citace: 9. květen 2025.] https://www.geeksforgeeks.org/what-are-the-different-types-of-plc/.
7. **IEC, International Electrotechnical Commission. 2013.** *IEC 61131-3:2013 Programmable controllers – Part 3: Programming languages.* Ženeva : IEC, 2013. IEC 61131-3:2013.
8. **Inteco Sp. z o.o. 2024b.** *Modular Servo System – USB2 Version: User's Manual.* Kraków : Inteco Sp. z o.o., 2024b. Interní dokumentace dostupná na pracovišti.
   —. 2024a. *PLC Interfaces – Inteco Servo USB.* Kraków : Inteco Sp. z o.o., 2024a. Interní dokumentace dostupná na pracovišti.
9. **MDPI. 2024.** The Development Trend of Programmable Logic Controller. [Online] 2024. [Citace: 9. květen 2025.] https://www.mdpi.com/2071-1050/16/14/6230.
10. **Mitsubishi Electric Corporation. 2023.** PLC engineering software (MELSEC) GX Works2. [Online] 2023. [Citace: 9. květen 2025.] https://www.mitsubishielectric.com/fa/products/cnt/plceng/smerit/gx_works2/index.html.
11. **PLCopen. 2016.** IEC 61131-3: a standard programming resource. [Online] 2016. [Citace: 9. květen 2025.] https://plcopen.org/sites/default/files/downloads/intro_iec_oct2016.pdf.

12. **PowerTechMax. 2024.** 18 Expert Future Prospects of Programmable Logic Controller PLC. [Online] 2024. [Citace: 9. květen 2025.] https://powertechmax.com/future-prospects-of-programmable-logic-controller/.
13. **ProjectSoft HK a.s. 2024.** *Laboratoř automatizace – rozváděč R0.* Hradec Králové : ProjectSoft HK a.s., 2024. Interní dokumentace dostupná na pracovišti.
14. **RL Consulting Inc. 2024.** The Future of PLC Programming: Trends and Innovations Shaping the Industry. [Online] 2024. [Citace: 9. květen 2025.] https://rlconsultinginc.com/the-future-of-plc-programming-trends-and-innovations-shaping-the-industry/.
15. **Rockwell Automation. 2023.** Studio 5000 Design Software. [Online] 2023. [Citace: 9. květen 2025.] https://www.rockwellautomation.com/en-us/products/software/factorytalk/designsuite/studio-5000.html.
16. **RS Components. 2023.** PLCs Programmable Logic Controllers - A Complete Guide. [Online] 2023. [Citace: 9. květen 2025.] https://uk.rs-online.com/web/content/discovery/ideas-and-advice/plcs-programmable-logic-controllers-guide.
17. **Siemens AG. 2025.** OPC UA – Structured data up to the cloud. [Online] 2025. [Citace: 7. květen 2025.] https://www.siemens.com/us/en/products/automation/industrial-communication/opc-ua.html.

    —. 2023. SIMATIC S7-1200 Basic Controllers. [Online] 2023. [Citace: 9. květen 2025.] https://new.siemens.com/global/en/products/automation/systems/industrial/plc/s7-1200.html.

    —. 2023. TIA Portal – Engineering Software. [Online] 2023. [Citace: 9. květen 2025.] https://new.siemens.com/global/en/products/automation/industry-software/automation-software/tia-portal.html.

    —. 2023. Totally Integrated Automation Portal. [Online] 2023. [Citace: 9. květen 2025.] https://www.siemens.com/us/en/products/automation/industry-software/automation-software/tia-portal.html.
18. **Šmejkal, Ladislav. 1999.** *PLC a automatizace 1.* Praha : BEN - technická literatura, 1999. ISBN 80-86056-58-9.