TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Propojení simulačního modelu dynamické soustavy s PLC

Bakalářská práce

Studijní program: B0714A270001 Mechatronika
Autor práce: **Tomáš Linhart**
Vedoucí práce: Ing. Petr Školník, Ph.D.
Ústav mechatroniky a technické informatiky

Liberec 2022

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Zadání bakalářské práce

Propojení simulačního modelu dynamické soustavy s PLC

Jméno a příjmení: **Tomáš Linhart**
Osobní číslo: M19000090
Studijní program: B0714A270001 Mechatronika
Zadávající katedra: Ústav mechatroniky a technické informatiky
Akademický rok: **2021/2022**

Zásady pro vypracování:

1. Seznamte se s možnostmi komunikace PLC řady Simatic S7-1500 (TCP/IP, MODBUS/TCP, OPC, IO-devices atd.) a možného využití pro komunikaci s prostředím Matlab+Simulink.
2. Realizujte v prostředí Matlab+Simulink simulační model dynamické soustavy s analogovými i digitálními I/O. Model doplňte o vizualizaci.
3. Simulační model propojte s PLC Simatic S7-1500. Pro komunikaci ověřte standardy TCP/IP, Modbus, OPC případně další.
4. Na PLC realizujte regulační algoritmus s ovládacím rozhraním na HMI panelu. Použijte bloky implementované v prostředí TIA portal a vytvořte návod pro konfiguraci bloků a nastavení komunikace.
5. Modifikujte úlohu tak, aby řízení úlohy respektive skupiny úloh bylo možno realizovat na jednom PC a řídit až osmi PLC (simulátor výrobního závodu). Ověřte funkčnost a vytvořte návod pro realizaci řízení a komunikací.

Rozsah grafických prací:	dle potřeby dokumentace
Rozsah pracovní zprávy:	30–40 stran
Forma zpracování práce:	tištěná/elektronická
Jazyk práce:	Čeština

Seznam odborné literatury:

[1] *MathWorks* [online]. [cit. 2021-10-10]. Dostupné z: https://www.mathworks.com/

[2] BERGER, Hans. Automating with SIMATIC S7-1500. 1. Germany: Publicis Publishing, 2014. ISBN 978-3-89578-404-0.

[3] BORDEN, Terry R., Richard A. COX a Richard A. COX. Technician's guide to programmable controllers. 6th ed. Clifton Park, NY: Delmar, Cengage Learning, c2013. ISBN 9781111544096. STENERSON, Jon a David DEEG.

[4] *Siemens Step 7 (TIA PORTAL) Programming, a Practical Approach, 2nd Edition*. Independently published, 2019. ISBN 978-1091474109.

Vedoucí práce:	Ing. Petr Školník, Ph.D. Ústav mechatroniky a technické informatiky
Datum zadání práce:	12. října 2021
Předpokládaný termín odevzdání:	16. května 2022

L.S.

prof. Ing. Zdeněk Plíva, Ph.D.
děkan

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

V Liberci dne 12. října 2021

Prohlášení

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

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

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

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

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

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

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

20. února 2022 Tomáš Linhart

Propojení simulačního modelu dynamické soustavy s PLC

Abstrakt

Práce se zaměřuje na možnosti komunikace průmyslového PLC S7--1500 s okolním světem. Její součástí je návrh simulačních modelů s digitálními i anologovými vstupy a výstupy v prostředí Matlab/Simulink, algoritmy řízení těchto modelů pomocí PLC a také následná komunikace využitím TCP/IP mezi těmito subjekty. Pro každý model je navíc realizovaná vizualizace, a to jak v Matlab/Simulink, tak na PLC HMI panelu. Následně se také zaměřuje na propojení více ovládacích PLC s jedním řídícím, díky čemuž je možné ovládat více simulací na jednom počítači.

Klíčová slova: PLC, MATLAB, SIMULINK, TCP/IP, TIA

Connection of simulation model of dynamic system with PLC

Abstract

This work is aiming at explaining different communication options of industrial PLC S7-1500 with other devices. Parts of thesis include designing simulation models with analog and digital inputs and outputs in Matlab/Simulink enviroment, controll algorithms of this models on PLC and main communication using TCP/IP protocoll running between them. There is also visualisation created for each model running in both Matlab/Simulink and HMI panel conected to the PLC. Lastly, thesis is focusing on connecting multiple controll PLCs with one master PLC, which allows running multiple simulation models on one computer.

Keywords: PLC, MATLAB, SIMULINK, TCP/IP, TIA

Poděkování

Rád bych poděkoval svému vedoucímu práce, panu Ing. Petru Školníkovi, Ph.D., za propůjčení potřebných zařízení k otestování funkčnosti komunikace a za odbornou pomoc při nastavování PID regulátorů.

Obsah

Seznam obrázků

Seznam zkratek

HMI	Human-Machine Interface - Rozhraní pro komunikaci s člověkem
HW	Hardware
ID	Identifier
IP	Internet Protocol
IO	Input/Output - Vstupy/Výstupy
PID	Proportional, Integral and Derivative - Proporcionální, Integrační a Derivační
PLC	Programmable logic controller -Programovatelný logický automat
TCP	Transmission Control Protocol
UDP	User Datagram Protocol

ÚVOD

Při učení se programování průmyslových PLC je velký problém s testováním funkčnosti naprogramovaného řešení. Pro toto ověření je většinou zapotřebí sestrojit fyzický model, který se tímto PLC dal ovládat. Sestrojení takového modelu je však finančně i časově náročné a většinou jej může využívat pouze jeden student najednou. Pokusím se tedy nabídnout řešení tohoto problému simulací trénovacích modelů v prostředí Matlab/Simulink a jejich následnou komunikaci s PLC. Díky možnosti běhu této simulace na běžném počítači je možné vyučovat programování těchto PLC efektivněji a také s menšími finančními náklady.

V této práci nejdříve představím různé metody komunikace zvoleného PLC, které by mohly být použity pro připojení k simulačnímu modelu. Následně nastíním postup navrhování takových modelů a to včetně grafické vizualizace a ovládání. Dalším krokem je zajištění samotné komunikace mezi PLC a Matlab/Simulink. Zaměřím se jak na nastavení HW nastavení PLC, tak i využití komunikačních bloků a jejich správnou konfiguraci na obou stranách spojení. Navrhnu také možné řídící algoritmy pro ovládání těchto úloh a to včetně vizuálního ovládacího rozhraní na HMI panelu. V poslední části se zabývám možnou modifikací komunikace pro více simulačních modelů běžících na jednom počítači a ovládaných více PLC najednou.

1 Komunikační možnosti PLC

PLC Siemens S7-1500 nabízí široký výběr možností komunikace s okolním světem. Ne všechny jsou však vhodné pro propojení s prostředím Matlab/Simulink. V této části představím výběr komunikačních možností a jejich případnou realizaci na obou stranách spojení.

1.1 TCP/IP

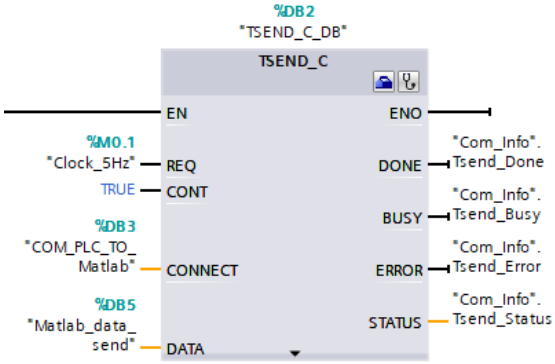
TCP/IP(Transmission Control Protocol) je spolu s UDP základním komunikačním protokolem. Narozdíl od UDP však před odesíláním informací naváže spojení s protistranou a spojení udržuje po celou dobu komunikace. Díky tomu je možné znovu odesílat data ztracená po cestě a zajistit tak spolehlivý přenos dat. Přenášená data jsou nejdříve rozdělena do paketů, které se následné odesílají samostatně. Pakety mohou cestovat sítí rozdílně a do výsledné zprávy se složí až u příjemce. Díky tomu lze při nedoručení paketu odeslat znovu pouze jeden a ne celou zprávu. Pro zahájení komunikace (ať už jde o odesílání či přijímání dat) je potřeba, aby se jedna strana chovala jako server a druhá jako klient. [1]

1.1.1 TCP/IP implementace v PLC

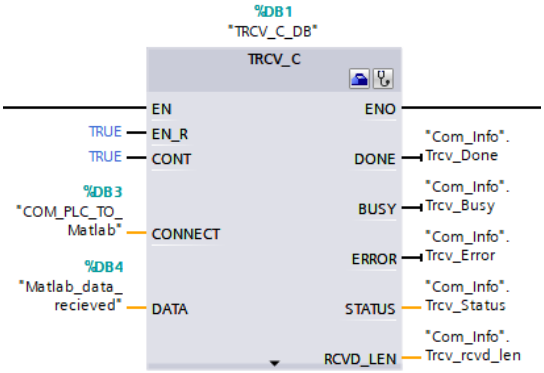
PLC Siemens S7-1500 (TIA portal V15) má pro tuto základní komunikaci přímo implementováno několik bloků. V záložce *communication/open user communication* nalezneme 2 bloky (**TSEND_C** a **TRCV_C**), které jsou pro tuto komunikaci s PC zapotřebí. Oba tyto bloky potřebují soubor pro nastavení komunikace, který se připojuje ke vstupu connect. Tento soubor obsahuje jak IP adresu a port koncového zařízení, se kterým má PLC spojení navázat, tak i další nastavení komunikace, jako například ID spojení nebo informaci o tom, kdo spojení aktivně vytváří. [2]

TSEND_C blok - po přijetí náběžné hrany signálu na vstup REQ odešle data (proměnnou, pole proměnných, nebo celý datablok) z portu DATA, příjemci, nastaveném v souboru Connect.

TRCV_C blok - nevyžaduje žádný hodinový signál, ale poslouchá neustále. Po přijetí informace z venkovní adresy nastavené v souboru Connect, zapíše přijatá data do databáze nastavené na vstupu DATA.



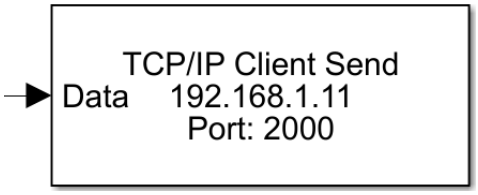
Obrázek 1.1: TSEND_C blok



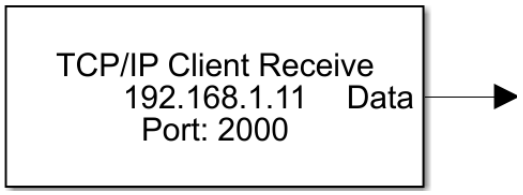
Obrázek 1.2: TRCV_C blok

1.1.2 TCP/IP implementace v Matlab/Simulink

V prostředí Simulink je využití tohoto protokolu realizováno bloky **TCP/IP Client Send** a **TCP/IP Client Recieve**, které se nachází v rozšíření Instrument Controll Toolbox. Pro jejich nastavení je třeba znát IP adresu a port na PLC, velikost i typ dat, které se odesílají a přijímají. Na rozdíl od bloků v PLC, nelze nastavit, jestli je blok klient nebo server, ale může být pouze klient. Přesnějším nastavením těchto bloků se zabývám v kapitole 3.2



Obrázek 1.3: TCP Client Send blok



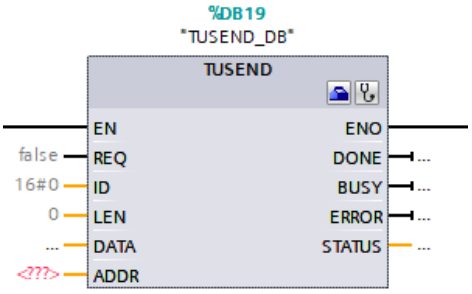
Obrázek 1.4: TCP Client Recieve blok

1.2 UDP

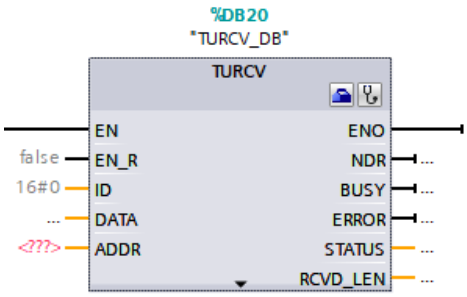
UDP (User Datagram Protocol) je méně spolehlivou alternativou k dříve zmiňovanému TCP/IP. Také data rozdělí na pakety, ale nezaručuje jejich doručení. Kvůli tomu, že před začátkem přenosu dat nenaváže spojení s protistranou, není možné znovu odeslat pakety ztracené po cestě. Z tohoto důvodu není UDP považováno za spolehlivý způsob komunikace.[3]

1.2.1 UDP implementace v PLC

Pro realizaci přenosu dat tímto protokolem má PLC implementované bloky **TUSEND** a **TURCV**, nacházející se v záložce *communication/open user communication/others*. Nastavení těchto bloků je mírně složitější, protože pro ně není v TIA Portal vytvořené grafické rozhraní jako pro bloky TSEND_C a TRCV_C, ale parametry potřebné ke zprovoznění komunikace jsou totožné jako u předchozího způsobu (TCP/IP).



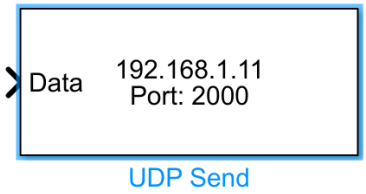
Obrázek 1.5: TURCV blok



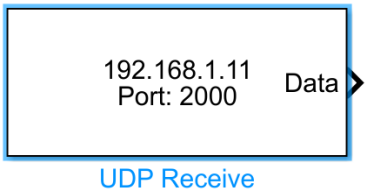
Obrázek 1.6: TUSEND blok

1.2.2 UDP implementace v Matlab/Simulink

Ovládací bloky pro tuto komunikaci se nachází taktéž v rozšíření Instrument Controll Toolbox a nazývají se velmi výstižně **UDP Send** a **UDP Recieve**. Tyto bloky se nastavují totožně s TCP/IP, až na velikost UDP zprávy, kterou je třeba dopočítat z odesílaných dat.



Obrázek 1.7: UDP Send blok



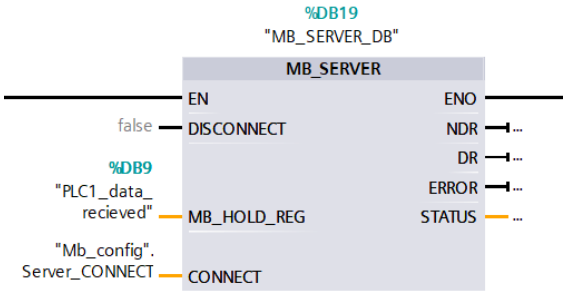
Obrázek 1.8: UDP Recieve blok

1.3 MODBUS/TCP

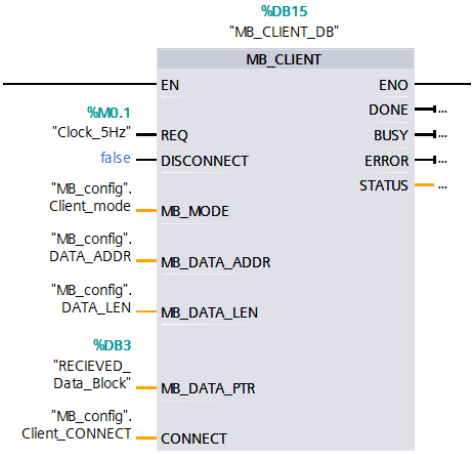
MODBUS/TCP je další komunikační protokol, který PLC Siemens řady S7 podporují. To ovšem neplatí pro Matlab/Simulink, který pro jeho využití potřeboval licenci k externímu programu (například Speedygoat).[4] Ověřil jsem proto alespoň jeho funkčnost pro komunikaci mezi dvěma PLC. V prostředí TIA portal jsou předpřipravené bloky přímo pro komunikaci využívající MODBUS/TCP. Ty se nachází v záložce *communication/others/Modbus TCP* a jsou jimi **MB Client** a **MB Server**.

Blok **MB Server** vytvoří, jak již z jeho jména vyplývá, server, ke kterému mohou následně bloky **MB Client** přistupovat. Na jeho vstup MB_HOLD_REGISTER se připojí Datablok(nebo přímo místo v paměti PLC), sloužící jako prostor, do kterého mohou klienti ukládat data, nebo z něj číst. Na vstup CONNECT se připojí soubor s nastavením komunikace, který obsahuje IP adresu, port klienta a ID připojení.

Blok **MB Client** má pak potřebných parametrů k nastavení více. Konkrétně vstup CONNECT, který jako u předchozího bloku obsahuje IP adresu a port protistrany (tentokrát serveru) a ID připojení. Dále však potřebuje mít na vstup MB_DATA_PTR připojený datablok, ze kterého se budou data odesílat, nebo do něj zapisovat. Funkce celého bloku se ale odvíjí od kombinace nastavení hodnot MB_MODE, MB_DATA_ADDR a MB_DATA_LEN. Pokud se například MB_MODE nastaví na 0, MB_DATA_ADDR na 40001 a MB_DATA_LEN na 20, přečte se po náběžné hraně vstupu REQ 20 byte dat ze serveru a zapíší se do připojeného databloku. Pro opačný směr (zapsání na server) lze využít MB_MODE = 1. V takovém případě se po náběžné hraně vstupu REQ odešle 20 byte z databloku do Hold registru serveru.[5]



Obrázek 1.9: MB Server blok



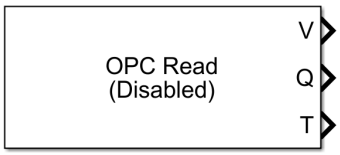
Obrázek 1.10: MB Client blok

Zapojení jedné úlohy využívající MODBUS/TCP je součástí přílohy.

1.4 OPC

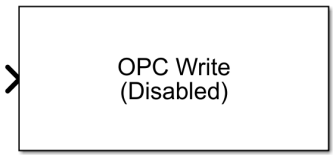
OPC UA je jeden z nejvyužívanějších způsobů komunikace a byl by jistě vhodný i pro komunikaci mezi PLC a simulačním modelem. Pro funkčnost této komunikace je však v PLC zapotřebí zakoupení runtime licence OPC UA, kterou v době psaní této práce nemám k dispozici. [6]

Nastavení OPC komunikace v Simulink je možné za použití celkem tří bloků tentokrát z rozšíření OPC Toolbox. Těmi jsou **OPC Config** , **OPC Read** a **OPC Write**. V bloku OPC Config se nastaví zařízení, se kterými mají ostatní bloky komunikovat. V nastavení **OPC Read** a **OPC Write** lze pak už pouze vybrat z proměnných nabízených klienty vhodnou proměnnou, kterou chceme číst, nebo do ní zapisovat.



Obrázek 1.11:
OPC Read blok

Obrázek 1.12:
OPC Config blok



Obrázek 1.13:
OPC Write blok

1.5 IO-devices

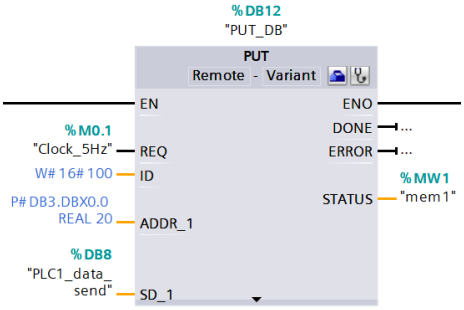
Protokol IO-devices je hojně využíván pro připojení externích Input/Output zařízení. Lze pomocí něho posílat Výstupní porty hlavního PLC na vstupní porty vedlejšího a naopak(propojení virtuálním drátem). Jeho využití pro komunikaci s MATLAB/Simulink však není možne. Otestoval jsem tedy alespoň komunikaci použitím IO-devices mezi dvěma PLC. Stačí aby byla ve stejné síti, jedno mělo zapnutý parametr IO controller a druhé IO device. Následně lze jednoduše nastavit, které BYTE z výstupů jednoho PLC se budou přeposílat na vstupy druhého.[7]

Příklad komunikace s využitím IO-devices je součástí přílohy.

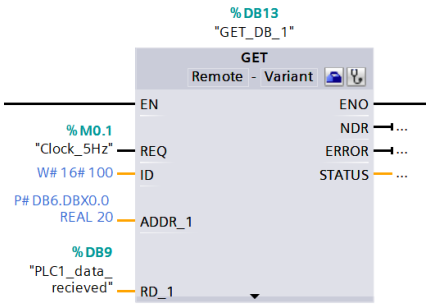
1.6 GET/PUT

Metoda GET/PUT je určená pouze pro komunikaci mezi PLC z rodiny S7 a je také jedním z nejjednodušších způsobů, jak mezi těmito PLC přesouvat data. V záložce *Device configuration/General/Protection &Security* na vedlejším PLC je zapotřebí povolit PUT/GET komunikaci, a Hlavní PLC pak může pomocí Bloků PUT a GET zapisovat i číst z paměťových adres či databloků vedlejšího PLC.

Tohoto způsobu využívám ve finálním kroku pro rozeslání dat z Master PLC, do jednotlivých PLC, které řídí samotný systém.



Obrázek 1.14: Blok PUT



Obrázek 1.15: Blok GET

2 Simulační modely v Matlab Simulink

K ukázce funkčnosti mého řešení jsem v prostředí Matlab/Simulink vytvořil tři demonstrační modely dynamické soustavy s analogovými i digitálními vstupy a výstupy. Všechny modely jsou navrženy pro řízení za pomoci stavového automatu realizovaném na PLC. Běží v reálném čase se simulačním krokem 0,2 sekundy. Tyto modely mají pouze nastínit funkčnost řešení a nemusí proto být ve všech ohledech totožné se samotným procesem, který simulují.

2.1 Model vstřikolisu

Jako první jsem si připravil demonstrační model vstřikolisu pro výrobu plastových dílů. Samotný model má dva analogové a pět digitálních výstupů a je ovládán pomocí jednoho analogového vstupu s rozsahem (0-100) a sedm digitálních vstupů.

Celá simulace se skládá ze dvou částí:

- **Funkční části**, která simuluje fungování fyzikálních vlastností
- **Vizualizační části**, která obsahuje všechny indikační a ovládací prvky

2.1.1 Funkční část vstřikolisu

Pro popsání funkční části vstřikolisu bych nejprve rád nastínil mou představu, jak samotný vstřikolis funguje a následně popsal jednotlivé části modelu, které simulují tyto vlastnosti.

Vstřikolis, jako takový, se skládá ze dvou hlavních částí:

- Extruderu, který zajišťuje přeměnu plastových pelet ze zásobníku do kapalného stavu a zajišťuje vstřikování roztaveného plastu do formy. V mém modelu je osazen těmito senzory a aktuátory:
  - Digitální aktuátor M1 - nasává pelety ze zásobníku do vnitřního prostoru extruderu
  - Digitální Senzor S1 - signalizuje naplnění extruderu
  - Topný prvek M5 - ovládaný analogovým signálem
  - Analogový senzor S4 - udává teplotu plastu uvnitř extruderu
  - Digitální aktuátor M3 - vstřikuje roztavený plast do formy

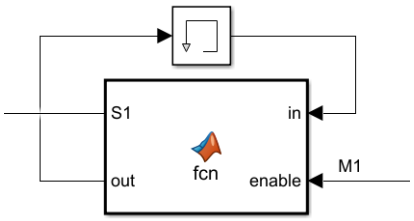
  - Analogový senzor S5 - udává objem plastu vstříknutého do formy

- Dvoudílné formy s možností sevření a rozevření pro vypadnutí hotového výrobku ven. Tato část obsahuje:
  - Digitální aktuátor M2 - zavírá formu
  - Digitální aktuátor M4 - otevírá formu
  - Digitální Senzor S2 - snímač koncové polohy pro zavřený stav
  - Digitální Senzor S3 - snímač koncové polohy pro otevřený stav

Následně se v modelu nachází ovládací prvky S6 a S7, přičemž S6 je tlačítko pro zapnutí/pozastavení výrobního procesu a S7 tlačítko pro okamžité zastavení celé funkčnosti vstřikolisu.

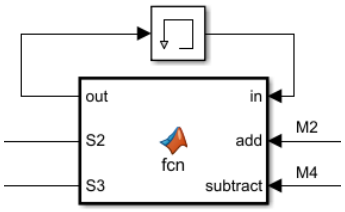
Úkolem funkční části modelu je tedy dynamicky simulovat výstupy modelu (senzory) v čase, na základě vstupů přicházejících z PLC. Jednotlivé výstupy jsou proto simulované následovně:

S1 - senzor nasátí plastových pelet do extruderu Simulace tohoto senzoru je realizována za použití vlastního funkčního bloku, který při signálu na vstupu enable odečítá od vniřní proměnné(při startu nastavené na určitou hodnotu) v každém taktu jedničku. Pokud tato proměnná bude menší než jedna, výstup S1 se nastaví na jedničku. Při příchodu nuly na vstup enable se vnitřní proměnná znovu nastaví na výchozí hodnotu. Kvůli tomu, že funkční bloky nedokáží udržet vnitřní paměť, bylo zapotřebí využít blok memory a pomocí něj přivést na vstup funkčního bloku výstup předchozího stavu.



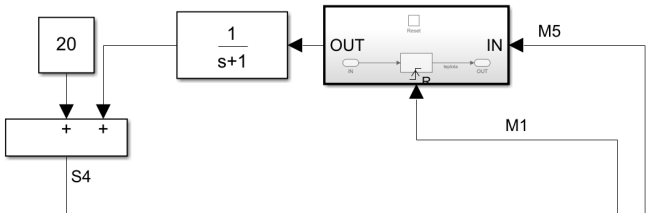
Obrázek 2.1: Simulace nasávání materiálu

S2 a S3 - koncové senzory pohyblivé části formy Jsou simulovány znovu za použití vlastního funkčního bloku, ale tentokrát má jeden vstup na přičítání hodnoty k vnitřní proměnné a druhý na odečítání. Samotné výstupy S2 a S3 jsou aktivní, pokud vnitřní proměnná klesne pod jedna nebo vzroste nad stanovenou mez.



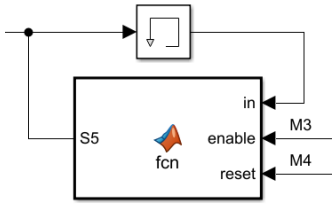
Obrázek 2.2: Simulace koncových poloh formy

S4 - senzor teploty Tento senzor je realizován za pomoci dvou bloků transfer function (systém druhého řádu), přičemž jeden simuluje teplotní setrvačnost lisu a druhý se náchází v bloku resetable function a simuluje teplotní odezvu sypaného materiálu (lze vyresetovat s materiálem novým). K výstupu tohoto bloku je dále přičtena konstantní hodnota 20 jako teplota vstupního materiálu. Je tedy řízen vstupem M5 a resetován při náběžné hraně vstupu M1.



Obrázek 2.3: Simulace teplotního snímače

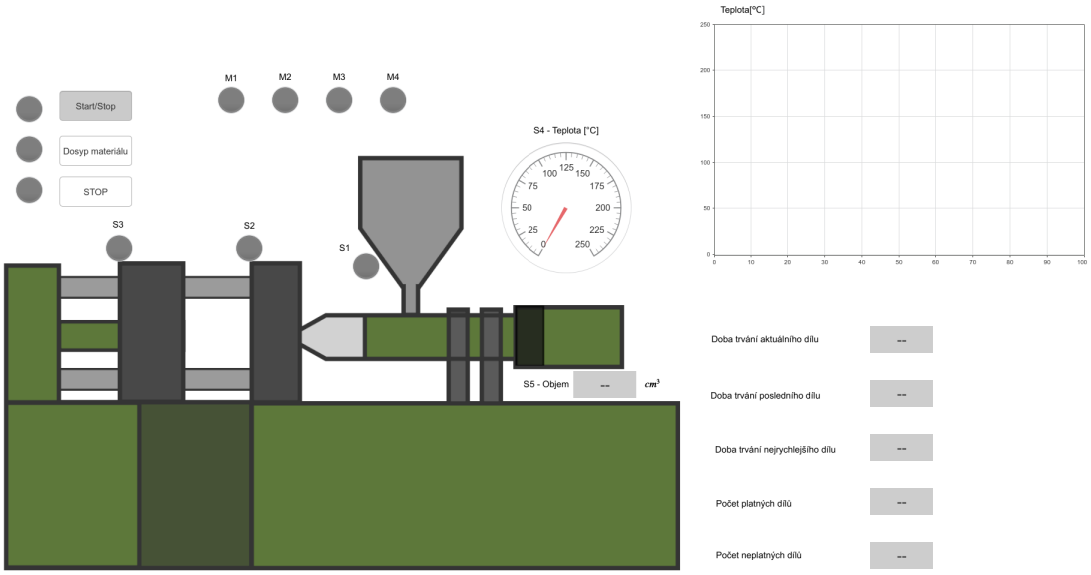
S5 - Analogový senzor objemu vstřikovaného materiálu Při signálu na enable vstupu se načítá hodnota na výstupu S5, která se zároveň vrací jako paměťový vstup. Signálem na vstupu reset se výstup nastaví na nulu.



Obrázek 2.4: Simulace objemového snímače

2.1.2 Vizualizační část vstřikolisu

Pro tento simulační model jsem za pomoci základní knihovny Simulink/Dashboard vytvořil vizualizační schema, obsahující ovládací prvky realizovány tlačítky, grafy analogových hodnot i signalizaci digitálních vstupů a výstupů. Vizualizace navíc obsahuje animaci sypání materiálu a otevírání/zavírání formy.



Obrázek 2.5: Vizualizace vstřikolisu

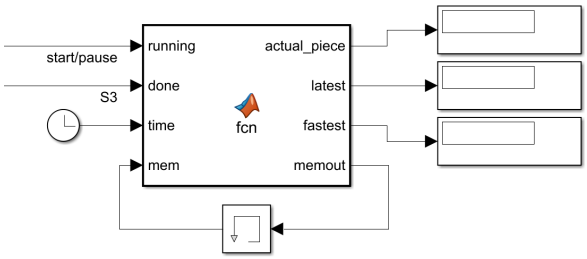
Stav každého digitálního vstupu i výstupu je zobrazen kontrolkou (dashboard/lamp). Analogový snímač teploty je pak zobrazován na ručičkovém ukazateli (dashboard/gauge) a v grafu (dashboard/scope). Indikaci objemu vstřikovaného materiálu zobrazuji pouze pomocí číslicového zobrazovače (dashboard/display). Pro snažší optimalizaci programu na ovládání tohoto systému jsem také přidal několik informativních prvků, díky kterým lze zjistit, jestli je například PID regulátor teploty správně nastaven.

Jsou jimi:

- čas výroby aktuálního, posledního a nejrychlejšího dílu - data užitečná pro optimalizaci výrobního procesu
- Počet dílů, splňujících/nesplňujících požadavky - slouží ke správnému naladění regulační smyčky

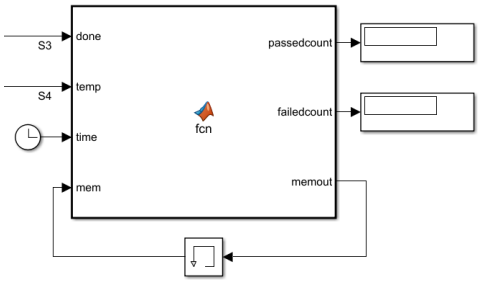
Tyto vedlejší prvky jsou realizovány následovně:

Doba trvání výroby dílu Tento blok využívá na vstupu time hodiny určující dobu od začátku simulace. Při signálu na vstupu running (S7-start/pause) uloží aktuální čas do vnitřní proměnné a signálem done (S3 - senzor otevření) získá odečtením času začátku od aktuálního času délku trvání výroby jednoho dílu. Tento čas si blok uloží a porovnává s dalšími díly, aby mohl zobrazit nejrychlejší čas.



Obrázek 2.6: Blok počítání času dílu

Hlídání kvality dílů Blok neustále sleduje teplotu na vstupu temp (S4). Pokud tato teplota přesáhne předem nastavenou hranici, v aktuálním případě 205 stupňů na dobu delší než dvě sekundy, zapíše se do vnitřní proměnné error stav 1. Po příchodu signálu na port done (S3 - senzor otevření), se v případě, že error nastal, díl počítá jako neplatný. Pokud byla teplota po celou dobu procesu v mezích, tento díl se bude počítat mezi platné.



Obrázek 2.7: Blok hlídání kvality dílů

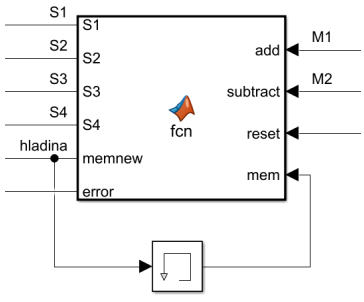
2.2 Model vodní nádrže

Jako další demonstrační úlohu jsem si připravil model vodní nádrže s topnou spirálou. Oproti předchozímu modelu má však menší počet vstupů i výstupů - konkrétně dva analogové a jeden digitální vstup, společně s dvěma analogovými a šesti digitálními výstupy. Výstupy a vstupy modelu jsou popsány následovně:

- Dvoustavový ventil M1 - zajišťuje přítok vody do nádrže
- Analogový ventil M2 - pracuje v rozmezí 0-100 a určuje rychlost odtoku vody z nádrže
- Topný prvek M3 - spirála na ohřev vody
- Digitální senzory výšky hladiny S1, S2, S3, S4 - Jsou umístěny v různých výškách nádrže a sepnou, pokud se hladina dostane na jejich úroveň
- Erorový výstup S5 - pokud je na ventil M2 přivedena hodnota mimo rozmezí, nastaví se tento výstup
- Analogový teploměr S6 - udává teplotu vody v nádrži
- Analogový potenciometr S7 - slouží k ručnímu nastavení rychlosti výtoku M2
- Tlačítko S8 - zapíná program

Jednotlivé reakce jsou pak simulovány takto:

Senzory výšky hladiny S1,S2,S3,S4 Tento blok si ve vnitřní proměnné udržuje výšku hladiny, ke které každý cyklus buď přičítá konstantu, pokud je signál na vstupu add, nebo odečítá analogovou hodnotu ze vstupu subtract. Na základě hodnoty této vnitřní proměnné se v určitých hladinách spínají jednotlivé výstupy.

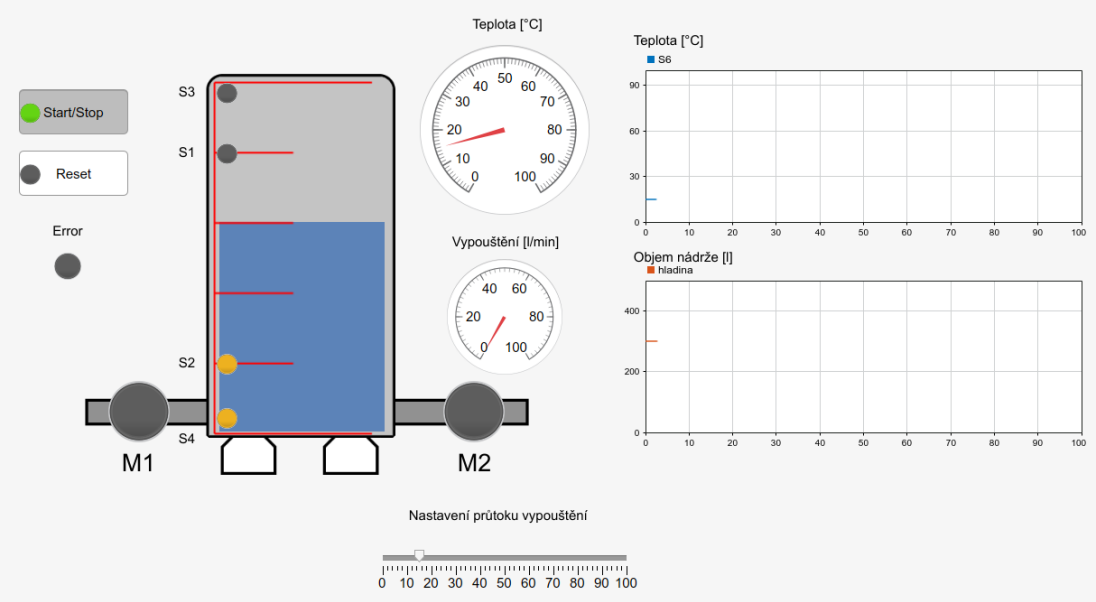


Obrázek 2.8: Simulace senzorů výšky hladiny

Analogový teploměr S6 Je opět realizován dvěma bloky transfer function, stejně jako v předešlém modelu.

Analogový potenciometr S7 Je vytvořen pomocí prvku z knihovny Dashboard, a to konkrétně blokem slider.

Vizuální zpracování nádrže Každý digitální vstup a výstup je signalizován kontrolkou s příslušným označením a na místě, kde by se měl onen senzor nebo ventil nacházet. Hladina vody je zobrazena jak přímo v nádrži, tak i v grafu. Pomocí grafu je také zobrazen průběh teploty vody v čase. Nakonec se zde nachází dva ručičkové ukazatele, a to konkrétně pro aktuální teplotu a průtok ventilu M2.



Obrázek 2.9: Vizuální zpracování nádrže

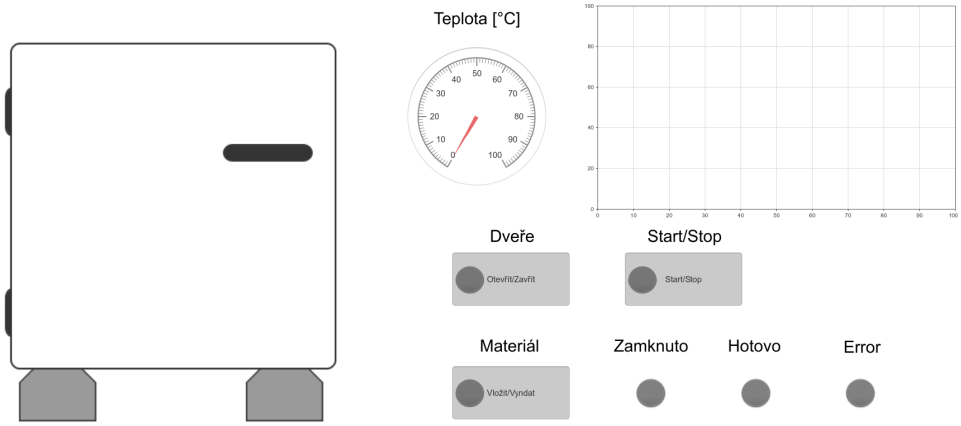
2.3 Model sušičky

Jako poslední ukázkový model jsem si připravil model sušičky plastových pelet. Je ovládán pouze pomocí dvou digitálních a jednoho analogového vstupu, přičemž obsahuje tři digitální tlačítka a jeden analogový teploměr.

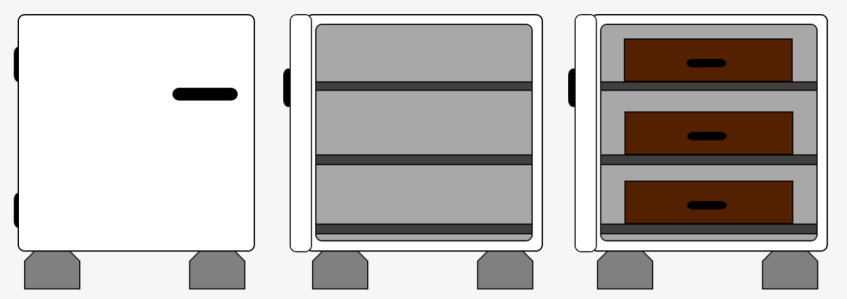
Jednotlivé vstupy a výstupy jsou označeny takto:

- Tlačítka S1,S2,S3 - slouží jako povely pro PLC o startu programu, zavření/otevření dveří, vložení/vyjmutí materiálu.
- Analogový teploměr S4 - udává teplotu v sušičce.
- Digitální vstup M1 - Mechanický zámek dveří
- Topný prvek M2 - slouží k ohřevu vzduchu v sušičce
- Digitální vstup M2 - Signalizace dokončení sušícího procesu

Po překročení určité kritické teploty (například teplota tání sušeného materiálu), se rozsvítí kontrolka Error, která se vypne opětovným zmáčknutím tlačítka Start.



Obrázek 2.10: Vizuální zpracování sušičky



Obrázek 2.11: Jednotlivé stavy sušičky

3 Vzájemná komunikace PLC/Simulink

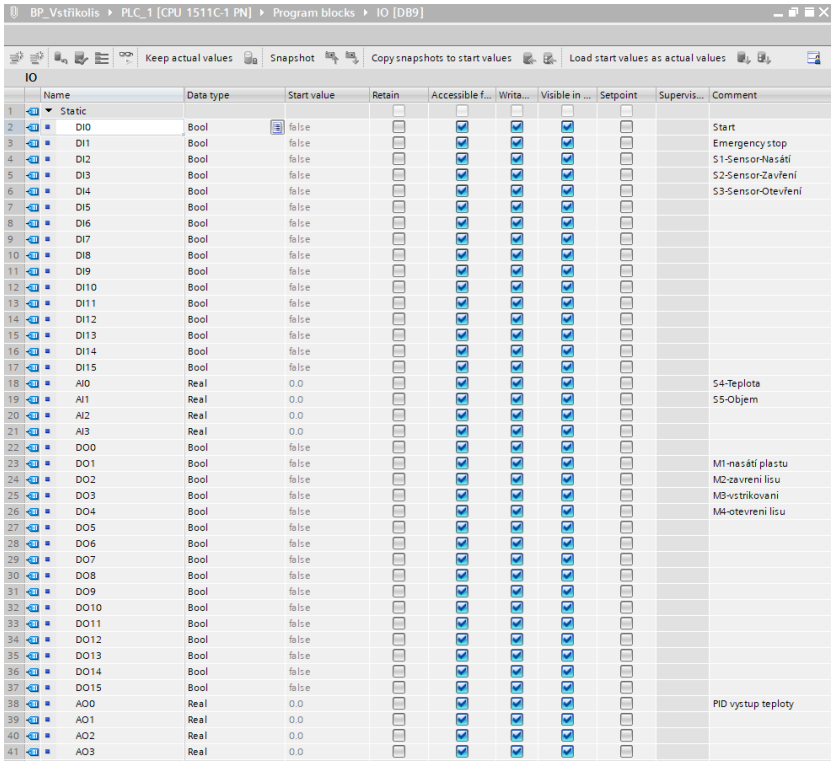
Samotnou komunikaci mezi PLC a Matlab/Simulink by tedy bylo možné realizovat čtyřmi způsoby:

- TCP/IP
- UDP
- Modbus TCP
- OPC

Bohužel, dvě z těchto možností se mi zprovoznit nepodařilo. Konkrétně Modbus TCP sice PLC plně podporuje, ale pro fungování v Matlabu je třeba licence externího programu. U využití OPC jsem narazil na přesně opačný problém. Matlab tuto komunikaci podporuje, ale pro správné fungování v PLC je třeba zakoupit runtime licenci OPC UA. Zbývají mi proto pouze dvě možnosti zprovoznění komunikace, a to TCP/IP a UDP. Mezi nimi není ve finální funkčnosti téměř žádný rozdíl. Proto se zaměřím hlavně na realizaci komunikace TCP/IP, která je proti UDP spolehlivější.

3.1 TCP/IP Strana PLC

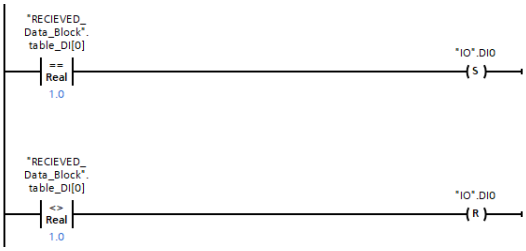
Před samotnou realizací komunikace je třeba nastínit, jaká data chceme posílat. V mém případě se jedná o digitální vstupy a výstupy (0/1) a analogové vstupy a výstupy (v tomto případě jakékoli reálné číslo). Jako simulaci reálných vstupů a výstupů PLC tedy vybral 16 digitálních a čtyři analogové vstupy a výstupy. Jako první jsem si vytvořil datablok, nazvaný IO. Ten obsahuje všechny proměnné, ke kterým bude regulační algoritmus přistupovat. V tomto databloku se tedy nachází všech 32 proměnných typu Bool a osm proměnných typu Real, přičemž polovina je vstupních a druhá polovina výstupních.



Obrázek 3.1: Datablok IO

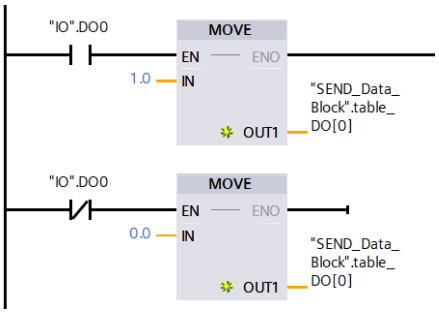
Pro snadnější zpracování dat na druhé straně (Simulink) jsem se rozhodl posílat pouze jeden typ dat. Převádím proto digitální signály v Tia portálu označované datovým typem Bool na stejný datový typ Real, používaný i pro analogové signály. Pro tento účel jsem si vytvořil dva PLC datatypy(jeden pro příjem a druhý pro odesílání dat), z nichž každý obsahuje dvě pole - jedno pro čtyři analogové proměnné a druhé pro 16 digitálních proměnných. Všechna tyto pole jsou datového typu Real. Do datatypu nelze přímo ukládat žádné hodnoty, ale slouží pouze jako nástroj k vytvoření databloku, který dokáže být odeslán. Pomocí těchto datatypů jsem vytvořil databloky Send_data_Block a Recieve_data_Block. Následně je potřeba přesouvat data z Databloku IO do Databloků sloužících k odesílání a přijímání. Vytvořil jsem proto Organizační blok program cycle pojmenovaný Communication. V do něj budu následně přidávat všechny potřebné komponenty k přesunu dat, i bloky určené pro samotnou komunikaci.

Při přijímání dat, které mají být digitální, je třeba přepsat proměnnou typu Real na proměnnou typu Bool. K tomu jsem využil základních compare instrukcí a logických operací set/reset.



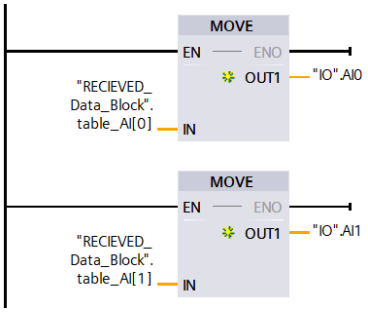
Obrázek 3.2: Převod proměnné typu Real na Bool

Převod v opačném směru - tedy z uložit proměnnou typu Bool jako typ Real, lze realizovat pomocí bloku Move, jehož vstup Enable se spíná digitální proměnnou (jeden blok move pro zapsání jedničky a jeden pro zapsání nuly).



Obrázek 3.3: Převod proměnné typu Bool na Real

Analogová data lze jednoduše přepsat pomocí bloku Move přímo do určeného databloku



Obrázek 3.4: Přesun analogových proměnných

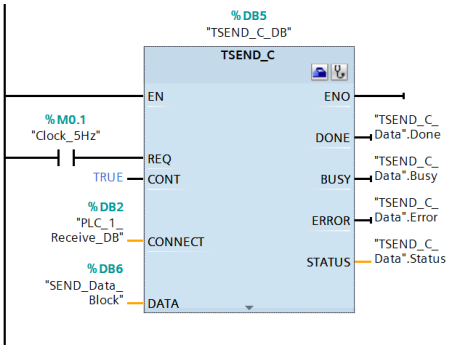
Po přesunu všech proměnných jsou databloky Send_data_Block a Recieve_data_Block připravené pro samotný přenos dat z a do simulace v Simulinku.

Přenos dat je realizován bloky **TSEND_C** pro odesílání a **TRCV_C** pro přijímání, které se nachází v záložce *communication/open user communication*. Oba bloky jsem umístil do Organizačního bloku Communication.

Blok TSEND_C Má několik základních vstupů, které je potřeba pro správnou funkčnost nastavit. Jsou jimi v tomto pořadí:

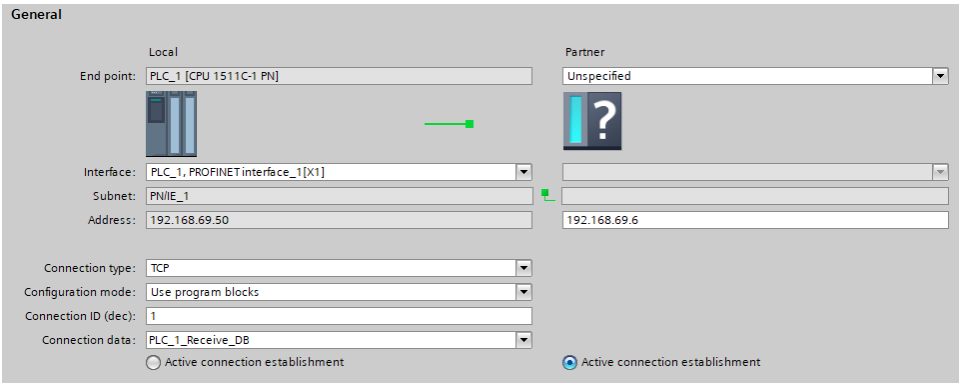
- EN - vstup, který povoluje bloku funkčnost(připojen na 1)
- REQ - vstup, díky kterému se na náběžnou hranu odešlou data(připojen na hodinový signál 5Hz pro periodu odesílání 0,2s)
- CONT - zajišťuje a udržuje komunikaci(nastaven na TRUE)
- CONNECT - je připojen na datablok, ve kterém je nastavení komunikace(nepřipojuje se, vytvoří se automaticky po dalším kroku)
- DATA - odkaz na datablok, který se má odeslat(připojen na SEND_Data_Block)

Výstupy bloku jsou pouze informativního charakteru a není je třeba pro funkčnost bloku zapojovat.



Obrázek 3.5: TSEND blok

Následně je zapotřebí nastavit parametry cílového počítače, na kterém běží simulace. Konkrétně v nastavení samotného bloku zvolit: Partner: Unspecified, Connection type: TCP, Partner adress: IP adresu počítače, na kterém běží simulace, Active connection establishment u partnera a local port 2000.

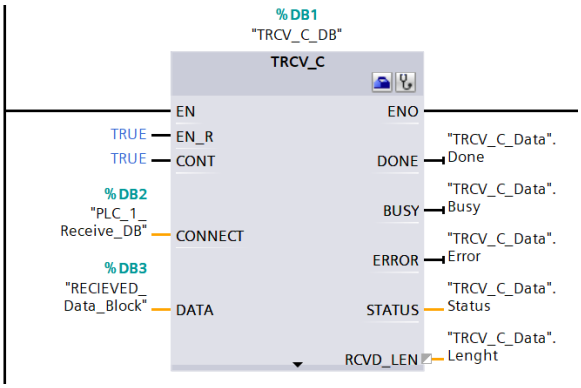


Obrázek 3.6: TSEND blok nastavení

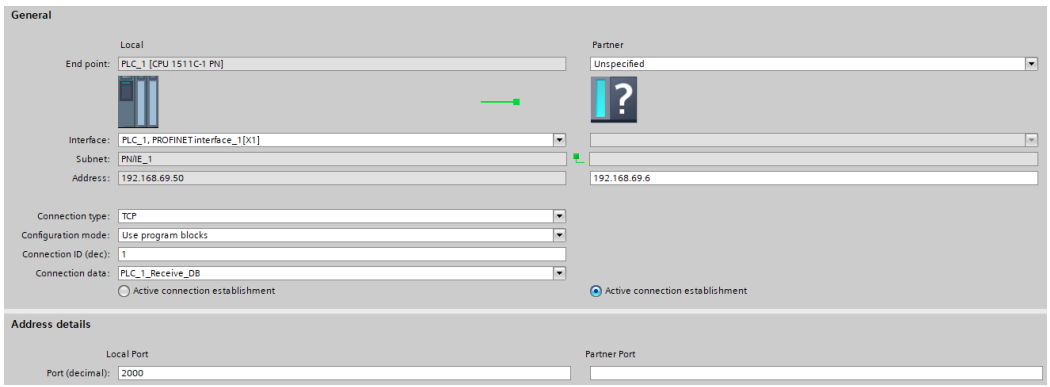
Blok TRCV_C Má také několik základních vstupů, které je potřeba nastavit. Ty jsou:

- EN - vstup, který povoluje bloku funkčnost(připojen na 1)
- EN_R - povoluje přijímání(nastaven na TRUE)
- CONT - zajišťuje a udržuje komunikaci(nastaven na TRUE)
- CONNECT - Vnitřní nastavení tohoto bloku je identické s nastavením bloku TSND_C. Není proto potřeba tyto údaje znovu vyplňovat a lze pouze připojit již vytvořený datablok na tento vstup.
- DATA - odkaz na datablok, kam se mají přijatá data zapsat(připojen na RECIEVED_Data_Block)

Výstupy jsou zde také pouze informativní a není je třeba pro funkčnost bloku zapojovat.



Obrázek 3.7: TRCV blok

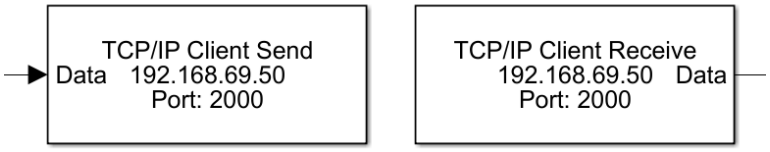


Obrázek 3.8: TRCV blok nastavení

Tímto krokem končí nastavování komunikace na straně PLC a na řadě je nastavení opačné strany v prostředí Matlab/Simulink. Je velmi důležité, aby perioda odesílání zpráv byla stejně dlouhá jako doba kroku nastaveného v simulaci. Pokud by byla perioda odesílání kratší, Simulink by nestíhal informace zpracovávat.

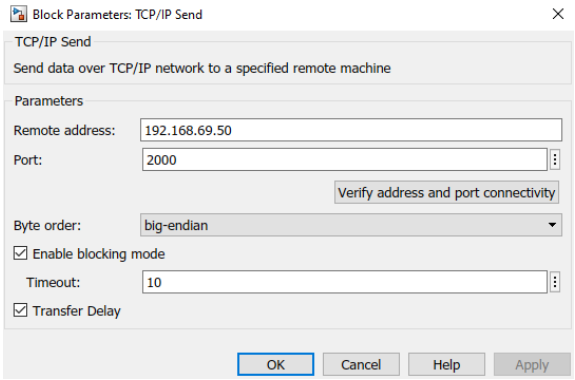
3.2 TCP/IP Strana Simulink

Před realizací přijímání a odesílání dat v prostředí Matlab/Simulink je potřeba vědět, jaký datový typ chceme posílat a jak se jeho označení liší oproti PLC. V mém případě posílám z PLC 20 proměnných typu Real, které Matlab označuje typ SINGLE s byte order big-endian. Celá práce je vytvořena ve verzi Matlab 2021b a doplňkem Instrument Control Toolbox verze 4.5 , který obsahuje bloky potřebné k vytvoření komunikace pomocí TCP/IP. Jsou jimi: **TCP/IP Client Send** a **TCP/IP Client Recieve**. Jak si lze již ze jména povšimnout, bloky se dají využít pouze jako klientská část spojení, což v praxi znamená, že ony spojení inicializují. Právě kvůli tomu jsem v PLC blocích komunikace nastavil, že spojení vytváří partner.



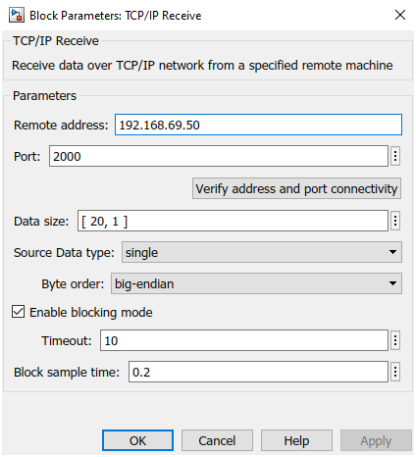
Obrázek 3.9: Bloky pro TCP/IP komunikaci

Nastavení bloku TCP/IP Client Send je minimalistické, protože parametry o počtu a typu dat se nastavují automaticky, podle dat příchozích na vstup bloku. Nastavuje se zde pouze adresa a port koncového serveru, Byte order - styl, v jakém formátu jsou čísla uložena, zaškrtávací políčko blocking mode -určuje, jestli má být simulace po dobu odesílání pozastavena, Timeout - doba, po kterou čeká blok na odpověď od serveru o přijetí paketu, Transfer delay-umožňuje prodloužit dobu čekání na odpověď o přijetí. V připravených simulacích je blok nastaven podle obrázku:



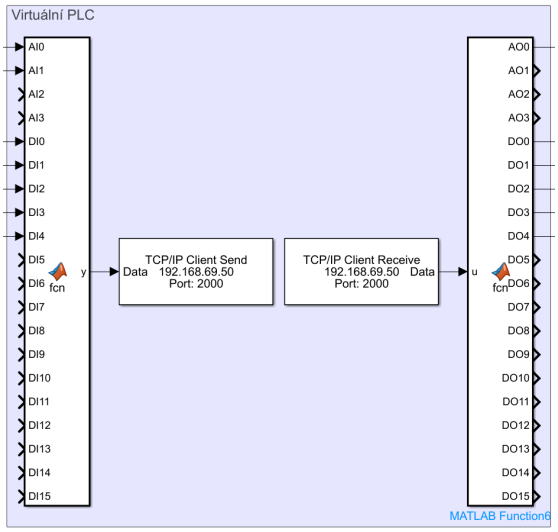
Obrázek 3.10: Nastavení bloku TCP/IP Client Send

Nastavení bloku TCP/IP Client Recieve již obsahuje položek více. Je zde potřeba nastavit vše, co v bloku TCP/IP Client Send, společně s informacemi o datech v paketu. Lze zde navíc nastavit: Data size - počet přijímaných dat(lze nastavit i jako vektor), Source data type - typ přijímaných informací a Block sample time - perioda, při které se mají data odesílat. Jako v případě předchozího bloku je v simulaci nastaven takto:



Obrázek 3.11: Nastavení bloku TCP/IP Client Recieve

Před využitím přijatých dat, je třeba je nejprve zpracovat. Rozhodl jsem se , že v celé simulaci budu využívat datový formát double. Proto jsem oba bloky pro přehlednost vložil do subsystemu a na jejich vstupy a výstupy připojil vlastní funkční bloky k přetypování dat. Díky využití subsystému, je také možné vložit do simulace virtuální PLC jako jeden blok, u kterého lze zvolit pouze vstupy a výstupy, které se budou využívat.



Obrázek 3.12: Celý blok PLC

4 Regulační algoritmy v PLC

Po zajištění funkční komunikace mezi PLC a simulací v Matlab/Simulink jsem se rozhodl vytvořit regulační algoritmy pro jednotlivé úlohy. Všechny úlohy lze řídit stavovým automatem a jedním regulátorem PID.

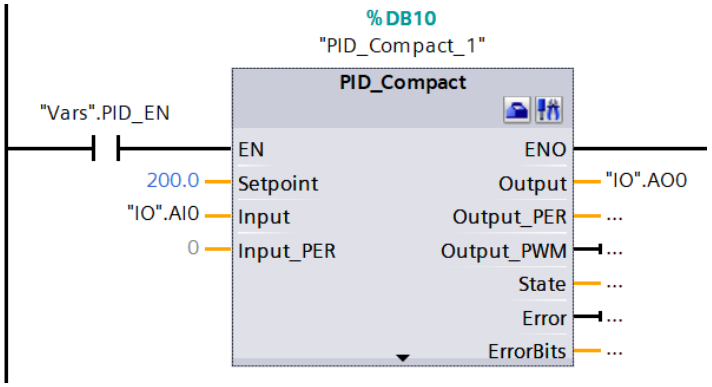
4.1 Algoritmus pro řízení vstřikolisu

Při programování algoritmu řízení modelu vstřikolisu jsem si zvolil několik stavů, mezi kterými se stavový automat přepíná na základě vstupů a vnitřních proměnných. Tyto stavy jsou:

- Stav 0 - Nic se neděje, čekání na zmáčknutí tlačítka start
- Stav 1 - Zapnutí PID regulátoru, zapnutí nasávání materiálu, čekání na senzor nasátí
- Stav 2 - Vypnutí nasávání, čekání až se senzor teploty dostane do určité meze a následně čeká čtyři sekundy
- Stav 3 - Zapnutí zavírání lisu, čeká se na senzor zavření
- Stav 4 - Vypnutí zavírání, zapnutí vstřikování, čeká na senzor objemu vstřikovaného materiálu a následně ještě dalších pět sekund (pro vychladnutí výrobku)
- Stav 5 -Zapnutí otevírání, čeká se na senzor otevření,vypnutí otevírání a následně čekání další tři sekundy před přepnutím znovu do Stavu 0

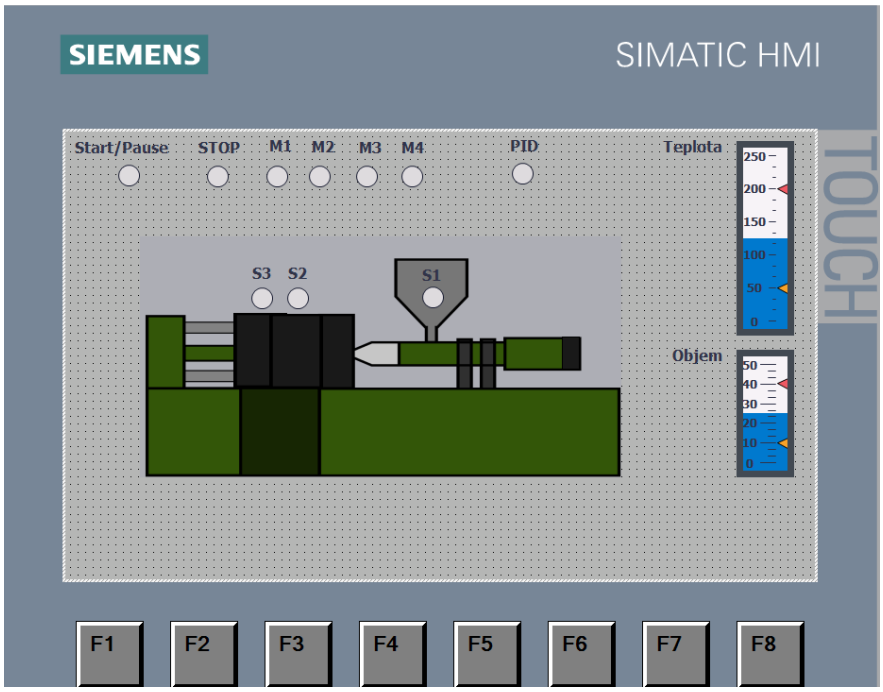
Podrobný diagram průběhu se nachází v příloze (A.3)

Všechny vstupní i výstupní proměnné se nachází v datablоku IO, vytvořeném při zprovozňování komunikace. Celý program, který tyto stavy přepíná, se nachází v hlavním programovém bloku Main. Pro regulaci teploty jsem si vytvořil programový blok cyklického přerušení(Cyclic interrupt) s periodou volání 100ms a do něj vložil **Blok PID compact**. Regulátor se zapíná pomocí proměnné PID_EN, připojené na vstup EN. Parametr setpoint(cílová hodnota) je v tomto případě pevně nastavená na 200 stupňů. Následně stačí pouze přivést proměnnou teplotního snímače na vstup Input a na výstupu Output dostaneme rovnou regulovanou veličinu. Samotné nastavení vnitřních proměnných je poměrně zdlouhavé a nebudu se jím v této práci zabývat. V přiložených zdrojových kódech je však tento blok nastavený a plně funkční pro všechny tři simulace.[8]



Obrázek 4.1: blok PID_Compact

Po vytvoření funkční části algoritmu, jsem vytvořil i vizuální část programu určenou pro HMI panel. Ta je velmi podobná vizualizační části simulace v Simulinku, akorát neobsahuje žádné řídící prvky a jen informuje uživatele o stavech jednotlivých vstupů a výstupů.



Obrázek 4.2: Vizualizace vstřikolisu na HMI

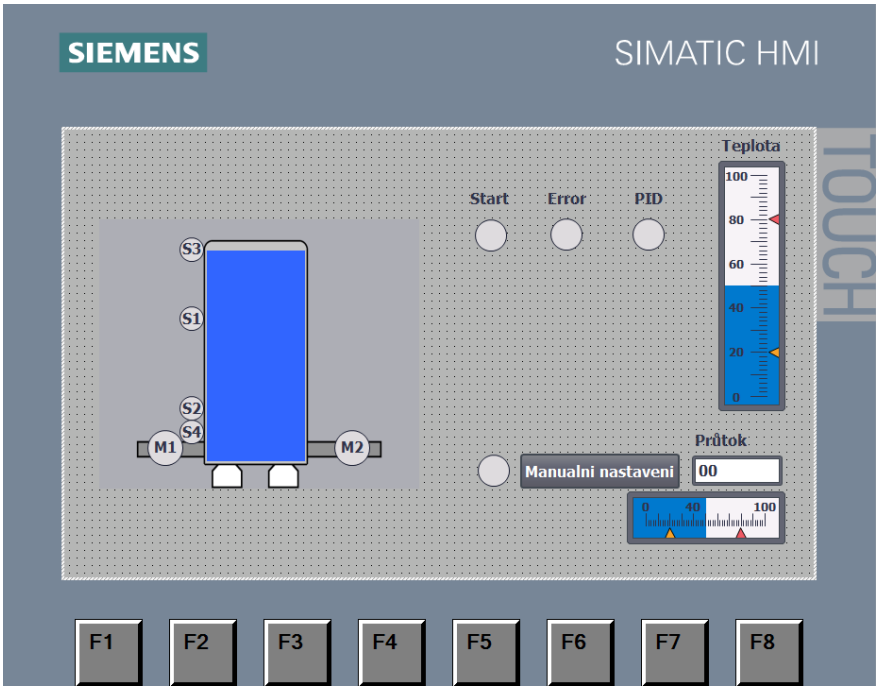
Indikátory teploty a objemu jsou vytvořené pomocí elementu bar, zatímco všechna indikace digitálních vstupů a výstupů je zobrazována změnou barvy kruhu. Pro správné fungování HMI panelu je třeba nastavit, které proměnné se do něj z PLC mají posílat. Nastavují se v záložce HMI tags a kromě jména proměnné v PLC lze zvolit i jak často se má tato proměnná obnovovat.

4.2 Algoritmus pro řízení vodní nádrže

Řízení simulace vodní nádrže je také realizováno za pomoci stavového automatu a jednoho PID regulátoru pro regulaci zahřívání vody. Stavy jsem definoval následovně:

- Stav 0 - Nic se neděje, čekání na zmáčknutí tlačítka start
- Stav 1 - Zapnutí napouštění, čeká se na napuštění do určité hladiny(S1)
- Stav 2 - Vypnutí napouštění, zapnutí zahřívání, čeká se, až se teplota vody ustálí na určené hodnotě po dobu 10 sekund.
- Stav 3 - Zapnutí vypouštění, čeká se na vypuštění pod určitou hladinu (S2), následně vypnutí vypouštění a přechod znovu do Stavu 0

Podrobný diagram průběhu se nachází v příloze (A.2) Změna oproti předchozí úloze se nachází v HMI panelu. Ten totiž neslouží jen k získávání informací o stavech, ale lze v něm také manuálně nastavit průtok vypouštěcího ventilu. Tlačítkem Manuální nastavení lze pak přepínat mezi hodnotou přicházející ze simulace a hodnotou nastavenou na HMI.



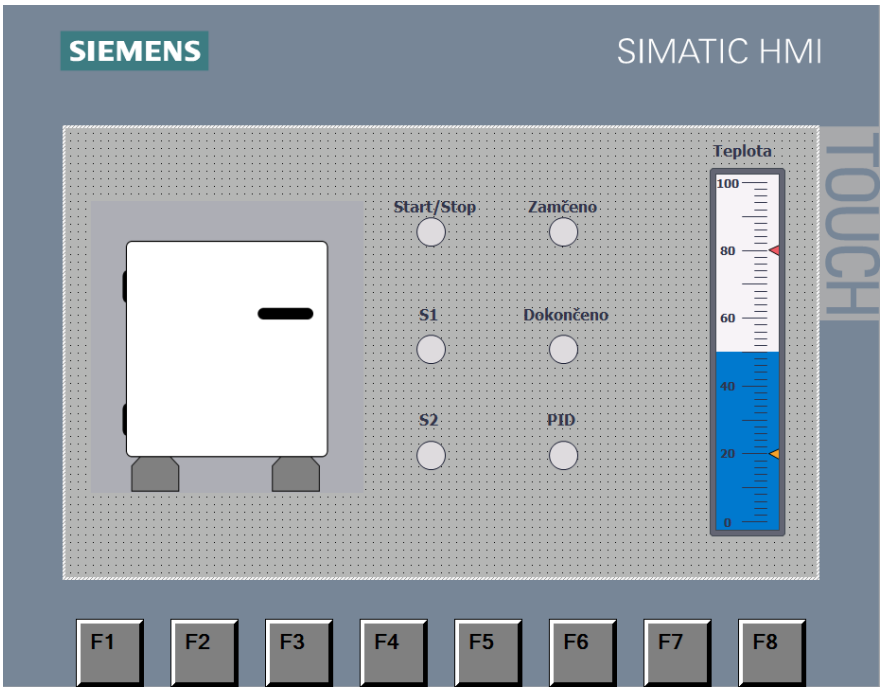
Obrázek 4.3: Vizualizace nádrže na HMI

4.3 Algoritmus pro řízení sušičky

I poslední simulační úloha lze řídit pomocí stavového automatu

- Stav 0 - Nic se neděje, čekání vložení materiálu a zavření dveří
- Stav 1 - Zamčení dveří, zapnutí vytápění, čeká se na ustálení teploty na určité hodnotě po dobu pět sekund
- Stav 2 - Odemknutí dveří, rozsvícení kontrolky Dokončeno, čekání na otevření dveří
- Stav 3 - Vypnutí vytápění, přepnutí do Stavu 0

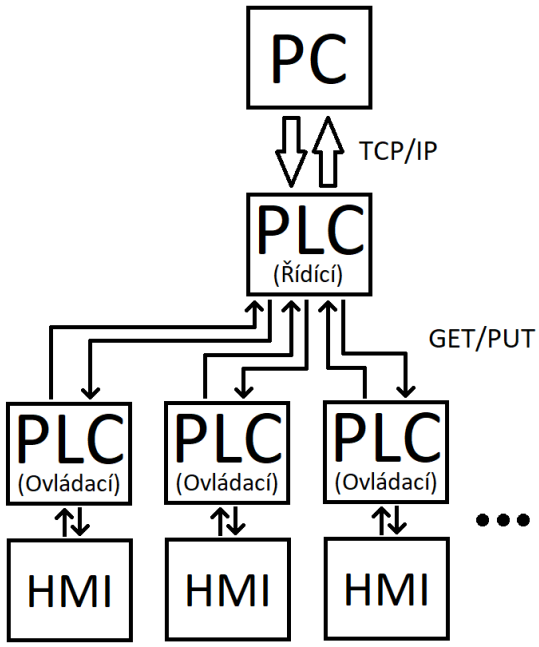
Podrobný diagram průběhu se nachází v příloze (A.1)



Obrázek 4.4: Vizualizace sušičky na HMI

5 Modifikace pro řízení více úloh na jednom PC

Pro běžný počítač by neměl být problém simulovat více těchto modelů najednou. Problém je však s optimalizací vizualizačních bloků v prostředí Simulink. Pokud chceme ke každé simulaci zobrazovat i vizualizaci, je potřeba výkonější PC. Rozhodl jsem se proto kvůli úspoře prostředků(stačí použít pouze jeden PC namísto počítače pro každou úlohu), vytvořit simulaci obsahující více simulačních modelů. Díky tomu také zmizí hlavní nevýhoda komunikace TCP/IP, kterou je, že musí běžet PLC, aby se mohla simulace v Simulinku spustit. Nevýhodou však je potřeba jednoho řídícího PLC navíc, které se bude chovat jako hlavní a bude komunikovat se všemi ostatními i Simulinkem. Strom komunikace by pak vypadal následovně:



Obrázek 5.1: Strom komunikace

Zatímco komunikace mezi řídícím PLC a počítačem by stále probíhala pomocí protokolu TCP/IP, jednotlivá ovládací PLC by si předávala data pomocí protokolu GET/PUT, který je přímo vytvořen pouze pro komunikaci PLC výrobce Siemens.

5.1 Přidání řídícího PLC

Do stejné sítě, jako jsou ovládací PLC jsem přidal řídící PLC, které komunikuje se Simulinkem totožně, jako dříve komunikovala jednotlivá PLC. Jediná změna je v množství odesílaných dat. Namísto odesílání a přijímání 20 proměnných se jich posílá 20x počet ovládacích PLC. V době zpracování této práce jsem neměl přístup k devět PLC (osm ovládacích a jedno řídící), a tak jsem se rozhodl otestovat funkčnost tohoto řešení pouze na třech úlohách běžících na jednom PC. V mojí demonstraci tedy řídící PLC odesílá a přijímá 60 proměnných.

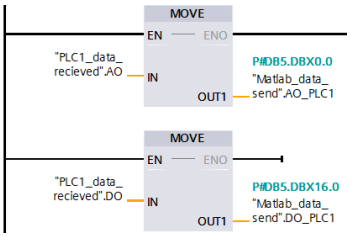
Nejprve je nutné vytvořit datatyp, který bude obsahovat všech 60 proměnných rozložených v jednotlivých polích. S jeho pomocí pak lze vytvořit datablok, který se připojí na vstup bloků **TSEND_C** a **TRCV_C**. Jejich další nastavení zůstává totožné s nastavením z jednotlivých úloh.

Matlab_data_rcv

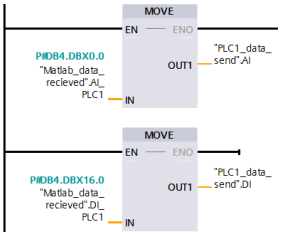
	Name	Data type	Default value	Accessible f...	Writa...	Visible in ...	Setpoint	Comment
1	AI_PLC1	Array[0..3] o...		☑	☑	☑	☐	
2	DI_PLC1	Array[0..15] of Real		☑	☑	☑	☐	
3	AI_PLC2	Array[0..3] of Real		☑	☑	☑	☐	
4	DI_PLC2	Array[0..15] of Real		☑	☑	☑	☐	
5	AI_PLC3	Array[0..3] of Real		☑	☑	☑	☐	
6	DI_PLC3	Array[0..15] of Real		☑	☑	☑	☐	

Obrázek 5.2: Datablok přijímání dat ze Simulinku

Následně je zapotřebí vytvořit datatypy a z nich databloky, které budou obsahovat zvlášť vstupní a zvlášť výstupní proměnné pro každé PLC (stejně jak tomu bylo u samostatných úloh). Pro prosté přesouvání dat mezi těmito databloky jsem využil základní instrukce MOVE.

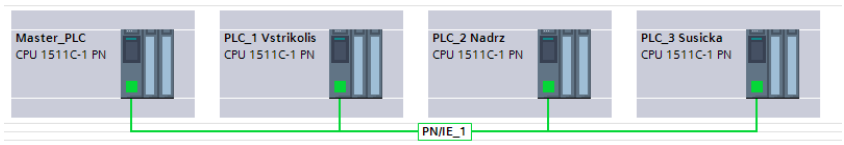


Obrázek 5.3:
Přesun dat k odeslání do matlabu



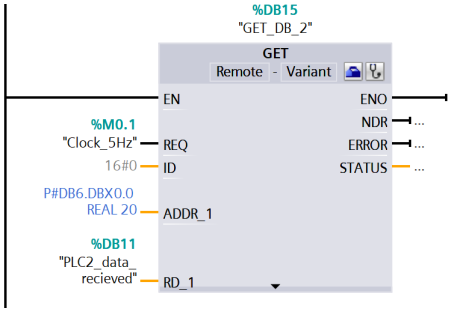
Obrázek 5.4:
Přesun dat přijmutých z matlabu

Před komunikací mezi PLC je zapotřebí, aby byla ve stejné síti. V záložce Devices and networks je tedy potřeba propojit všechna využívaná PLC do jedné sítě (PN/IE_1).



Obrázek 5.5: Síť PLC

Dále je možné využití bloků GET a PUT. Pomocí těchto bloků lze bez dalšího nastavování vkládat nebo číst data přímo z paměti partnerského PLC. V této práci jsem si vybral použití pouze bloku GET, protože nesprávně použitý blok PUT může být nebezpečný. Zatímco za pomoci GET, se pouze čte paměť jiného PLC, blok PUT může vnutit data jinému PLC i bez jeho svolení.

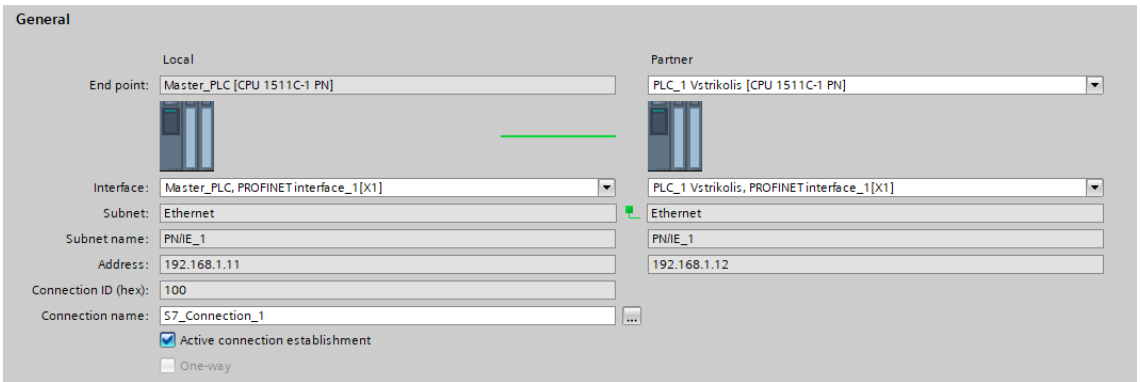


Obrázek 5.6: Blok GET

Vnější nastavení bloku:

- EN - zapnutí funkčnosti bloku
- REQ - při náběžné hraně přečte data z ADDR_1 a vloží je na RD_1
- ADDR_1 - na tomto vstupu je uloženo místo v paměti partnerského PLC, ze kterého bude blok GET číst
- RD_1 - datablok dat, do kterého se mají přečtená data zapsat

Ve vnitřním nastavení bloku je pak nutné akorát nastavit jako partnera příslušné PLC, se kterým má blok komunikovat, a všechny ostatní kolonky se dovyplní samy.

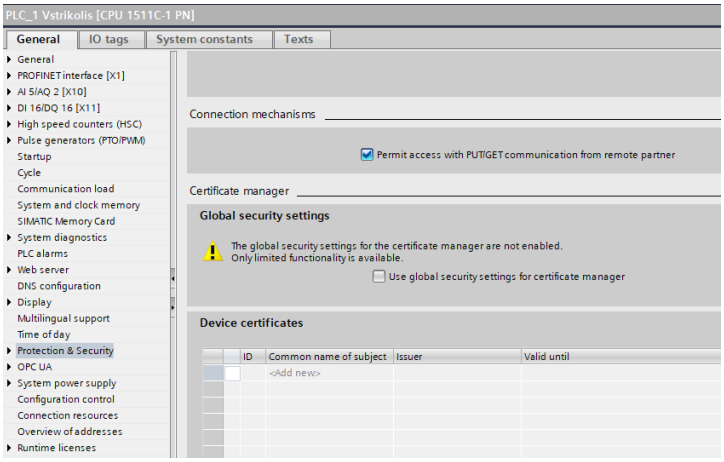


Obrázek 5.7: Vnitřní nastavení bloku GET

5.2 Modifikace ovládacích PLC

Hlavní program ovládacího PLC zůstává stejný. Je potřeba pouze upravit několik nastavení a odstranit komunikační bloky **TSEND_C** a **TRCV_C**. Využití těchto bloků pro komunikaci s Matlabem již zajišťuje řídící PLC a nejsou tak zapotřebí.

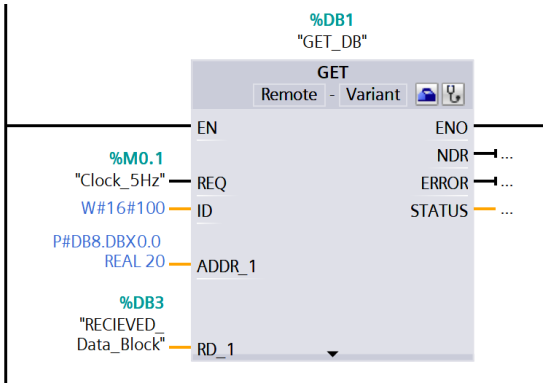
Dále je nutné v záložce Device configuration/Protection and security povolit komunikaci prostřednictvím PUT/GET (Permit acces with PUT/GET communication from remote partner)



Obrázek 5.8: Povolení komunikace GET/PUT

Následně se musí ve vlastnostech databloků, do kterých se mají ukládat nebo z nich odesílat data, v záložce atributes odškrtnout položku optimized block access. Databloky díky tomu budou mít pevně uložené místo v paměti a komunikace PUT/GET s nimi může pracovat.

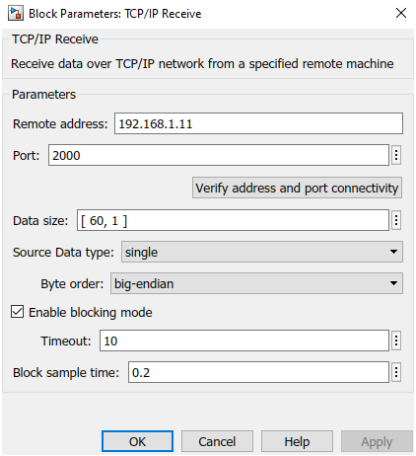
Odesílání výstupů je již zprostředkováno blokem GET v řídícím PLC, ale pro příjem vstupů jsem znovu využil blok GET, který čte data z řídícího PLC a ukládá si je do vlastní paměti.



Obrázek 5.9: Blok GET

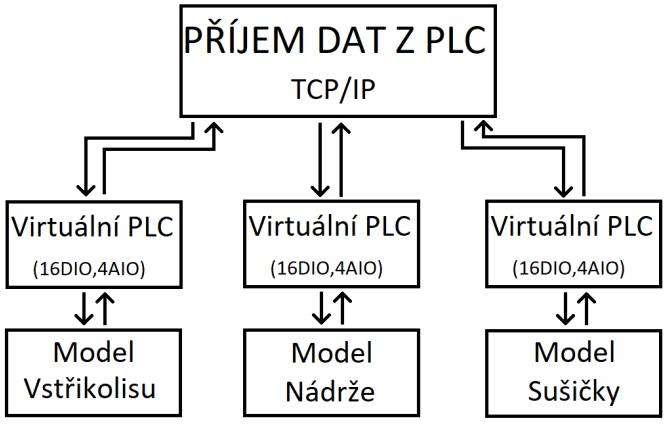
5.3 Modifikace simulace

Pro komunikaci všech modelů v simulaci s jejich příslušnými ovládacími PLC najednou je zapotřebí pouze jeden blok TCP/IP Client Recieve a jeden TCP/IP Client Send. Jedinou změnou v nastavení těchto bloků je množství dat, které budou odesílat a přijímat. Blok pro odesílání dat nemá tuto možnost nastavení, protože automaticky počítá potřebnou velikost paketu z velikosti dat, která mu přijdou na vstup data. U přijímacího bloku je však potřeba upravit položku data size.



Obrázek 5.10: Modifikace bloku TCP/IP Recieve

Tato data je následně zapotřebí rozdělit příslušným virtuálním PLC, která už budou obsluhovat jednotlivé modely. Protože se přijatá data chovají jako jednorozměrné pole, stačí využít blok Matlab Function a pomocí základních funkcí toto pole rozdělit na úseky posílané jednotlivým PLC. Spojování všech odchozích dat probíhá obdobně. U samotných Virtuálních PLC pak stačí odebrat komunikační bloky a místo nich navést do subsystému výstupy komunikačního bloku.



Obrázek 5.11: Struktura pohybu dat v simulaci

Závěr

V práci jsem prozkoumal různé způsoby vnější komunikace PLC a zhodnotil jejich využitelnost pro potřebné propojení. Některé z nich jsem také otestoval, konkrétně Modbus TCP a IO devices, ale nakonec jsem se rozhodl využít protokolu TCP/IP pro komunikaci se simulací a GET/PUT pro vzájemnou komunikaci mezi PLC.

Dále jsem v prostředí Matlab/Simulink vytvořil tři simulační modely dynamických soustav, včetně jejich vizualizace. Modely jsou v základním principu podobné, ale snažil jsem se, aby každý z nich ukázal něco, co není využito v ostatních.

Vytvořil jsem také pro každou úlohu řídící algoritmus, běžící na PLC, a využívající vestavěného PID regulátoru k regulaci teploty. Každá úloha má také svou vizualizaci na HMI panelu.

Tyto dvě instance jsem následně propojil pomocí funkčních bloků, využívajících komunikačního protokolu TCP/IP.

V poslední části jsem spojil všechny simulační modely do jednoho programu a přidal jedno řídící PLC, které s tímto programem komunikuje. Ostatní ovládací PLC si následně vyměňují data pouze s řídícím PLC a díky tomu může více PLC obsluhovat pouze jeden PC. Narazil jsem však na výkonovou limitaci PC, kdy při větším počtu současně běžících modelů s vizualizací narůstá latence spojení. Je proto vhodné využít výkonnější PC, nebo omezit rozsáhlost simulačního programu.

Všechny body zadání tak byly splněny a je možné využít toto řešení při výuce programování PLC.

Použitá literatura

[1] *What is a Transmission Control Protocol TCP/IP Model* [online]. 2022 [cit. 2022-05-06]. Dostupné z: https://www.fortinet.com/resources/cyberglossary/tcp-ip.

[2] *TIA Portal: Open User Communication TSEND + TRCV / PLC-PLC Communication* [online]. 2019 [cit. 2022-05-06]. Dostupné z: https://www.youtube.com/watch?v=fVLVvqn03qM&ab_channel=Hegamurl.

[3] *User Datagram Protocol (UDP)* [online]. 2022 [cit. 2022-05-06]. Dostupné z: https://www.techtarget.com/searchnetworking/definition/UDP-User-Datagram-Protocol.

[4] *Modbus TCP* [online]. 2022 [cit. 2022-05-06]. Dostupné z: https://www.speedgoat.com/products-services/i-o-connectivity/protocols/modbus-tcp.

[5] STENERSON Jon, David DEEG. *Siemens Step 7 (TIA PORTAL) Programming, a Practical Approach*. 2. vyd. Nezávisle publikováno, 2019. ISBN 978--1091474109.

[6] *SIMATIC S7-1500 OPC UA Server with Companion Specifications and SiOME* [online]. 2018 [cit. 2022-05-06]. Dostupné z: https://www.youtube.com/watch?v=GpMQgdqy3ow.

[7] *TIA Portal: IO-Devices / PLC-PLC Communication* [online]. 2019 [cit. 2022-05-06]. Dostupné z: https://www.youtube.com/watch?v=ZcgyhDg97oM&ab_channel=Hegamurl.

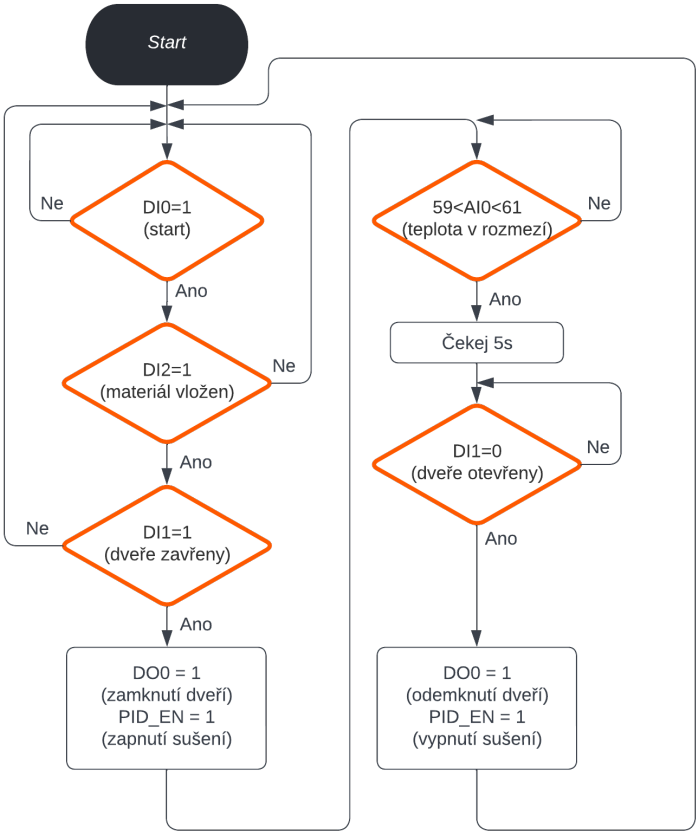
[8] BORDEN Terry R., Richard A. COX a Richard A. COX. *Technician's guide to programmable controllers*. 6. vyd. Clifton Park, NY: Delmar, Cengage Learning, 2013. ISBN 9781111544096.

Obsah volných příloh

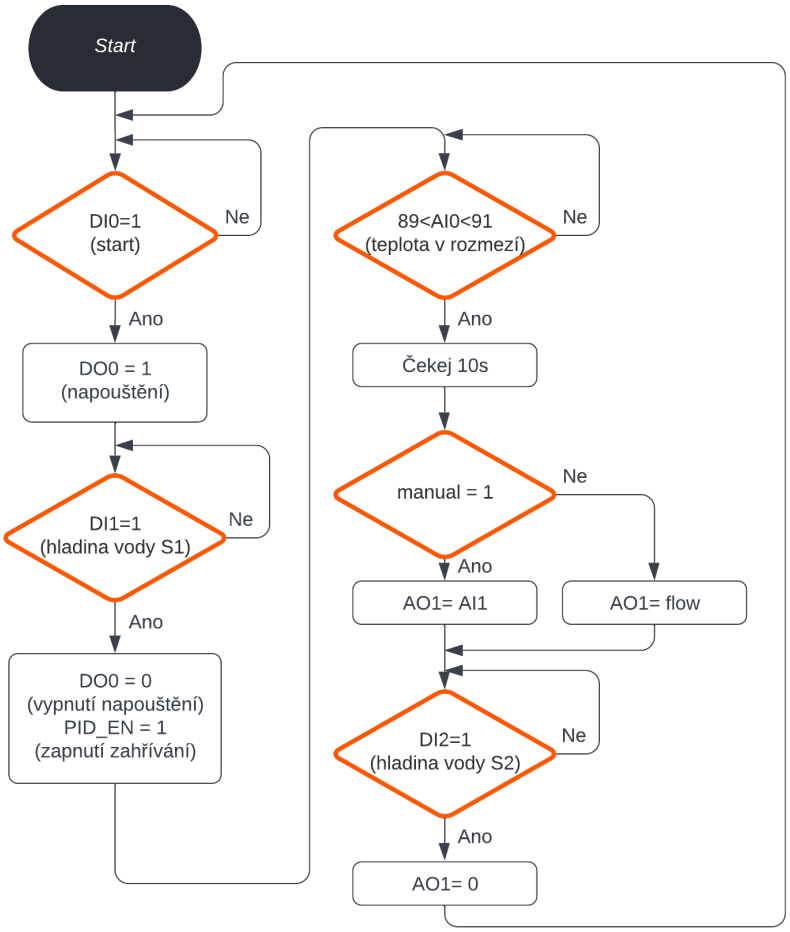
- Složka: „Matlab“
  - Programy pro jednotlivé modely, vytvořené v prostředí Matlab/Simulink pro každou úlohu zvlášť.
  - Program Kombinace, obsahuje spojení všech modelů do jednoho programu.
  - Program Kombinace_bez_vizualizace, obsahuje také spojení všech modelů do jednoho programu, avšak s co nejmenším počtem vizualizačních prvků.
- Složka: „TIA“
  - Projekty pro regulační algoritmy jednotlivých úloh (včetně HMI panelu), vytvořené v prostředí TIA portal.
  - Projekt Kombinace, obsahující všechna ovládací PLC společně s řídícím.
  - Projekt Modbus_TCP, obsahující ukázku komunikace za použití protokolu Modbus_TCP
  - Projekt IODevices, obsahující ukázku komunikace za použití protokolu IODevices

Přílohy

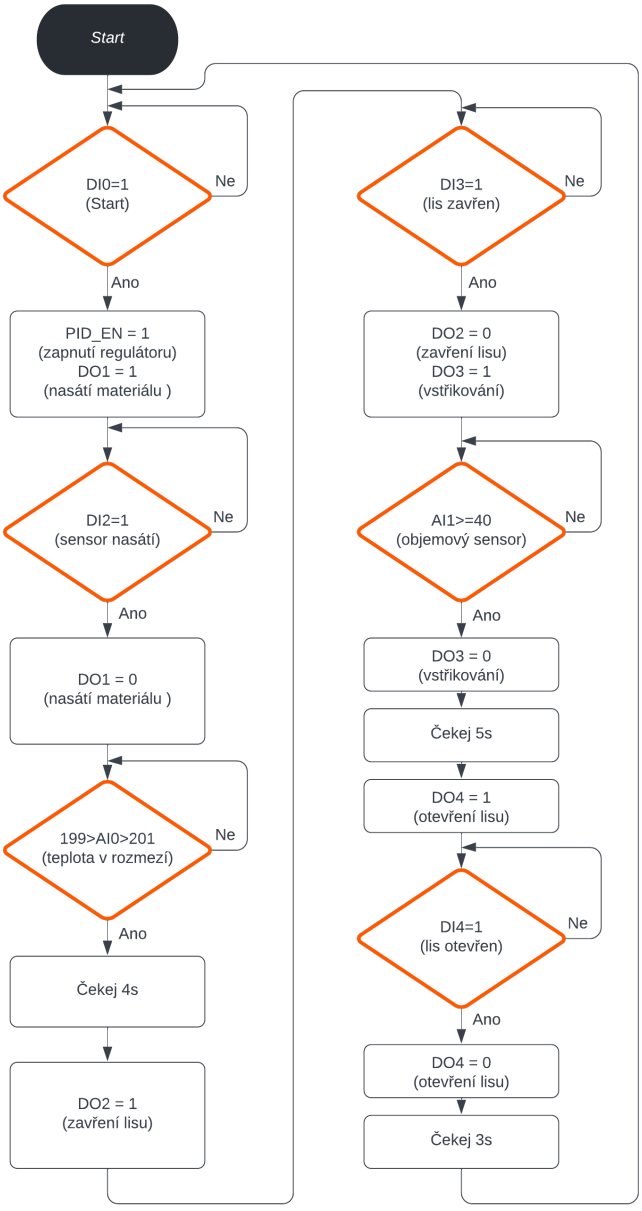
A Stavové diagramy



Obrázek A.1: Stavový diagram pro sušičku



Obrázek A.2: Stavový diagram pro nádrž



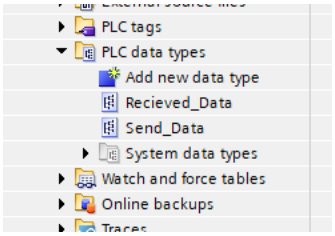
Obrázek A.3: Stavový diagram pro vstřikolis

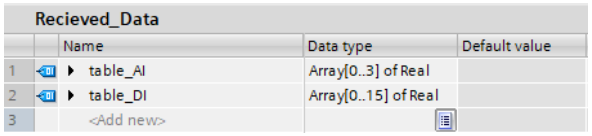
B Podrobný návod nastavení komunikace

B.1 Samostatná úloha

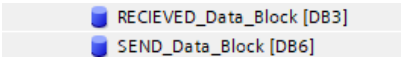
Pro vytvoření spojení PLC s modelem je jako první nutné znát IP adresu počítače, na kterém simulace běží. V tomto návodu je to konkrétně **192.168.1.200** Vše začíná vytvořením projektu a přidáním vhodného PLC. Samotné vytvoření projektu nebudu popisovat a přejdu rovnou k prvkům, potřebným pro komunikaci.

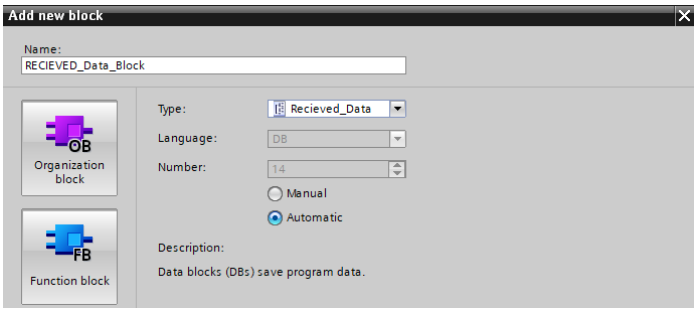
1. Vytvoření Datatypů - v levé záložce, PLC data types - add new data type. Je potřeba vytvořit dva datatypy o velikosti odesílaných a přijímaných dat. Konkrétně jeden datatyp pro příjem dat (obsahuje čtyřprvkové pole AI typu Real a 16 prvkové pole DI typu Real) a jeden datatyp pro odesílání dat (obsahuje pole 4 AO typu Real a 16 DO typu Real) - typ musí být stejný pro snažší komunikaci, lze následně přetypovat v PLC.





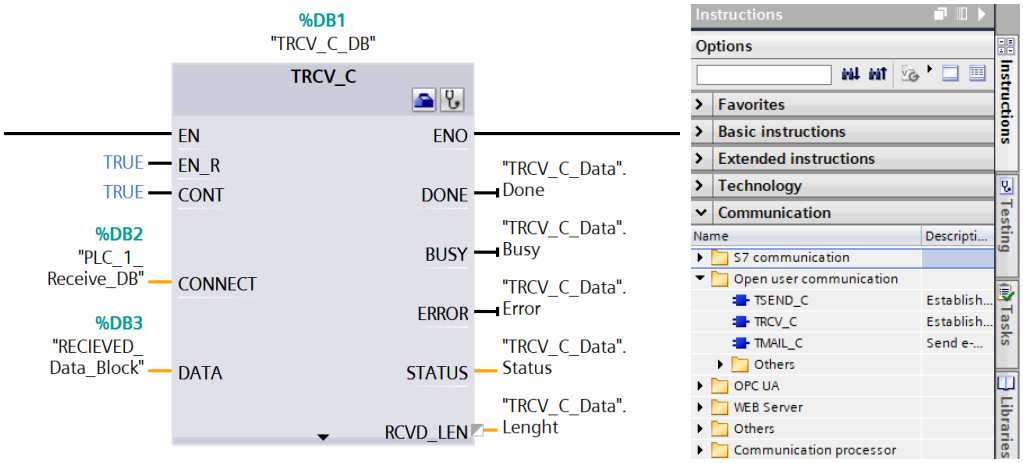
2. Vytvoření dvou Databloků - v levé záložce, program blocks - add new block. Vytvoření dvou data blocků, do typu se zadá příslušný datatyp. Znovu jeden pro odesílání a jeden pro přijímání - (SEND_Data_Block a RECIEVED_Data_Block)



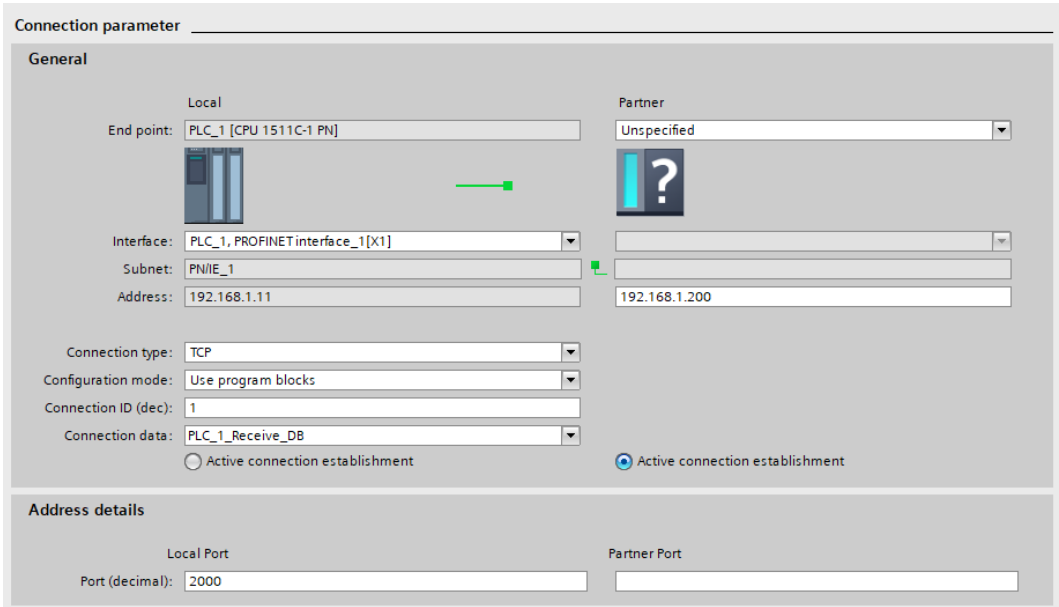


3. Vytvoření Organizačního bloku - v levé záložce, program blocks - add new block. Vytvoření bloku program cycle jazyku LAD. Tento krok je zde spíše pro organizaci. Lze ho přeskočit a bloky vložit do OB Main.

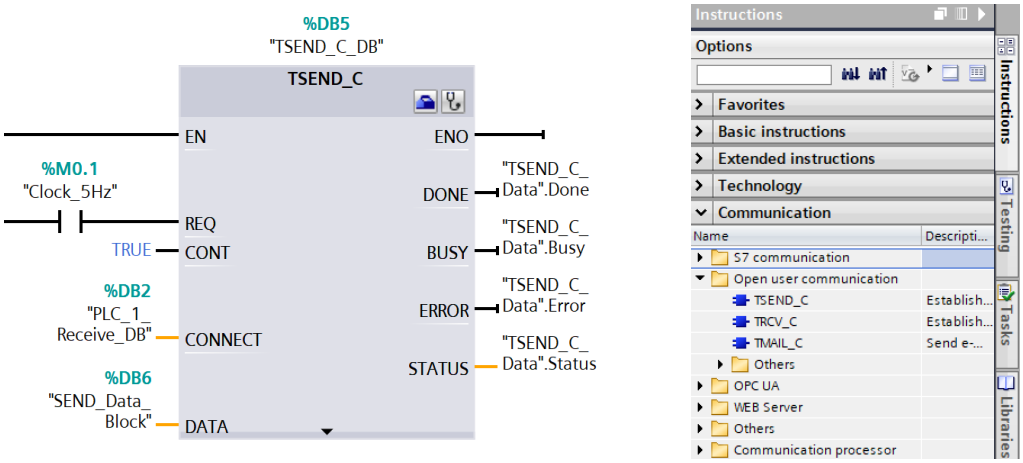
4. Vložení bloku **TRCV_C** - v levé záložce, Open user communication - přetáhnout blok **TRCV_C** do vytvořeného Organizačního bloku.



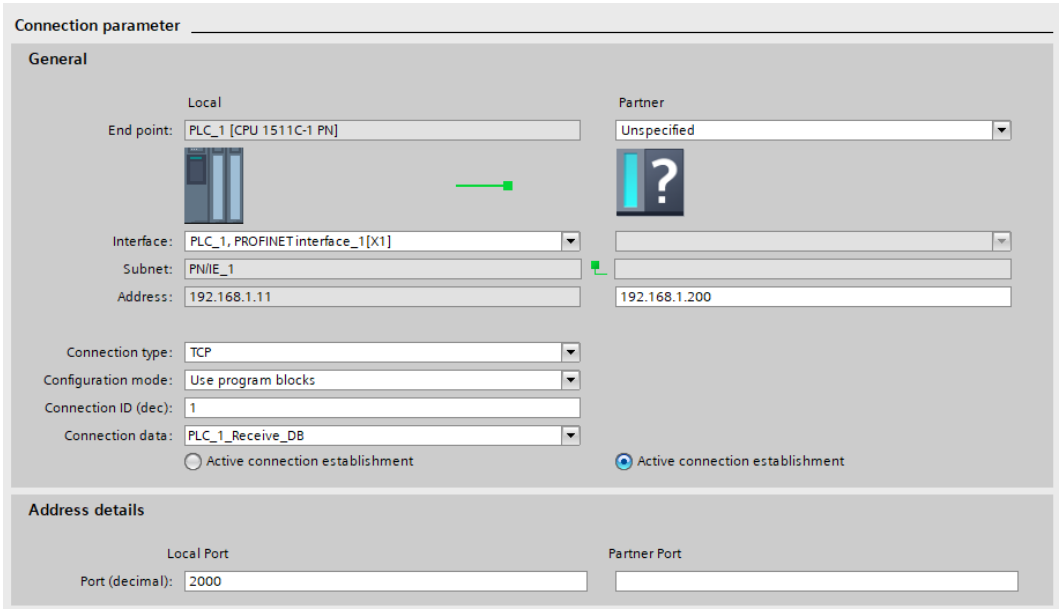
5. Nastavení **TRCV_C** - na vstup DATA vložit datablok RECIEVED_Data_block, EN_R: TRUE, výstupy jsou pouze informativní. Ve vnitřním nastavení - Connection data: <new> ,partner: unspecified, partner address: ip počítače (192.168.1.200), active connection establishment zaškrtnuto u partnera, local port: 2000



6. Vložení bloku **TSEND_C** - stejný postup jako u předchozího bloku

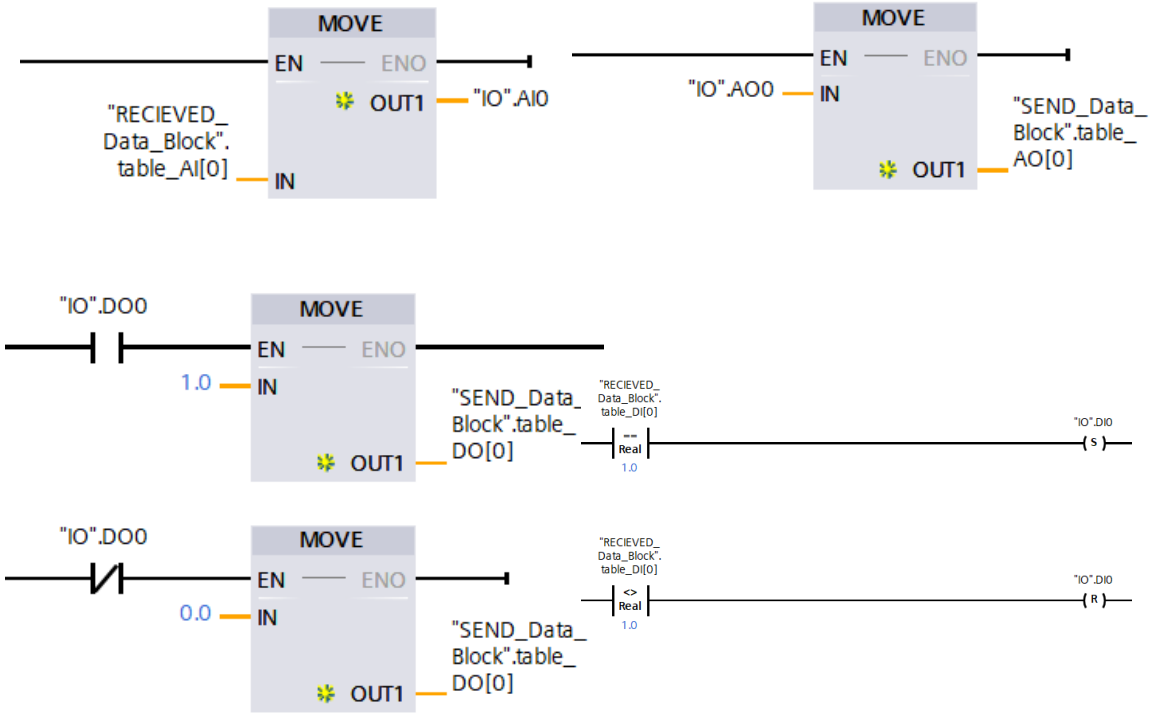


7. Nastavení **TSEND_C** - na vstup DATA vložit datablok SEND_Data_block, na vstup REQ připojit hodinový signál s příslušnou periodou odesílání(5Hz) - zpráva se odešle vždy na náběžnou hranu tohoto signálu, výstupy jsou pouze informativní. Ve vnitřním nastavení - partner: unspecified, Connection data: zvolit proměnnou vytvořenou v předešlém kroku. Všechna potřebná data se doplní



8. Přetypování dat - spojení je nyní funkční, ale všechna data jsou uložená v typu real. Je proto potřeba přetypovat je na bitovou logiku. Vytvořil jsem si datablok IO, do kterého ukládám všechny využívané proměnné vstupů a výstupů. Přesun analogových hodnot lze vyřešit blokem Move, digitální jedním směrem spínáním bloku move a druhým porovnáváním reálného čísla a nastavováním SET/RESET.

14	DI12	Bool	false	
15	DI13	Bool	false	
16	DI14	Bool	false	
17	DI15	Bool	false	
18	AI0	Real	0.0	
19	AI1	Real	0.0	
20	AI2	Real	0.0	
21	AI3	Real	0.0	
22	DO0	Bool	false	
23	DO1	Bool	false	
24	DO2	Bool	false	
25	DO3	Bool	false	
26	DO4	Bool	false	

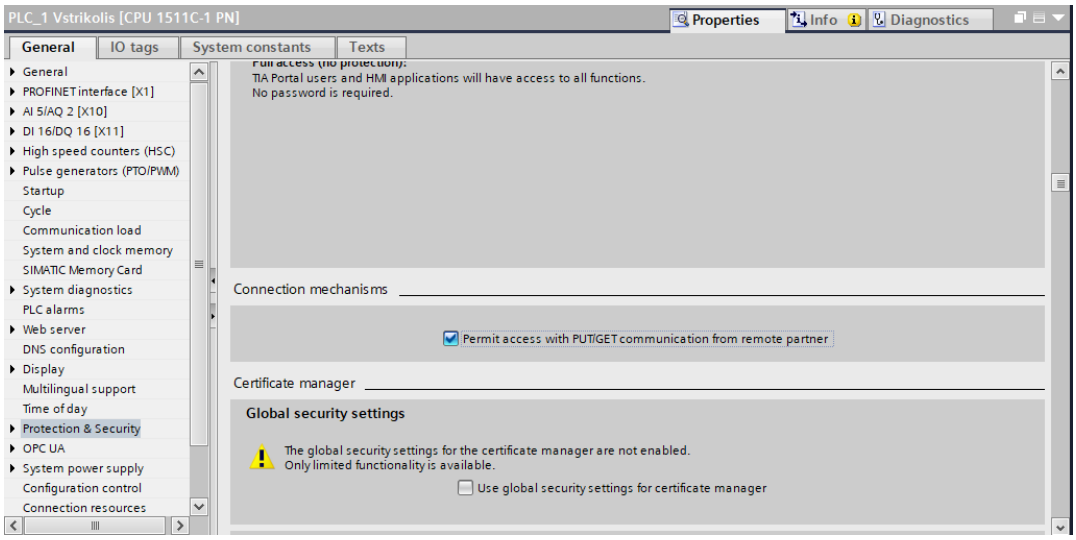


Nyní lze zapisovat a číst z proměnných v databloku IO a data z něj se budou automaticky odesílat do Simulinku.

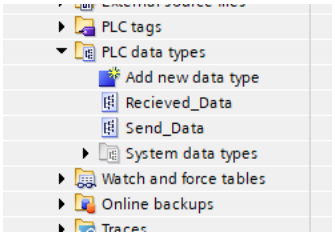
B.2 Úloha s řídícím PLC

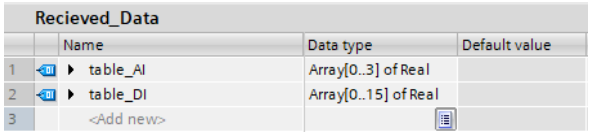
Vše začíná vytvořením projektu a přidáním vhodného PLC. Samotné vytvoření projektu nebudu popisovat a přejdu rovnou k prvkům, potřebným pro komunikaci.

1. Povolení funkcí GET/PUT - Device configuration - Protection and security - Connection mechanisms - zaškrtnout Permit acces with PUT/GET communication from remote partner.

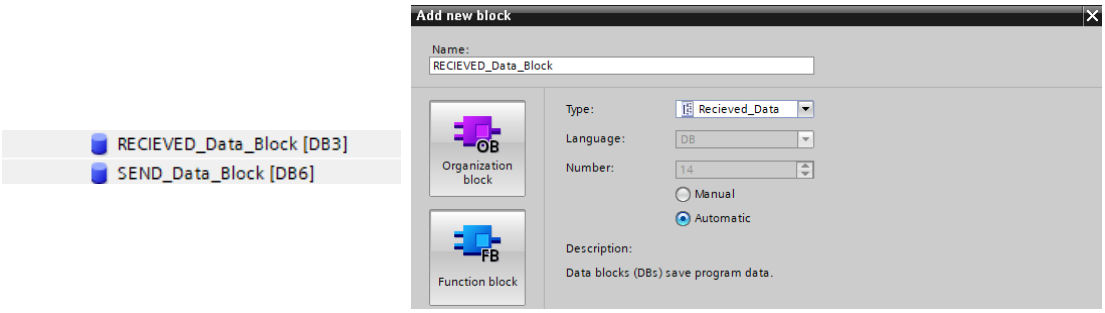


2. Vytvoření Datatypů - v levé záložce, PLC data types - add new data type. Je potřeba vytvořit dva datatypy o velikosti odesílaných a přijímaných dat. Konkrétně jeden datatyp pro příjem dat (obsahuje čtyřprvkové pole AI typu Real a 16 prvkové pole DI typu Real) a jeden datatyp pro odesílání dat (obsahuje pole 4 AO typu Real a 16 DO typu Real) - typ musí být stejný pro snažší komunikaci, lze následně přetypovat v PLC.

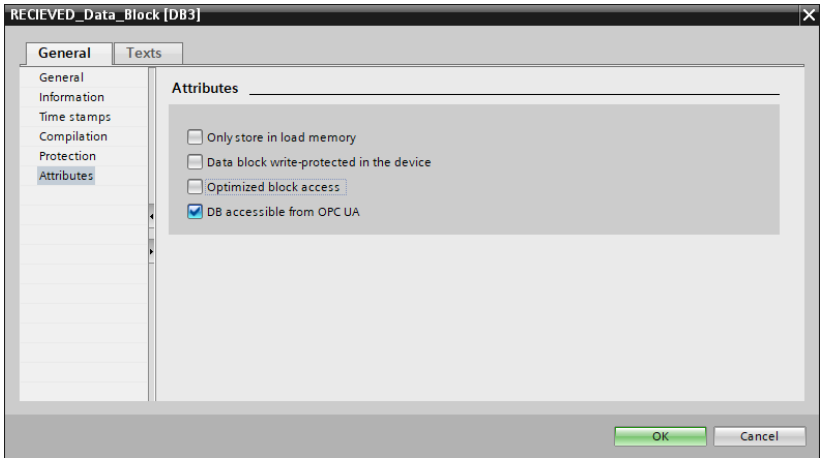




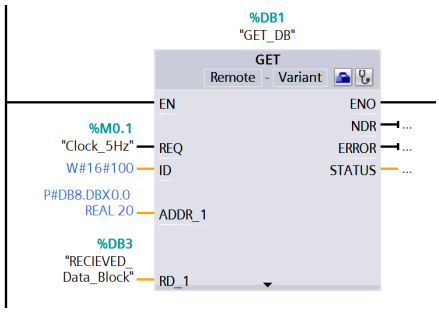
3. Vytvoření dvou Datablok - v levé záložce, program blocks - add new block. Vytvoření dvou data blocků, do typu se zadá příslušný datatyp. Znovu jeden pro odesílání a jeden pro přijímání - (SEND_Data_Block a RECIEVED_Data_Block)

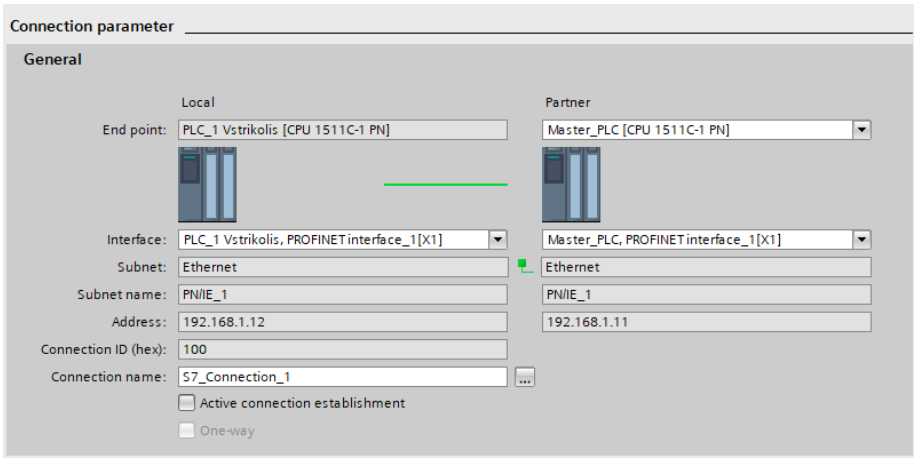


4. Vypnutí Optimized block access - Pro oba databloky(přijímání i odesílání) je potřeba vypnout Optimized block access. Blok Properties - Attributes - Optimized block access.



5. Čtení přijímaných dat pomocí GET - v pravé záložce - Communication - S7 communication - blok GET. Na vstup REQ řipojit hodinový signál s příslušnou periodou odesílání(5Hz) - zpráva se odešle vždy na náběžnou hranu tohoto signálu, na vstup ADDR_1 adresa místa, ze kterého chceme číst, typ čteného čísla a počet čtených čísel(REAL 20). Na vstup RD_1 se připojí datablok RECIEVED_Data_Block. Ve vnitřním nastavení se pak pouze zvolí správný partner - Master_PLC a všechny ostatní údaje se vyplní automaticky.





6. Zapisování výstupních dat pomocí PUT - v pravé záložce - Communication - S7 communication - blok PUT. Jeho nastavení je totožné jako u bloku GET, jediný rozdíl je vstup ADDR_1, na který se zapíše adresa místa druhého PLC, kam se mají data zapisovat a vstup SD_1, na který se připojí datablok SEND_Data_Block.