ČESKÉ VYSOKÉ
UČENÍ TECHNICKÉ

V PRAZE

FAKULTA
STROJNÍ

BAKALÁŘSKÁ PRÁCE

2018

ARTSIOM
PUNKO

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Punko** Jméno: **Artsiom** Osobní číslo: **438315**

Fakulta/ústav: **Fakulta strojní**

Zadávající katedra/ústav: **Ústav přístrojové a řídicí techniky**

Studijní program: **Strojírenství**

Studijní obor: **Informační a automatizační technika**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Algoritmy sběru dat z PLC do SCADA

Název bakalářské práce anglicky:

PLC to SCADA data acquisition algorithms

Pokyny pro vypracování:

1) Proveďte rešerši možností měření času událostí v PLC podle IEC 1131-3 i mimo tuto normu a pro několik variant vytvořte kód pro PLC a prakticky ověřte.
2) Proveďte rešerši možností organizace ukládání dat v PLC podle IEC 1131-3 i mimo tuto normu a pro několik variant vytvořte kód pro PLC a prakticky ověřte..
3) Popište komunikační protokol sběrnice Modbus, jeho varianty a zprovozněte komunikaci mezi vybraným PLC a SCADA systémem pomocí této sběrnice.
4) Proveďte rešerši algoritmů sběru dat z PLC do SCADA, vybrané algoritmy realizujte na daném PLC.
5) Sebraná data ukládejte do databáze.

Seznam doporučené literatury:

[1] L. Šmejkal a J. Černý, "Esperanto programátorů PLC", roč. 2014, č. 3, s. 40-43, 2014.
[2] L. Šmejkal a M. Martinásková, PLC a automatizace. 1. díl, 1. díl,. Praha: BEN - technická literatura, 1999.
[3] Firemní dokumentace PLC Tecomat, https://www.tecomat.cz/
[4] Firemní dokumentace MySCADA, https://www.myscada.org/cs/
[5] Dokumentace ke komunikačnímu standardu MODBUS, http://www.modbus.org/

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Mgr. Jakub Jura, Ph.D., U12110.3

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

Datum zadání bakalářské práce: **18.04.2018** Termín odevzdání bakalářské práce: **15.06.2018**

Platnost zadání bakalářské práce: _____

Ing. Mgr. Jakub Jura, Ph.D. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Ing. Michael Valášek, DrSc. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

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

18.04.2018
Datum převzetí zadání

Podpis studenta

© ČVUT v Praze, Design: ČVUT v Praze, VIC

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně s tím, že její výsledky mohou být dále použity podle uvážení vedoucího bakalářské práce jako jejího spoluautora. Souhlasím také s případnou publikací výsledků bakalářské práce nebo její podstatné částí, pokud budu uveden jako spolu autor.

Dne ……. *Podpis …….*

Poděkování

Na tomto místě chci poděkovat vedoucímu mé bakalářské práce Mgr. Ing. Jakubu Jurovi, PhD. za odborné vedení, cenou pomoc, rady a bezmeznou trpělivost při zpracování této práce.

Název: Algoritmy sběru dat z PLC do SCADA

Abstrakt

Práce se věnuje algoritmům sběru dat z programovatelného automatu PLC do SCADA (systému dispečerského řízení a sběru dat). Teoretická se část se zabývá programovatelnými automaty PLC, sloužícími k řízení technologických procesů. Za účelem jejich rozvoje vznikla norma IEC EN 61131. Pro komunikaci mezi PLC a SCADA systémem byl použit komunikační protokol Modbus. Navázání komunikace pomocí protokolu Modbus s využitím reálného PLC automatu Tecomat Foxtrot se věnuje praktická část práce. Došlo k naprogramování PLC v režimu podřízeného zařízení s využitím programovacího prostředí Mosaic.

Klíčová slova:

PLC, SCADA, Modbus TCP, Mosaic,, komunikace, algoritmus sběru dat

Title: PLC to SCADA data acquisition algorithms

Abstract

The following Bachelor thesis deals with data acquisition algorithms from the Programmable Logic Controller (PLC) to the Supervisory Control And Data Acquisition (SCADA). The theoretical part deals with programmable controllers (PLCs), which provided for control of technological processes. For their development, IEC EN 61131 standard was used. For communication between the PLC and the SCADA system, the Modbus communication protocol was used. Practical part of the thesis is devoted to establishing communication with PLC Tecomat Foxtrot using protocol Modbus. PLC was programmed in slave mode using the Mosaic developmental environment.

Keywords:

PLC, SCADA, Modbus TCP, Mosaic, communication, data acquisition algorithms

Obsah

Seznam použitých zkratek

PLC - Programmable logic controller

LD - Ladder diagram

FBD - Function block diagram

IL - Instruction List

ST - Structured text

SFC - Sequential function chart

PDU - Protocol data unit

ADU - Application data unit

SCADA - Supervisory control and data acquisition

RTU - Remote terminal unit

HMI - Human -machine interface

0 Úvod

Tématem této práce je přenos dat z řídicího systému programovatelného automatu Programmable Logic Controller (dále jen „PLC“) do nadřazeného SCADA systému za použití průmyslové sběrnice Modbus TCP. Práce uvádí, co PLC je, včetně principu a základního popisu normy IEC 61131-3, popisující tento řídicí systém. Zmiňuje zástupce PLC od firmy Teco a popisuje vývojové prostředí Mosaic pro programovaní PLC od této společnosti. Poté se zabývá sběrnicí Modbus, popisuje ji a uvádí příklad použiti pro komunikace PLC Tecomat Foxtrot a SCADA. Dále popisuje systém SCADA a dále konkrétně mySCADA od společnosti mySCADA Technologies s.r.o., ve kterém byla zhotovena praktická část této práce. Obsahuje kompletní popis algoritmu sběru dat z PLC do SCADA a také kompletní popis nastavení systému. V práci je uveden postup zprovoznění komunikace mezi PLC Foxtrot CP-1008 a MySCADA, popisuje nastavení PLC jako podřízené zařízení.

1 Programovatelný automat – PLC

Programovatelný automat PLC je průmyslový řídicí systém používaný pro automatizaci technologických procesů v reálném čase. Je postaven na mikroprocesorové bázi. Na rozdíl od běžných počítačů vykonává program v cyklech.

1.1 Historie

Všechny výrobní procesy, které musí probíhat bezpečně a efektivně z hlediska nákladů a hospodárnosti, vyžadují systém řízení. V 70. letech se pro tyto účely používala elektromechanická relé. V roce 1969 americká společnost Bedford Asociace spolu s General Motors vyvinula mikroprocesorové zařízení, které umožnilo zapojení signálních vodičů, které jsou k němu připojeny, v různých kombinacích. Tyto kombinace byly nastaveny pomocí řídicího programu, který byl kompilován na obrazovce počítače a poté vložen do paměti řadiče. Šlo o první programovatelný logický automat (PLC), tedy průmyslový regulátor používaný pro automatizaci technologických procesů.

1.2 Princip PLC

Od prvních verzí PLC zůstala jejich struktura prakticky beze změny. Fyzicky je PLC jednotkou, která má určitou sadu vstupů a výstupů pro připojení snímačů a akčních členů.

Obrázek č. 1: Schématická struktura PLC

Algoritmy řízení PLC jsou uloženy v paměti uživatelského programu, který je cyklicky vykonáván. Pro změnu algoritmu nejsou vyžadovány žádné hardwarové úpravy. Hlavním provozním režimem je dlouhodobé autonomní řízení procesů, často v náročných podmínkách a bez větší údržby.

1.3 Běh programu

Řídicí program programovatelného automatu je zapsán v podobě posloupnosti instrukcí paměti uživatelského programu. „Centrální jednotka postupně čte z této paměti jednotlivé instrukce, provádí příslušné operace s daty v zápisníkové paměti a zásobníku, případně provádí přechody v posloupnosti instrukcí. Jsou-li provedeny všechny instrukce požadovaného algoritmu, provádí centrální jednotka aktualizaci výstupních proměnných do výstupních periferních modulů a aktualizuje stavy ze vstupních periferních modulů do zápisníkové paměti. Tento děj se stále opakuje a nazýváme jej cyklem programu.“ [4]

Jednorázová aktualizace stavů vstupních proměnných během celého cyklu programu odstraňuje rizika vzniku hazardních stavů při chodu algoritmu řízení, takže během výpočtu nemůže dojít ke změně vstupních proměnných.

Obrázek č. 2: Cyklus řešení uživatelského programu [4]

1.4 Norma IEC EN 61131

Základy programovatelných logických řadičů jsou upraveny mezinárodní normou IEC 61131. Vývojový proces tohoto standartu začal v roce 1979, první verze byla vydána roku 1982. Po prvním seznámení s normou se ukázalo, že standard je ve formě jednotného dokumentu příliš složitý pro práci. Proto byl rozdělen do několika části:

- Část 1: Obecné informace
- Část 2: Požadavky na vybavení a zkoušky
- Část 3: Programovací jazyky
- Část 4: Pokyny pro uživatele
- Část 5: Komunikace

- Část 6: Funkční bezpečnost
- Část 7: Programování fuzzy řízení

1.4.1 Programovací jazyky normy

Standard IEC 61131-3 definuje programovatelné jazyky pro programovatelné automaty PLC. Při vývoji normy existovalo mnoho variací jazyků pro programovatelné automaty, takže nebylo možné vybrat jednu z existujících variant jako společný jazyk. Proto norma IEC 61131 popisuje hned pět programovacích jazyků: dva grafické jazyky: Ladder Diagram (LD) a Function Block Diagram (FBD), dva textové jazyky: Instruction List (IL) a Structured text (ST) a zbylý Sequential Function Chart (SFC).

Jeden z pěti jazyků podporovaných normou IEC 61131-3 je jazyk zvaný „Instruction List“. Program v něm je složen z posloupnosti jednoduchých operací tvořených základními instrukcemi. Daný způsob dovoluje maximálně optimalizovat program z hlediska jeho velikosti i rychlosti vykonávání, protože jej lze vytvořit bez všech zbytečných přesunů dat, avšak nehodí se pro programování složitých aplikací.

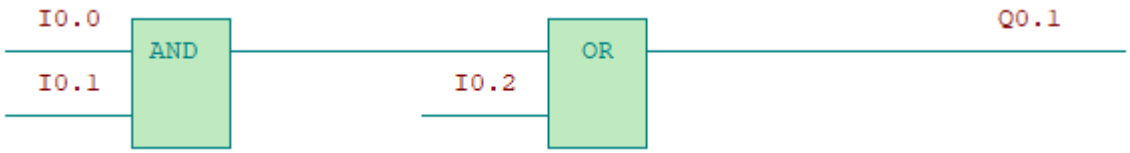
```
1.  LD I1.0
2.  O  M2.3
3.  =  M5.3
4.  R  I0.2
```

Obrázek č. 3: Příklad využití jazyka IL

Jazyk Ladder Diagram (LD) je založen na grafické reprezentaci reléové logiky. Program v jazyce reléové logiky má intuitivní grafické uživatelské rozhraní, které zobrazuje logické operace, jako elektrický obvod s uzavřenými a otevřenými kontakty. Hlavní výhodou tohoto zápisu je jeho dobrá přehlednost. Mezi nevýhody naopak řadíme nevhodnost pro aritmetické operace a práci s daty.

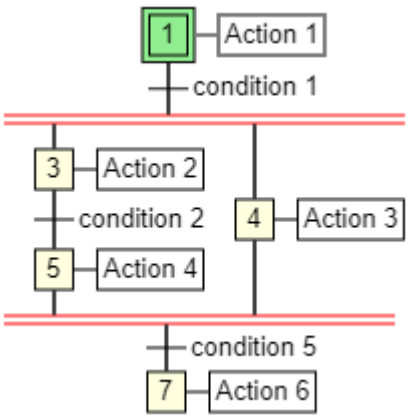
Obrázek č. 4: Příklad využití jazyka LD

Function Block Diagram (FBD) je grafický jazyk funkčních bloků. Program se v jazyce FDB vytváří pomocí logického blokového schématu. Vyjadřuje chování funkcí, funkčních bloků a programů jako souboru vzájemně provázaných grafických bloků, obdobně jako tomu je v elektronických obvodových diagramech. Pro vyjádření logických funkcí se zde používají standardní funkční bloky a dále čítače, časovače, komunikační bloky nebo podle potřeby speciální bloky. Jazyk má přehledný zápis kódu. Jazyk je však méně vhodný pro složitější zpracování analogových signálů a složitých algoritmů.



Obrázek č. 5: Příklad využití jazyka FBD

Sequential Function Chart (SFC) popisuje sekvenční chování řídicího programu. Jazyk SFC je odvozen ze symboliky Petriho sítí a umožňuje rozložit úlohu řízení na zvládnutelné části při zachování přehledu o chování celku. Sekvenční funkční diagram je složen z kroků a přechodů. Každý krok představuje stav řízeného systému a je k němu sobě přiřazen blok akcí. Přechod je spojen s podmínkami pro deaktivaci předešlého kroku, a naopak aktivaci kroku, který bude následovat. Každý prvek, tedy přechod i blok akcí, může být naprogramován v libovolném jazyku definované normou IEC 61131-3. Základem prvků SFC je norma dále IEC 848, jež je mezinárodní verzí francouzského standardu „Grafcet“.



Obrázek č. 6: Příklad využití jazyka SFC

Textový jazyk ST je výkonný vyšší programovací jazyk s kořeny v jazycích Pascal a C. Syntaxe jazyka je definována povolenými příkazy a výrazy. V jazyce je definováno několik typů příkazů, například přiřazení, vyvolání funkce, návrat, výběr a další. Příkazy jsou oddělovány středníkem a může jich být více na jednom řádku. Jazyk ST je vhodným nástrojem pro definování komplexních funkčních bloků, které mohou následně být užity v libovolném programovací jazyku.

```
IF pulz = false THEN
timer_start := true;
END_IF;
```

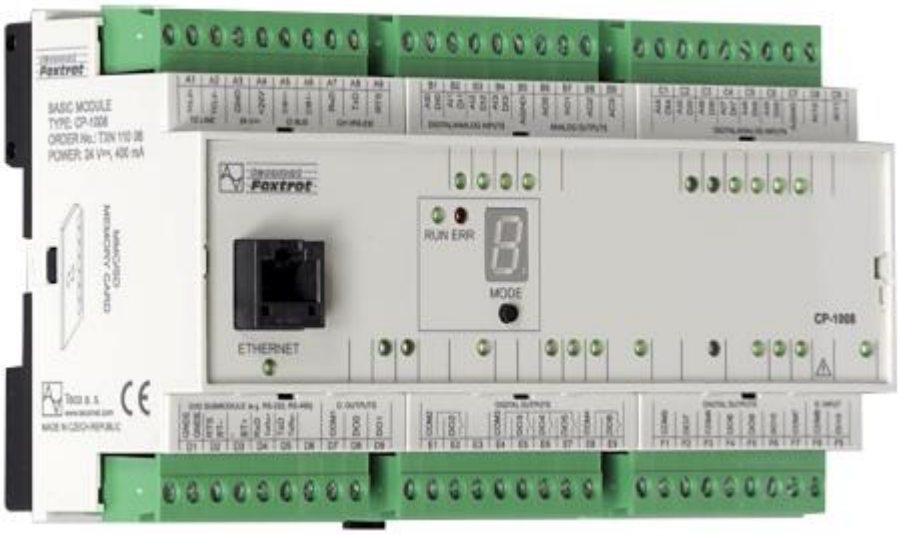
Obrázek č. 7: Příklad využití jazyka ST

1.5 Teco a.s.

Společnost Teco a.s. je významným českým výrobcem průmyslových řídicích systémů v kategorii systémů PLC, které jsou vyvíjeny, vyráběny a testovány podle výše zmíněných standardů mezinárodní normy IEC EN 61131. Společnost vyrábí automaty určené pro řízení nejrůznějších strojů z oblasti průmyslu, především však strojírenského, chemického nebo dřevozpracujícího průmyslu. Teco a.s. dodává na trh široký výběr PLC, počínaje automaty s několika vstupy a výstupy vhodnými například pro chytré domy a velkými systémy s tisíci vstupy a výstupy, určenými pro složité technologické operace konče. Hlavní výhodou výrobce je software, který společnost poskytuje ke svým automatům zdarma.

1.5.1 Tecomat Foxtrot CP-1008

Ve své práce jsem použil Tecomat Foxtrot CP-1008 českého výrobce Teco a.s., a proto zde popíši jeho funkce a základní údaje o něm.



Obrázek č. 8: Tecomat Foxtrot CP-1008 [4]

Společnost Teco a.s. k tomuto automatu uvádí: „CP-1008 je základním modulem modulárních programovatelných automatů řady Foxtrot. Jednotlivé základní moduly se liší počtem nebo typem vstupů a výstupů a indikačními a ovládacími prvky. CP-1008 je vybaven deseti víceúčelovými vstupy, z nichž každý je využitelný buď jako analogový, nebo jako binární, dvěma

analogovými vstupy, využitelnými pro měření napětí nebo pro připojení termočlánků a lambda sondy, jedním binárním vstupem pro napětí 230 V AC, čtyřmi triakovými výstupy, využitelnými jako výstupy PWM, čtyřmi analogovými výstupy 0 až 10 V a sedmi reléovými výstupy (dva reléové výstupy jsou propojeny s triakovými výstupy a umožňují řízení směru AC motorů).“[4]

Základní modul CP-1008 je vybaven centrální jednotkou (CPU) řady K, která je vhodná pro aplikace s výraznějšími požadavky na výkon. Obsahuje zálohovanou paměť CMOS RAM pro uživatelské programy, data, tabulky, uživatelské registry a DataBox, paměť Flash pro zálohování uživatelského programu, slot pro běžné typy paměťových karet, obvod reálného času, síťové rozhraní Ethernet, dva sériové kanály (jeden s pevným rozhraním RS-232, druhý s pozicí pro volitelné submoduly), jeden komunikační kanál s rozhraním CIB pro připojení externích periferií a systémové rozhraní TCL2 určené pro připojení rozšiřovacích modulů, které zvyšují počet I/O systému. [4]

1.5.2 Programování Tecomatu

Hlavní výhodou výrobce je software, který firma poskytuje zdarma. Prostředí označované Mosaic je komplexním vývojovým nástrojem pro programování běžných i náročnějších aplikací pro systémy Tecomat. Mosaic umožňuje snadnou tvorbu a optimalizaci programů. V prostředí Mosaic lze vytvářet i rozsáhlejší projekty, které čítají větší množství řídicích systémů nebo vzdálených I/O modulů. Mosaic využívá moderní technologie a architektura prostředí i jeho nástroje zohledňují normu IEC 61131-3.

1.6 Flexibilita programovacích jazyku dle normy IEC

Přestože norma obsahuje velké množství standardních funkcí a funkčních bloků, vývojáři normy nemohli vyhovět všem požadavkům a přáním programátorů. Proto IEC 1131-3 podrobně popisuje mechanismy, pomocí nichž mohou výrobci a uživatelé definovat nové datové typy, funkce a funkční bloky. Tím byla normě dána vysoká flexibilita a možnost dalšího rozvoje. Programátoři pomocí rozsáhlé sady standardních funkcí a funkčních bloků, svých vlastních prvků a adaptability normy mohou řešit širokou škálu problémů. Využití standardu se díky tomu „rozšiřuje“ a lze předvídat, že bude schopen sloužit mnoha generacím nových technologií řízení. Jako potvrzení můžeme zmínit například měření času události, které je realizováno pomocí prvku normy.

1.6.1 Měření času události

Všechny technologické procesy využívají čas. Téměř žádný aplikační program monitorovacích a řídicích systémů nemůže pracovat bez operací s časem. Spolu s logickými výrazy a zpracováním bitových linek měření a vytváření časových intervalů jsou nejdůležitějšími akcemi.

Metoda pomocí časovače TON

Časovač TON se používá tam, kde potřebujeme časové zpoždění. Časovač se spustí, když vstup IN se změní z „0“ na „1“ (pozitivní hranice signálu). Po odpočítávání casu PT výstup Q změní stav signálu na „1“. Když se stav signálu na vstupu IN změní z hodnoty „1“ na „0“, výstup Q se resetuje. Funkce časovače se znovu spustí, když je na vstupu startu zjištěna nová změna pozitivního signálu. Tento blok můžeme použít pro měřeni času události. Vystup ET ukazuje aktuální čas, který uplynul po spuštění časovače.

```
timer : TON; // deklarace TON

timer_start := tlacitko;

timer(IN := timer_start, PT := TIME#20d,
          ET => time_on, Q => pulz);

   IF timer_start = true THEN
   saved_time := time_on;
   END_IF;
```

V tomto příkladu jsem použil časovač TON pro měření doby stisknutí tlačítka. Po stisknutí tlačítka se vstup časovače změní z "0" na "1". Na výstupu časovače ET bude zobrazen aktuální čas, který uplynul od stisknutí tlačítka. Dále tento čas bude uložen do proměnné „saved_time“. Metoda je vhodná pro měření malých a středních časových intervalů. Maximální možná hodnota se rovná 24 dní 2 hodiny 31 minut a 23 sekund. Přesnost je omezena dobou cyklu PLC, která je obvykle kolem 10 ms.

Metoda pomocí funkci systémového času

Metoda měření času pomoci funkci *Rego_GetTime* a *Rego_GetDateTime*, které vrací aktuální systémový čas nebo datum a čas. Metoda je vhodná pro měření velkých úseku času. Maximální měřená hodnota je omezena pouze maximálním systémovým časem PLC.

```
start := tlacitko;

    IF start = false THEN
    start_time := Rego_GetTime();
    END_IF;

    IF start = true THEN
    time := Rego_GetTime()-start_time;
    END_IF;
```

Pro získaní času události odečteme čas začátku události od času jejího skončení. Hlavní nevýhodou je potřeba použití aritmetiky pro výpočet času.

2 Možností organizace ukládání dat v PLC

2.1 Typy dat

V rámci společných prvků jsou definovány typy dat. Definování datových typů předchází vzniku chyb v samém počátku tvorby projektu, proto je nutné definovat typy všech použitých parametrů. Běžně používanými datovými typy jsou: **BOOL**, **BYTE**, **WORD**, **INT**, **REAL**, **DATE**, **TIME**, **STRING** a další. Elementární datové typy jsou charakterizované šířkou dat (počtem bitů) a případně i rozsahem hodnot. U objektů elementární datových typy nemůže programátor pracovat s vnitřní strukturou proměnné. Z těchto základních datových typů je pak možné odvozovat vlastní uživatelské datové typy, tzv. odvozené datové typy. Tímto způsobem můžeme např. definovat jako samostatný datový typ analogový vstupní kanál a opakovaně ho používat pod definovaným jménem. [1]

2.2 Proměnné

Proměnné jsou používány pro ukládání a zpracování informací. Každá proměnná je definována jménem proměnné a datovým typem. Datový typ určuje velikost proměnné v paměti a zároveň do značné míry určuje způsob zpracování. Pro definice proměnných jsou k dispozici standardní datové typy (BOOL, BYTE, INT, …) a některé z odvozených. Použití těchto typů závisí na tom, jaká informace bude v proměnné uložena (např. typ BOOL pro informace typu ANO-NE, typ INT pro uložení celých čísel se znaménkem apod.).

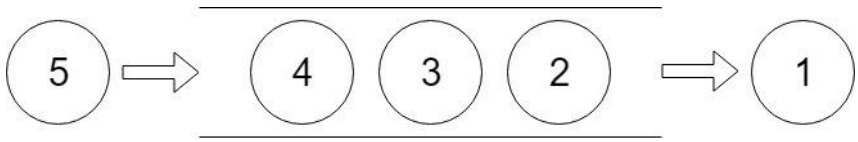
Zápis hodnot proměnných do paměti PLC systému je automaticky zajišťován programovacím prostředím. Pokud je to potřeba, může umístění proměnné v paměti definovat i uživatel. Proměnné mohou být přiřazeny explicitně k hardwarovým adresám (např. vstupům, výstupům) pouze v konfiguracích, zdrojích nebo programech. Tímto způsobem je dosaženo vysokého stupně hardwarové nezávislosti a možnosti opakovaného využití softwaru na různých hardwarových platformách. Oblast působnosti proměnných je běžně omezena pouze na tu programovou organizační jednotku, ve které byly deklarovány (proměnné jsou v ní lokální). To znamená, že jejich jména mohou být používána v jiných částech bez omezení. Tímto opatřením dojde k eliminaci řady dalších chyb. Pokud mají mít proměnné globální působnost, např. v rámci celého projektu, pak musí být jako globální deklarovány (VAR_GLOBAL). Aby bylo možné správně

nastavit počáteční stav procesu nebo stroje, může být parametrům přiřazena počáteční hodnota při startu nebo studeném restartu. [1]

2.3 Buffer

V oblasti výpočetní techniky je vyrovnávací paměť (anglicky „buffer“) částí paměti, která slouží k dočasnému ukládání dat při jejich výměně, tedy vstupu nebo výstupu. Výměna dat může proběhnout jak s externím zařízením, tak s procesem v mikropočítači. Buffery mohou být implementovány v hardwaru nebo softwaru, avšak převážná většina vyrovnávacích pamětí je implementována v softwaru. Buffery se používají, když existuje rozdíl mezi rychlostí, jakou jsou data přijata, a rychlostí zpracování, nebo když jsou tyto rychlosti proměnné.

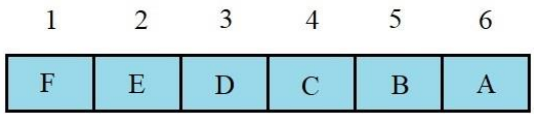
2.3.1 Lineární fronta



Obrázek č. 9: Schematické zobrazeni lineární fronty

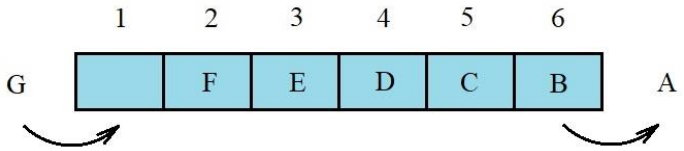
Fronta (anglicky „queue“) je jedním ze základních typů bufferu a slouží k ukládání a výběru dat takovým způsobem, aby prvek, který byl uložen jako první, byl také jako první vybrán. Fronta je také nazývána strukturou typu FIFO (první v, první, první, první). Výhodami této metody jsou úspory paměti ve srovnání s jinými metodami a snadné použiti. Oproti tomu nevýhody můžeme spatřovat omezení maximálního počtu položek velikostí pole.

Lineární fronta funguje způsobem, kdy se vytvoří prázdná lineární vyrovnávací paměť s určitou předem stanovenou délkou. Jedná se například o vyrovnávací paměť se šesti elementy. Poté jsou po jednom přidávány další prvky. Paměť je plná, obsahuje-li 6 prvků.



Obrázek č. 10: Schématické zobrazení plné lineární fronty

Pro zápis nové hodnoty v plném bufferu posuneme hodnoty vpravo, jak je zachyceno na obrázku č. 11.



Obrázek č. 11: Posunuti hodnot lineární fronty pro zápis nové hodnoty

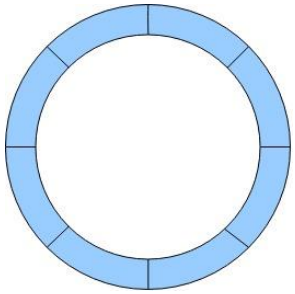
Realizace lineární fronty v jazyce ST:

```
IF podminka = true THEN
      PoleBuffer[5] := PoleBuffer[4];
      PoleBuffer[4] := PoleBuffer[3];
      PoleBuffer[3] := PoleBuffer[2];
      PoleBuffer[2] := PoleBuffer[1];
      PoleBuffer[1] := value;

         END_IF;
```

V případě, že dojde ke splnění podmínky, hodnoty budou posunuty a do uvolněného místa bude zapsána hodnota, tedy proměnná *value*.

2.3.2 Kruhový buffer

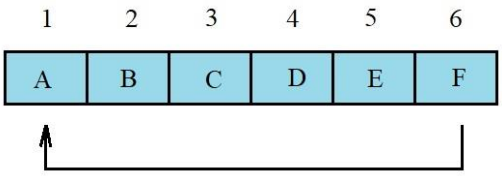


Obrázek č. 12: Schematické zobrazeni kruhového bufferu

Princip kruhového bufferu spočívá v postupu, kdy při zaplnění vyrovnávací paměti jsou nová data zapsána opět na začátek. Tato struktura snadno poskytuje schopnost vyrovnávat datové toky.

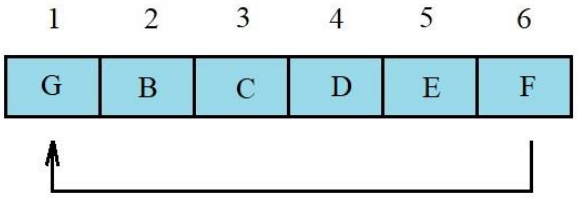
Kruhové vyrovnávací paměti se často používají k uspořádání různých front zpráv a přijímání a přenosu vyrovnávacích pamětí různých komunikačních rozhraní. Popularita KB je způsobena tím, že se jedná o jeden z nejjednodušších a nejúčinnějších způsobů, jak organizovat FIFO.

Kruhový buffer stojí na postupu, kdy se vytvoří prázdná kruhová vyrovnávací paměť s určitou předem stanovenou délkou. Jedná se například o vyrovnávací paměť se šesti elementy. Pak jsou jednotlivě přidány další prvky. Pokud je ve vyrovnávací paměti 6 prvků, je plná:



Obrázek 12: Schematické zobrazeni plného bufferu

Pokud budeme pokračovat v zapisování hodnot, nová data přepíšou stará. V našem případě přidáním prvků G přepíšeme A v buňce 1:



Obrázek č. 13: Přepsaní nejstarší hodnoty

Realizace kruhového bufferu v jazyce ST:

```
9:  VAR
10: Pole : array [0..6] of INT;
11: END_VAR
12:
13: counter (CU := pulz, R := reset, PV :=6, Q => Quit, CV =>
    counter_out);

14:   IF Quit = true THEN
15:   reset := true;
16:   END_IF;
```

```
17:    IF pulz = true THEN
18:     Pole [counter_out] := counter_out2;
19:     END_IF;
20:
21:     IF reset = true and Quit = false THEN
22:     reset := false;
23:     END_IF;
```

3 Komunikační protokol Modbus a jeho varianty

Modbus je otevřený komunikační protokol založený na architektuře master-slave. Je široce používán v průmyslu pro organizování komunikace mezi elektronickými zařízeními. Modbus byl vyvinut společností Modicon (v současné době vlastněn společností Schneider Electric) pro komunikace programovatelných logických automatu. Specifikace protokolu byla poprvé publikována v roce 1979. Jednalo se o otevřený standard, který popisuje formát zpráv a jejich přenos v síti tvořené různými elektronickými zařízeními. Vývoj a aktualizaci protokolů Modbus od dubna 2004 řídí organizace Modbus. Organizace Modbus je sdružení uživatelů a dodavatelů standardu Modbus, která se zasazuje o další používání této technologie.

3.1 Výhody protokolu

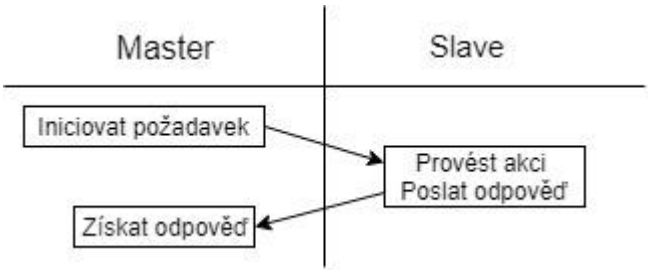
Hlavními výhodami protokolu Modbus je otevřenost a hromadný charakter. Výrobci nyní produkují tisíce typů a modelů senzorů, ovládacích zařízení, modulů pro zpracování a normalizaci signálu, které podporují daný protokol. Téměř všechny průmyslové řídicí a monitorovací systémy mají softwarové a hardwarové možnosti pro práci se sítěmi využívajícími protokol Modbus. Další výhodou je, že Modbus nevyžaduje speciální rozhraní, kdežto Profibus a CAN vyžadují speciální čipy. To vše snižuje náklady na zvládnutí standardů pro vývojáři zařízení. Další výhodami jsou snadné nasazení a údržba.

3.2 Nevýhody protokolu

Protokol Modbus byl vyvinut v roce 1979 s přihlédnutím k potřebám a výpočetním výkonům té doby. Je třeba poznamenat, že nedostatek možností je důsledkem jednoduchosti protokolu, která ale usnadňuje studium a urychluje nasazení. Protokol určuje způsob přenosu pouze dvou typů dat. Protokol nepopisuje počáteční inicializaci systému. Přiřazení sítové adresy a předepisování systémových parametrů pro každé konkrétní zařízení se provádí ručně ve fázi přizpůsobení a programování systému. Délka dotazu je omezena, data mohou být požadovány pouze z registrů po sobě jdoucích. Modbus neposkytuje způsob, kterým by mohl podřízený přístroj detekovat ztráty komunikace s nadřízeným.

3.3 Princip

Řídicí jednotky na sběrnici Modbus komunikují pomocí modelu master-slave založeného na transakcích, které sestávají z požadavku a odpovědi.



Obrázek č. 14: Transakce sběrnice MODBUS

Obvykle je v síti pouze jedno hlavní, takzvané „master“ zařízení a několik podřízených označovaných jako „slave“. Master zařízení iniciuje transakci vysláním požadavků. Master může žádost adresovat individuálně, nebo poslat zprávy pro všechna podřízená zařízení. Slave, který rozpozná svoji adresu, reaguje na požadavek, který mu byl adresován.

Specifikace Modbus popisuje strukturu požadavků a odpovědí. Jejich základem je balíček základních protokolů, tzv. PDU (Protocol Data Unit). Struktura PDU je nezávislá na typu propojení a zahrnuje funkční kód a datové pole.

Obrázek č. 15: Schéma PDU

Pokud dojde k odesílání požadavku, kód funkce udává, jaký typ operace slave provede. Kód funkce je kódován jednobajtovým polem a může mít hodnoty v rozsahu 1 až 127. Rozsah hodnot 128 až 255 je vyhrazen pro chybové kódy. Datové pole může mít proměnnou délku. Velikost paketu PDU je omezena na 253 bajtů. Je-li prováděná operace bezchybná, server odpoví zprávou, která v poli „kód funkce“ obsahuje kód provedené, tedy požadované, funkce jako indikátor úspěšného provedení požadavku. Požadovaná data, pokud nějaká jsou, předá klientovi server v datové části

odpovědi. V případě operace vykazující chybu, obsahuje pole „Kód funkce“ kód chyb, který je vrácen klientovi. Chybový kód obsažený v datové části upřesňuje důvod neúspěchu operace. Kód chyby lze získat přičítáním hodnoty 128 ke kódu funkce.

Obrázek č. 16: Srovnání odpovědi po bezchybné operaci a operaci s chybou

Podle typu sítě, ve které je protokol použit, je PDU rozšířena o další části a díky tomu tvoří zprávu na aplikační úrovni. Tento balíček se nazývá ADU (Application Data Unit). Formát ADU závisí na typu komunikační linky. Existuje několik variant ADU pro přenos dat přes asynchronní rozhraní a pro sítě TCP/IP. Pro Modbus TCP vypadá ADU takto:

Obrázek č. 17: Schéma ADU

V paketu ADU jsou ID transakce a ID protokolu obvykle nulové bajty. Délka paketu je dva bajty, které obsahují délku další části paketu. Adresa slave je adresou zařízení, jemuž je žádost adresována. Kód funkce je příkazem hlavního zařízení, například pro čtení hodnot z několika digitálních vstupů se používá funkce 02 (0x02). V oblasti dat jsou adresy a počet požadovaných elementu.

3.4 Verze protokolu

Existují tři hlavní verze protokolu Modbus, dva pro přenos dat na sériových komunikačních linkách (Modbus RTU a Modbus ASCII) a jeden pro přenos dat přes ethernetové sítě pomoci TCP/IP (Modbus TCP).

Modbus ASCII se používá v sériové komunikaci. V režimu ASCII je každý 8bitový byte posílán jako dvojice ASCII znaků. Pro kontrolu integrity se používá jednobajtový kontrolní součet. Začátek a konec zprávy jsou označeny speciálními symboly. Oproti režimu RTU je tedy pomalejší, ale umožňuje vysílat znaky s časovými rozestupy až 1 sekunda. Režim ASCII má také zjednodušený systém dekódování dat.

Modbus RTU se používá v sériové komunikace a využívá kompaktní binární reprezentaci dat, tj. data přenášena jako 8bitové binární symboly což poskytuje vyšší datovou rychlost. Integrita zpráv je zajištěna pomocí kontrolního součtu typu CRC. Modbus RTU je nejběžnější implementace pro Modbus. Zpráva Modbus RTU musí být spojité bez mezer mezi znaky.

Modbus TCP/IP nebo Modbus TCP je variantou Modbus používanou pro komunikaci přes sítě TCP/IP, připojující portem 502. Nevyžaduje výpočet kontrolního součtu, protože to poskytují nižší vrstvy.

3.5 Nastavení komunikace mezi PLC Foxtrot CP-1008 a SCADA systémem pomocí sběrnice MODBUS

Jak bylo uvedeno výše v kapitole 1.5.1 PLC od výrobce Teco Foxtrot CP-1008 může komunikovat pomoci Modbus TCP. Ve své práci jsem použil PLC v roli slave. Tento typ komunikace lze jednoduše nastavit v programovacím prostředí Mosaic pomocí knihovny ModbusRTUlib. Knihovna je sbírkou funkcí a funkčních bloku používaných pro vývoj softwaru. MOSAIC vždy obsahuje knihovny s vestavěnými funkcemi, které jsou zabudovány v překladači. Další knihovny mohou být přidány nebo odebrány programátorem prostřednictvím průzkumníku knihoven. Pro komunikace mezi vybraným PLC a SCADA stačí použit jeden funkční blok ModbusTCPslave knihovny ModbusRTUlib.

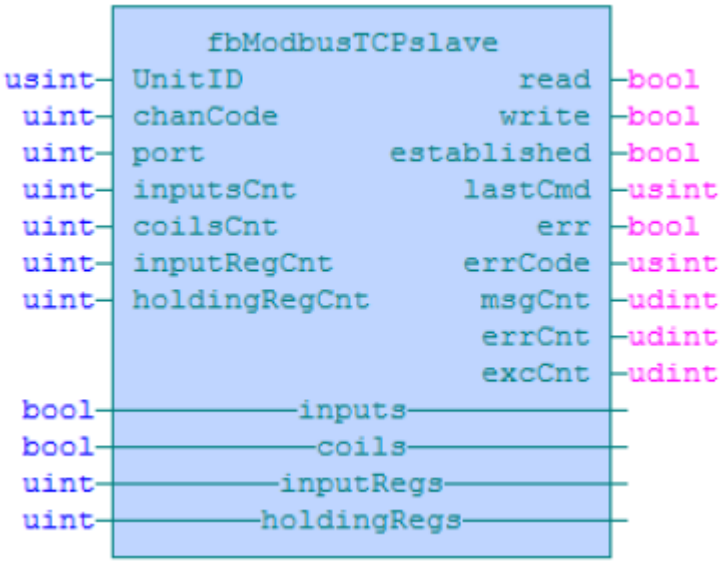
3.5.1 Knihovna ModbusRTUlib

Pro kanály a spojení, které jsou nastaveny v režimu UNI, lze realizovat režim Modbus. Knihovna ModbusRTUlib dovoluje provést komunikaci v obou množných režimech, tedy PLC Tecomat v roli serveru (master) i v roli klienta (slave). Součástí knihovny ModbusRTULib jsou následující funkce a funkční bloky:

- ModbusCmd a ModbusCmdTCP – funkce pro sestavení příkazu popisujícího jednu komunikaci
- ModbusRTUmas a ModbusTCPmas – FB provádějící komunikace podle příkazů
- GetModbusErrTxt – funkce pro převod chybového kódu na text

Pro nastavení komunikace sériovým kanálem jsou určeny další funkční bloky ModbusRTUmas a fbModbusRTUslave. Pro připojení protokolem Modbus TCP je určen funkční blok ModbusTCPmas, kde IP adresa koncových slave zařízení je zadávána v každém příkazu samostatně a jejich počet není omezen. IP adresy se dynamicky mění za chodu programu. Slave stranu protokolu řeší blok fbModbusTCPslave. Pokud je potřeba paralelní připojení více klientů je možné založit více instancí bloku z nichž každá bude pracovat na svém spojení. [12]

3.5.2 fbModbusTCPslave



Obrázek č. 18: Funkční blok fbModbusTCPslave [12]

Funkční blok ModbusTCPslave vytváří komunikační relace a interpretuje příkazy. Skrze zvolený komunikační kanál umožňuje tento funkční blok vysílání dat a příjem příkazů od masteru. Funkční blok nastavuje automaticky vlastnosti sériového kanálu v režimu UNI, aby se choval jako Modbus. Spojení je specifikované vstupem chanCode cílené na adresu danou vstupem UnitID. Ukazateli datových polí jsou: inputs (diskrétní vstupy), coils (cívky), inputRegs (vstupní registry) a holdingRegs (vnitřní registry). Počty objektů v polích jsou dány vstupy inputsCnt, coilsCnt, inputRegCnt a holdingRegCnt. Počty musí být rovny nebo menší, než je velikost proměnných, na které odkazují. Pokud tyto podmínky nejsou dodrženy, může dojít k zápisu do jiné části paměti. Zóny se mohou vzájemně překrývat. Překrytí zón umožňuje ke stejné paměti přistupovat jak po 16 bitových slovech tak i po bitech. [12]

Tabulka č. 1: Popis vstupních proměnných funkčního bloku:

Proměnná	Typ	Význam
UnitID	USINT	Adresa 1–247
chanCode	UINT	Kód kanálu (CH1_uni, ..., CH10_uni)
Port	UINT	Číslo místního portu (502)
inputsCnt	UINT	Počet diskrétních vstupů (počet BOOLů)
coilsCnt	UINT	Počet diskrétních výstupů (počet BOOLů)
inputRegCnt	UINT	Počet vstupních registrů (počet WORDů)
holdingRegCnt	UINT	Počet registrů (počet WORDů)

Tabulka č. 2: Popis výstupních proměnných funkčního bloku:

Proměnná	Typ	Význam
Read	BOOL	Byla čtena data
Write	BOOL	Byla zapsána data
Established	BOOL	TCP spojení navázáno
lastCmd	USINT	Poslední přijatý příkaz (viz. Kap. 1.3 Tab. 1)
Err	BOOL	Chyba

errCode	USINT	Chybový kód
msgCnt	UDINT	Počet zpracovaných zpráv
errCnt	UDINT	Počet chyb
excCnt	UDINT	Počet výjimek

Tabulka č. 3: Popis vstupních i výstupních proměnných v poli funkčního bloku:

Proměnná	Typ	Význam
inputs	BOOL	První diskrétní vstup v poli (musí být BOOL)
coils	BOOL	První diskrétní výstup v poli (musí být BOOL)
inputRegs	BOOL	První vstupní registr v poli
holdingRegs	USINT	První registr v poli

4 Sběr dat z PLC do SCADA

Systém SCADA (Supervisory Control And Data Acquisition) představuje hardwarový a softwarový komplex pro sběr dat a následné dispečerské řízení. Jde o hromadný sběr technických dat, například z výrobní linky, a jejich následné vyhodnocování. Výsledky jsou poté zobrazeny přes webové aplikace nebo ve webovém prohlížeči. SCADA systémy můžou sledovat velký počet objektů od 1 do 10 000, někdy vzdálených tisíce kilometrů od sebe. Takovými zařízeními jsou ropovody, plynovody, vodovody, rozvody elektrické energie, přívody vody, stanice diesel generátoru atd.

4.1 Účel a úkoly systémů SCADA

Hlavním úkolem systémů SCADA je shromažďování informací o řadě vzdálených objektů pocházejících z kontrolních bodů a zobrazení těchto informací v hlavním terminálu. Systém SCADA také poskytuje dlouhodobou archivaci přijatých dat. Dispečer často má schopnost nejen pasivně pozorovat objekt, ale také jej řídit a reagovat na různé situace.

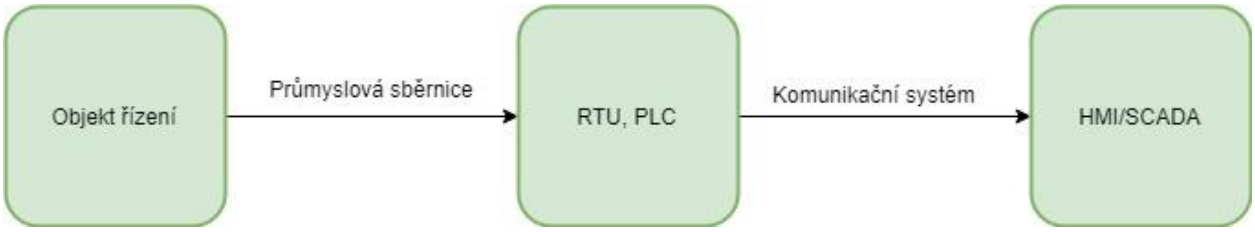
Díky tomuto systému lze online vidět vše, co se ve sledovaném zařízení děje v reálném čase, jaké hodnoty jsou aktuálně nastaveny a jaké jsou aktuální. Rovněž je možné dálkové řízení tohoto zařízení nebo objektu. Kromě možnosti dálkového řízení a nastavování vstupních parametrů lze dále regulovat kupříkladu otáčky motoru nebo rychlost výroby na výrobní lince.

Úkoly systémů SCADA jsou:

- výměna dat s PLC nebo RTU v reálném čase;
- zpracování informací v reálném čase;
- zobrazování informace na obrazovce v srozumitelné podobě;
- udržování databáze s technologickou informací v reálném čase;
- alarmy;
- příprava a generování zpráv o běhu technologického procesu;
- poskytování komunikace s externími aplikacemi (IMS, tabulky atd.).

4.2 Hlavní komponenty SCADA

SCADA systém obvykle obsahuje následující prvky:



Obrázek č. 19: Obecné schéma SCADA

HMI je rozhraní mezi strojem a člověkem, jak vypovídá zkratka HMI = Human Machine Interface, které poskytuje interakce operátora s jím řízeným strojem. HMI je například vizualizační software, který zobrazuje informace o procesu, umožňuje proces ovládat buď přímo z aplikace přímo v PC či terminálu na stanovišti, nebo také vzdáleně z okna webového prohlížeče. V HMI také lze nastavovat určité vstupní a výstupní veličiny. Může zobrazovat grafické průběhy daného procesu nebo dat z databáze, kam se data archivují.

RTU je vzdálený terminál, který zpracovává úkol v reálném čase. Má široké spektrum aplikací od jednoduchých senzorů až po speciální víceprocesorové počítačové systémy odolné proti chybám. Jeho realizace je určena pro konkrétní aplikací. Zařízení je využíváno pro nízko úrovňové zpracování informací, což snižuje požadavky na kapacitu kanálů pro komunikaci s centrálním řídicím počítačem.

Komunikační systém (komunikační kanály) je potřebný pro přenos dat ze vzdálených bodů do centrálního rozhraní operátora a přenos řídících signálů.

4.3 Možnosti sběru dat

Sběr dat je jedním z hlavních úkolu systému SCADA. Lze jej provádět manuálně, kdy se jedná o pochůzkový sběr dat, kdy člověk v pravidelném čase sleduje stav měřiče. Tato metoda se dnes již v podstatě nevyužívá z důvodu zavedení automatizovaného sběru dat.

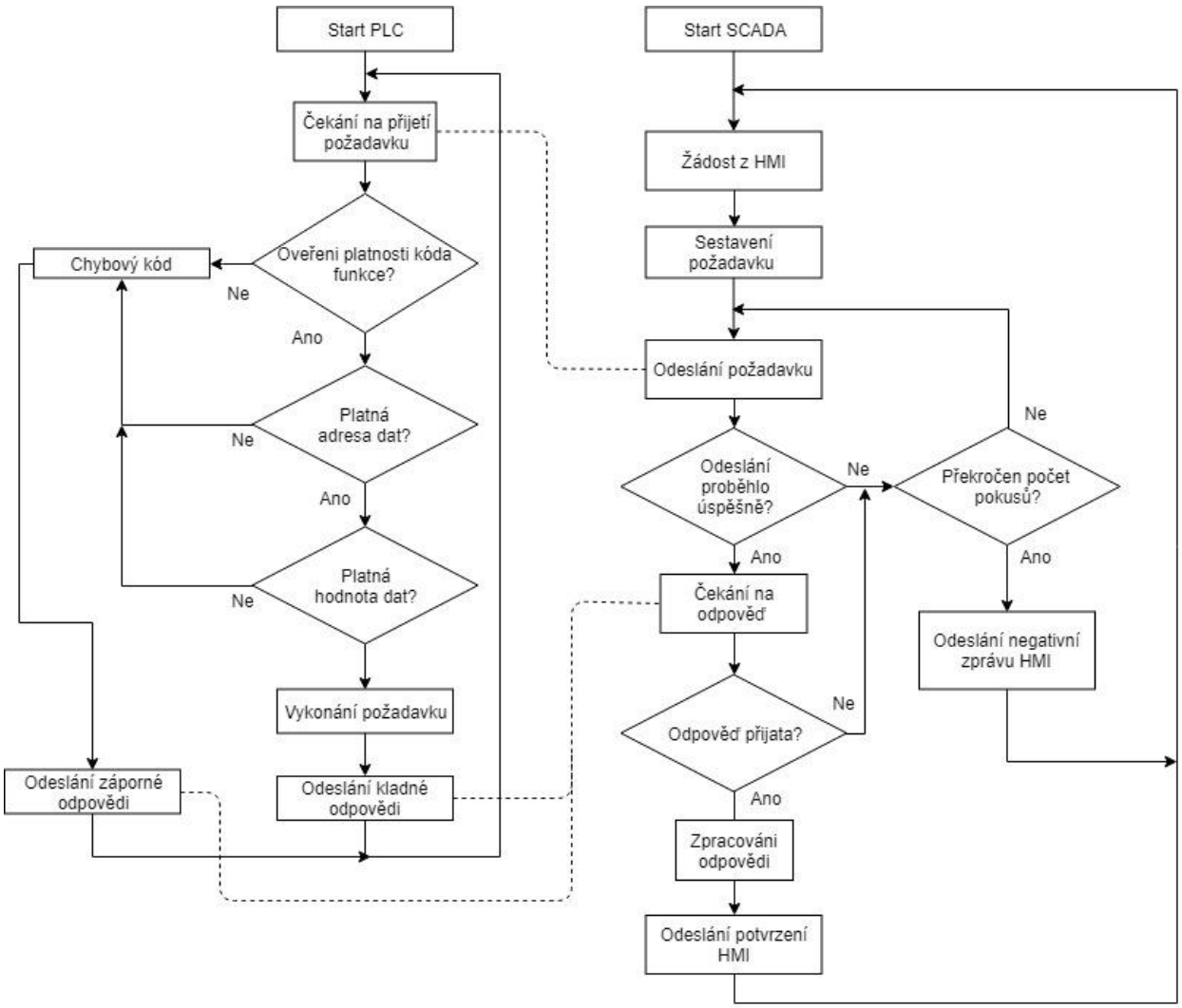
Při automatickém sběru dat jsou data dostupná online a umožňují řízení i sběr dat v reálném čase. PLC komunikují s počítači pomocí komunikačního protokolů. Mezi nejčastější patří ModBus, Profibus a CanBus. Systém SCADA lze napojit přímo k PLC za použití driveru pro daný protokol, anebo využít OPC serveru, který data z PLC standardizuje a poté s nimi může pracovat jakýkoliv software SCADA.

4.4 Algoritmus sběru dat z PLC do SCADA

Algoritmus lze rozdělit na dvě části: část probíhající v PLC a část probíhající ve SCADA. Obe části začínají spuštěním systému. Nejprve je třeba se věnovat popisu obecného postupu zpracování MODBUS požadavku na straně serveru.

Po spuštěni přejde PLC do pohotovostního stavu a bude čekat požadavek. Jakmile server dostane požadavek, začne ho zpracovávat, tj. ověří platnost kódu funkce, platnost adresy dat, platnost hodnoty dat. Jestliže kontrola proběhla úspěšně, bude vykonán požadavek, nejčastěji čtení nebo zápis nových hodnot. Po zpracovaní požadavku sestaví se odpověď a bude odeslaná klientovi. V závislosti na výsledku zpracování požadavku bude vytvořena jedna ze dvou možných odpovědí: pozitivní odpověď nebo negativní odpověď (viz kapitola 2.3.4). Pozitivní odpověď, když všechny přijata data jsou platná a vykonání požadavku prošlo úspěšné. Negativní odpověď v případě neplatných dat nebo neúspěšného vykonaní požadavku.

Algoritmus na straně SCADA systému je mnohem složitější. Po spuštění SCADA bude čekat na žádost z HMI. Po přijetí žádosti bude sestaven a odeslán požadavek (PDU + ADU). V případě úspěšného odesílání bude SCADA čekat na odpověď. Dojde-li k neúspěchu, bude požadavek odeslán znovu. Když po uplynutí určité doby SCADA odpověď nedostane, bude požadavek odeslán znovu. Počet možných opakování odeslání je omezen. Je-li možný počet překročen, odešle SCADA negativní zprávu HMI. V případě, že SCADA požadavek přijala, bude zpracován. Nejčastěji se jedná o aktualizaci tagu, či zápis do data-logu. Jestli zpracovávaní proběhlo úspěšně, bude odesláno potvrzení aplikaci.



Obrázek č. 20: Schéma průběhu algoritmu sběru dat

4.5 MySCADA

MySCADA je profesionální SCADA/HMI systém sloužící k monitorování a ovládání široké škály průmyslových technologií a automatizaci budov. Výrobcem systému mySCADA je česká firma zabývající se vizualizací průmyslových procesů. Jejich SCADA/HMI řešení je univerzální pro všechna průmyslová odvětví. MySCADA je naprogramovaná v C++ a vysoce optimalizované pro maximální výkon. MySCADA lze spustit na všech hlavních operačních systémech, Windows, Linux nebo Mac OS. V MySCADA data jsou získávána z řídicích systémů pomocí komunikačních driverů buď přímo, nebo zprostředkovaně, kdy je systém SCADA napojený na OPC server, který komunikuje s konkrétním hardwarem. Data jsou zpracovávána, ukládána do databází

a zobrazována koncovým uživatelům v grafické formě v podobě schémat, grafů, tabulek či různých ukazatelů. [9]

5 Praktická část

Pro praktickou ukázku jsem popsal zprovoznění komunikace mezi PLC Tekomat Foxtrot CP-1008 a SCADA systémem. Jako SCADA systém jsem vybral software od firmy mySCADA. Používal jsem program myDESIGNER pro vytvoření projektu, dále pak myPRO, který slouží jako runtime. Komunikace bude probíhat pomocí sběrnice Modbus TCP.

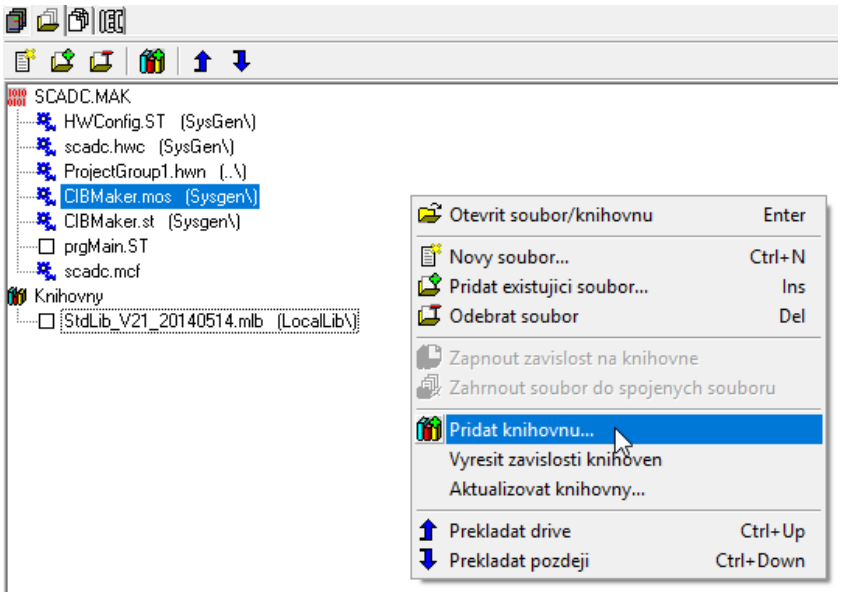
5.1 Nastavení PLC

Jak bylo uvedeno v kapitole 3.5, pro nastavení PLC Tecomat v roli klienta (slave) je nutné použít funkční blok fbModbusTCPslave, který obsahuje Knihovna ModbusRTULib.

5.1.1 Přidání knihovny do projektu

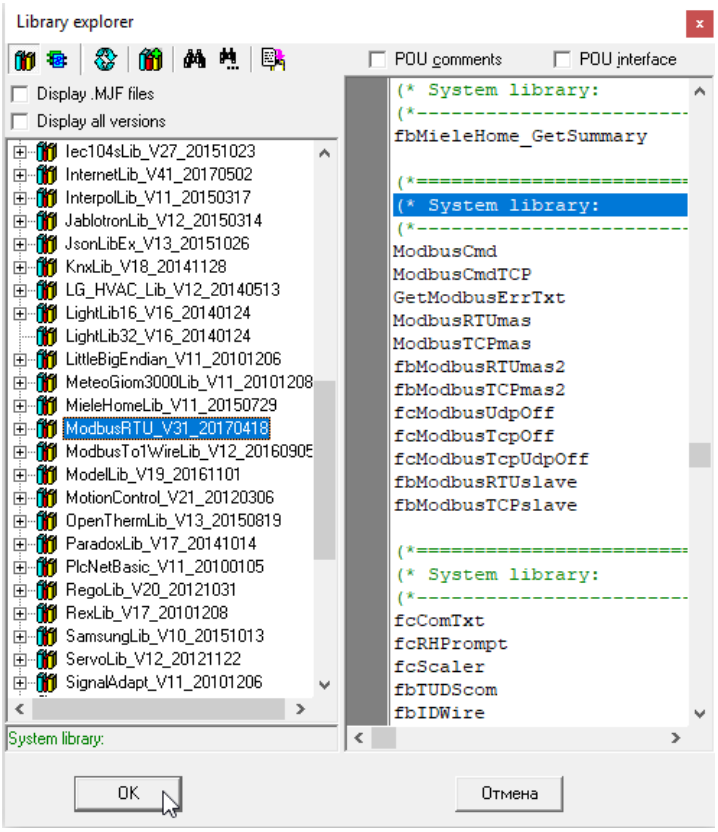
Knihovnu ModbusRTUlib přidáme do projektu následovně:

Stiskem pravého tlačítka v okně „Soubory projektu“ vyvoláme lokální menu a zvolíme možnost „Přidat knihovnu“:



Obrázek č. 21: Volání průzkumníka knihoven

Po stisknutí tlačítka „Přidat knihovnu“ se otevře Průzkumník knihoven. V okně „Průzkumník knihoven“ vybereme knihovnu ModbusRTUlib a stiskneme OK:

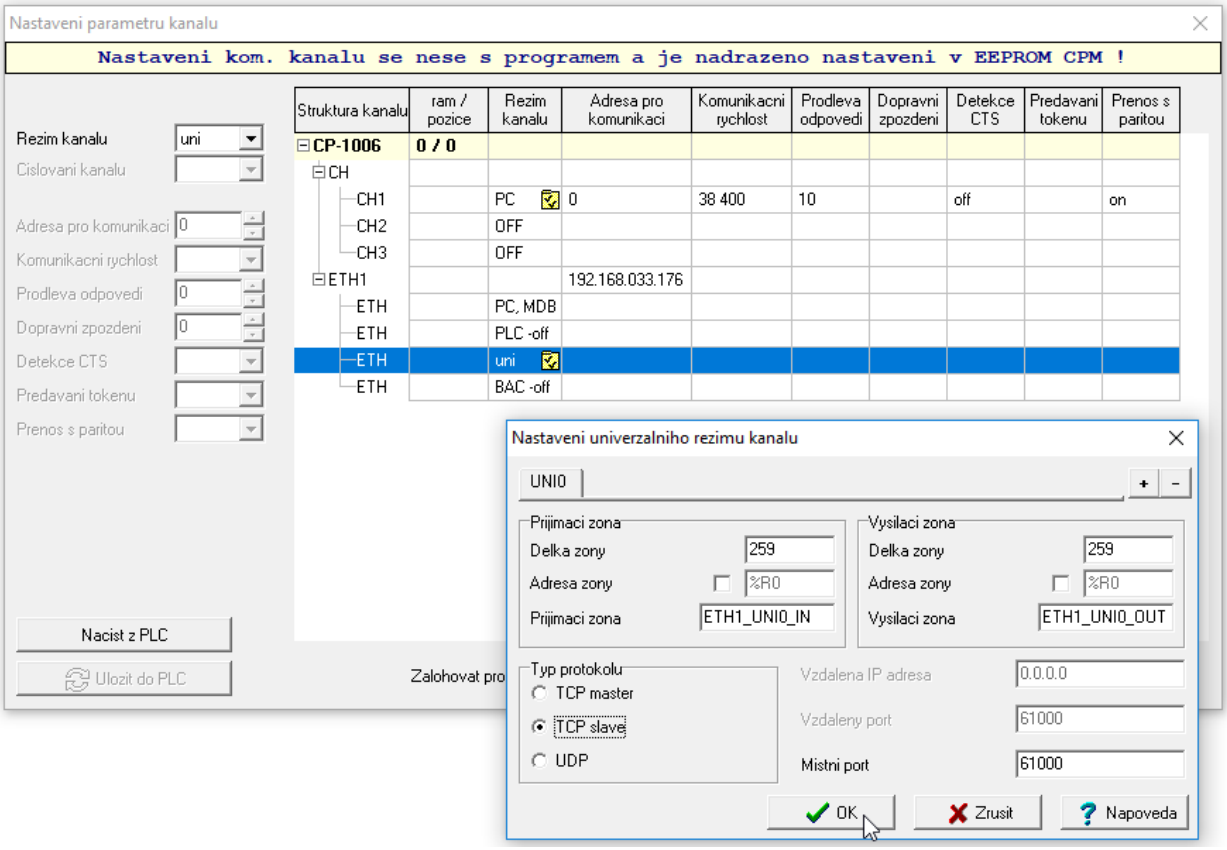


Obrázek č. 22: Průzkumník knihoven

Přidaná knihovna ModbusRTUlib bude vidět v seznamu souborů projektu.

5.1.2 Nastavení kanálu

Pro správnou činnost protokolu Modbus TCP musí být nastaven komunikační kanál. Zvolené spojení v režimu UNI musí být nastaveno na typ protokolu TCP slave. Minimální délka přijímací a vysílací zóny je 259 bytů. Místní port může být nastaven na libovolnou hodnotu, protože bude doplněn dle vstupu portu funkčního bloku.



Obrázek č. 23: Nastavení komunikačního kanálu

5.1.3 Realizace připojení v jazyce ST

Po přidáni potřebné knihovny a nastavené kanálu lze pokračovat v nastaveni Foxtrot CP-1008 v režimu slave. V této kapitole uvádím ukázku kódu v jazyce ST. Začátek programu:

```
1: PROGRAM prgMain
2:    VAR_INPUT
3:    END_VAR
4:    VAR_OUTPUT
5:    END_VAR
```

Deklarace funkčního bloku fbModbusTCPslave, který jsem použil v programu:

```
6:    VAR
7:    // deklarace funkcniho bloku fbModbusTCPslave
8:    propojeni : fbModbusTCPslave;
```

Deklarace vstupu funkčního bloku:

```
9:    UnitID : USINT :=1; // adresa PLC
10:   chanCode : UINT := ETH1_UNI0; // kód kanálu
11:   port : UINT :=502; // číslo místního portu
12:   inputsCnt : UINT:=10; // počet diskrétních vstupů
13:   coilsCnt : UINT:=10; // počet diskrétních výstupů
14:   inputRegCnt : UINT:=1; // počet vstupních registrů
15:   holdingRegCnt : UINT:=1; // počet registrů
```

Deklarace výstupu funkčního bloku:

```
16:   read : BOOL; // byla čtena data
17:   write : BOOL; // byla zapsána data
18:   established : BOOL; // TCP spojení navayano
19:   lastCmd : USINT; // poslední přijatý příkaz
20:   err : BOOL; // chyba
21:   errCode : USINT; // chybovy kod
22:   msgCnt : UDINT; // počet zpracovaných zpráv
23:   errCnt : UDINT; // počet chyb
24:   excCnt : UDINT; // počet výjimek
```

Deklarace poli diskrétní vstupu, výstupu, diskrétních výstupů a vstupních registru:

```
25:    // pole diskrétních výstupů
26:    PoleCoils : array [0..10] of Bool;
27:    // pole vstupních registrů
28:    PoleInputReg : array [0..10] of UINT;
29:    // pole registrů
30:    PoleHoldingReg : array [0..10] of UINT;
31:    // pole vstupu
32:    PoleInputs : array [0..10] of Bool;
33:
34:    END_VAR
35:    VAR_TEMP
36:    END_VAR
```

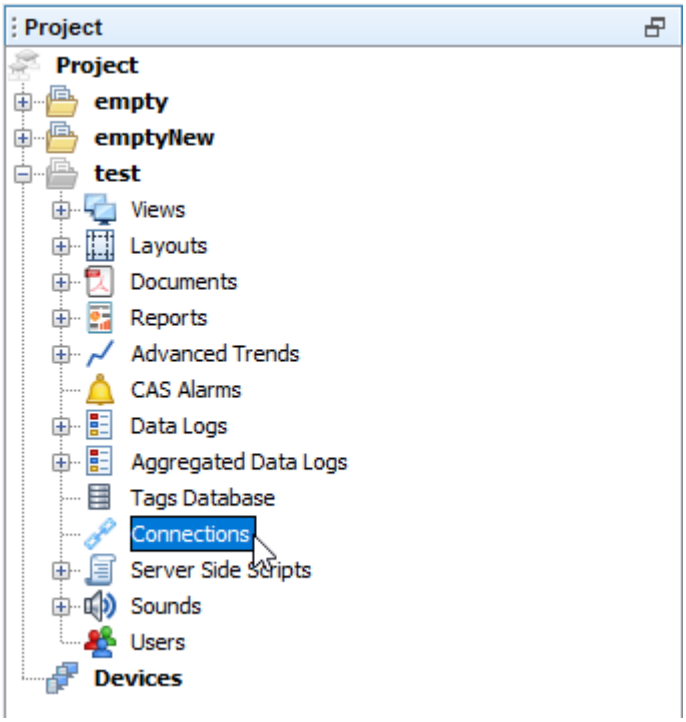
Volání funkčního bloku:

```
37: propojeni( UnitID := UnitID, chanCode := chanCode,
38:            port := port,inputsCnt := InputsCnt,
39:            coilsCnt := coilsCnt, inputRegCnt := inputRegCnt,
40:            holdingRegCnt := holdingRegCnt,
41:            read => read, write => write,
42:            established=>established, lastCmd => lastCmd,
43:            err => err,errCode => errCode, msgCnt => msgCnt,
44:            errCnt => errCnt, excCnt => excCnt,
45:            coils := PoleCoils[0], inputRegs:=PoleInputReg[0],
46:            holdingRegs:=PoleHoldingReg[0], inputs := b1 );
47:
48: END_PROGRAM
```

Po zavolání funkčního bloku je nastaveni komunikace dokončeno a může PLC pomocí sběrnice Modbus přijímat a odesílat data. V poli PoleCoils, PoleInputReg, PoleHoldingReg lze zapisovat data, které budou posílaná pomoci Modbus TCP.

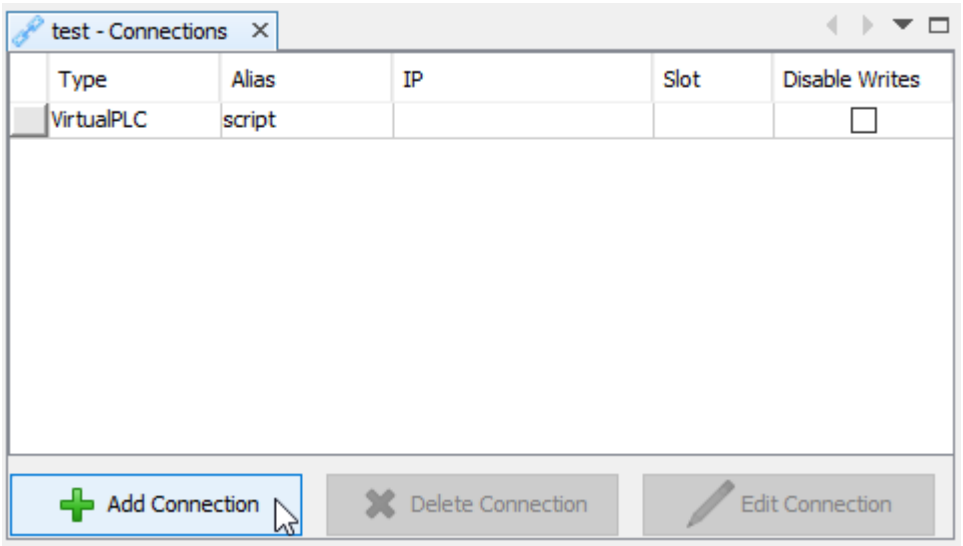
5.2 Nastavení komunikace Modbus ze strany SCADA systému

Pro zobrazení existujících připojení nebo pro tvorbu nového připojení vybereme složku „Připojení“ v okně „Projekt“:



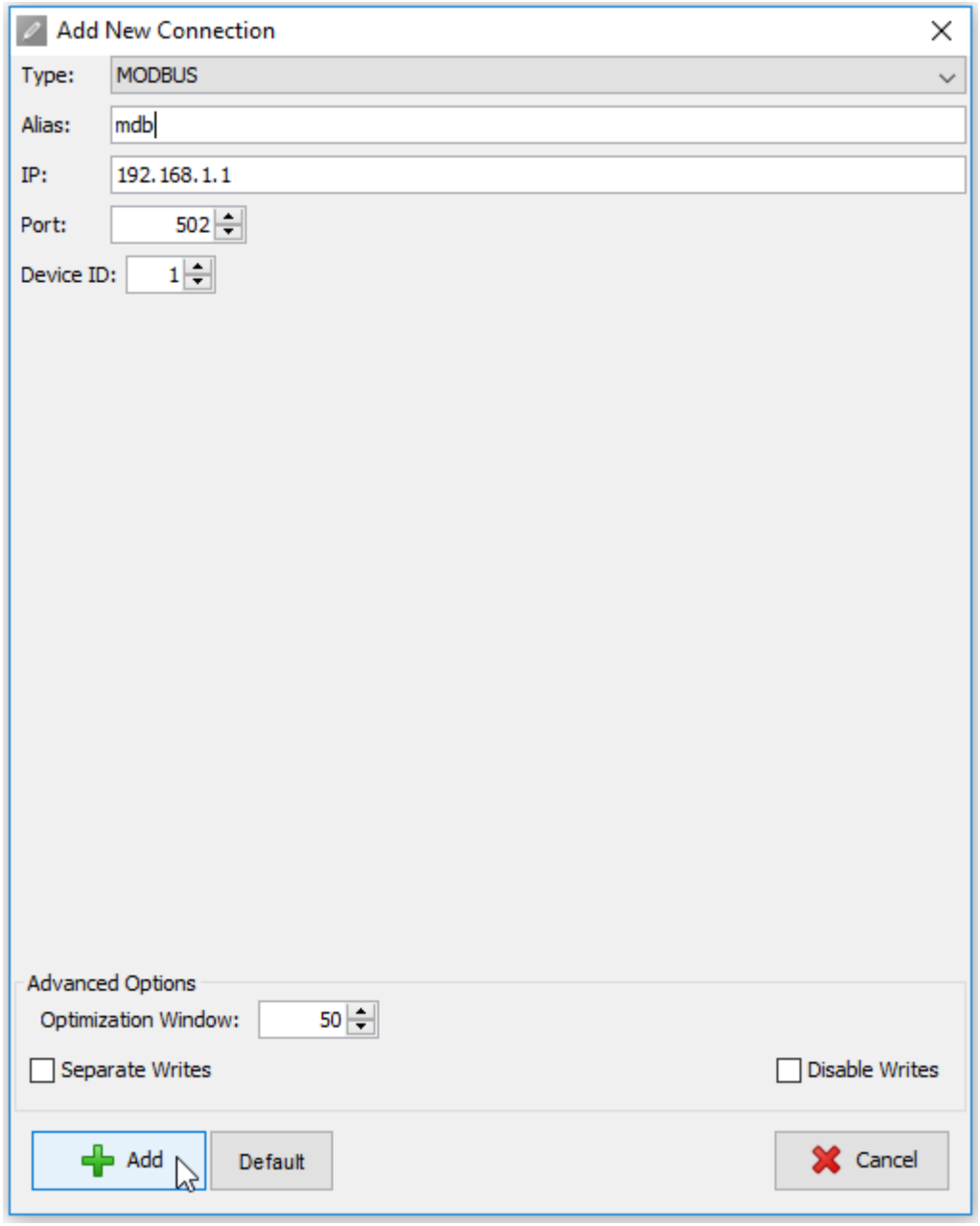
Obrázek č. 24: Okno s možnostmi projektu

V tomto okně můžete vidět všechna definovaná připojení, jak je zobrazeno na obrázku č. 25:



Obrázek č. 25: Přehled definovaných připojení

Po stisknutí tlačítka „Add Connection“ se otevře okno, ve kterém můžeme zvolit typ komunikace: komunikace s PLC, databáze nebo s IoT. Po výběru PLC se otevře okno s nastavením nové komunikace:



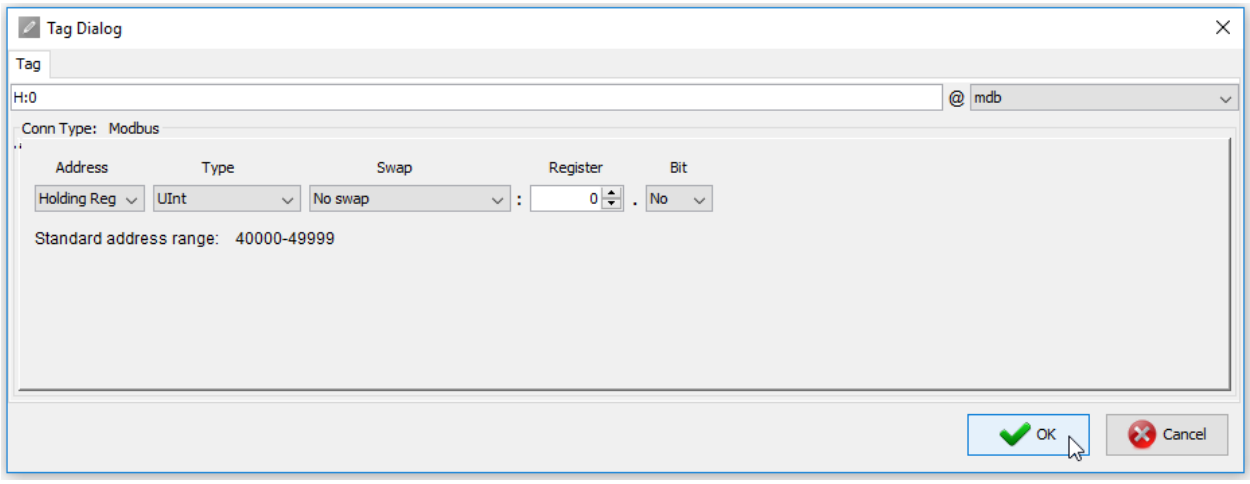
Obrázek č. 26: Okno s předvolbami nového připojení

V tomto dialogovém okně lze zadat specifické parametry připojení. V poli „Type“ jsem vybral typ komunikace, tj. Modbus. V poli „Alias“ byl zadán název tohoto zapojeni, dále uvedena IP adresa PLC, se kterým bude SCADA komunikovat, Port a Device ID.

Po stisknutí tlačítka „Add“ se vytvoří nové zapojení Modbus, které lze použít pro komunikaci s PLC Tecomat Foxtrot CP-1008.

5.2.1 Databáze tagů

Databáze tagů je užitečnou funkcí pro správu všech tagů. Nejdůležitější funkcí databáze tagů je možnost spravování použitých proměnných na jednom místě. V databázi tagů lze jednoduše měnit tagy a změna se projeví napříč projektem. Všude, kde se používá tag, je stará hodnota nahrazena aktualizovanou hodnotou. V okně „Projekt“ lze otevřít „Tag database“ a vytvořit nový tag:

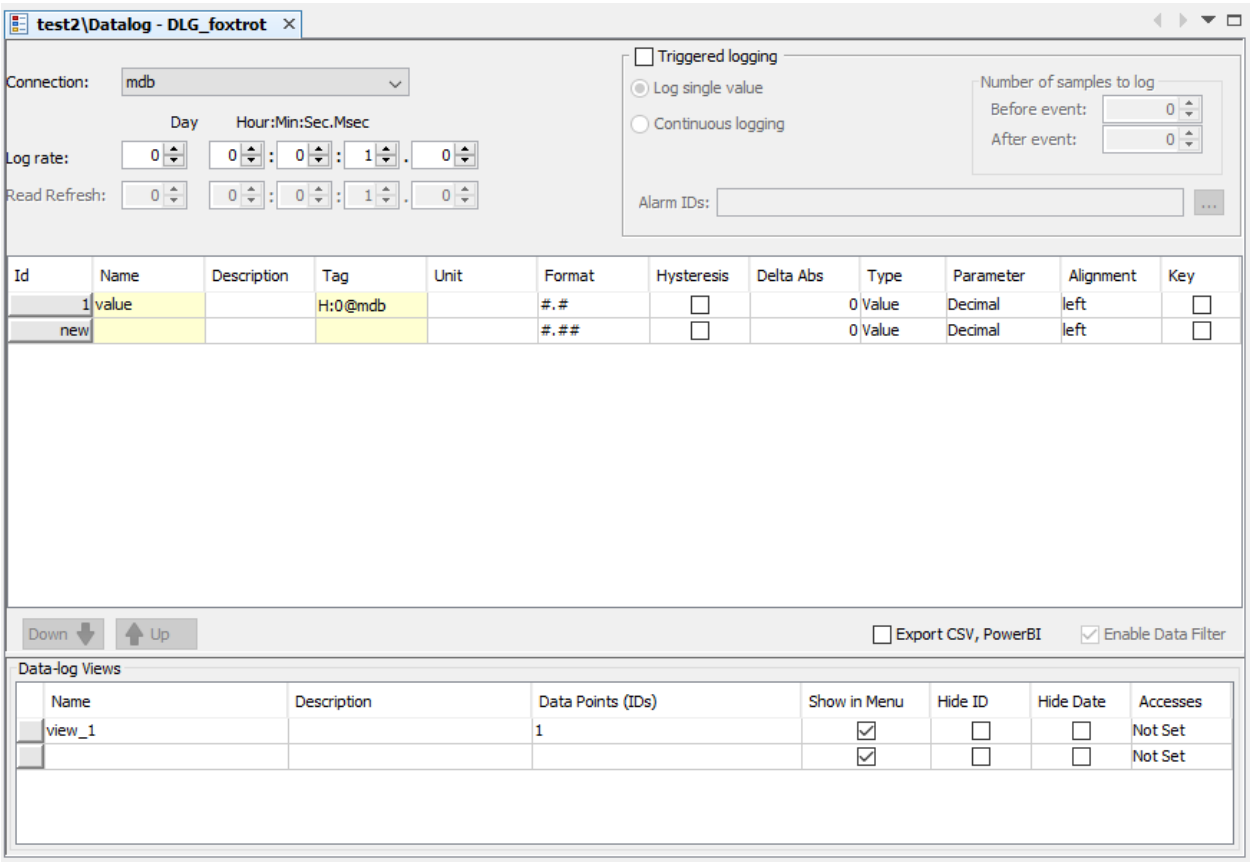


Obrázek č. 27: Vytvoření nového tagu

V okně Tag Dialog lze nastavit zapojení, adresu nového tagu, jeho typ, registr a další parametry.

5.2.2 Data-log

Data-log se používá pro ukládaní dat. Existují dva typy Data-logu. Prvním je “Continuous Data-logs“ zapisující data pravidelně bez přerušení. Tento typ je užitečný pro trvalé procesy. Druhým typem je „Triggered Data-logs“ závisející na určité události či stavu. Podmínka je určena ID alarmu (nebo ID více alarmů). Tento typ je užitečný pro náhodné procesy, u kterých můžete určit spouštěcí podmínku. Pro vytvoření nového data logu v okně „Projekt“ stiskneme pravým tlačítkem „Data-Logs“ a zvolíme možnost „Add new“, což vytvoří nový data-log.



Obrázek č. 28: Okno data logu

Okno data-logu lze rozdělit na tři části. V horní třetině lze nastavit typ připojení, rychlost zápisu a případně událost, která tento data-log spustí. Uprostřed okna jsou tagy, které tento data-log zapisuje. Ve spodní části lze nastavit zobrazení data-logu. Pro ukázku jsem nastavil zápis tagu H:0@mbd každou sekundu. Data budou zobrazeny v Data-log Views.

Tímto jsem dokončil nastaveni mySCADA pro přijetí dat z PLC pomocí sběrnice Modbus.

6 Závěr

Cílem práce bylo nastavit přenos dat z řídicího systému PLC do nadřazeného SCADA systému, a to za použití průmyslové sběrnice Modbus TCP.

První dvě kapitoly práce jsou věnované programovatelnému automatu PLC. Uvedl jsem, co PLC je, a to včetně principu a základního přehledu normy IEC 61131-3, popisující tento řídicí systém. Pro další práci byl vybrán a popsán řídicí automat od firmy Teco a bylo popsáno vývojové prostředí Mosaic.

Třetí kapitola byla věnována sběrnici Modbus a jejím variantám. Dále byl popsán systém SCADA, struktura a kompletně popsán algoritmus sběru dat z PLC do SCADA. Pro praktickou část byl vybrán systém mySCADA od společnosti Technologies s.r.o.

V praktické části práce jsem zprovoznil komunikaci mezi PLC Foxtrot CP-1008 a MySCADA. Komunikace byla reálně navázána a proběhla úspěšně.

Seznam obrázků

Seznam tabulek

Seznam použité literatury

[1] MARTINÁSKOVÁ, Marie a Ladislav ŠMEJKAL. *Řízení programovatelnými automat*y. Praha: Vydavatelství ČVUT, 2003. ISBN 80-01-02804-6.

[2] *Teco DVD INFO 10/2008.* Kolín: Teco a. s., 2008.

[3] MEYERS, S. Effective STL: *50 Specific Ways to Improve Your Use of the Standard Template Library*. Addison-Wesley, 2001.

[4] *Programovatelné automaty Tecomat Foxtrot CP-1008, CP-1018, CP-1028, CP-1038, TXV 004 38.* 2017, 7. vydání. [online]. [cit. 2018-05-17]. Dostupné z: https://www.tecomat.cz/ke-stazeni/prirucky/

[5] ZEZULKA, František. *Automatizační prostředky*. Brno: PC-DIR Real, 1999. Učební texty vysokých škol. ISBN 80-214-1482-0.

[6] KOCOUREK, Petr a Jiří NOVÁK. *Přenos informace*. Praha: Vydavatelství ČVUT, 2004. ISBN 80-010-2892-5.

[7] *Programování dle normy IEC 61 131* [online]. [cit. 2018-06-11]. Dostupné z: www.spszl.cz/soubory/plc/programovani_dle_normy_iec61131.pdf

[8] Firemní dokumentace PLC Tecomat [online]. [cit. 2018-06-05]. Dostupné z: https://www.tecomat.cz/

[9] Firemní dokumentace MySCADA [online]. [cit. 2018-06-05]. Dostupné z https://www.myscada.org/cs/

[10] Dokumentace ke komunikačnímu standartu MODBUS [online]. Dostupné z: http://www.modbus.org/

[11] ŠMEJKAL, Ladislav a Marie MARTINÁSKOVÁ. *PLC a automatizace*. Praha: BEN – technická literatura, 1999. ISBN 80-86056-58-9.

[12] *Knihovna ModbusRTUlib* [online]. [cit. 2018-06-11]. Dostupné z: https://www.tecomat.cz/ke-stazeni/prirucky/