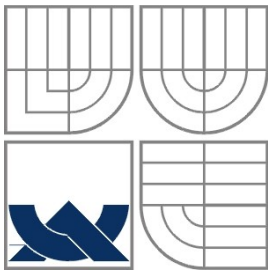
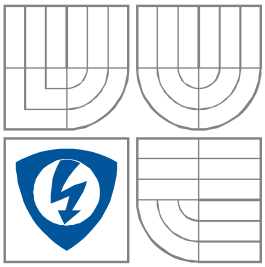


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF CONTROL AND INSTRUMENTATION

KOMUNIKAČNÍ OVLADAČ A SIMULÁTOR VÁHY METTLER TOLEDO VE STEP7

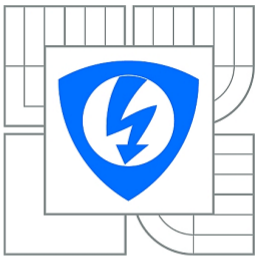
COMMUNICATION DRIVER AND SIMULATER FOR METTLER TOLEDO WEIGHT IN STEP7

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — VÍT PŘICHYSTAL
AUTHOR

VEDOUCÍ PRÁCE — ING. JAN PÁSEK
SUPERVISOR

BRNO 2012



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav automatizace a měřicí techniky

Bakalářská práce

bakalářský studijní obor
Automatizační a měřicí technika

Student:	Vít Přichystal	***ID:***	125610
Ročník:	3	***Akademický rok:***	2011/2012

NÁZEV TÉMATU:

Komunikační ovladač a simulátor váhy Mettler Toledo ve Step7

POKYNY PRO VYPRACOVÁNÍ:

Vytvořit funkční blok který bude zajišťovat obsluhu komunikace s vážním procesorem připojeným na síti Ethernet pomocí protokolu TCP/IP. Vytvoření druhého funkčního bloku jako simulátor reálné váhy, která není pro BP v danou chvíli k dispozici.

DOPORUČENÁ LITERATURA:

1 STEP7 - Manuály firmy Siemens
2. METTLER TOLEDO Balance Support Site, dostupné na: www.mettler-toledo-support.com/...
3. IND560x Intrinsically Safe Weighing terminal - METTLER TOLEDO, dostupné na: se.mt.com/...

Termín zadání:	6.2.2012	***Termín odevzdání:***	28.5.2012

Vedoucí práce: Ing. Jan Pásek, CSc.
Konzultanti bakalářské práce: Ing. Martin Mierva

doc. Ing. Václav Jirsík, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

Autor bakalářské práce nesmí při vytváření bakalářské práce porušit autorská práva třetích osob, zejména nesmí zasahovat nedovoleným způsobem do cizích autorských práv osobnostních a musí si být plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č.40/2009 Sb.

Abstrakt

V úvodu je přehled programovacích jazyků pro PLC. Práce se zabývá vytvořením driveru pro komunikaci s váhovým terminálem firmy Mettler Toledo, konkrétně jde o typ IND560. Dále se práce zabývá vytvořením simulátoru tohoto váhového terminálu, který není k práci k dispozici. Vše je tvořeno pomocí jazyka SCL a softwaru firmy Siemens. Také byla vytvořena vizualizace projektu.

Klíčová slova

jazyky pro PLC, komunikace, Mettler Toledo IND560, ovladač, PLC, SCL, Siemens, simulátor, Step7, TIA Portal V11, váhový terminál, WinCC

Abstract

In the beginning of the thesis I mention a list of PLC programming languages. The thesis itself deals with a problem of implementing a driver for Mettler Toledo weight terminal,concretely the type IND560. Moreover, the simulator of this terminal is created (the terminal itself was not available). For the implementation, the language scl was used, together with the Siemens company software. At the end of the thesis, I created a visualization of the project.

Keywords

communication, driver, languages for PLC, Mettler Toledo IND560, PLC, SCL, Siemens, simulator, Step7, TIA Portal V11, weighing terminal, WinCC

Bibliografická citace:

PŘICHYSTAL, V. *Komunikační ovladač a simulátor váhy Mettler Toledo ve Step7*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2012. 46s. Vedoucí bakalářské práce byl Ing. Jan Pásek, CSc.

Prohlášení

„Prohlašuji, že svou bakalářskou práci na téma Komunikační ovladač a simulátor váhy Mettler Toledo ve Step7 jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.
Jako autor uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

V Brně dne: **28. května 2012** ……
podpis autora

Poděkování

Děkuji vedoucímu bakalářské práce Ing. Jan Páskovi, CSc. a konzultantovi Ing. Martinu Miervovi za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé bakalářské práce.

V Brně dne: **28. května 2012** ……

podpis autora

Obsah

Seznam obrázků

Seznam tabulek

1 ÚVOD

Zadání této práce bylo vytvořeno díky spolupráci firmy COMPAS automatizace, kde jsem byl také na praxi. Firma COMPAS automatizace se zabývá průmyslovou automatizací a výrobními informačními systémy MES. Používá převážně produkty firmy Siemens a vlastní MES systém COMES. V oblasti programového vybavení pro řídící systémy SIMTAIC S7 a PCS7, včetně vizualizace WinCC patří zaměstnanci firmy k největším týmům v Evropě. Zabývají se pružnou recepturovou automatizací šaržových výrob potravin a léčiv a dále automatizací sériových výrob ve strojírenství a automobilovém průmyslu. Během praxe jsem měl možnost seznámit se s programováním pomocí Step7 a s vizualizací v programu WinCC. Oba softwary jsou od firmy Siemens. Díky těmto zkušenostem mi byl přidělen úkol vytvořit simulátor a driver pro komunikaci s vážním terminálem.

1.1 Hlavní cíl

Hlavním cílem projektu je vytvoření komunikačního ovladače pro váhu Mettler Toledo a zároveň vytvořit simulátor této váhy, která není momentálně k dispozici. Vážní procesor měl být připojen na síti Ethernet pomocí protokolu TCP/IP. Řešení komunikačního driveru a simulátoru bylo vytvořeno v programu Step7 od firmy Siemens, konkrétně v TIA Portalu V11, (totally integrated automation). Tomu předcházela analýza komunikace váhy a driveru. Programy, resp. datové bloky, byly napsány převážně programovacím jazykem SCL, který je v tomto případě nejvýhodnější. Konkrétní důvody proč právě SCL, bude vysvětleno v další kapitole. Komunikace v mém projektu neprobíhá samozřejmě pomocí TCP/IP, ale pomocí řetězců v jednom datovém bloku. Důvod je zřejmý, váhu nemám k dispozici, pracuji jen s její simulací, a řetězec znaků lze snáze přenášet různými protokoly. Dále byla vytvořená jednoduchá vizualizace HMI panelu pro operátora v tomtéž programu, tedy TIA Portalu V11.

2 POROVNÁNÍ JAZYKŮ

Programy, jak už bylo zmíněno, byly psány v TIA Portalu. Zde můžeme volit mezi různými jazyky vhodnými pro PLC automaty. Lze tedy použít: STL (statement list), LAD (ladder), FBD (function block diagram), GRAFCET (graphe gonctionnel de connexion etapes transitions), SCL (structured control language). První tři jmenované jazyky jsou již dlouhá léta zaběhlými jazyky pro PLC. Mezi mladší jazyky lze zařadit SCL a GRAFCET. Tyto jazyky jdou s trendem programování PLC. SCL je strukturovaný text podobný známým moderním jazykům Pascal, C++, atd. GRAFCET je zase sekvenční jazyk, který je programován grafickou formou.

2.1 STL

Jazyk STL je instrukční kód. Je to assembler pro PLC. Teto kód je základní kód, do kterého se všechny ostatní jazyky PLC překládají. Jedná se o vyjádření logické sítě, ale jsou zde i další instrukce navíc. Ne vše, co je možné napsat v jazyce STL, lze převést do FBD nebo LAD, atd., ale vše co lze napsat v LAD, FBD, SCL nebo GRAFCET, lze převést do STL. Je blízký všem programátorům, kteří začínali strojovými kódy. Dnes se už převážně nepoužívá, protože psaní kódu je v tomto jazyce poněkud nepřehledné.

Krátký program v jazyce STL uvedený níže převádí proměnnou typu int na proměnnou typu real.

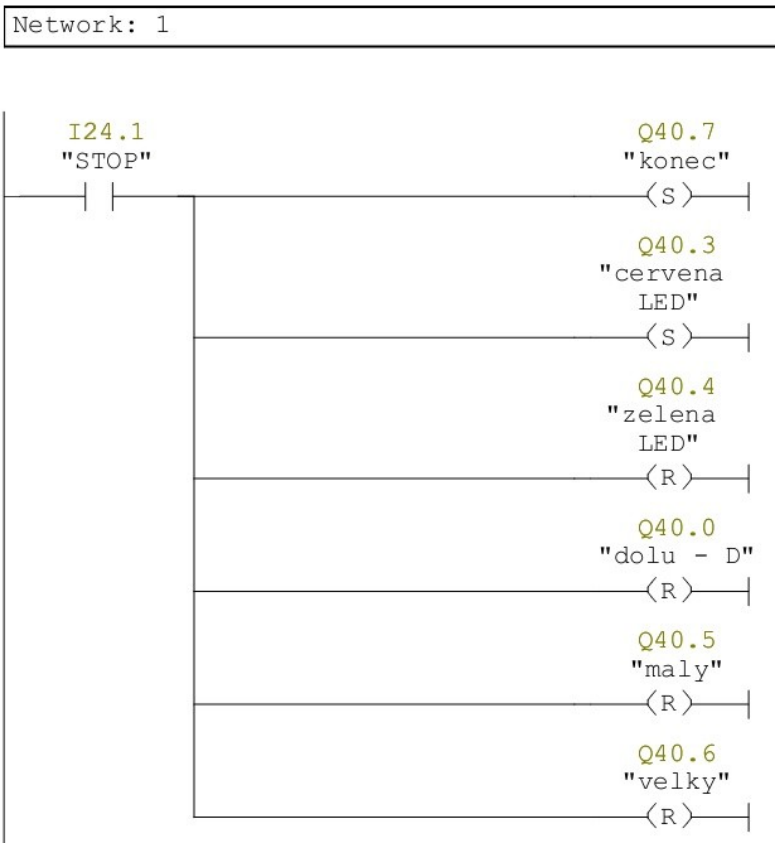
```
L INT
ITD
DTR
T REAL
```

Program na prvním řádku načte proměnnou INT. Na druhém převede tuto hodnotu na typ dint a na třetím na real. Na posledním řádku je převedená hodnota uložena do proměnné REAL. Z uvedeného kódu je patrné, že tento jazyk používá jednoduché instrukce, díky tomu je psaní programu poněkud zdlouhavé.

2.2 LAD

LAD, čili žebříčkový zápis programu vychází z logických sítí, které byly tvořeny fyzickými relé, tlačítky, časovými relé, apod. Proto v sedmdesátých letech pro tvoření PLC programů byla tato symbolika převzata. Jedná se tedy o jakési zobrazení, nahrazení elektrických prvků jejich symboly. LAD je nejbližší elektrikáři.

Pro představu uvedu ukázku v jazyce LAD:



Obr. 2.1: Program v jazyce LAD, Network 1

Zde spínací tlačítko po aktivaci přepne bity Q40.7 „konec“ a Q40.3 „cervena LED“ do stavu jedna. Bity Q40.4 „zelena LED, Q40.0 „dolu – D“, Q40.5 „maly“ a Q40.6 „velky“ budou nastaveny na nulu. Na obr. 2.1 můžeme vidět napodobení elektrického obvodu. Vlevo je rozvedeno „napájení“ a vpravo je „zem“. Mezi těmito různými potenciály je „tlačítko“ a „žárovky“.

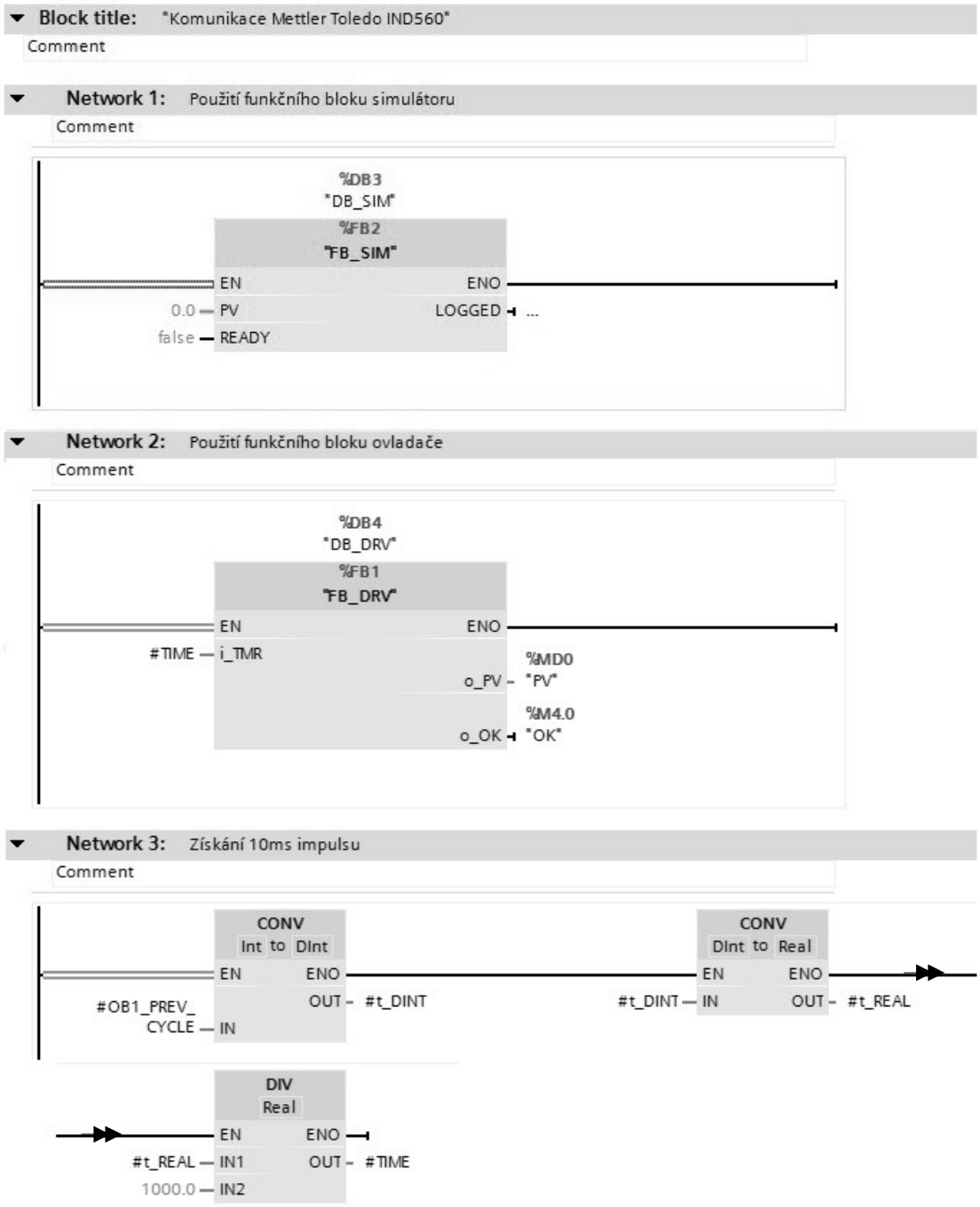
Tato část programu je převzata z protokolu s názvem Model vrtačky. Úloha byla měřena 6.4.2011. Spoluautor protokolu je Sodomka Petr.

2.3 FBD

FBD je tzv. jazyk funkčních bloků. Je to tedy jiný zápis logické sítě, tvořený pomocí funkčních bloků. Tento jazyk je blízký tvůrci logických funkcí. V programu vidíme vstupy a výstupy použitých bloků a také funkci těchto bloků. Jak je daná funkce vytvořena nás v tomto případě nezajímá, zajímá nás jen jak se tato funkce chová vůči vstupním a výstupním proměnným. Jednotlivé bloky se skládají za sebe a tvoří, či řekněme zobrazují, booleovskou logiku.

Ukázka programu v jazyce FBD je z mé bakalářské práce. Na obr. 2.2 je vidět použití dvou vlastních funkčních bloků. Funkční blok simulátoru váhy a blok ovladače (network 1 a 2). Network 3 ukazuje použití dvou bloků, které provádí konverzi

datových typů a třetí blok je dělení. Tímto postupem a díky proměnné OB1_PREV_CYCLE získáme 10ms časový impuls. Ten je pak vstupem do bloku ovladače.



Obr. 2.2: Program v jazyce FBD

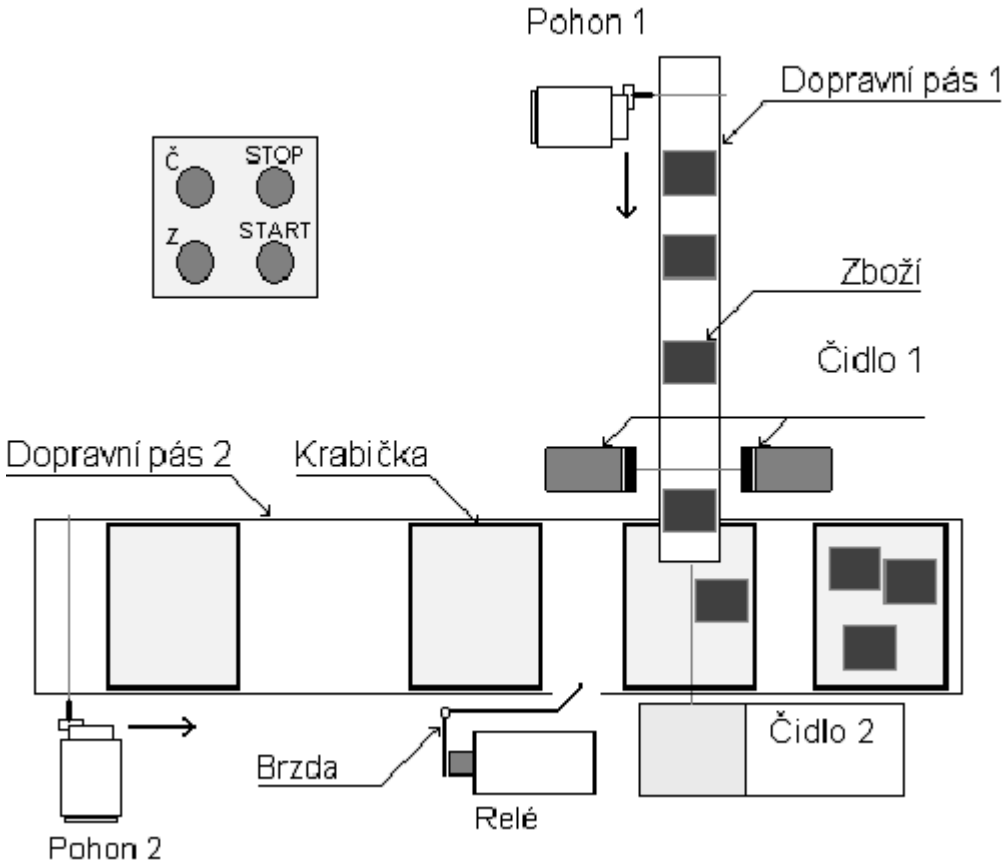
2.4 GRAFCET

GRAFCET (graphe gonctionnel de connexion etapes transitions) byl standardizován ve Francii, proto francouzská zkratka. Tato by se dala přeložit jako: Funkční graf přechodů. Jazyk vychází z Petriho sítí[2]. Je tedy založen na rozdělení úlohy na jednotlivé kroky. Je to jazyk, který by měl být svojí skladbou srozumitelný technologovi (procesnímu inženýrovi). To, že technolog rozumí kódu, který popisuje celý technologický proces, je velmi důležité, protože je hlavním architektem daného procesu. Procesní inženýr může pak takovýto program přímo tvořit.

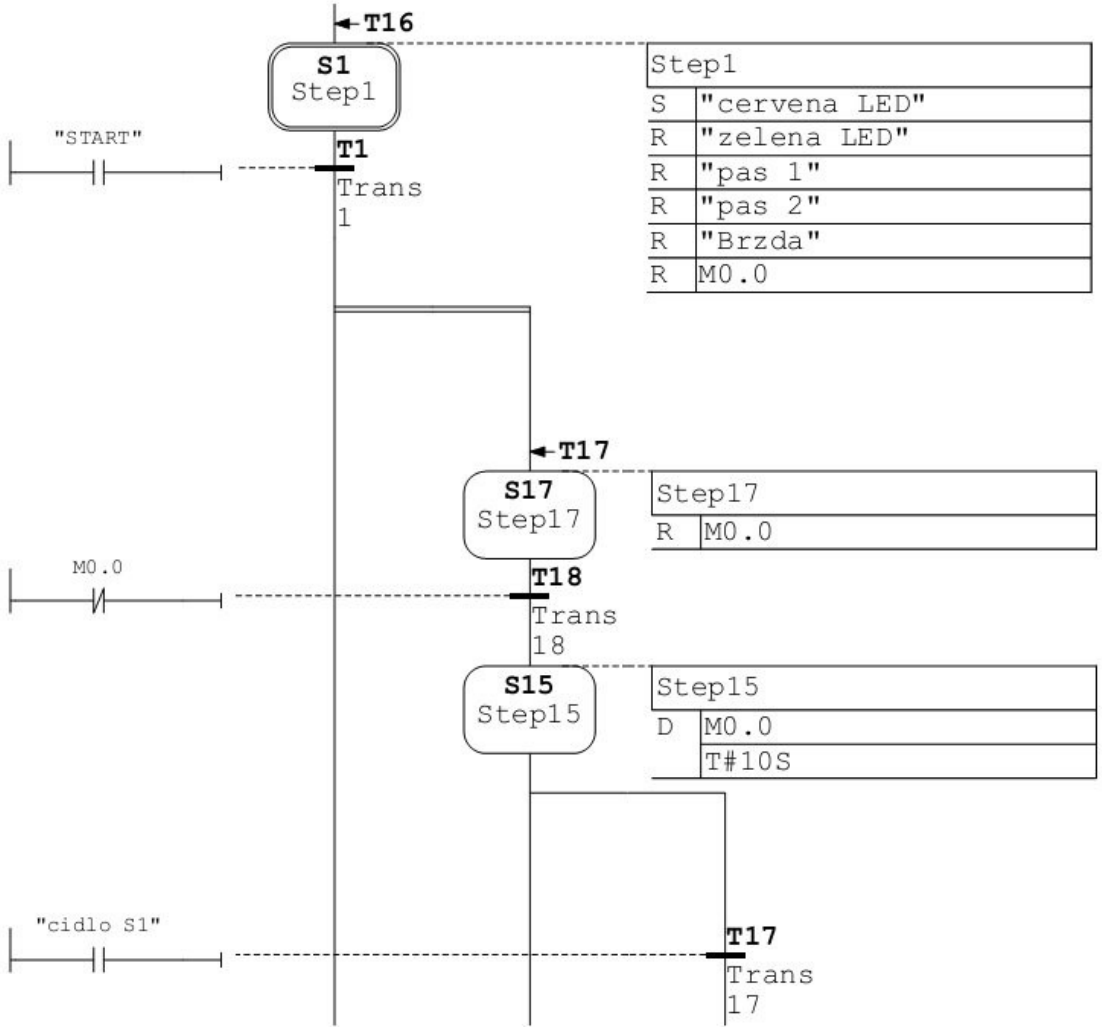
Na obr.2.4, 2.5 a 2.6 je uveden malý program, který je napsaný jazykem Graph. K programu se vztahuje obr. 2.3, který ukazuje rozložení jednotlivých prvků a osvětluje představu řešeného problému. Jedná se o dopravník, který počítá kusy zboží a rovná je do krabiček. Zboží přijíždí v nepravidelných intervalech po dopravním pásu 1 a je indikováno čidlem 1. Krabičky přijíždí po dopravním pásu 2 a následující krabice je zastavována brzdou, protože čidlo 2 neregistruje krabičku ve správné poloze. Po naplnění krabičky třemi kusy zboží krabička odjede a je brzdou uvolněna prázdná krabička. Cyklus se opakuje. Dopravní pás 1 musí být vypnut v případě, že po dobu 5 s nedorazí zboží. Systém je uveden do chodu tlačítkem START. Během chodu svítí zelená LED. Systém se zastaví tlačítkem STOP. Pokud bylo stlačeno tlačítko STOP, je rozsvícena červená LED.

Pohon 1 a 2 je v programu použit jako: „pas 1“ a „pas 2“. Č a Z na obr. 2.3 značí červenou a zelenou LED, ty jsou v programu nazvány: „cervena LED“ a „zelena LED“. Dále čidlo 1 a 2 je pojmenováno: „cidlo S1“ a „cidlo S2“.

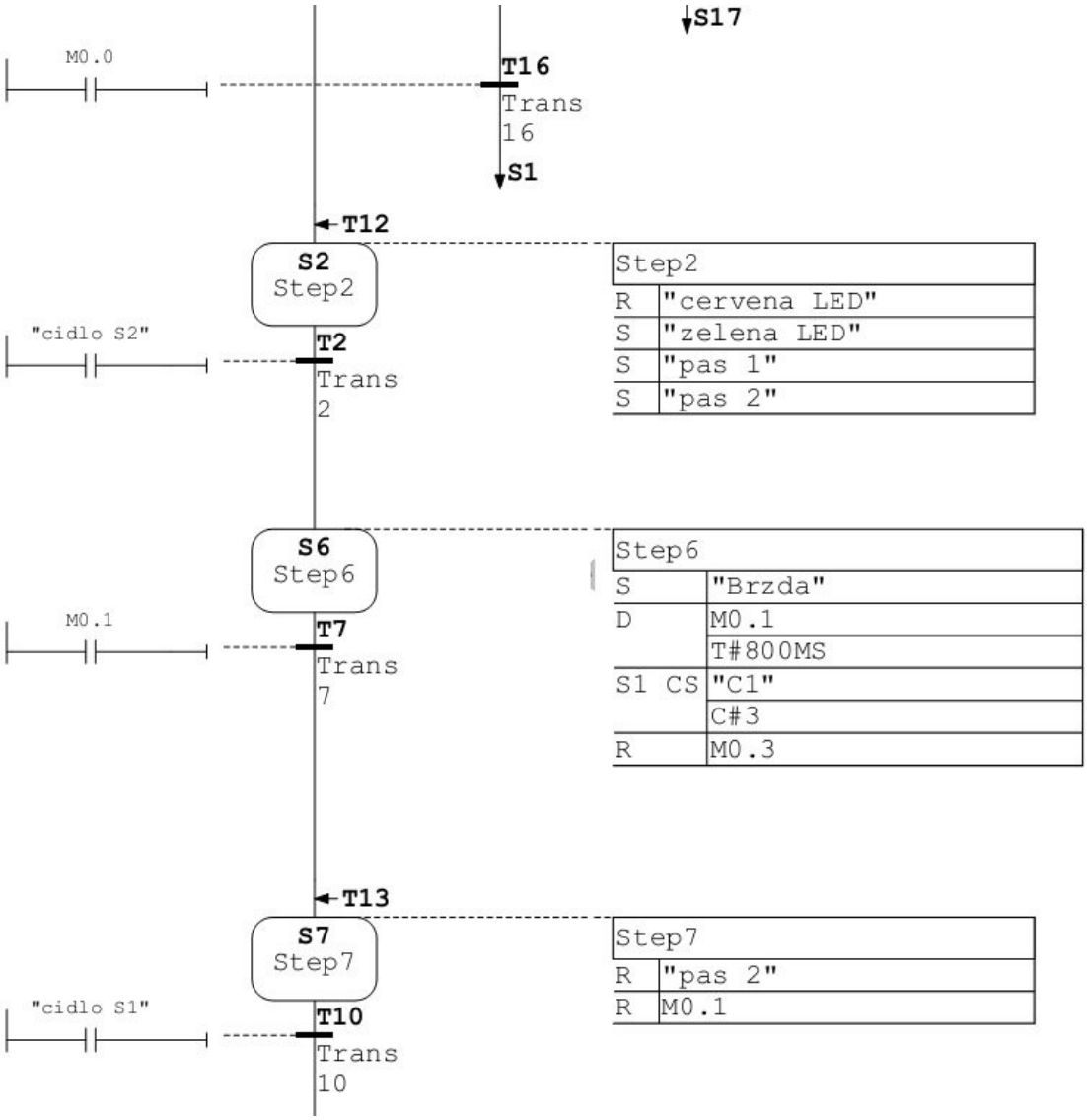
Program je převzat z protokolu s názvem Dopravník – počítání kusů do beden. Úloha byla měřena 20.4.2011. Spoluautor protokolu je Sodomka Petr.



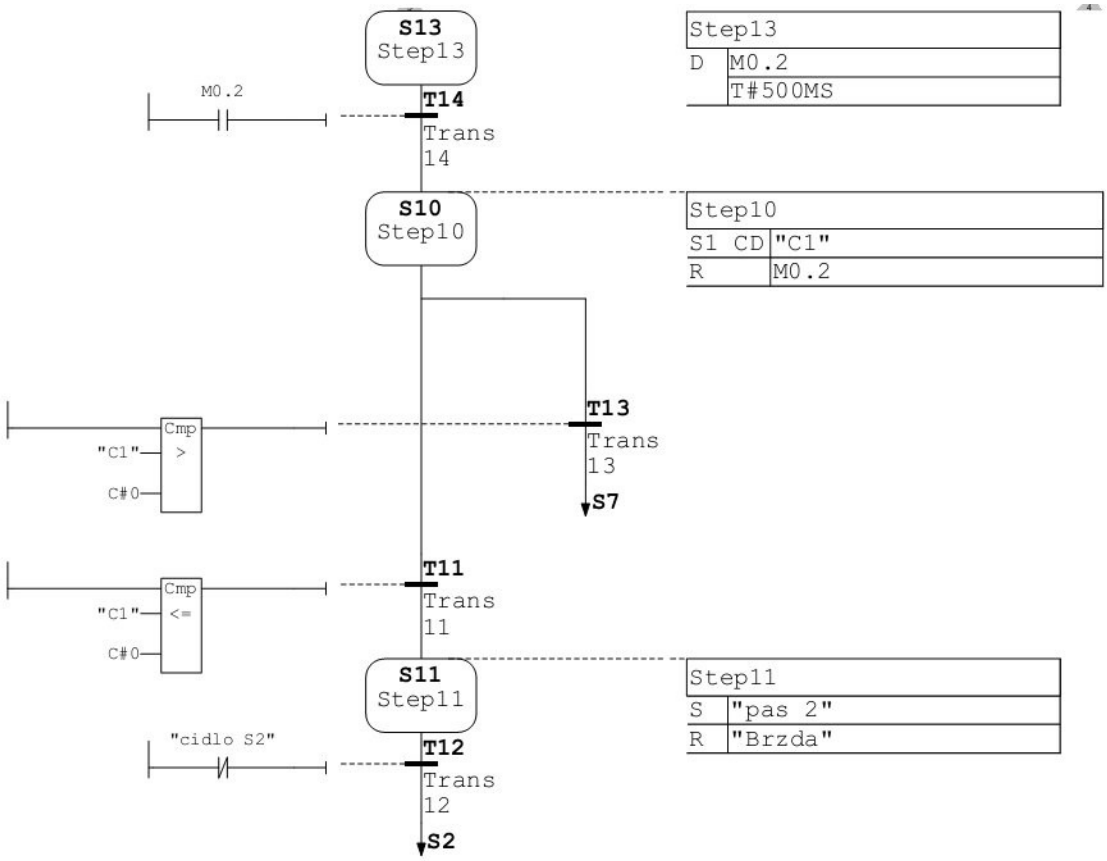
Obr. 2.3: Náčrt dopravníku s bednami



Obr. 2.4: Program v Graphu, část 1



Obr. 2.5: Program v Graphu, část 2



Obr. 2.6: Program v Graphu, část 3

2.5 SCL

SCL (structured control language) je strukturovaný moderní vyšší programovací jazyk, který je svou skladbou podobný ostatním moderním programovacím jazykům. Od původních jazyků PLC se SCL výrazně liší. Původní jazyky vycházejí z elektrických obvodů, složených z relé, tlačítek, atd. SCL vychází z Pascalu a zde je vidět posun toho, jak se moderně programují PLC automaty.

Jako ukázku programu v jazyce SCL jsem vybral vlastní funkci FC_RAL použitou v projektu. Tato funkce má za úkol vyjmout ze vstupního řetězce konec zprávy a má jej převést na hodnotu typu real. Reálná hodnota je pak výstupem této funkce. Z níže uvedeného kódu je patrné, že se jedná o strukturovaný programovací jazyk.

```
#t_string := #i_str;
#t_string := MID(IN:=#t_string, L:=#i_cfr, P := (LEN
   (#t_string)-(#i_cfr)));
#t_p := FIND(IN1:=#t_string, IN2:='.');
#t_string := DELETE (IN:=#t_string, L:=1, P:=#t_p);
#t_real := STRING_TO_INT (#t_string);

FOR #t_int := 1 TO #i_cfr-#t_p DO
  #t_real := #t_real/10;
END_FOR;

#o_REAL := #t_real;
```

3 METTLER TOLEDO IND560

Mettler Toledo je průmyslový váhový terminál, viz obr. 3.1. Vážicí rozsah je od miligramů po tuny. Terminál může vážit buď klasicky, na základě tenzometrů, nebo pomocí velmi přesné technologie vážení, která využívá obnovení elektromagnetických sil. Díky terminálu lze řídit plnicí a dávkovací aplikace v ručním, poloautomatickém a automatickém režimu. IND560 je navržen pro průmysl se spojitou regulací. Je to farmaceutický, chemický, potravinářský a nápojový průmysl. Náročným podmínkám odolá díky velmi odolnému krytí IP69K, což znamená odolnost vůči vysokotlaké tryskající horké vodě a prachu. Odpovídá celosvětovým standardům UL, CE, NTEP a OIML. Má několik rozhraní pro komunikaci. Mezi tato rozhraní patří sériová linka RS232, Profibus, Ethernet TCP/IP, Modbus TCP, atd. Všechny důležité vlastnosti jsou shrnuty v tabulce 3.1, která je uvedena na stránkách výrobce.

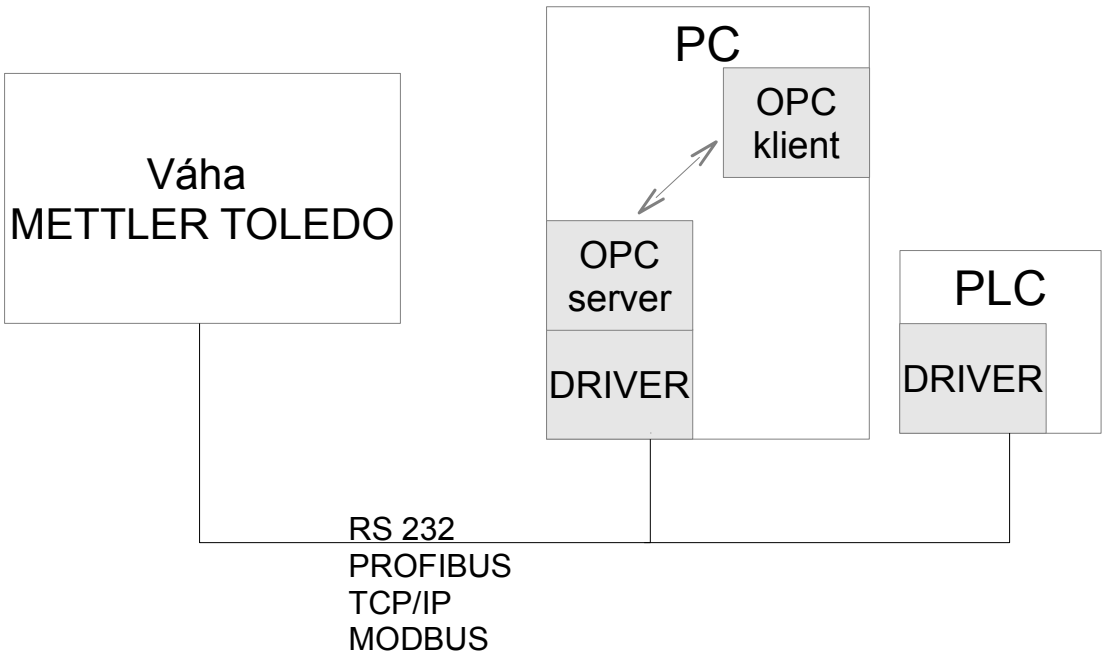
Obr. 3.1: Váhový terminál IND560

Specifikace - IND560 váhový terminál	
Keyboard	25-klávesová číselná doteková klávesnice, navigačními klávesami, funkčními klávesami
Šasí - konstrukce	Náročné prostředí - nerezová ocel; Panel – přední panel z nerezové oceli, hliníková šasi
Hazardous area approval	Třída 1, div 2, zóny 2/22, skupiny A-D, F, G a evropská kategorie 3
Displej	Vakuum-fluorescenční, 21 mm
Rozhraní	Sériová linka, PLC A-B, vzdálené V/V, Profibus DP, DeviceNet, analogové výstupy, 4-20 mA, 0-10 V
Stupeň ochrany	Náročné prostředí – IP69K, Panel – Typ 4/12
Number of attachable platforms	Až osm analogových snímačů 350 ohm
Resolution	100 000 dílků
Mounting options	Na zeď, plošinu, do panelu, na stojan
Signal processing	TraxDSP™ filtrování
Digital Input/Output	Maximálně 12 vstupů/18 výstupů
Rozměry	Náročné prostředí - 16,0 x 26,5 x 17,0 cm; Panel - 16,0 x 26,5 x 9,2 cm
Resolution (approvable)	Analogové - 10,000e OIML, 10,000d NTEP; IDNet – 100,000d OIML, NTEP
Options	Analogový výstup, Ethernet TCP/IP, EtherNet/IP, Modbus TCP, Profibus, A-B RIO, digitální diskrétní V/V, Fill-560 aplikační software
Applications	Standardní procesní/pohyblivé vážení, kontrolní vážení nad/podváhy, plnění
A/D rate (int./ext)	více než 366 Hz interní, 50 Hz porovnání žádané hodnoty, 20 Hz PLC výstup
Suitable platforms / weigh modules	Analogové, IDNet (MMR)

Tabulka 3.1: Specifikace váhového terminálu IND560 [4]

3.1 Možnosti komunikace váhy

Váha může komunikovat pomocí několika různých protokolů, jak už bylo zmíněno. Zaměřme se nyní na komunikaci váhy z hlediska koncového uživatele, resp. operátora, a z hlediska toho, kdo uvádí váhu do provozu. Propojení a komunikaci váhy naznačuje obrázek 3.2.



Obr. 3.2: Možnosti komunikace váhy

Z tohoto obrázku je patrné, že se nám nabízejí dvě možnosti jak komunikovat s váhou. Můžeme využít nabídky výrobce, který nám dodá k váze OPC server, jež pracuje přes OPC klienta, a nebo můžeme vytvořit vlastní komunikační ovladač.

Komunikace s váhou pomocí OPC serveru a OPC klienta má tu výhodu, že je to velmi rychlé. Hotový software se obdrží od výrobce a jen se nainstaluje. Může být k výrobku zdarma nebo se za něj připlatí. OPC server a klient vyžaduje Microsoft Windows, musíme mít PC s daným operačním systémem. To může být někdy problém. V místě použití nemusejí být pracovní podmínky pro PC a je tu také vyšší pořizovací cena. Další celkem podstatnou nevýhodou je údržba takové komunikace. Jedná se o nevlastní software a pokud se vyskytne nějaký problém, je složitější a náročnější hledat chybu v cizím ovladači než ve vlastním. To může také hrát roli v zavedení systému do provozu, mám na mysli čas zavedení systému. Nicméně výhodou je, že lze PC, které má dostatek paměti a výpočetní kapacity, využít i k jinému využití než jen ke komunikaci s váhovým terminálem.

Druhou možností je vytvořit vlastní ovladač. Tento způsob má zase několik výhod i nevýhod. Hlavní nevýhodou je čas, který programátor potřebuje k prostudování komunikace terminálu a dále k vytvoření a odladění tohoto ovladače. Celý proces je velmi časově náročný. K vytvoření kvalitního a robustního ovladače je tedy potřeba nejen dokumentace používaných zpráv a povelů v komunikaci, ale také určité znalosti, zkušenosti a kvalifikace programátora. I když vlastní ovladač je časově náročný a je zapotřebí určité programátorské kapacity přináší i některé podstatné výhody. Vzhledem k znalosti ovladače není takový problém odhalit některé chyby či nedostatky. V některých případech stačí i jednodušší ovladač, který využívá pouze ty příkazy, které

jsou potřebné. Další výhodou je nezávislost ovladače na přístroji, na kterém je použitý. Záleží na nás pro co tento ovladač přizpůsobíme. Může být použit na PC, kde výhody i nevýhody byly popsány, nebo na průmyslovém PLC. Použití PLC je vhodnější v náročném prostředí průmyslu a plně stačí ke komunikaci a zpracování či zobrazení dat z váhového terminálu. Tato varianta je i levnější. V tomto projektu byl právě řešen druhý případ, tedy tvorba vlastního ovladače pro komunikaci váhového terminálu a PLC.

3.2 Zapojení váhy v provozu

Přístroj má pracovat ve farmaceutické firmě. Farmaceutický průmysl patří k těm odvětvím, která mají velmi přísné požadavky na přesnost technologií a na používané přístroje. Přístroje musejí splňovat různé normy a nařízení zákazníka. Při realizaci je nutné postupovat následovně:

1. V první řadě je třeba definovat požadavky zákazníka, tzv: URS. Zákazník pak předá URS, ve kterém je napsáno, co má program dělat. Je velmi důležité porozumět těmto požadavkům zákazníka, aby později nedocházelo k omylům.
2. Dalším bodem je vytvoření funkční specifikace FS. Zde se definuje pohled firmy, která bude realizovat zadávaný projekt. FS je vytvořena na základě URS a je následně vysvětlena zákazníkovi a on ji musí schválit.
3. Třetím bodem je vytvoření programu, který řeší daný problém. Programátor pracuje podle FS.
4. Dalším neméně důležitým bodem je testování. Vyzkoušení celého projektu je nutné před předáním zákazníkovi.
5. Pátým bodem se už zabývá zákazník. Ten vyzkoušený projekt ověří v praxi a hlavně proběhne validace. Validací zákazník potvrdí, že daný projekt odpovídá zadaným požadavkům a vyhovuje specifikovanému provozu.

Váha Mettler Toledo IND560 splňuje veškeré požadavky pro náročný provoz. Váha disponuje velmi odolným krytím IP69K a odpovídá standardům, díky kterým je vhodná pro farmaceutický průmysl. U váhy tedy problém s validací nebyl. Pro komunikaci s váhou a zařízeními od firmy Siemens by bylo nejjednodušší použít Profibus. Toto rozhraní však neprošlo validací. Bylo zamítnuto proto, protože neobsahuje certifikát. Není tedy zaručen správný chod v daných náročných podmínkách. Proto bylo zvoleno rozhraní TCP/IP, které je standardizováno a validací bylo schváleno. V reálném provozu bude tedy váha komunikovat s okolím pomocí protokolu TCP/IP. V této práci je však komunikace řešena přes řetězce znaků, protože není k dispozici váhový modul. A však řetězec znaků lze celkem jednoduše převést a upravit pro přenos pomocí protokolu TCP/IP či pomocí jiných protokolů.

4 KOMUNIKACE S VÁHOVÝM TERMINÁLEM

V dokumentaci váhového terminálu se můžeme dozvědět o několika funkcích, pomocí kterých můžeme s váhou komunikovat. V našem případě však vystačíme s užším výběrem funkcí. Budeme využívat toho, abychom se mohli k váze správně přihlásit a měli tak přístup k funkcím, dále budeme chtít znát aktuální hodnotu měřené hmotnosti a budeme kontrolovat stav připojení, zda komunikujeme správně.

Od zaměstnanců Compasu jsem dostal zápis požadované komunikace s reálnou váhou. Zde uvádím její přepis:

```
  53 Ready for user
  >user admin
  12 Access OK
  >r wt0101
  00R001~       0.000~
  >r wt0103
  00R002~kg~
  >r wt0101
  00R003~       0.000~
  >r sp0100
  00R004~6x                   500g                    SLOZKA
1^1^^78^500.000000^0^0^6^470.000000^480.000000^20.000000^~
  >r wt0101 wt0103
  00R005~        15.5~kg~
  >callback wt0101
  00B006~OK
  >
  00C007~wt0101=        15.0
  >
  00C008~wt0101=        15.5
  >
  00C010~wt0101=        16.0
  >
  00C011~wt0101=        17.5
  >
  00C012~wt0101=        20.0
  >
  00C013~wt0101=        23.0
  >
```

```
00C014~wt0101=        26.5
>
00C015~wt0101=        30.0
>
00C016~wt0101=        34.0
>
00C017~wt0101=        37.5
>
00C018~wt0101=        41.5
>
00C019~wt0101=        45.0
>
00C053~wt0101=        92.0
>
00C054~wt0101=        92.5
>
00C055~wt0101=        93.0
>
00C056~wt0101=        93.5
>
00C057~wt0101=        94.0
>
00C058~wt0101=        94.5
>
00C071~wt0101=        81.0
>
00C072~wt0101=        73.5
>
00C073~wt0101=        65.5
>
00C074~wt0101=        57.5
>
00C075~wt0101=        50.0
>
00C094~wt0101=         6.0
>
00C095~wt0101=         6.5
>
00C096~wt0101=         7.5
>
00C097~wt0101=         8.0
```

```
>
00C098~wt0101=          9.0
>
00C107~wt0101=         15.0
>
00C108~wt0101=         15.5
>
00C109~wt0101=         16.0
>
00C110~wt0101=         15.5
>
00C111~wt0101=         15.0
>
00C112~wt0101=         15.5
>xcallback wt0101
00X113~OK
>
```

Pokud se spojení přeruší a odešlete po obnovení místo přihlášení příkaz, vrátí se vám:

```
83 Command not recognized
```

a musíte se přihlásit:

```
>user admin
12 Access OK
```

Tuto komunikaci jsem převedl do přehledného schématu, tak jak je použita v mé simulaci této komunikace, viz obr. 4.1.

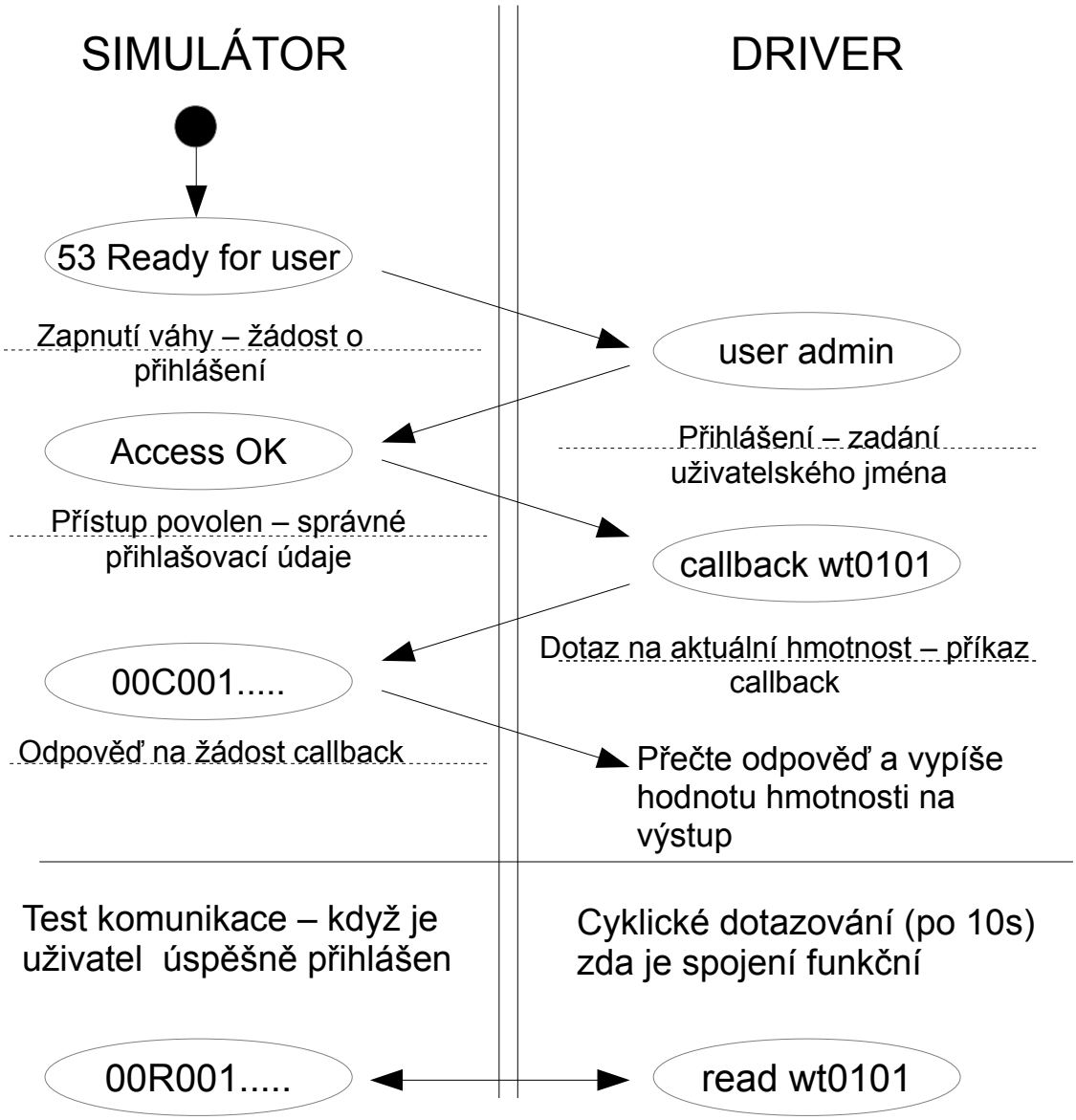
Komunikace by tedy měla probíhat takto:

- Po zapnutí napájení váhy pošle váha zprávu pro ovladač: „53 Ready for user“.
- Driver po obdržení této zprávy odpoví přihlašovacím jménem: „user admin“. Po tomto příkazu by mohlo následovat vyžádání hesla, příkazem: „51 Enter Password“ a odpověď: „pass heslo“. Já se však držel dodaného přepisu komunikace, proto je přihlášení bez hesla. Přidání hesla není ale problém.
- Pokud proběhne úspěšné přihlášení k váze, váha odpoví zprávou: „Access OK“, jsme k váze úspěšně přihlášeni a můžeme využívat její funkce.

- Po správném přihlášení vyšle ovladač žádost o poslání aktuální hodnoty měřené hmotnosti. Vyšle tedy žádost: „callback wt0101“. Callback je příkaz, který má tu funkci, že váha vyšle zprávu o dané hodnotě hned po zadání příkazu a vždy, když se daná hodnota změní. Druhá část příkazu: „wt0101“ určuje váze typ dané hodnoty. V tomto případě to znamená odeslání informace o zaokrouhlené hrubé hodnotě měřené hmotnosti.
- Váha odpoví na tento příkaz ve tvaru: „00C001~wt0101= 15.5“. „00C“ značí odpověď na žádost callback, další tři číslice znamenají číslo odpovědi. Dále následuje již zmíněný typ hodnoty o kterou se jedná. Poslední číslice znamenají naměřenou hmotnost.
- Driver tuto odpověď zpracuje tak, že na jeho výstupu se objeví již číselná hodnota typu real, která udává změřenou hmotnost.

Toto je tedy jedna část komunikace, kterou řeší driver. Dále má ovladač za úkol kontrolovat, zda je spojení s váhou funkční a tedy, zda výstupní hodnota hmotnosti, kterou ukazuje driver na svém výstupu, je opravdu naměřená hodnota váhy. Je to řešeno tak, že driver vyšle každých 10s žádost ve tvaru: „r wt0101“, což je zkrácený zápis: „read wt0101“. Tato žádost znamená, že chceme číst jednorázově aktuální naměřenou hodnotu. Odpověď váhy by měla být ve tvaru: „00R001~ 15.5“. Vysvětlení jednotlivých částí zprávy je obdobné jako u předchozí odpovědi na callback. „00R“ je odpověď na žádost typu „read“, další tři čísla udávají číslo odpovědi a poslední tři čísla nesou informaci o naměřené hmotnosti. Driver tuto zprávu zpracuje tak, že zkontroluje její tvar. Pokud odpověď je relevantní žádosti a daná hmotnost je v mezích, je spojení váhy a ovladače považované za správné.

Předchozí popis ukazuje komunikaci za správného funkčního stavu. Pokud váhu odpojíme a tedy nekomunikuje, ovladač čeká na nová data a na vyzvání k přihlášení. Na výstupu driveru je předchozí naměřená hodnota, ale kontrolní bit o_OK je v nule. Tento nás informuje o správnosti údaje. Pokud váha odpoví nedefinovaným způsobem, je komunikace vyhodnocena jako nefunkční. Bit o_OK je v nule. Ovladač pak kontroluje každých 10s zprávu od váhy, pokud neodpovídá správně je stav stále stejný, jestliže odpoví správně je komunikace vyhodnocena jako správná. Bit o_OK je v jedničce. Ovladač se dotazuje až 60s, jestliže není potvrzeno spojení. Pak se resetuje a zkouší vše celé znovu od přihlášení. Takto je to vytvořeno pro případ, kdyby váha neodpovídala nebo byla resetována.



Obr. 4.1: Náčrt komunikace mezi váhou a ovladačem

5 TVORBA PROGRAMU

Program, který jsem vytvářel, se skládá ze dvou hlavních funkčních bloků. Je to funkční blok simulátoru váhy a funkční blok ovladače, který má s touto váhou komunikovat. V mém případě jsem oba funkční bloky použil v jednom společném projektu tak, abych mohl jejich komunikaci simulovat. V praxi by byl ovladač spuštěn na samostatném PLC. Další důležitou částí jsou dva datové bloky. Jeden obsahuje používané instrukce. Na tento datový blok je odkazováno vždy, když je použita některá definovaná zpráva. Tento datový blok by měl být obsažen v každém PLC, které by mělo komunikovat s váhou. Druhý datový blok slouží pro vlastní komunikaci. V tomto bloku si vyměňují ovladač a váha informace. Je vytvořen jakoby komunikace probíhala ve fullduplexním režimu.

5.1 Datové bloky

Datové bloky jsou dvou typů: vlastní a datové bloky funkčních bloků.

Vlastní datové bloky jsou vytvořeny samostatně a obsahují statické proměnné, kterým můžeme přednastavit hodnotu. K těmto proměnným mají přístup všechny funkce a bloky. Jsou to globální proměnné používané ve více blocích. Tyto budou níže popsány.

Datové bloky funkčních bloků jsou vytvořeny kdykoliv je funkční blok použit. Obsahuje proměnné funkčního bloku. Takových datových bloků může být více pro jeden funkční blok, v tom případě se jedná o použití stejného typu vícekrát. Např.: Použití více simulátorů váhy se stejnou funkcí by obsahovalo jeden funkční blok simulátoru a více datových bloků, kterých by byl stejný počet jako počet simulátorů.

5.1.1 Datový blok instrukcí - DB_INS

Datový blok instrukcí, v projektu nazvaný DB_INS, obsahuje všechny používané instrukce. Byl vytvořen především pro jednodušší správu instrukcí. Jednotlivé instrukce zde můžeme snadno změnit, aniž bychom zasahovali do programu. Máme samozřejmě možnost i přidávání a mazání instrukcí. Tento datový blok zdaleka neobsahuje celou instrukční sadu váhy, ale jen instrukce, které se využívají v tomto projektu. Celý výpis datového bloku ukazuje tabulka 5.1.

Name	Data type	Offset	Start value
access	String	0.0	12 Access OK'
ready	String	256.0	53 Ready for user'
notrec	String	512.0	83 Command not recognized'
read	String	768.0	r wt0101'
callback	String	1024.0	callback wt0101'
user	String	1280.0	user admin'

Tabulka 5.1: Výpis datového bloku instrukcí - DB_INS

Tabulka ukazuje všechny statické proměnné instrukčního datového bloku. V prvním sloupci je uveden název proměnné tak, jak je použitý v programu. Ve druhém sloupci vidíme typ této proměnné, v tomto případě se jedná u všech proměnných o stejný typ string. Další sloupec ukazuje adresu této proměnné a v posledním sloupci je uvedena konkrétní hodnota proměnné, tedy konkrétní instrukce. Tabulka byla vytvořena na základě datového bloku DB_INS, tak jak je zobrazena v programu TIA Portal V11.

5.1.2 Komunikační datový blok – DB_COM

Tento datový blok, který je v projektu nazván DB_COM, není obsahově velký, ale je velmi důležitý. Přes tento blok probíhá vlastní komunikace váhy a ovladače. Obsahuje čtyři proměnné, viz tabulku 5.2. Jsou to dva řetězce znaků, které nesou zprávu ze simulátoru váhy do ovladače a z ovladače do simulátoru. Další dvě proměnné jsou typu bool a označují novou nepřečtenou zprávu. Tento datový blok v podstatě nahrazuje vlastní komunikační protokol. V praxi by byly zprávy posílány například přes TCP/IP. V projektu je to řešeno tak, že simulátor i ovladač buď reagují na zprávu nebo kontrolují, zda nepřišla nová zpráva. Vždy po přečtení zprávy je bit SD_new_data nebo DS_new_data nastaven do nuly a vždy po vyslání zprávy nastaven na hodnotu jedna.

Name	Data type	Start value	Comment
SD_new_data	Bool	false	Bit upozorňující na nová data ze simulátoru do driveru
SD_data	String		Data ze simulátoru do driveru
DS_new_data	Bool	false	Bit upozorňující na nová data z driveru do simulátoru
DS_data	String		Data z driveru do smulátoru

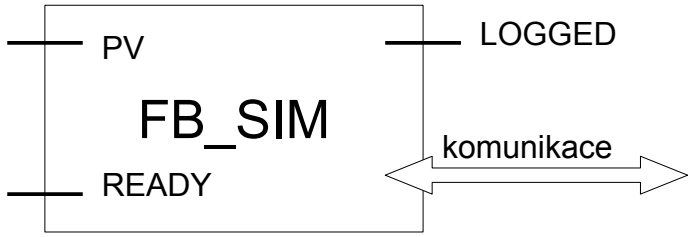
Tabulka 5.2: Výpis komunikačního datového bloku - DB_COM

5.2 Funkční bloky

Funkční blok tvoří funkci. Obsahuje vstupní, statické, dočasné, vstupně-výstupní a výstupní proměnné. Sled instrukcí tvoří danou funkci. Funkční blok může používat funkce nebo může být napojen na další bloky. Použitím funkčního bloku v projektu se vytvoří z něj vycházející datový blok. Tímto se právě liší funkční blok od funkce.

5.2.1 Funkční blok blok simulátoru váhy – FB_SIM

FB_SIM neboli funkční blok simulátoru váhy nahrazuje chování reálné váhy. Obrázek 5.1 nám má pomoci představit si jak je simulátor řešen v projektu. Simulátor má dvě vstupní proměnné. První z nich je PV (process value). Proměnná PV simuluje vstupní naměřenou hmotnost. Je typu real. Druhá proměnná je typu bool a značí připojení napájení váhy. Pokud je tento bit v jedničce, znamená to, že váha může komunikovat a vyšle zprávu: „53 Ready for user“. Po úspěšném přihlášení k váze je nastavena třetí proměnná LOGGED na hodnotu jedna. Tato proměnná je výstupní a má velmi důležitý informační charakter.



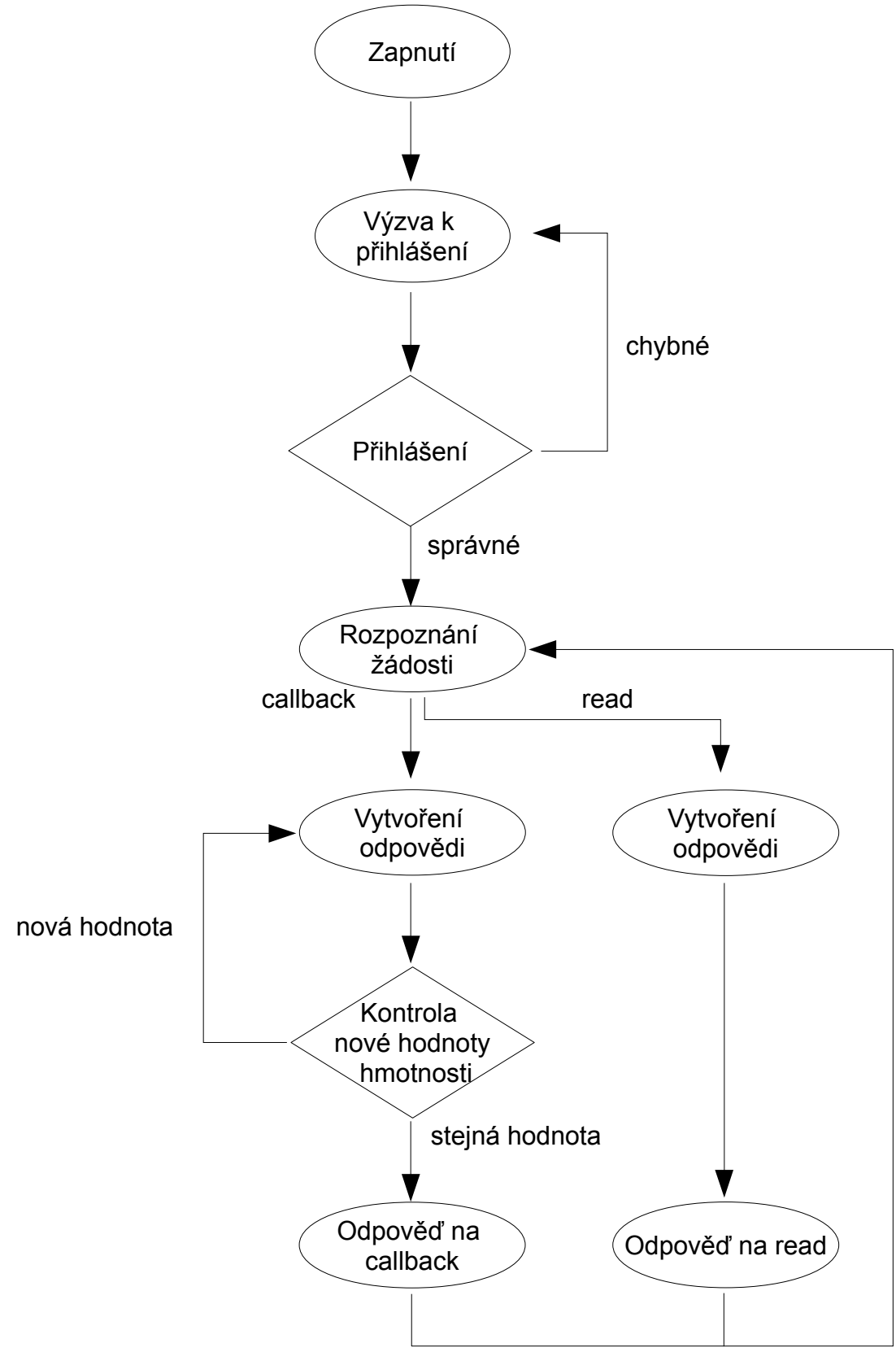
Obr. 5.1: Blokové zobrazení simulátoru váhy

Hlavním jádrem datového bloku je funkce FC_STR, která má za úkol složit a upravit jednotlivé části zprávy do jednoho řetězce. Tyto části jsou různého typu, proto je tato funkce celkem složitá. Využívá několik funkcí na převod a úpravu typu proměnné.

Simulátor v klidovém stavu, kdy je READY nulové, má výstupní bit LOGGED v nule a vysílá zprávu: „83 Command not recognized“. Pokud nastavíme bit READY na hodnotu jedna, vyšle váha zprávu: „53 Ready for user“ a čeká na přihlášení uživatele. Po správném přihlášení je LOGGED nastaveno do jedničky. Správné přihlášení proběhne jestliže zadáme uživatelské jméno, které váha má v databázi (v našem případě pokud je obsaženo v datovém bloku DB_INS), a případně heslo. Heslo je vyžadováno příkazem: „51 Enter Password“. Když zadáme správné udaje, váha odpoví po zadání uživatelského jména: „12 Access OK“, stejná zpráva se objeví i po zadání správného hesla. Simulátor je vytvořen tak, že pokud je zadáno špatné jméno, uživatel se nepřihlásí a je vyzván k opětovnému zadání jména. V případě že je READY a LOGGED v hodnotě jedna, váha čeká na další příkazy. Váha reaguje na dva typy příkazů: „read wt0101“ a „callback wt0101“. V obou případech je ke statické proměnné s_CNT přičtena jednička. s_CNT je typu int. Značí číslo odpovědi na žádosti read a callback. Je společná pro obě žádosti. Nuluje se vždy při přihlášení uživatele. Dále je volána funkce FC_STR. Jako vstupní parametry jsou jí předány typ odpovědi (00C nebo 00R). Je to tedy řetězec znaků. Druhý parametr je pořadí odpovědi, tedy celé číslo s_CNT. Třetí vstupní hodnotou je naměřená hmotnost PV, typ real. Poslední vstupující proměnnou je s_cfr. To je statická proměnná typu int a značí počet cifer hmotnosti v posílané zprávě. Čtyři cifry je implicitní hodnota pro s_cfr. Funkce FC_STR vstupní proměnné zpracuje tak, že výsledkem je jeden řetězec znaků vždy se stejnou délkou (18 znaků pro odpověď typu 00R a 26 znaků pro odpověď typu 00C) dle dodaného

záznamu komunikace, viz kapitolu 4. Jestliže simulátor odpovídá na žádost callback, zároveň kontroluje vyslanou hodnotu PV a aktuální hodnotu. Pokud se hodnoty liší, vyšle novou odpověď s aktuální naměřenou hmotností.

Stavový digram simulátoru je uveden na obr. 5.2. Ten jednoduše shrnuje předchozí popis simulátoru.



Obr. 5.2: Stavový diagram simulátoru váhy

Níže uvádím kód simulátoru. Z příkladu vidíme, že se jedná o strukturovaný text. Obdobným způsobem je tvořen i blok ovladače, proto uvádím pouze kód simulátoru. Celý program je v elektronické verzi.

```
  IF NOT #READY THEN
    "DB_COM".SD_data:= "DB_INS".notrec;
    "DB_COM".SD_new_data:= true;
    #LOGGED := false;
  END_IF;

  //PŘIHLÁŠENÍ
  IF #READY AND NOT #LOGGED THEN
    "DB_COM".SD_data:= "DB_INS".ready;
    "DB_COM".SD_new_data:= true;
    IF "DB_COM".DS_new_data  THEN
      IF "DB_INS".user = "DB_COM".DS_data THEN
        "DB_COM".DS_new_data := false;
        "DB_COM".SD_data:= "DB_INS".access;
        "DB_COM".SD_new_data:= true;
        #s_CNT := 0;
        #LOGGED:= true;
      ELSE
        #LOGGED := false;
      END_IF;
    END_IF;
  END_IF;

  //PŘÍKAZY
  IF #READY AND #LOGGED THEN
    IF "DB_COM".DS_new_data THEN
      //READ
      IF "DB_INS".read = "DB_COM".DS_data THEN
        "DB_COM".DS_new_data := false;
        #s_CNT := #s_CNT + 1;
          "FC_STR"(i_TYP:='00R', i_CNT:=#s_CNT, i_PV:=#PV,
i_cfr := #s_cfr);
        "DB_COM".SD_new_data := true;
      END_IF;

    //CALLBACK
     IF "DB_INS".callback = "DB_COM".DS_data  THEN
       "DB_COM".DS_new_data := false;
```

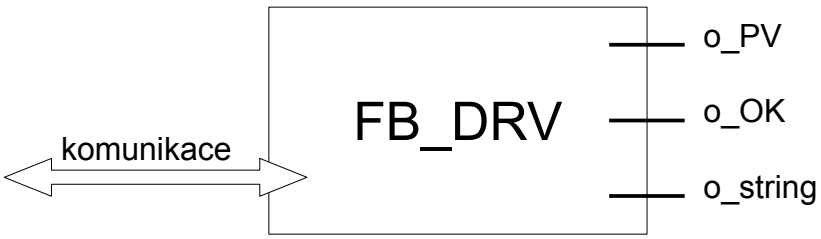
```
        #s_CNT := #s_CNT + 1;
           "FC_STR"(i_TYP:='00C',  i_CNT:=#s_CNT,  i_PV:=#PV,
i_cfr := #s_cfr);
        #s_PV := #PV;
        "DB_COM".SD_new_data := true;
       END_IF;

     END_IF;
   //OPAKOVANÉ ODESLÁNÍ HODNOTY PŘI ZMĚNĚ
     IF "DB_COM".DS_data = "DB_INS".callback THEN
       IF #s_PV <> #PV THEN
         #s_CNT := #s_CNT + 1;
           "FC_STR"(i_TYP:='00C',  i_CNT:=#s_CNT,  i_PV:=#PV,
i_cfr := #s_cfr);
         #s_PV := #PV;
         "DB_COM".SD_new_data := true;
       END_IF;
     END_IF;

   END_IF;
```

5.2.2 Funkční blok ovladače – FB_DRV

Funkční blok ovladače/driveru je v projektu nazvaný FB_DRV. Ovladač komunikuje s váhou dle předem stanovených pravidel komunikace, viz obr. 4.1. Driver má tři výstupní proměnné. Reálná hodnota o_PV ukazuje naměřenou hmotnost. Druhá proměnná označená o_OK je typu bool. Značí správnost spojení mezi váhou a ovladačem. Komunikace je označena za správnou, tedy funkční, je-li na příkaz read odpověď simulátoru v definovaném tvaru a měřená hmotnost v daných mezích. Tato kontrola probíhá každých 10s. Informaci v jakém stavu je komunikace nese třetí výstupní proměnná o_string. Ta textovou zprávou informuje operátora, zda je váha zapnuta a zda bylo spojení potvrzeno. Blokové zobrazení ovladače je na obr. 5.3. V bloku ovladače je vytvořena jedna složitější funkce FC_RAL. Ta by se dala nazvat inverzní k funkci FC_STR. Ze vstupního řetězce znaků vyjme hodnotu naměřené hmotnosti. Výstupem je o_PV, tedy typ real.

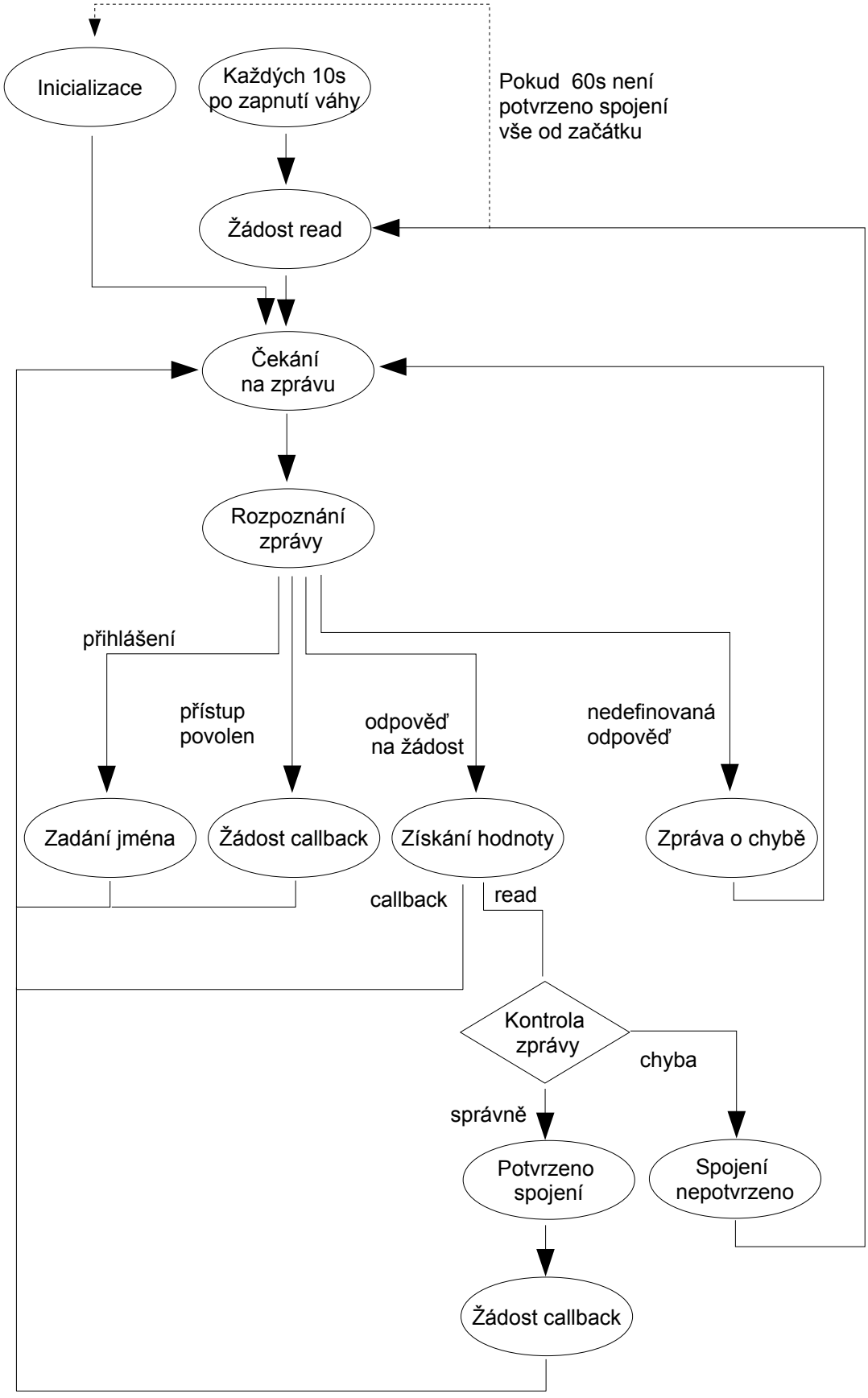


Obr. 5.3: Blokové zobrazení ovladače

Ovladač nejdříve čeká na vyzvání k přihlášení uživatele. Když ovladač obdrží žádost váhy o přihlášení, vypíše do výstupní proměnné o_string zprávu: „Váha zapnuta, spojení nepotvrzeno.“. Dále se ovladač přihlásí pomocí předem nastaveného uživatele příkazem: „user admin“. Pokud vše proběhne v pořádku, driver vyšle žádost callback. Jakmile odešle předchozí žádost a přijdou nová data pro ovladač, volá funkci FC_RAL a získává hodnotu naměřené hmotnosti. Nezávisle na předchozím cyklu vyšle ovladač každých 10 vteřin žádost read. Po odeslání žádosti kontroluje příchozí zprávu. Pokud zpráva začíná: „00R“ a udávaná hodnota hmotnosti je v mezích je spojení považováno za funkční a bit o_OK je nastaven do jedničky a na výstupu driveru se objeví zpráva: „Váha zapnuta, spojení potvrzeno.“. Jestliže tomu tak není je bit o_OK v nule a spojení není správně funkční a výsledek o_PV nelze považovat za správnou hodnotu. V řetězci o_string je předešlá zpráva. Meze hodnoty o_PV lze nastavit pomocí statické proměnné s_max, ta je typu int. Je jedním z parametrů ovladače. Do funkce FC_RAL vstupuje jako proměnná řetězec příchozí zprávy určené pro driver a druhou proměnnou je celočíselná hodnota i_cfr, která udává počet cifer čísla hmotnosti ve zprávě. Počet cifer je přednastaven na čtyři. Výstupní proměnnou je o_REAL, která udává hodnotu naměřené hmotnosti.

Funkce FC_RAL vyjme ze vstupního řetězce poslední část zprávy, která nese informaci o hmotnosti. Vyjmutý řetězec nelze převést přímo na typ real, protože tato funkce v jazyce SCL vyžaduje vyjádření čísla v exponenciálním tvaru a takto není hmotnost v řetězci napsána. Proto je nejdříve z řetězce odstraněna desetinná tečka. Dále je řetězec převeden na typ int. Díky informacím o celkovém počtu cifer čísla a pozice desetinné tečky je hodnota typu int dělena deseti tolikrát, abychom získali správnou hodnotu. Výsledné číslo je tedy typu real.

Na obr. 5.4 je uveden stavový diagram vytvořeného ovladače, jehož popis je uveden výše.



Obr. 5.4: Stavový diagram ovladače

Zde je výpis kódu ovladače:

```
start:
#t_cfr := #s_cfr + 1;                                  // K počtu
cifer přičteme jedničku kvůli desetinné tečce

  IF "DB_COM".SD_new_data THEN
     #t_string := MID (IN:="DB_COM".SD_data, L:=3, P:=1);
//Řetězec pro kontrolu typu přijaté zprávy
    //PŘIHLÁŠENÍ
    IF "DB_INS".ready = "DB_COM".SD_data THEN
      "DB_COM".SD_new_data := false;
      "DB_COM".DS_data := "DB_INS".user;
      "DB_COM".DS_new_data := true;
      #o_string := 'Váha zapnuta, spojení nepotvrzeno.';

    //ŽÁDOST CALLBACK
    ELSIF "DB_INS".access = "DB_COM".SD_data THEN
      "DB_COM".SD_new_data := false;
      "DB_COM".DS_data := "DB_INS".callback;
      "DB_COM".DS_new_data := true;

    //ČTENÍ ZPRÁV S ÚDAJEM O HMOTNOSTI
    ELSIF #t_string = '00R' OR #t_string = '00C' THEN
      "DB_COM".SD_new_data := false;
      "FC_RAL"(i_str:="DB_COM".SD_data,                  //
Funkce, která vyjme ze zprávy hodnotu hmotnosti PV
               i_cfr := #t_cfr,
               o_REAL=>#o_PV);
       IF #t_string = '00R' AND 0 <= #o_PV AND #o_PV <=
#s_max THEN
         #o_OK := true;
         #o_string := 'Váha zapnuta, spojení potvrzeno.';
         "DB_COM".DS_data := "DB_INS".callback;
         "DB_COM".DS_new_data := true;
          #t_TMR := 0.0;
      END_IF;

    ELSE
      "DB_COM".SD_new_data := false;
      "DB_COM".DS_new_data := false;
      "DB_COM".DS_data := '';
```

```
      #o_string := 'Váha vypnuta, nekomunikuje.';
      #o_OK := false;
   END_IF;

END_IF;

//KONTROLA FUNKČNÍHO SPOJENÍ S VÁHOU
IF #t_TMR < 10.0 THEN
   #t_TMR := #t_TMR+#i_TMR;
 ELSIF #t_TMR>= 10.0 THEN
    "DB_COM".DS_data := "DB_INS".read;
    "DB_COM".DS_new_data := true;

   ELSIF #t_TMR>=60 THEN
     #t_TMR := 0;
     GOTO start;

 END_IF;
```

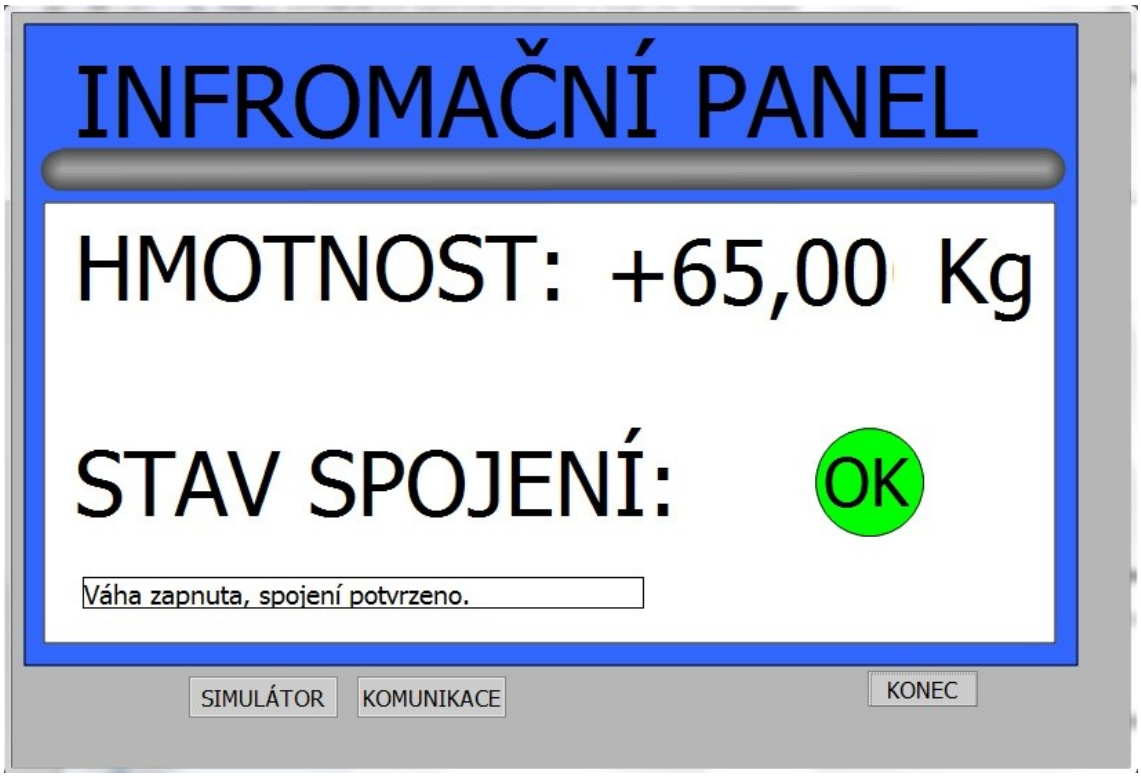
6 OVLÁDACÍ PANEL

Pro přehlednost jsem vytvořil ve vizualizačním programu TIA Portal V11 WinCC obrazovky, které ukazují použití programů. Tato vizualizace může být realizována v PC nebo v panelech různých typů. Panely nemusejí být výhradně od firmy Siemens, ale musí podporovat WinCC Comfort. Od Siemense můžeme vybírat z panelů SIMATIC HMI o rozměrech 4"-12". Já bych navrhoval použít dotykový panel SIMATIC HMI TP700 Comfort, který je sedmipalcový.

6.1 Informační panel

První panel, který byl vytvořen byl nazván informační. Ten by mohl sloužit operátorovi na pracovišti jako kontrola měřené hmotnosti. Na panelu je zobrazena naměřená hmotnost a stav spojení, zda je údaj správný. Dále je zobrazena textová zpráva ovladače, která informuje o tom, zda je váha zapnuta a zda je spojení v pořádku. Viz obr. 6.1 a 6.2.

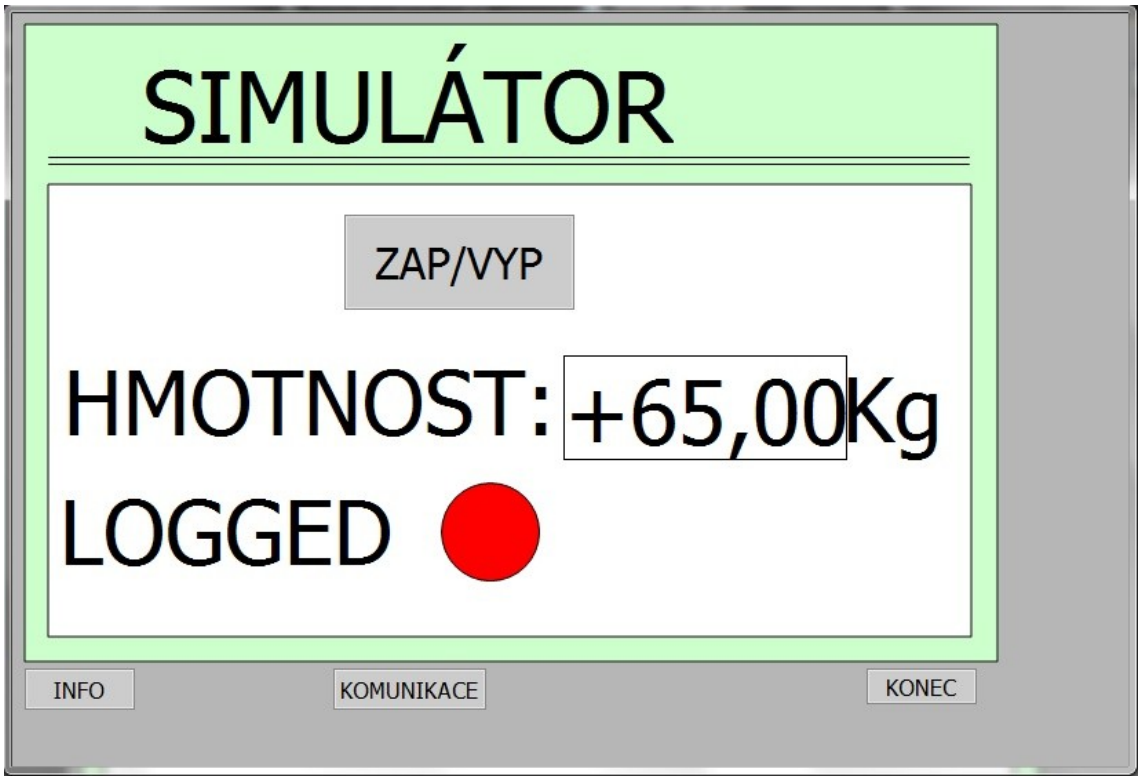
Obr. 6.1: Informační panel - váha vypnuta



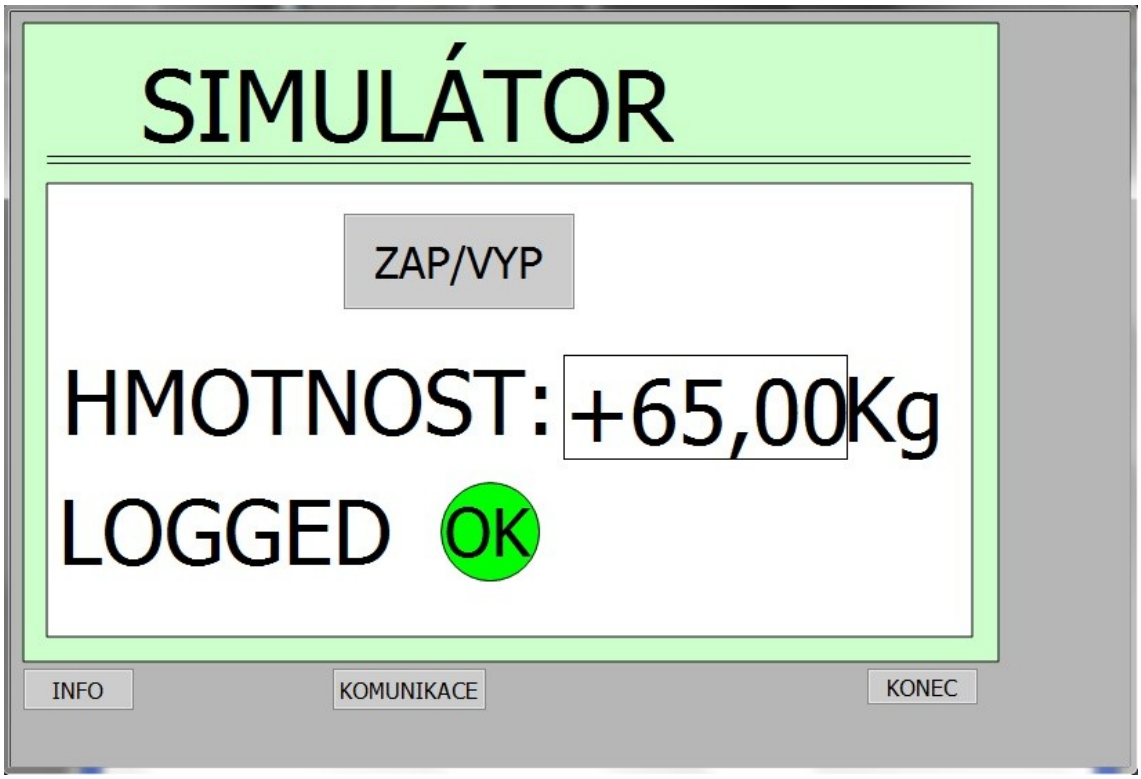
Obr. 6.2: Informační panel - váha zapnuta, spojení ok

6.2 Simulátor váhy

Okno simulátoru váhy umožňuje jednoduše simulovat reálnou váhu. Ve vrchní části okna je tlačítko pro zapnutí/vypnutí váhy. Dále je okno pro vložení simulované hmotnosti. Ve spodní části je zobrazena kontrolka. Ta je v počátečním stavu červená. Pokud se barva změní na zelenou a objeví se nápis OK, znamená to, že je k váze správně přihlášen ovladač. Viz. obr. 6.3 a 6.4.



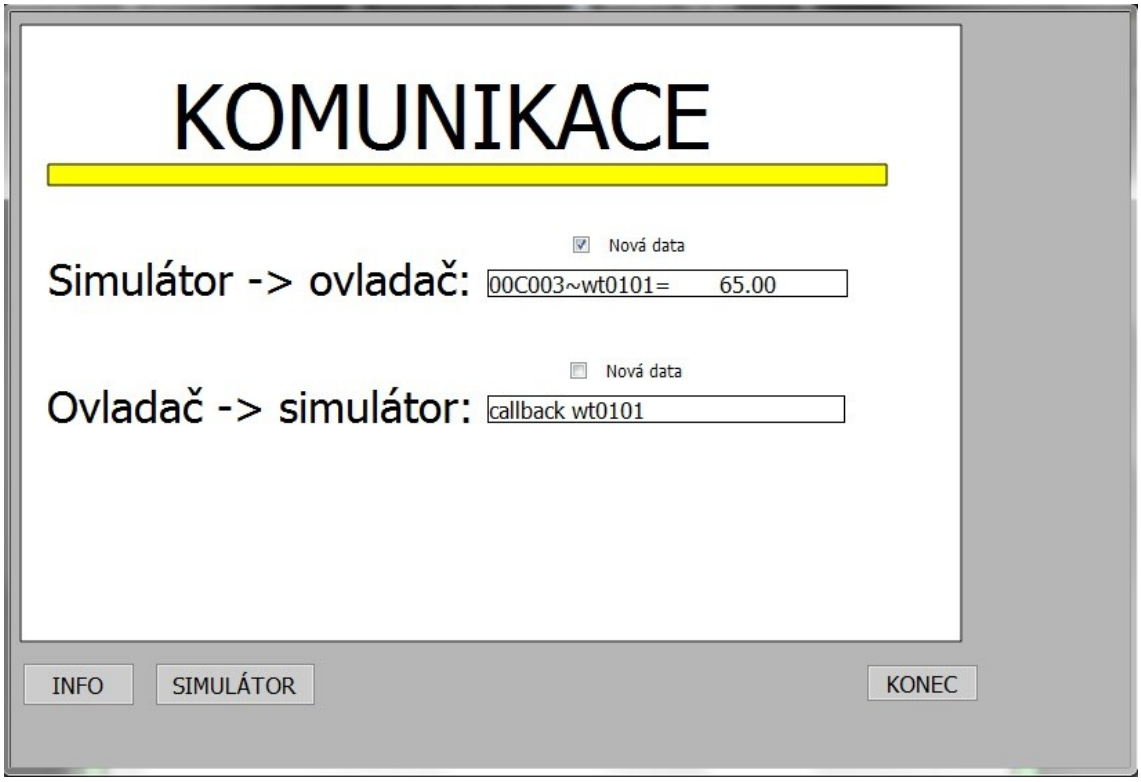
Obr. 6.3: Simulátor váhy - vypnuto



Obr. 6.4: Simulátor váhy - zapnuto

6.3 Komunikace

Třetí okno zobrazuje zprávy, které si vyměňují simulátor a ovladač. Do těchto oken je možno poslat i vlastní zprávu a simulovat tak poruchu nebo vyzkoušet vlastní přihlášení. Viz. obr. 6.5.



Obr. 6.5: Komunikační okno

7 ZÁVĚR

Práce se zabývá vytvořením komunikačního ovladače k váze Mettler Toledo IND560 a také jejího simulátoru. Ovladač i simulátor jsem programoval pomocí jazyka SCL. Tento jazyk pochází od firmy Siemens. Z důvodu horší podpory jazyka vývojovými prostředími nebylo zatím jeho rozšíření nijak velké. S novou verzí TIA Portalu, která obsahuje Step7 V11, je použití vyššího programovacího jazyka pohodlnější a snazší. Je to hlavně díky tomu, že lze odlaďovat programy i v jednotlivých funkčních blocích.

V první části práce se věnuji srovnáním jazyků používaných k programování PLC od firmy Siemens. Jednotlivé jazyky vznikaly tak, jak se vyvíjel trend programování PLC. Od strojového textu přes zobrazení elektrického schématu a funkční bloky až k moderním strukturovaným jazykům a sekvenčnímu grafickému programování, zde se domnívám, že se trend programování PLC dělí na dvě odlišné větve. Pohled programátora a pohled technologa. V kapitole 2 jsou vytyčeny hlavní přednosti těchto jazyků. Pro lepší představu jsem přiložil i části kódu napsaných v jednotlivých jazycích.

Další část práce jsem věnoval váze Mettler Toledo IND560, se kterou má ovladač komunikovat. V kapitole 3 jsem vyčetl hlavní parametry váhy a popsal možnosti propojení a komunikace váhy v praxi. Zde jsem také uvedl stručný přehled realizace projektu tak, jak by měla probíhat v praxi.

V kapitole 4 je vložen záznam komunikace váhy s ovladačem. Tento záznam jsem obdržel ve firmě COMPAS automatizace. Záznam mi byl vodítkem k tomu jak zrealizovat simulátor váhy a také vlastní ovladač. K analýze záznamu mi také posloužila dokumentace k váze [5] a [6].

Hlavní náplní práce bylo samotné vytvoření simulátoru a ovladače. Jak bylo zmíněno programoval jsem pomocí jazyka SCL ve Step7 V11. S tímto jazykem jsem měl možnost se setkat již ve svém semestrálním projektu. Ale i přes to jsem při bakalářské práci nastudoval mnoho funkcí knihoven jazyka SCL. Bylo to pro mě z určité části objevování nového. Výsledkem toho je simulátor váhy IND560 a ovladač k váze, jejichž funkce jsou popsány v kapitole 5.

V kapitole 6 jsem pracoval na vizualizaci simulátoru a ovladače. Navrhl jsem informační obrazovku pro operátora, která by mohla být na pracovišti např. v sedmipalcovém dotykovém panelu TP700 Comfort firmy Siemens. Dále jsem vytvořil obrazovku pro simulátor. Tato obrazovka slouží k simulaci reálné váhy. Poslední obrazovka, která byla vytvořena, je určena k zobrazení vlastních zpráv komunikace ovladače i simulátoru. Okno slouží pro kontrolu zpráv a také je tu možnost posílání vlastních zpráv.

Díky této práci jsem obohacen o mnoho zkušeností, hlavně s prací v jazyce SCL.

Použité zkratky

TIA – totally integrated automation
MES - manufacturing execution systém
HMI – human machine interface
PLC - programmable logic controller
STL - statement list
FBD - function block diagram
GRAFCET - graphe gonctionnel de connexion etapes transitions
SCL - structured control language
OPC – open process control
URS – user request
FS – functional specifications

Literatura

[1] Rábová, Z., Hanáček, P., Peringer, P., Přikryl, P., Křena, B.: Užitečné rady pro psaní odborného textu [online]. Brno: c1993-2008, aktualizováno 2008-11-01. Dostupné z: http://www.fit.vutbr.cz/info/statnice/psani_textu.html

[2] GRAFCET. Togaware [online]. 1.1.2005. Dostupné z: http://www.togaware.com/linux/survivor/GRAFCET.html

[3] IND560 - váhové terminály. *METTLER TOLEDO* [online]. 2012-03-09. Dostupné z: http://cs.mt.com/cz/cs/home/products/Industrial_Weighing_Solutions/terminals_indicators/IND560_Family.html

[4] METTLER TOLEDO: IND560 váhový terminál. *METTLER TOLEDO* [online]. 2.7.2007 [cit.2012-03-30].Dostupné z: http://cs.mt.com/cz/cs/home/products/Industrial_Weighing_Solutions/terminals_indicators/IND560_Family/IND560.html?cnty=cz

[5] METTLER TOLEDO. IND560 Ethernet komunikace. 2005. vyd. Polaris Parkway, Columbus, Ohio, 2005, 34 s.

[6] METTLER TOLEDO. IND560 Terminal: Shared Data Reference. 2010. vyd. Polaris Parkway, Columbus, Ohio, 09.2010, 95 s.

[7] PÁSEK, Jan. VUT BRNO. Programovatelné automaty v řízení technologických procesů, Brno, 30.11.2007. Dostupné z: https://www.vutbr.cz/www_base/priloha.php? dpid=45795

[8] SIMATIC HMI Comfort Panels - SIMATIC HMI, the leading Human Machine Interface solution - Siemens. Siemens [online]. 2010. Dostupné z: http://www.automation.siemens.com/mcms/human-machine-interface/en/operator-interfaces/hmi-comfort-panels/Pages/Default.aspx

Seznam příloh

Příloha 1.CD s elektronickou verzí