KATEDRA ELEKTRICKÝCH POHONŮ A TRAKCE

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

FAKULTA ELEKTROTECHNICKÁ

OVLÁDÁNÍ SIMULÁTORU VÝTAHU POMOCÍ PLC

BAKALÁŘSKÁ PRÁCE

KVĚTEN 2021

JIŘINA ŠVAMBERGOVÁ

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Švambergová** Jméno: **Jiřina** Osobní číslo: **468043**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra elektrických pohonů a trakce**

Studijní program: **Elektrotechnika, energetika a management**

Studijní obor: **Aplikovaná elektrotechnika**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Ovládání simulátoru výtahu pomocí PLC

Název bakalářské práce anglicky:

Control of Lift Simulator

Pokyny pro vypracování:

1) Seznam te se s vlastnostmi a způsoby řízení technologických celků pomocí programovatelných logických automatů
2) Navrhněte logické řízení pro model čtyřpodlažního výtahu
3) Na přípravku s PLC SIMATIC a HMI realizujte navrženou logiku řízení

Seznam doporučené literatury:

[1] Weidauer J., Messer R. Electrical Drives, Publics Erlangen, 2014
[2] SCE Training Curriculum. Siemens AG, 2016
[3] Durry B. The Control Techniques Drives and Controls Handbook 2nd ed., IeT, 2009
[4] Pavelka J., Kobrle P. Elektrické pohony a jejich řízení. 3. přepracované vydání. Praha: České vysoké učení technické v Praze, 2016. ISBN 978-80-01-06007-0.

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Jan Bauer, Ph.D., katedra elektrických pohonů a trakce FEL

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

Datum zadání bakalářské práce: **28.01.2021** Termín odevzdání bakalářské práce: _____

Platnost zadání bakalářské práce: **30.09.2022**

Ing. Jan Bauer, Ph.D. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

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

Datum převzetí zadání

Podpis studentky

PODĚKOVÁNÍ

Děkuji především vedoucímu mé bakalářské práce Ing. Janu Bauerovi, Ph.D. za cenné připomínky a věcné rady. Ing. Martinovi Kozákovi, za technickou podporu. A v neposlední řadě bych chtěla poděkovat svému příteli, rodině a všem kteří mě podporovali.

PROHLÁŠENÍ

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

V Praze dne 12. května 2021

.....

ABSTRAKT

Tato práce obsahuje seznámení se s problematikou programování PLC od společnosti Siemens ve vývojovém prostředí SIMATIC TIA Portál (STEP 7, WinCC, Startdrive). Návrh jednoduchého logického řízení čtyřpodlažního výtahu a vizualizaci v HMI panelu.

Klíčová slova: Siemens, PLC, TIA Portál, výtah, HMI panel

ABSTRACT

This document contains an introduction to Siemens PLC programing in the Engineering Tool - SIMATIC TIA Portal (STEP 7, WinCC, Startdrive). My simple elevator control system project with visualizations in HMI panel.

Keywords: Siemens, PLC, TIA Portal, lift, HMI panel

OBSAH

SEZNAM OBRÁZKŮ

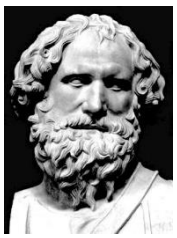
ÚVOD

Mým cílem bylo seznámit se s vlastnostmi a způsoby řízení pomocí programovatelných logických celků. Na základě těchto znalostí jsem poté měla navrhnout řízení čtyřpodlažního výtahu pomocí PLC. Doposud jsem neměla zkušenosti s programováním PLC. Díky projektu mojí bakalářské práce jsem se naučila pracovat s vývojovým prostředím Siemens TIA Portál a hardwarem sloužícím pro řízení výtahu. Velkou pozornost v mé práci věnuji právě seznámení s danými funkcemi TIA Portálu a následnému využití v mé práci.

KAPITOLA 1: HISTORIE ŘÍZENÍ VÝTAHŮ.

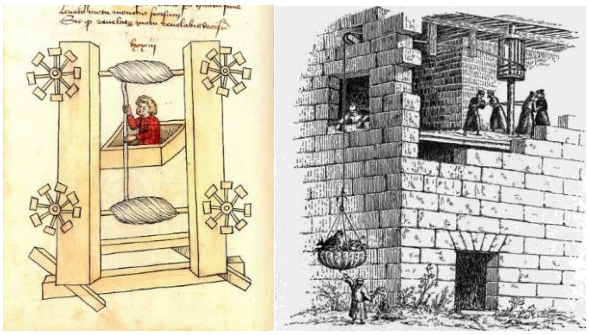
1.1 Začínáme od A, tedy Archimédes.

Když jsem se zamýšlela, jak uchopit téma výtahu, a kde jsou počátky zdvihacích mechanismů, s překvapením jsem zjistila, že na počátku stál samotný Archimédes. Stalo se to ve třetím století před naším letopočtem. Tento výtah byl velice podobný tomu současnému. Jeho kabina byla zavěšena na konopném laně a do výšky byl vytahován pomocí ručního vrátku. [5]



Obr. 1-1 Archimédes [6]

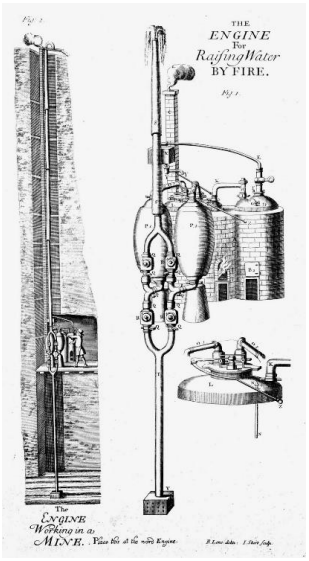
Dalším hrdým vlastníkem výtahu byl císař Nero r. 37–68 našeho letopočtu. Zajímavostí tohoto výtahu byly nafouklé kožené vaky, které měly zbrzdit výtah v případě pádu. Za zmínku také stojí Ludvík XV., který si roku 1743 nechal postavit výtah pro přístup do své soukromé komnaty ve Versailles. [7] U tohoto výtahu bylo prvně použito protizávaží. Ve Francii bylo používání výtahu výsadou krále, a nikdo jiný ho nesměl používat. Pravděpodobně první výtah, který byl zřízen v obytném domě, zkonstruoval Erhard Weigela v roce 1670 v Jeně. [8]



Obr. 1-2 Výtahy poháněné lidskou [8] nebo zvířecí silou [5]

1.2 První výtah na parní pohon.

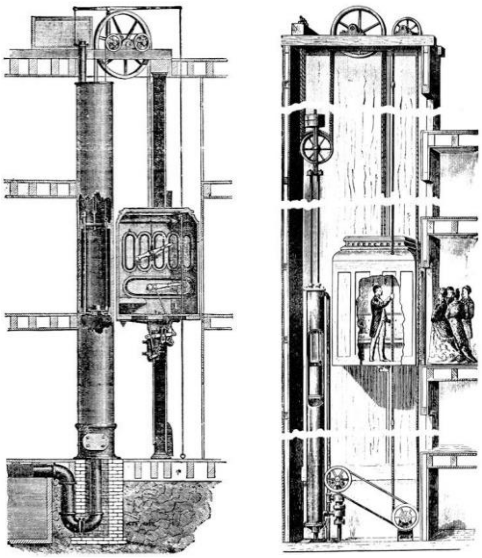
Angličané Frost a Strutt mají na svém kontě první výtah na parní pohon, který byl v roce 1830 postaven ve městě Derby. [9]



Obr. 1-3 Výtah na parní pohon [9]

1.3 Hydraulické a pneumatické pohony.

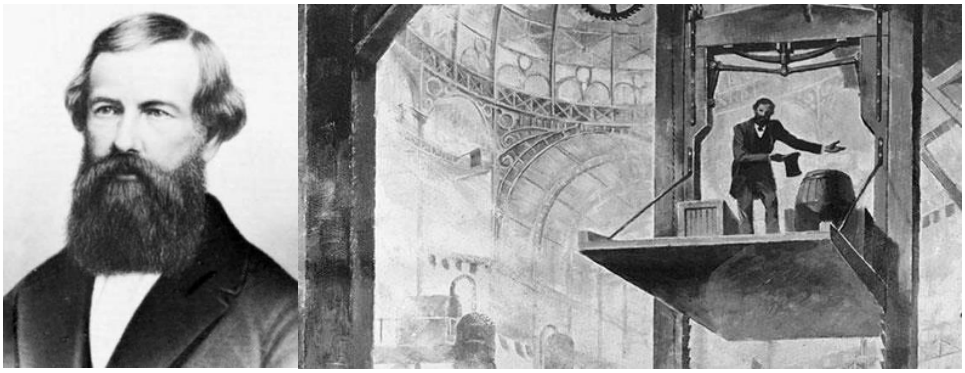
Pneumatický výtah byl vynalezen v roce 1845, následně byl v roce 1846 zprovozněn hydraulický pohon výtahu. Konstrukce byla provedena tak, že válec byl umístěn pod zemí, a píst byl zvedán vodou, kterou v současnosti nahradil hydraulický olej. Píst se zvedal nebo klesal podle tlaku kapaliny. Z počátku byl tok vody obsluhován lany, později pákovým mechanismem, který byl přesnější a plynulejší. U tohoto výtahu nebylo potřeba protizávaží, jelikož kabina výtahu klesala jen takovou rychlostí, jak rychle odtékala voda z pístu. První instalace tohoto výtahu byla v císařském paláci ve Vídni. [5]



Obr. 1-4 Výtah na vodní pohon a Hydraulický výtah [5]

1.4 Epocha výtahů s použitím lana, kladky a protizávaží.

V roce 1853 vynalezl Elisha G. Otis konstrukci výtahu, která se s malými obměnami používá dodnes. Tento výtah měl po stranách vodící lišty, které měly umožnit v případě přetržení lana zbrzdit pád kabiny pomocí takzvaných zachycovačů. Tento výtah byl poprvé použit v New York Crystal Palace. [5]



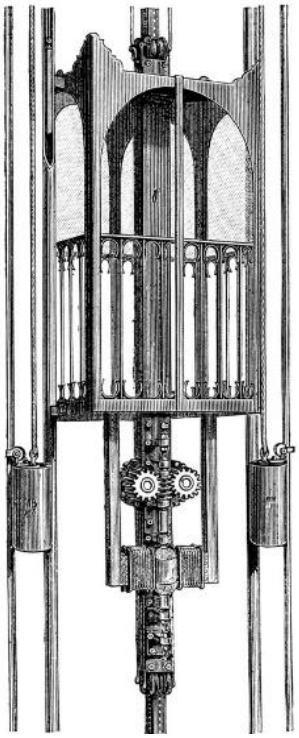
Obr. 1-5 Elisha G. Otis [10] a konstrukce Otisova výtahu [5]

1.5 Dalším milníkem byl výtah pana Werner von Siemense.

Na průmyslové výstavě v Mannheimu v roce 1880 představil Siemens výtah poháněný elektromotorem umístěným přímo pod podlahou kabiny, který otáčel ozubeným pastorkem dosedajícím na vodící hřeben.

Obr. 1-6 Werner von Siemense [11]

V této chvíli jsem si uvědomila, že konstrukce modelu, který používám ve své bakalářské práci se velice podobá první konstrukci Siemensova výtahu. Nakonec však převládlo trakční řešení s použitím lana přehozeného přes lanovnici, kde je na jedné straně lano kabiny a na druhé lano závaží. Toto staronové řešení ušetřilo energii a umožnilo používat výtah ve výškových budovách.[12]



Obr. 1-7 Konstrukce Siemensova výtahu [12]

1.6 První výtahy v českých zemích

První nákladní výtah byl zkonstruován firmou Breitfeld – Daněk v roce 1876. Tato firma se věnovala především průmyslovým výtahům, a byla jedním z předchůdců ČKD. Tento výtah byl umístěn v Litoměřickém pivovaru.

První elektrický výtah s tlačítkovým ovládáním byl použit v hotelu Modrá hvězda v Praze. Dalším pak byl výtah na zámku Konopiště zřízený pro Františka Ferdinanda d'Este. [12]

Obr. 1-8 Strojovna v zákopech [12] a ovládání Konopiště [12]

Obr. 1-9 Kabina výtahu Konopiště [12]

KAPITOLA 2: PLC

Programmable Logic Controller (PLC) je průmyslový počítačový kontrolní systém, který neustále sleduje stav vstupních veličin, a provádí změny na základě požadovaného stavu výstupních veličin v reálném čase.

PLC byly vyvinuty jako náhrada reléových logických systémů pro automobilový průmysl. Dnes slouží k řízení všech výrobních procesů, jako jsou například montážní linky a robotika, kde je vyžadována vysoká spolehlivost, snadná programovatelnost a diagnostika poruch.

Hlavní rozdíl mezi PLC a jinými počítačovými zařízeními je jejich odolnost vůči vlivu prostředí jako je prach, vlhkost, teplo a mráz. Výhodou PLC je jejich snadná údržba a rychlé opravy, které jsou umožněny modulovým uspořádáním. Další velkou výhodnou je možnost kdykoliv změnit program k danému procesu, nebo nahrát stávající program do nového PLC. [4]

Obr. 2-1 PLC Siemens [13] a příklad reléové logiky [14]

2.1 Hlavní části PLC

2.1.1 Napájecí zdroj

Napájecí zdroj přivádí proud do centrální procesorové jednotky (CPU) a ostatních modulů na montážní liště. Velikost proudu závisí na velikosti PLC modulu. Pro menší moduly se proud pohybuje od dvou do deseti ampér, pro větší moduly může být až padesát ampér. Většina PLC je vybavena také záložní baterií, která v případě výpadku proudu zajistí krátkodobou dodávku energie.

2.1.2 CPU s integrovaným displejem

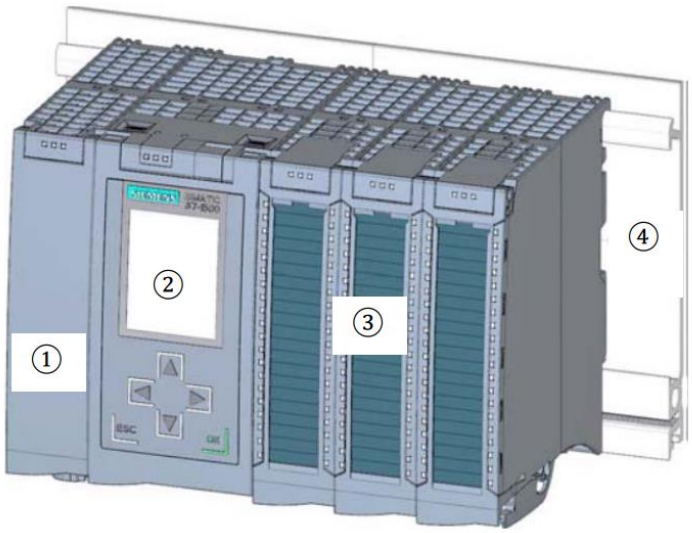
Central Processing Unit (CPU) je centrální procesorová jednotka, která je tvořena mikroprocesorem, který provádí kontrolu a výpočty dat, a jejich následné ukládaní do paměti. CPU také provádí komunikaci mezi jednotlivými komponenty PLC.

2.1.3 Vstupně výstupní karty

Vstupně výstupní karty slouží k propojení PLC s ostatními zařízeními jako jsou regulační ventily, teplotní čidla, tlaková čidla, průtoková čidla apod. Vstupy a výstupy mohou být buď analogové (spojité) nebo digitální (diskrétní). Analogový signál se může měnit v závislosti na změně tlaku, teploty a podobně.

2.1.4 Montážní lišta

Montážní lišta slouží ke snadnému upevnění modulů. Výhodou tohoto způsobu uchycení je snadná údržba a možnost rozšíření systému o další moduly.



Obr. 2-2 Hlavní část PLC [13]

2.2 Programování PLC

2.2.1 Programovací jazyky

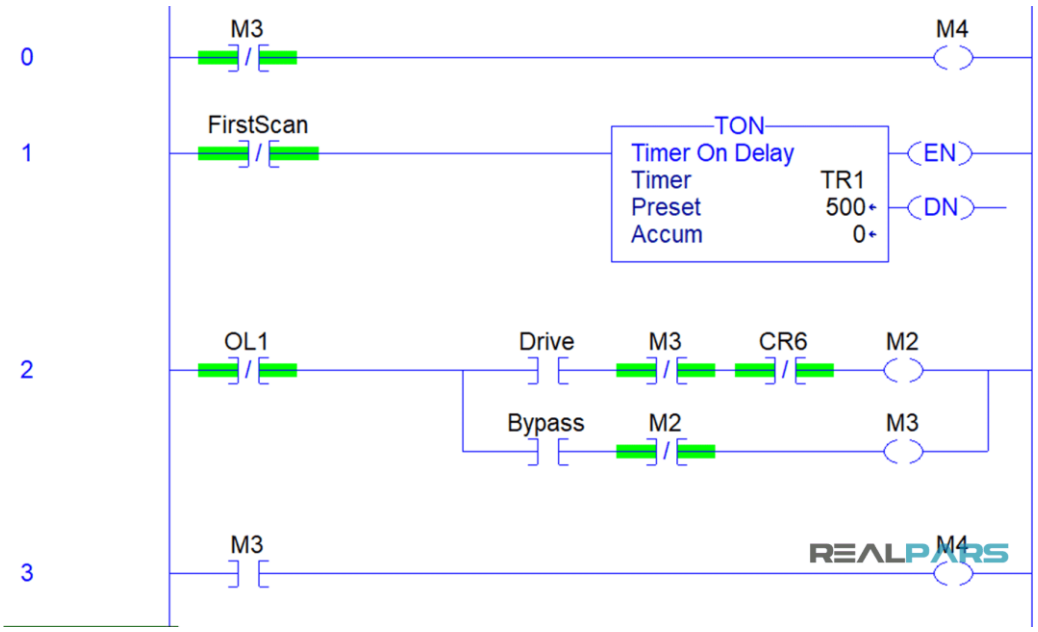
Existuje pět typů programovacích jazyků, kterými lze PLC naprogramovat: ladder diagram, functional block diagram, sequential function chart, instruction list a structured text.

2.2.2 Ladder diagramy

Ladder diagram (LD) je grafický jazyk určený k programování automatických systémů. Jedná se o nejstarší a zároveň nejpopulárnější programovací jazyk učený k programování automatických systémů. Stejně jako u liniových schémat pro reléovou logiku se ladder diagram skládá ze dvou vertikálních linií, které znázorňují napájení, a horizontálních linií (řádků), které reprezentují řídící obvody.[15]

Rozeznáváme dva způsoby využití ladder diagramů: kombinační a sekvenční logiku. Při použití kombinační logiky je výstup stejný bez ohledu na to, zda se změnila vstupní hodnota. Sekvenční logika nám umožňuje zpětnou vazbu. V ladder diagramech je to zajištěno tak, že výstup jednoho obvodu slouží k ovládání vstupu jiného.

Výhodou ladder diagramu je právě jeho podobnost s liniovými schématy pro reléovou logiku, jelikož je díky tomu přehledný pro elektrotechnika. Nevýhodou je nepřehlednost, která nastává při vytváření složitějšího programu. [16]

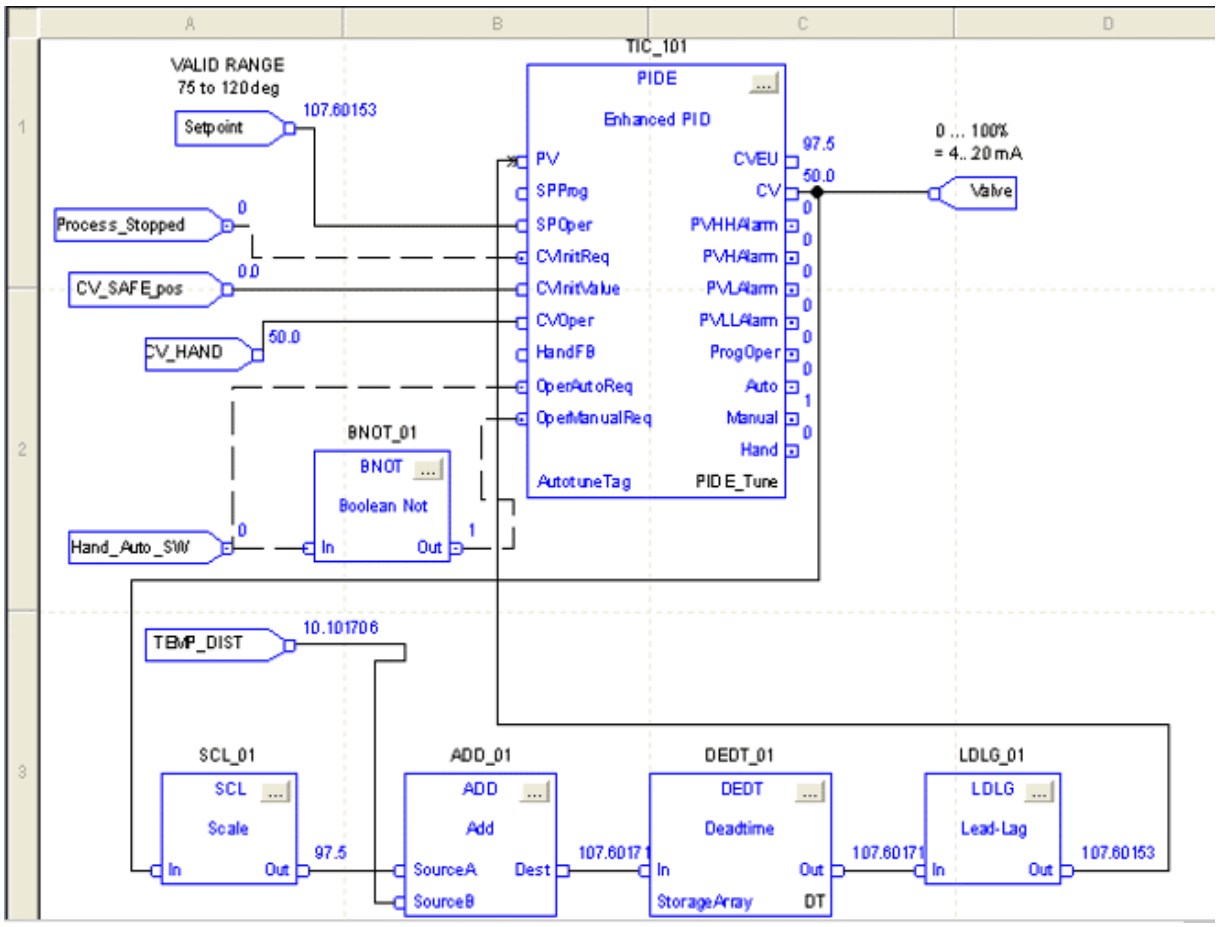


Obr. 2-3 Příklad ladder diagramu[15]

2.2.3 Function Block Diagram

Function Block Diagram (FBD) je grafický jazyk sloužící pro programování PLC, který pomocí bloků popisuje závislost (funkce) mezi vstupními a výstupními veličinami. FBD využívají stejně jako ladder diagramy kombinační logiku. To znamená, že pomocí logických operátorů jako OR a AND, operátorů přiřazení a negace, bistabilních funkčních bloků, časových bloků a detekce hran, definuje závislosti mezi jednotlivými vstupy a výstupy.[17]

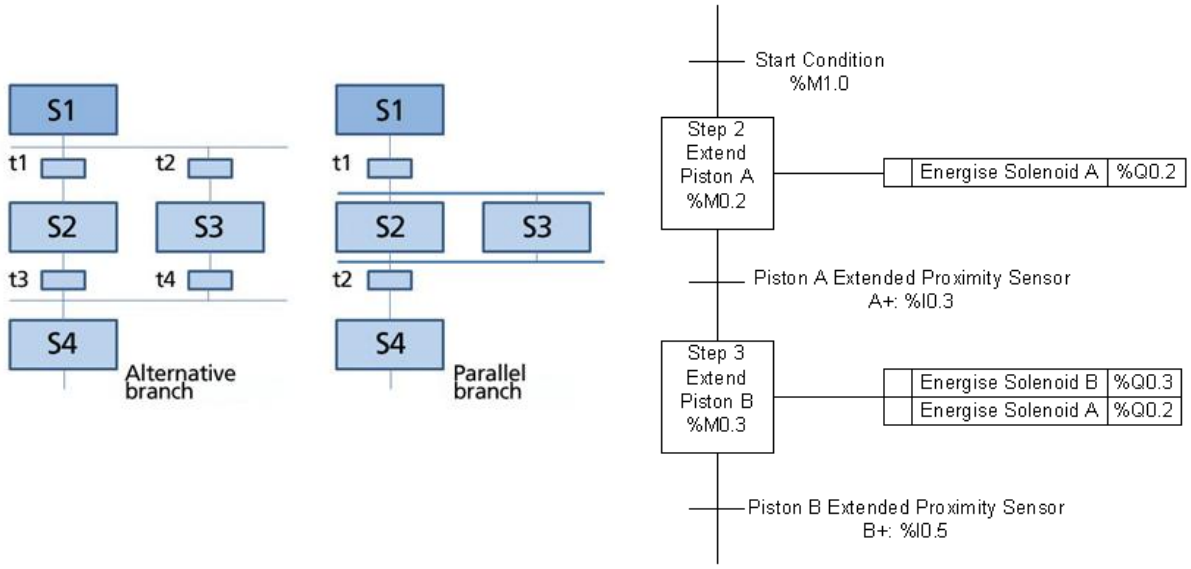
Jednotlivé bloky jsou propojené pomocí linií. Tyto linie jsou orientované, to znamená, že informace se v nich šíří zleva doprava. Díky tomuto uspořádání se snadno sledují cesty mezi jednotlivými vstupy a výstupy. Další výhodnou FBD je jeho bloková struktura, jelikož jednotlivé funkční bloky lze kopírovat a používat vícekrát v jednom programu. [18]



Obr. 2-4 Příklad Function Block Diagramu [17]

2.2.4 Sequential function charts

Sequential function charts (SFC) je další grafický programovací jazyk sloužící k programování PLC. Hlavní části SFC jsou kroky a přechody mezi nimi. Kroky představují jednotlivé funkce v systému. Přechody pak ukazují, jak probíhá změna z jednoho kroku, případně stavu, do druhého. SFC také obsahuje standardní programovací techniky jako je zpětná vazba a větvení. Tím, že se jedná o grafický programovací jazyk, si lze složitější úlohy rozdělit na dílčí celky, které se pak snáze řeší. Díky tomu jsou grafické jazyky obecně přehlednější než jazyky čistě textové. [21]



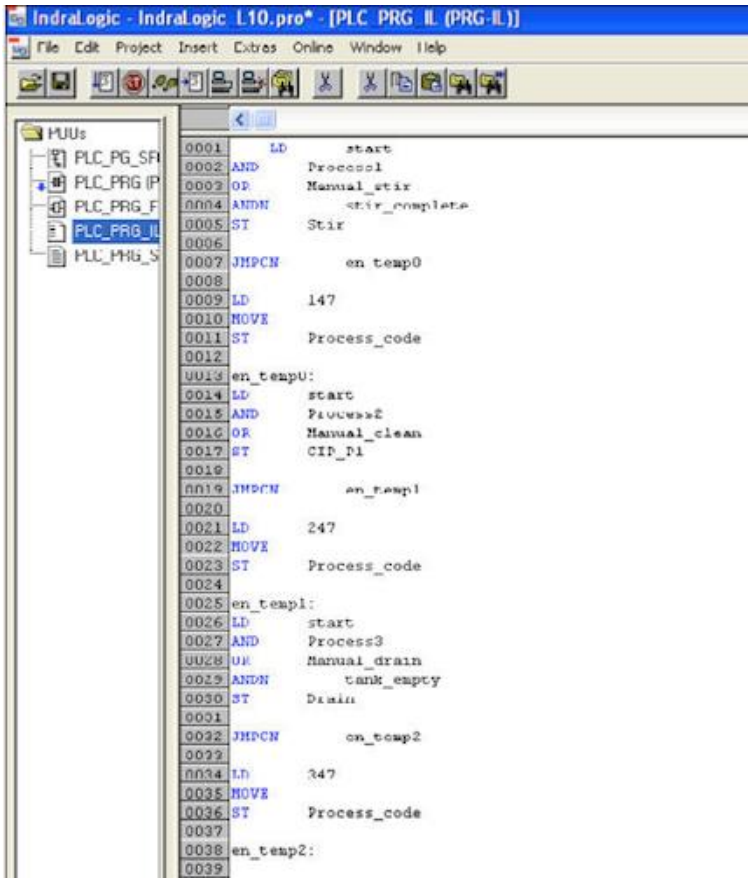
Obr. 2-5 Paralelní a sériové zapojení SFC [20] a část SFC [21]

2.2.5 Instruction list

Instruction list (IL) je textový programovací jazyk, to znamená, že pro programování používá text, čísla a interpunkci stejně jako přirozený jazyk. Spolu s ladder diagramem to byl jeden z prvních programovacích jazyků pro PLC. Jedná se o nižší programovací jazyk, což znamená, že je velmi blízký strojovému kódu. [19]

V instruction list je každá instrukce nebo strojový příkaz psán na nový řádek. Instrukce se skládají z operátorů, operandů a modifikátorů. Pro označení operátorů se používají mnemotechnické pomůcky např. A pro AND, MOV pro move (posuň).

Jednou z výhod nižších programovacích jazyků je jejich rychlost a efektivita, a to hlavně v porovnání s grafickými programovacími jazyky. Další výhodou je, že zabírají méně paměti. Z těchto důvodů se instruction list používá hlavně v aplikacích, které vyžadují velmi rychlé zpracování dat jako např. polohový regulátor (control loops). Nevýhodou IL je jeho náchylnost k run-time errors, což způsobuje nekonečné smyčky (zacyklení programu) nebo nedovolené aritmetické operace.[17]



Obr. 2-6 Příklad Instruction listu [19]

2.2.6 Structured text

Structured text (ST) je stejně jako instruction list, textový programovací jazyk. ST je vyšší programovací jazyk, který umožňuje snadnější programování, a s jeho pomocí mohou být některé části počítačového systému (např. správa paměti) zautomatizovány nebo úplně skryty. ST má blokovou strukturu, která syntaxí připomíná Pascal, podle kterého byl vytvořen. To znamená, že ST obsahuje cykly, větvení a operátory. Je to standardizovaný programovací jazyk pomocí kterého lze programovat všechna PLC.

Jednou z výhod textových programovacích jazyků je, že složitější programy zabírají méně místa v paměti. Další výhodou je možnost provádět kombinace programovacích jazyků PLC. Například je možné vytvořit blok ve FBD do něhož je vepsána funkce ve formátu ST. Jejich nevýhodou v porovnání s grafickými programovacími jazyky, je složitější ladění (debuggování).[22]

FindGreaterThan

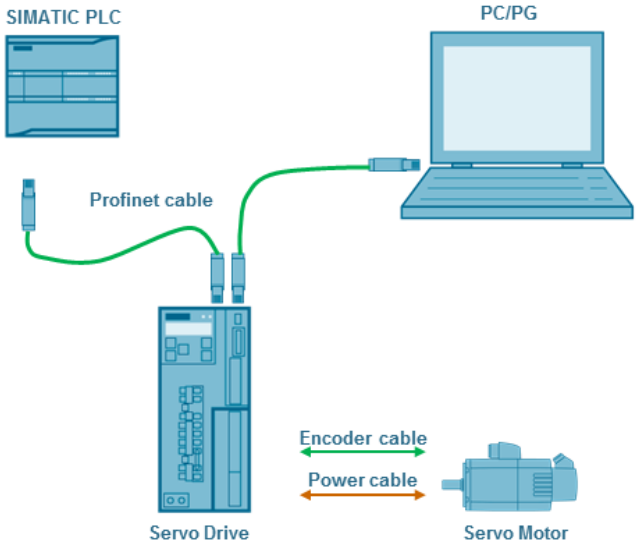
```
1  FUNCTION FindGreaterThan : INT
2  VAR_INPUT
3      Threshold : REAL;
4  END_VAR
5  VAR_IN_OUT
6      Values : ARRAY[1..1000] OF REAL;
7  END_VAR
8  VAR
9      Found : BOOL;
10     Index : INT;
11 END_VAR
12
```

```
1  Found := FALSE;
2  Index := 1;
3  WHILE NOT(Found) AND Index <= 1000 DO
4      IF Values[Index] > Threshold THEN
5          Found := TRUE;
6      ELSE
7          Index := Index + 1;
8      END_IF;
9  END_WHILE;
10
11 IF Found THEN
12     FindGreaterThan := Index;
13 ELSE
14     FindGreaterThan := 0;
15 END_IF;
16
```

Obr. 2-7 Příklad Structured listu [22]

KAPITOLA 3: POUŽITÝ HARDWARE

Moje sestava se skládá z pěti komponentů. PLC S7 1200 s CPU 1215c dc/dc/dc, které slouží pro řízení této soupravy. Dále je zde měnič SINAMICS V90 PN, který spolu se servomotorem SIMOTICS S-1FL6, uvádí výtah do pohybu. Důležitou součástí je model pro simulaci výtahu s vyznačenými patry, který mi umožňuje si řízení vytvořené v TIA portálu otestovat. Celý tento popsaný komplet je ovládán pomocí HMI panelu KTP700 Basic, ve kterém jsou vytvořena tlačítka pro řízení výtahu.



Obr. 3-1Řídící a výkonová část mé soupravy [24]

3.1 Servomotor SIMOTICS S-1FL6

Servomotor SIMOTICS S-1FL6 je synchronní servomotor s permanentními magnety. Jedná se další vývoj motorů po bezkartáčových DC servomotorech. Tyto synchronní motory se vyznačují proudem se sinusovým průběhem a velmi vyhlazeným hnacím momentem. [1] Tomuto typu motoru stačí jen jeden enkodér pro měření pozice a rychlosti. Tím se liší od bezkartáčových DC servomotorů, které potřebují enkodéry dva. [2]

3.2 Měnič SINAMICS V90 PN

Jedná se o verzi měniče SINAMICS V90, který má již integrované rozhraní pro PROFINET. Toto rozhraní umožnuje snadné propojení měniče s automatizačním systémem, a zároveň nabízí automatickou optimalizaci systému. [23]

3.3 PLC S7 1200

PLC S7 1200 spolu s centrální procesorovou jednotkou CPU 1215c dc/dc/dc umožňuje automatizaci řízení, a má zastoupení v mnoha aplikacích. Jednotka CPU má také zabudovaný PROFINET port, který umožnuje komunikaci všech zařízení v sestavě přes PROFINET.

3.4 HMI panel KTP700 Basic

Jedná se o HMI panel základní řady Basic, který slouží pro řešení jednodušších úloh. KTP700 označuje key touch panel o velikosti 7 palců. Tento panel umožnuje vytvoření online grafu sledovaných veličin, a zároveň umožňuje hlášení alarmů v podobě textových zpráv. Dále je zde možnost archivace až deseti sledovaných veličin, např. aktuální pozice, a alarmů na USB. Panel nabízí práci jak se zabudovanými funkční tlačítky, tak s virtuálními tlačítky na obrazovce panelu.

KAPITOLA 4: POUŽITÁ KOMUNIKACE – FIELDBUS

Fieldbus (Sběrnice) je označení pro rodinu komunikačních protokolů pro řízení průmyslových aplikací v reálném čase. Dříve měly automatizační systémy jedno centrální řízení, do kterého vedly všechny signály a hodnoty. Nyní je řízení tzv. decentralizované, a „inteligentní část" automatizačního systému už netvoří pouze centrální řídící jednotka. Současná používaná metoda řízení má „Inteligentní část" mimo centrální řídící jednotku, a je umístěna přímo uvnitř stroje např. decentralizované pohony (měniče). Souběžně s vývojem tohoto decentralizovaného řízení vzrostl požadovaný počet signálů, které bylo nutné přenést. Nyní se pro přenos dat používá fieldbus, který umožňuje větší tok informací něž klasické jednovodičové vedení. [2]

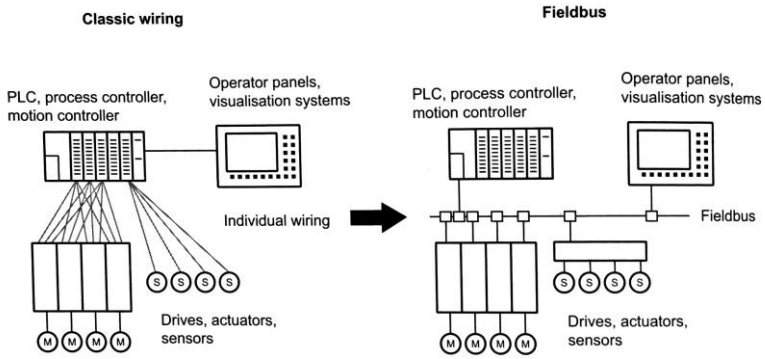


Figure 9.1 From single-conductor wiring to fieldbus

Obr. 4-1 Srovnání spojení pomocí jednoho vodiče a Filedbusu [2]

4.1 PROFINET I/O

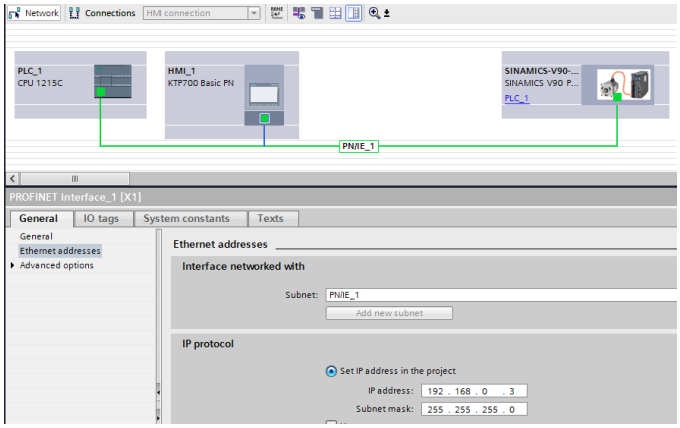
PROFINET vychází z ethernetu, neprovádí však změny již existujících funkcí, pouze přidává další funkce do protokolů na všech úrovních. Existují dvě verze PROFINETU: PROFINET CBA a PROFINET I/O. Pro komunikaci mé sestavy používám PROFINET I/O. Výhodou této verze PROFINETU je to, že pracuje přímo na úrovní ethernetu. Díky tomu má až 100x rychlejší přenos než klasické sběrnice (Fieldbusy). Dále díky celosvětové dostupnosti komponentů ethernetu, jako jsou kabely, konektory atd., dochází k standardizaci hardwaru pro komunikaci. Tím, že ale pracuje na úrovni sběrnice, je nutné větší zabezpečení, například pomocí firewallu. PROFINET I/O umožňuje velmi rychlou komunikaci v reálném čase, a proto se hodí k mé sestavě se servomotorem, který slouží pro řízení výtahu na základě jeho současné polohy. [2]

KAPITOLA 5: TIA PORTÁL

Totally Integrated Automation Portal (TIA Portal) lze volně přeložit jako úplně sjednocený automatizační portál. To znamená, že TIA Portál propojuje hardware, software a servis. TIA Portál nabízí řadu funkcí, jako jsou například simulace, které jsem využila ve své práci.[25]

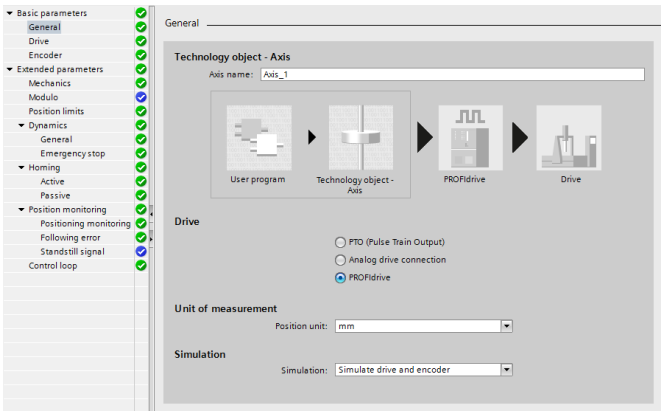
5.1 Vytváření projektu – základní nastavení osy

Pro svou práci na projektu jsem použila TIA Portál V16, ke kterému jsem měla přístup ve virtuálním prostředí programu VMware. Projekt jsem zahájila vložením servomotoru SIMOTICS S-1FL6 a měniče SINAMICS V90. Komponenty jsem podle jejich sériových čísel vybrala z katalogu v TIA Portálu. Poté jsem stejným způsobem přidala PLC a HMI panel. Zásadním krokem bylo nastavit rozdílné IP adresy pro všechny komponenty. V dalším kroku jsem propojila všechna uvedená zařízení pomocí PROFINETu.



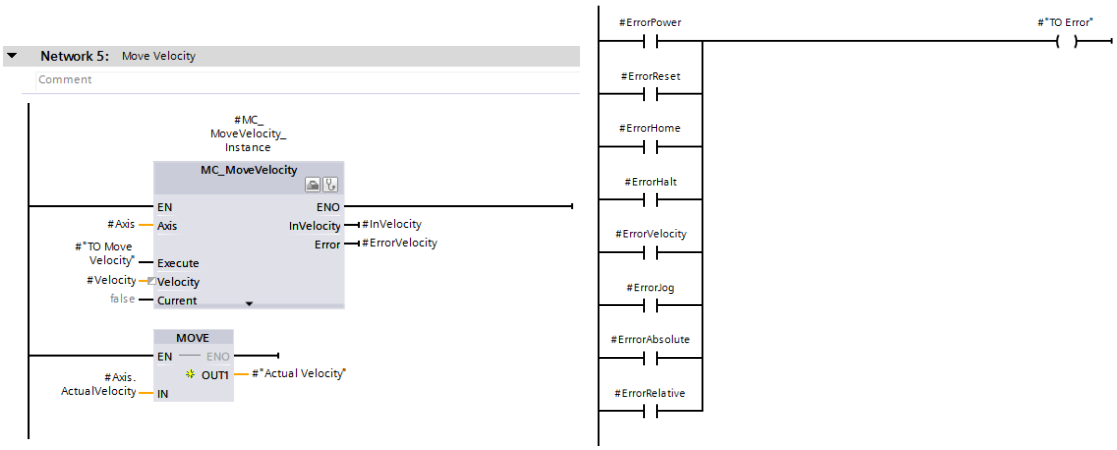
Obr. 5-1 Propojení hardwaru pomocí PROFINETu

V další fázi je možné přidat technologické objekty (TO). V mé práci jsem v tomto kroku přidala a poté nakonfigurovala synchronní osu. Nejprve jsem ose přidělila připravený pohon a enkodér. Pro komunikaci osy s pohonem a enkodérem byl automaticky zvolen standartní telegram 3. Konfigurace se neobejde bez nastavení mechaniky a dynamiky osy, hlavně pak stoupání 8 mm/s, maximální rychlost 150 mm/s, rampy 1 s, nouzové zastavení 0.1 s. Pro nastavení polohovacích limitů jsem vytvořila tzv. PLC tagy. Pokračovala jsem nastavením funkce homing. Rozlišujeme homing aktivní a pasivní, já jsem použila pouze aktivní, který jsem nastavila přes digitální vstup.



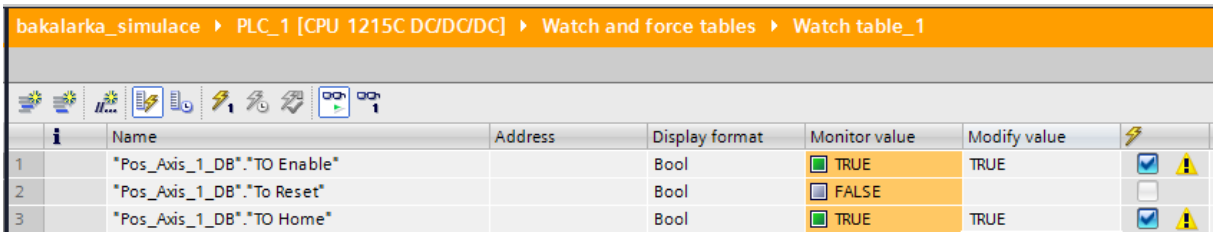
Obr. 5-2 Nastavení konfigurace polohovací osy

Po nastavení hardwaru bylo potřeba vytvořit program k dané ose pomocí Funkčního bloku. Nastavila jsem bloky: Power, Reset, Home a Halt, a dále bloky řídící pohyb: Move Velocity, Jog, Move Absolute a Move Relative. Ke každému bloku je nutné nastavit chybový výstup, a poté vytvořit žebříčkové schéma, které obsahuje všechny errory a spojuje je s proměnnou TO Error.



Obr. 5-3 Programové bloky v TIA portálu

Po dokončení nastavení a vytvoření programu jsem vše zkompilovala a nahrála do PLC. Nakonec jsem funkčnost osy vyzkoušela ovládáním pomocí Watch tabulky, kde jsem nejprve spustila pohon, který osu pohání pomocí proměnné TO Enable. Při prvním spuštění je nutné osu zkalibrovat, neboli použít homing, který osu dopraví do výchozí pozice. Pomocí změny proměnné TO Move Velocity z FALSE na TRUE jsem osu, a tím i výtah rozpohybovala, a pomocí proměnné TO Halt zastavila. Pro změnu směru a rychlosti pohybu jsem použila proměnné Velocity.



	Name	Address	Display format	Monitor value	Modify value
1	"Pos_Axis_1_DB"."TO Enable"		Bool	TRUE	TRUE
2	"Pos_Axis_1_DB"."To Reset"		Bool	FALSE	
3	"Pos_Axis_1_DB"."TO Home"		Bool	TRUE	TRUE

Obr. 5-4 Watch tabulka pro sledování a změnu hodnot proměnných

KAPITOLA 6: NASTAVENÍ HMI PANELU

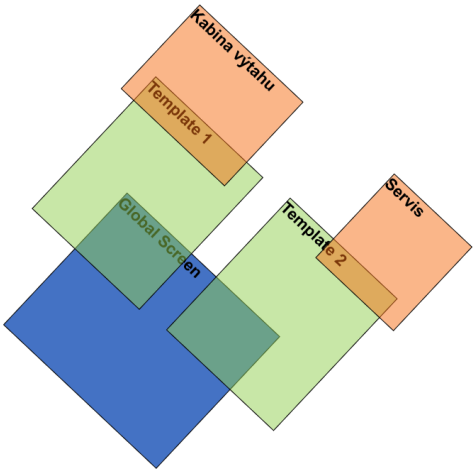
Důležitým nástrojem pro vytváření vizualizačních projektů v TIA portálu je WinCC. Panely typu Basic slouží hlavně pro sledování a ovládání řízené technologie. Pro ukládání a archivaci dat jsou používány panely vyšších řad. Panely typu Basic umožňují použití pouze jednoho komunikačního rozhraní. Také nemají paměťovou kartu, to znamená, že sledované veličiny lze ukládat pouze do paměti nebo pomocí USB na flash disk. Tyto panely lze pořídit za nízkou cenu, a pro mou práci nepředstavovaly jejich nevýhody žádné omezení. Pro můj projekt s Basic HMI panelem jsem použila vývojové prostředí WinCC Basic, které je součástí TIA Portálu.

6.1 Napojení HMI panelu na PLC

Pro napojení HMI panelu na PLC je nutné mít již vytvořený projekt v TIA portálu. Propojení funguje pomocí ethernetu. Pro snazší nastavení HMI panelu jsem využila pomůcku „Start device wizard“, pomocí které jsem vytvořila základní kostru vizualizačního projektu. Jako první jsem v HMI device wizard přiřadila konkrétní PLC k mému HMI panelu. Dále lze nastavit například vzhled obrazovek, okna pro alarmy a jiné. V další části je možné definovat strukturu obrazovek, jejich počet lze v průběhu vytváření projektu měnit. Ve své práci jsem využila strukturu s jednou centrální „root“ obrazovkou, ze které vycházejí ostatní obrazovky. Poté systém automaticky vytvořil tlačítka pro přechod mezi obrazovkami. Dále můžeme přidat i systémové obrazovky jako je správa uživatelů, systémové informace, přepínání jazyků. Nakonec se v HMI device wizard nastaví základní tlačítka například „domů“ nebo „změna jazyků“ pomocí přetažení. Provedla jsem pouze nastavení tlačítka „domů“. „Start device wizard“ jsem ukončila tlačítkem „dokončit“, a vytvořila se část projektu pro HMI panel. Poté jsem nastavila stejnou IP adresu jako má HMI panel v projektu v TIA portálu na skutečném HMI panelu, který jsem měla k dispozici. Nastavení IP adresy jsem provedla v záložce Síťová rozhraní.

6.2 Obrazovka

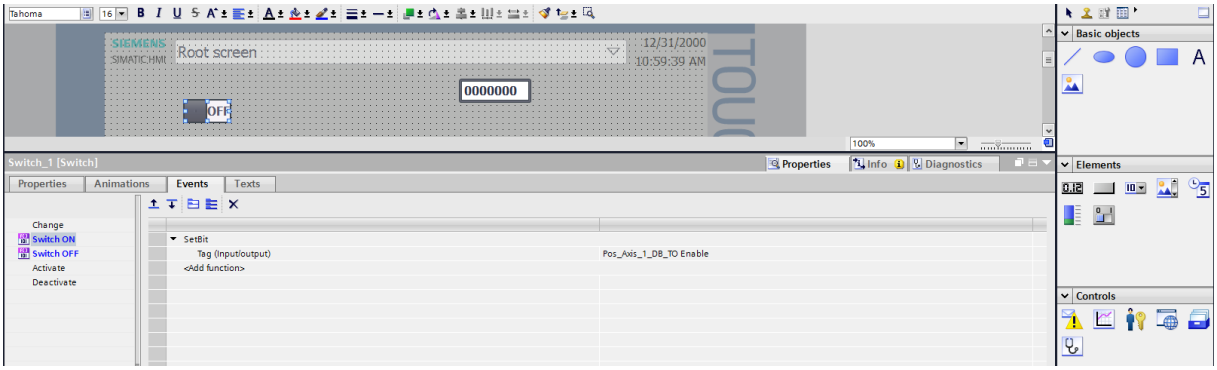
Když si představíme obrazovky jako vrstvy, tak jako první máme Globální obrazovku, kde jsou pouze alarmy a alarm indikátory. Nad touto vrstvou máme takzvané Template. Jedná se o obrazovku, která má v sobě objekty společné pro nějakou skupinu obrazovek. Další vrstva nad Template jsou už konkrétní obrazovky. Jedná se například o situaci, kde máme nějakou Globální obrazovku, v níž bude Template 1, 2 a 3. V Template 1 bude Obrazovka A, B a C, a v Template 2 bude Obrazovka D a E. Pro můj projekt byl dostačující Template 1, kde je obrazovka pro řízení výtahu z kabiny a Template 2, kde je obrazovka externího servisu výtahu.



Obr. 6-1 Uspořádání obrazovek v HMI panelu

To, k jakému Template daná obrazovka náleží, jsem nastavila v Properties (Vlastnostech). Dále jsem nastavila přechod z jedné obrazovky na druhou. Toto je možné provést několika způsoby. Jedním ze způsobů je vytvořit si na jedné obrazovce přechod na obrazovku druhou pomocí vybraného tlačítka. Zde je přes záložku Events (Události) možné nastavit způsob spouštění tlačítka, například klik nebo přidržení, a přiřadit aktivní obrazovku danému tlačítku. Já jsem využila jednodušší možnost. Ze složky Screens (Obrazovky), kde jsou všechny mé obrazovky vypsané, jsem přetáhla zástupce obrazovky Kabina výtahu do obrazovky Servis, a tlačítko pro přechod mezi těmito obrazovkami se automaticky vytvořilo.

Dále přichází krok nastavení konkrétních tlačítek. Pro jejich vytvoření se využívá panelu Toolbox, ve kterém jsou různé objekty, klasická tlačítka, přepínače, stupnice, apod. Objekty z Toolboxu lze využít i pro grafické znázornění sledovaných veličin. U každého objektu lze nastavit pomocí jeho záložek konkrétní vlastnosti jako jsou např. barva, způsob ovládání, atd. V záložce Vlastnosti se upravuje například barva, text, apod. V záložce Animace určujeme například změnu viditelnosti a vzhledu tlačítek při určitých podmínkách. Velmi důležitou záložkou jsou Události, ve které stanovujeme, která proměnná je spojena s daným objektem, a podle jaké funkce se bude případně měnit. Jako jeden z objektů jsem použila I/O field, který zobrazí aktuální polohu kabiny výtahu pro kontrolu absolutního polohování výtahu.



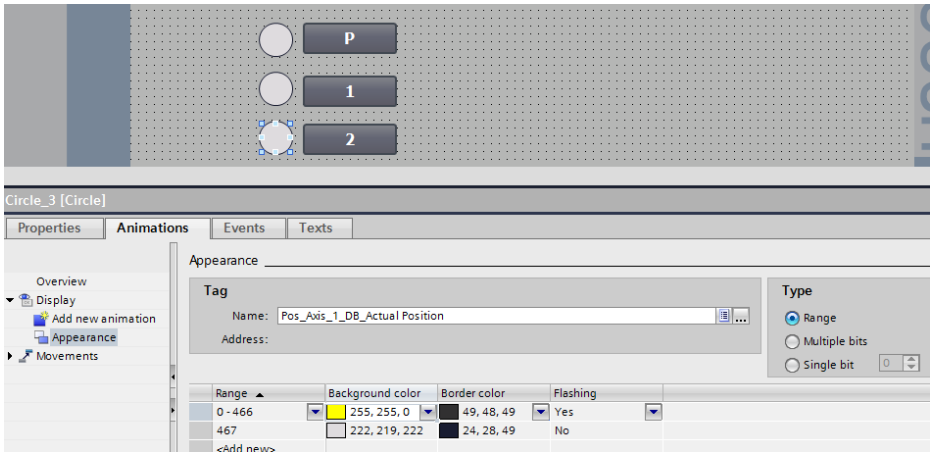
Obr. 6-2 Vytváření objektů na obrazovce

Animace objektů jsou velmi důležité, protože nám pomohou signalizovat stav našich proměnných. Pokud chceme měnit viditelnost objektů, musíme zvolit v závislosti na jaké proměnné se bude měnit. Když máme proměnnou například typu int, tak zvolíme interval pro který bude tlačítko viditelné, případně neviditelné. A pokud máme proměnnou typu bool, tak stačí zvolit, že například při hodnotě 1 je tlačítko viditelné, a při 0 je tlačítko neviditelné.

Dalším druhem animace je změna vzhledu objektu při změně hodnoty proměnné. Nejjednodušší způsob změny vzhledu objektu je změna jeho barvy. Toto se dá skvěle využít právě k signalizaci proměnných typu bool. Například objekt bude zelený, pokud je hodnota proměnné 1, to znamená, pokud je TRUE. Toho lze v mém projektu využít pro proměnné TO Enable, TO Home a TO Move Absolute, které je možné sledovat na servisní obrazovce, a signalizují, že je výtah připraven, zkalibrován, a zda je v pohybu.

Dále jsem změnu barvy využila pro signalizaci příjezdu výtahu do konkrétního patra na hlavní obrazovce výtahu. Tuto animaci pro signalizaci lze vytvořit více způsoby. Můj první návrh této animace můžeme vidět na Obr. 6-3. Zde jsem zvolila kruh vedle tlačítka pro patro, který mění barvu v závislosti na poloze výtahu. To znamená, pokud výtah jede například do druhého patra, tak kruh vedle tlačítka pro druhé patro bliká žlutě, dokud výtah do druhého patra nedojede. Pro můj druhý návrh signalizace jsem změnu barvy řídila pomocí části programu ve Funkci výtah.

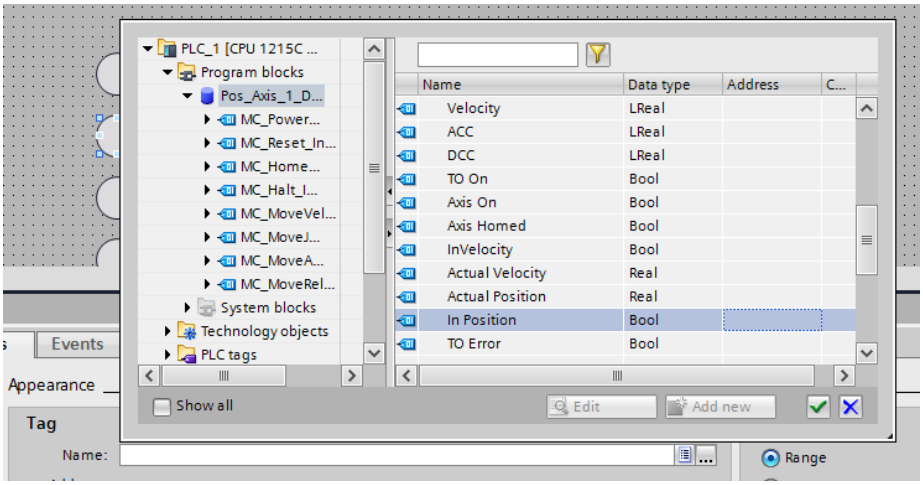
Tento postup jsem popsala v Kapitola 8: a můžeme ho vidět na Obr. 8-11 a Obr. 8-12. Ve svém projektu jsem upřednostnila druhý návrh, protože signalizace se více blížila skutečnosti.



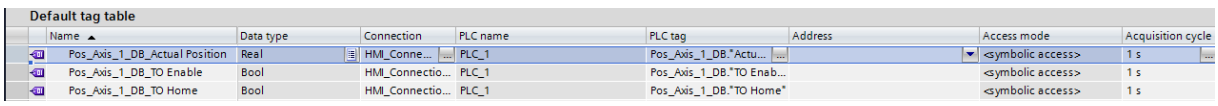
Obr. 6-3 Signalizace v HMI panelu

6.3 HMI Tag

Jako proměnné pro HMI panel se mohou využívat PLC Tagy a proměnné z datových bloků, které mám již v TIA Portálu vytvořené. Je nutné je při jejich nastavení zviditelnit a zpřístupnit pro práci v HMI panelu. Poté již HMI tagy nemusíme programovat, ale vytvoří se sami. V mém případě jsem si vzala pro signalizaci proměnnou Actual Position z programového bloku polohovací osy, a HMI tag se poté sám vytvořil.



Obr. 6-4 Práce s proměnnou z datového bloku v HMI panelu



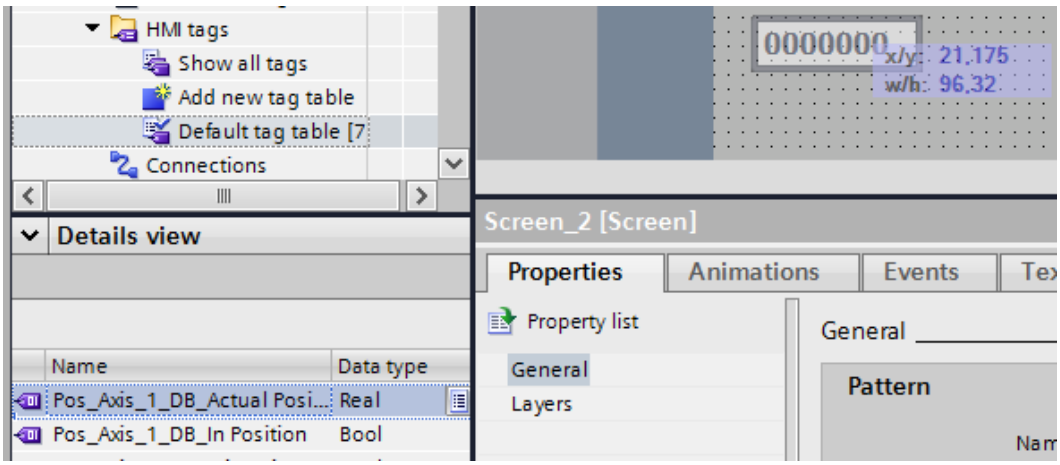
Obr. 6-5 Automatické vytvoření HMI tagu

Pro HMI tag je poté možné nastavit způsob jeho čtení. Ve vlastnostech daného tagu je záložka nastavení, kde si lze vybrat ze tří způsobů čtení proměnné. Ve výchozím nastavení je čtení periodické, které načítá proměnnou pokud jsme na obrazovce, kde je tato proměnná použita. Můžeme si dále nastavit periodu čtení proměnné. Další možností je neustálé periodické čtení proměnné, u kterého nezáleží, která obrazovka je aktivní. Posledním způsobem čtení proměnné je na zadaný požadavek. U tohoto způsobu načítání musíme použít nějakou systémovou funkci, abychom proměnnou načetli. Rychlost čtení není dobré stanovovat pod jednu vteřinu, neboť poté dochází ke zpomalení programu.

6.4 Vytváření objektů

Na obrazovce lze vytvářet různé objekty pomocí Toolboxu jako například tlačítko, graf, apod. V mém projektu jsem použila objekty I/O field, tlačítka, obrázek s volitelnou grafikou a textové pole. Pro správné fungování tlačítek je nutné nastavit jejich parametry v jejich jednotlivých záložkách. Nejdůležitější nastavení tlačítka je v záložce Události, kde tlačítku přiřadíme konkrétní proměnou. Dále zde můžeme nastavit způsob spouštění tlačítka jako je kliknutí, přidržení, uvolnění, aktivování, deaktivování a změna. Při spouštění kliknutím se proměnná změní podle zvolené funkce/povelu. Při použití funkce SetTag můžeme proměnnou nastavit na konkrétní hodnotu. Pokud ale máme proměnnou typu bool, tak je jednodušší použít funkci SetBit, která nám proměnnou nastaví na 1 neboli na TRUE. Další důležitou funkcí je ResetBit, která nám booleovskou proměnnou vrátí do její původní hodnoty. Ve svém programu používám funkci SetBit pro tlačítka pater. Další zajímavé funkce jsou pak IncreaseTag a DecreaseTag, pomocí kterých mohu proměnnou navyšovat nebo zmenšovat. Velmi univerzální je pak funkce LinearScaling, do které je možné vepsat jakoukoliv lineární funkci, podle které chci proměnnou měnit. Dalšími objekty mohou být sloupcové grafy, Text listy a Graphic listy.

Důležitým objektem v mém projektu je I/O field, který slouží pro sledování žádané hodnoty. Konkrétně pro sledování aktuální polohy výtahu. Tu jsem vytvořila přetažením HMI Tagu Actual_Position na Obrazovku. Mohu tak sledovat zda aktuální poloha odpovídá příslušnému patru. Objekt je možné použít ve třech módech. Nejběžnější je Output, který umožnuje proměnnou z PLC jen sledovat. Dále Input/Output, kterým lze veličinu sledovat a také ji měnit ze strany HMI panelu. A jako poslední je Input, který umožňuje pouze veličinu měnit ze strany HMI panelu, ale pokud se mění na straně PLC, tak se na obrazovce nezobrazuje. Já jsem pro moji proměnnou zvolila nastavení Output, protože pozici potřebuji jen sledovat nikoliv ji měnit.



Obr. 6-6 Vytvoření objektu pro čtení požadované hodnoty

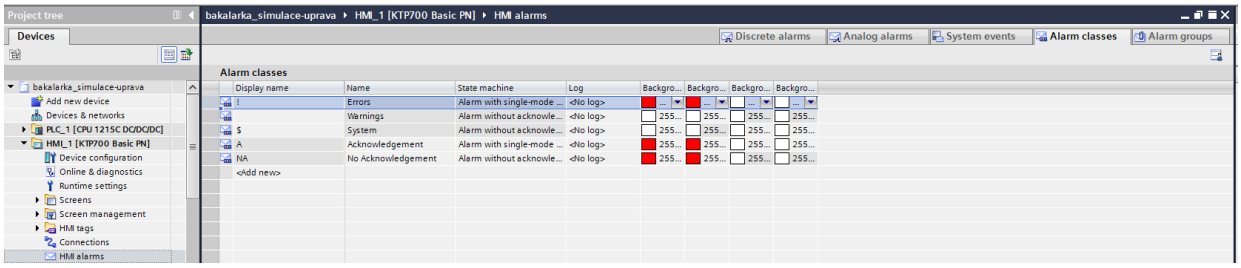
6.5 Alarmy

Alarmy jsou textové zprávy upozorňující na nějakou vzniklou situaci v technologii, například překročení nosnosti kabiny určené výrobcem. Nejedná se o automatickou záležitost, je nutné je v programu pro řízení technologie vytvořit. Mojí technologií byl model pro simulaci výtahu. Tato technologie přes karty digitálních vstupů posílá informaci o svém stavu do PLC. Pokud dojde k nějaké poruše této technologie, je potřeba uvědomit obsluhu. Vytvoření alarmu je důležité, protože pouze varovná signalizace poruchy by se nám nikam neuložila, a neexistovala by zpětná kontrola kolikrát, a při jakých podmínkách k dané poruše došlo. Objekty sloužící pro zobrazování alarmů jsou Alarm View a Alarm Window. Alarmy lze posílat přes SMTP server mimo řídící systém ve formě emailu pro vzdálenou údržbu.

Existuje více variant zasílání alarmů. Hlavními typy alarmů jsou diskrétní, analogové a telegramové. Diskrétní alarmy fungují tak, že je v PLC uložená proměnná, a každý bit této proměnné představuje nějaký alarm. HMI panel tuto proměnnou sleduje, a pokud je nějaký bit v 1, tak aktivuje alarm odpovídající příslušnému bitu. Analogový alarm sleduje, zda byla překročena nějaká mez, nebo zda sledovaná veličina pod danou mez klesla. Tento typ je běžný například pro sledování teploty. U těchto dvou typů alarmů vzniká časová značka až v HMI panelu, čímž je způsobena časová prodleva několik milisekund. Pro aplikace, které vyžadují větší přesnost je třeba zvolit telegramové alarmy, které jsou celé vytvářené v PLC. U těchto alarmů PLC spolu s textovou zprávou zašle do HMI panelu i přesný čas udání poruchy.

Alarmová hlášení se skládají ze sedmi komponentů. Jako první je důležitost alarmu, pomocí které rozlišujeme, zda se jedná o varování, pohotovost nebo jiný stupeň poruchy. Na Obr. 6-7 je vidět, jak jsou rozděleny alarmy podle důležitosti do tříd, jako je například již zmiňované varování (Warning). Dalšími položkami alarmového hlášení jsou číslo alarmu, čas a datum alarmu a status. Ve statusu je zaznamenáno, zda v tento čas došlo ke vzniku poruchy, jejímu potvrzení obsluhou, nebo k jejímu vyřešení. Písmeno I značí vzniklou poruchu, O značí vyřešenou poruchu, A poruchu potvrzenou obsluhou. Potvrzení obsluhou není vždy vyžadováno. Další položkou alarmového hlášení je text alarmu, který upozorňuje k jaké konkrétní poruše došlo. Důležitost alarmů můžeme také rozlišit pomocí barev, například červená oznamuje nejzávažnější poruchy, oranžová středně závažné poruchy, a bílá, pokud se jedná pouze o varování.

Při nastavení alarmů se jako první stanovuje důležitost alarmů, neboli jejich zařazení do tříd. Toto se provádí v HMI alarm v záložce Alarm Class. Několik tříd je již vytvořeno ve výchozím nastavení alarmů, ale lze přidat i vlastní třídy a současné třídy upravovat. Zde lze zvolit požadavek na potvrzení obsluhy (Acknowledgment). Také se zde stanovuje uložiště alarmu. Pokud uložiště není zvoleno, tak se alarmy ukládají automaticky do paměti panelu, kde je místo na 256 posledních alarmů.



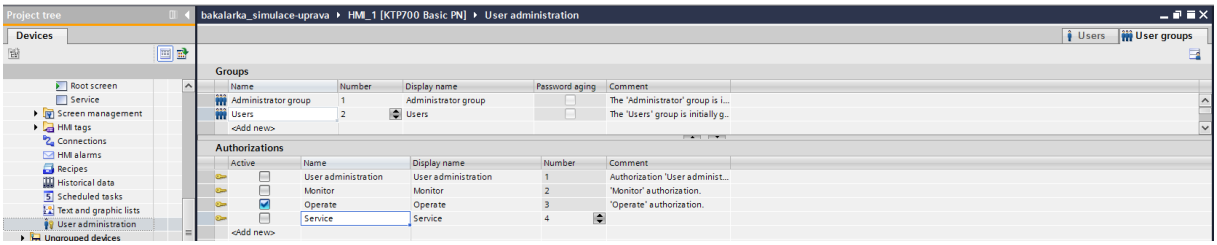
Obr. 6-7 Nastavení tříd alarmů v HMI panelu

Další nastavení se poté provádí v Runtime setting, kde lze zvolit například zda využijeme výchozího nastavení barev tříd alarmů, čas zobrazení alarmu na Globální obrazovce, a zda budeme požadovat systémovou diagnostiku z PLC. Dále je nutné vytvořit Trigger Tag, což bude proměnná, jejíž bity budou odpovídat konkrétním poruchám. V HMI alarm v záložce Discreate alarms (diskrétní alarmy) zvolíme pro daný bit našeho Trigger tagu text alarmu a jeho třídu. Do textu alarmu je možné vložit hodnotu konkrétní proměnné, nebo Text list. Podobně můžeme nastavit i analogové alarmy v záložce Analog alarms.

Zobrazení alarmů se provádí dvěma způsoby, a závisí na typu obrazovky, na které bude alarm zobrazen. První možností je Alarm view, který se používá pro zobrazení na klasických obrazovkách. Druhou možností je Alarm window, který je možný použít jen na Globální obrazovce. Alarm window se zobrazí pokaždé, když zaznamená nový alarm. Alarm view se zobrazuje pouze na aktivní obrazovce. V Alarm view lze určit jaké třídy alarmů budou zobrazeny. V obou způsobech lze zvolit zdroj alarmu, tím může být současný stav alarmů nebo alarmy uložené v paměti.

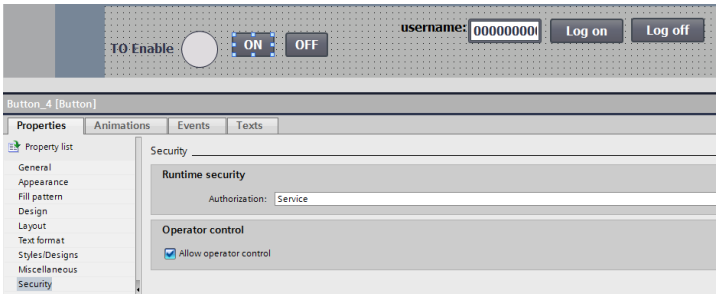
6.6 Uživatelé a přístupová práva

V HMI panelu lze vytvořit skupiny uživatelů, například programátor, servis a operátor výroby. Každé této skupině přísluší jiná přístupová práva. Pracovník obsluhy má přístupová práva omezená jen na funkce, které jsou nutné pro vykonávanou práci. Pracovník servisu má nastavena vyšší přístupová práva, která mu umožňují provádět opravy zařízení. Nejvyšší a zároveň neomezená práva má programátor. V této práci jsem udělila práva pro skupinu Servis, která jsou definovaná ve stejnojmenné autorizaci. U této skupiny jsem určila konkrétního uživatele, který bude mít přístupová práva pro výkon servisu.



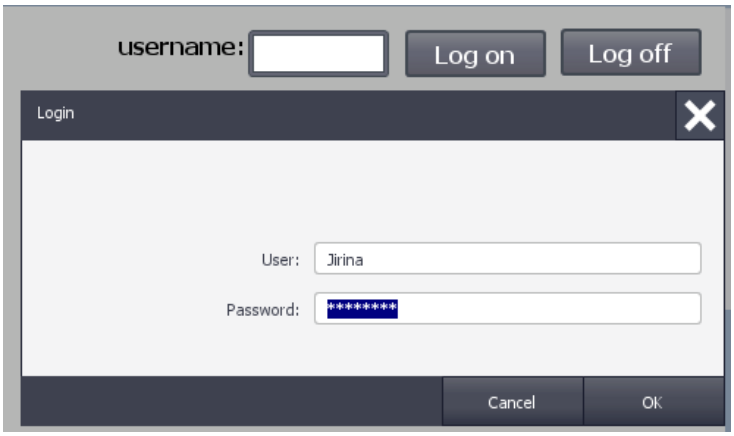
Obr. 6-8 Nastavení autorizace pro konkrétní skupiny uživatelů

Takto lze připravit bezpečnostní autorizace, které ale sami o sobě nic nedělají. Je třeba autorizace přiřadit konkrétním objektům, jako jsou například tlačítka. Tudíž je třeba stanovit, jaká autorizace je potřeba pro ovládání konkrétních objektů. V mé práci jsou všechna tlačítka na obrazovce Servis zpřístupněna pouze osobám s autorizací Servis. Jedná se o tlačítka a objekty, pomocí kterých lze výtah řídit manuálně, a tím zjistit, zda se jedná o chybu v technologii nebo v programu. Jednotlivým objektům na této obrazovce jsem ve Vlastnostech v Security přiřadila autorizaci Servis.



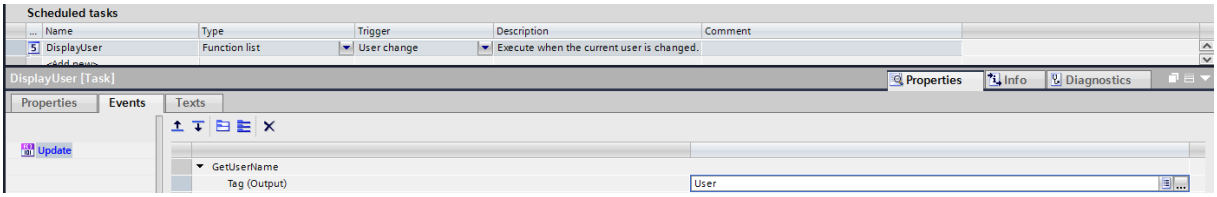
Obr. 6-9 Nastavení bezpečnostní autorizace

Na obrazovku lze přidat i tlačítka pro přihlášení a odhlášení jednotlivých uživatelů. V Událostech pak těmto tlačítkům přiřadíme funkce ShowLogonDialog a Logoff. Po zmáčknutí tlačítka se bude zobrazovat klasická tabulka pro zapsání uživatelského jména a hesla.



Obr. 6-10 Přihlášení uživatelů k HMI panelu

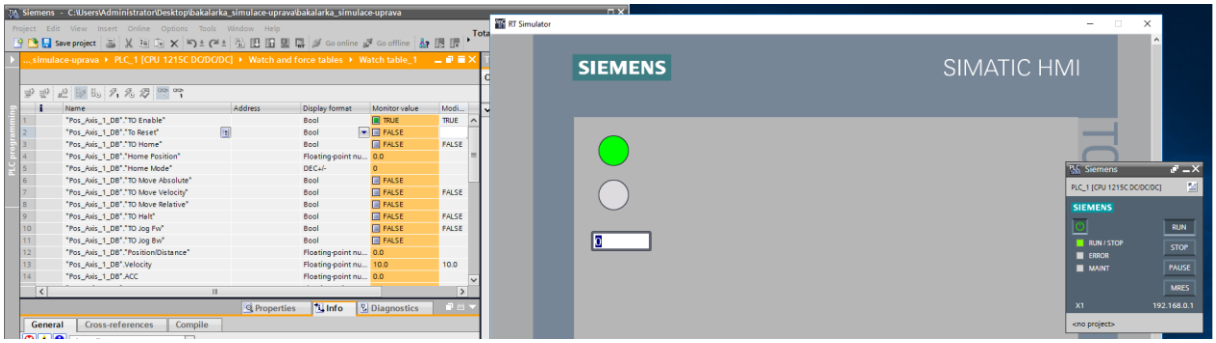
Pro zobrazení přihlášeného uživatele jsem nejprve vytvořila proměnnou User datového typu WString v HMI Tags. Poté jsem si v Scheduled tasks vytvořila úlohu na pozadí, kde budu využívat funkce GetUserName. Pro zobrazení na obrazovce jsem použila I/O field, kde jsem přiřadila proměnnou User.



Obr. 6-11 Nastavení zobrazení přihlášeného uživatele

KAPITOLA 7: SIMULACE

Velkou výhodou TIA portálu je možnost simulace jak PLC, tak HMI panelu. Tyto simulace spolu dokáží komunikovat, a díky tomu umožňují replikovat situaci, která by vznikla v reálném hardwaru. Pro úspěšný průběh simulace je nutné nejprve spustit simulaci PLC a poté nahrát program do našeho virtuálního PLC. Následně se zapíná simulace HMI panelu. U mého PLC S7 1200 bylo nejprve třeba smazat programové bloky MC-Servo a MC-Interpolar, zapnout v nastavení osy simulaci pohonu a enkodéru, a poté programové bloky MC-Servo a MC-Interpolar znovu nahrát. Tento problém vznikl v důsledku toho, že jsem při nastavování hardwaru nezvolila možnost simulace pohonu a enkodéru před vložením těchto programových bloků. Tím nevznikla pro tyto programové bloky simulace, a nemohlo dojít k jejich nahrání do virtuálního PLC.

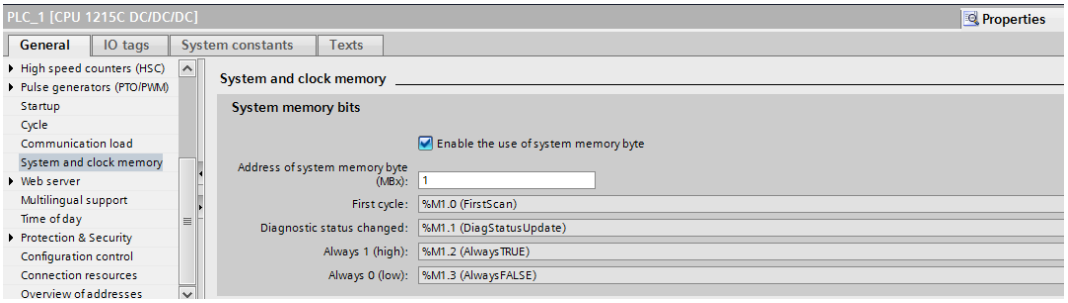


Obr. 7-1 Simulace PLC a HMI panelu

Simulace mohou být i mnohem složitější. Například lze simulovat robota, který je namodelován v NX Mechatronics Concept Designer podle výkresu vytvořeném například v AutoCADu. Jedná se o velmi výkonné systémy, a jejich programování je už velmi náročné.

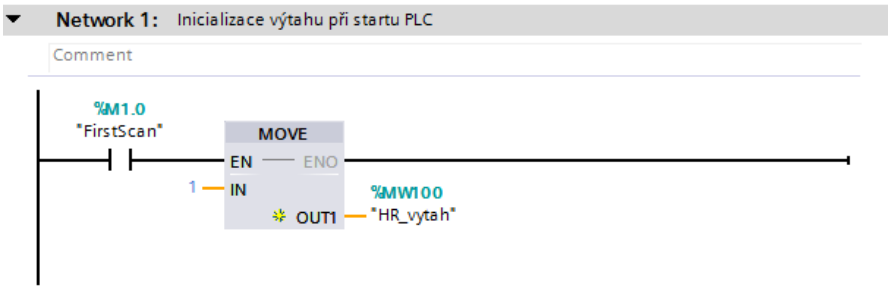
KAPITOLA 8: PROGRAM V TIA PORTÁLU

Mým úkolem bylo vytvořit program pro řízení výtahu pomocí tlačítek na displeji HMI panelu. Program jsem vytvářela pomocí Funkce, ve které jsem navázala na program pro otáčení jedné osy. Kvůli funkcím FirstScan a AlwaysTRUE jsem nejprve v PLC povolila použití System memory bits. FirstScan probíhá pouze při prvním průchodu programu. AlwaysTRUE jsem využila pro bloky, které chci vždy v jedničce. Ve stejné záložce jsou i Clock memory bits, které využívám pro signalizaci jízdy výtahu do patra.



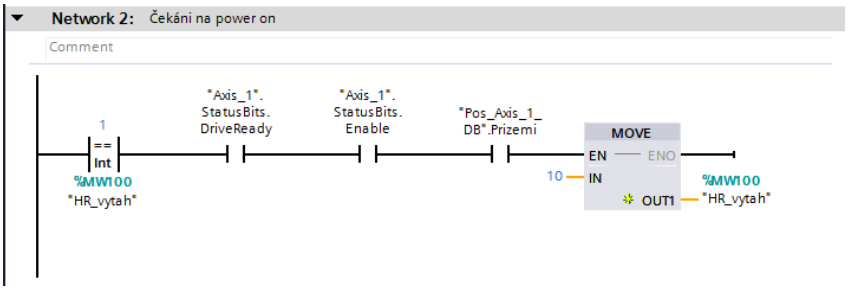
Obr. 8-1 Povolení System a clock memory

Na Obr. 8-2 je vidět, že pokud proběhne úspěšně FirstScan, mohu přiřadit mé pomocné proměnné HRvýtah hodnotu 1. Přiřazení provedu pomocí funkce Move.



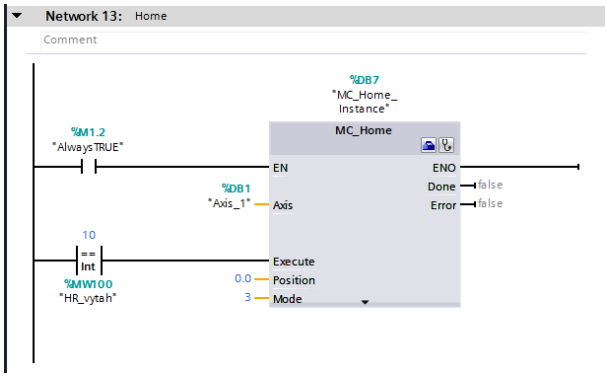
Obr. 8-2 Inicializace výtahu pro start PLC

Dále jsem stanovila čtyři podmínky pro rozjezd výtahu. Musela být splněna podmínka z předchozího Network. Tutu podmínku jsem vytvořila pomocí funkce Compare, která slouží k porovnání „1“ a hodnoty v proměnné HRvýtah. Pro další dvě podmínky jsem využila statusů osy pro ošetření chyb (DriveReady) a Enable. Nakonec jsem vložila tlačítko pro přízemí z HMI panelu. Pokud budou tyto podmínky splněny, pomocné proměnné HRvýtah přiřadím hodnotu 10.



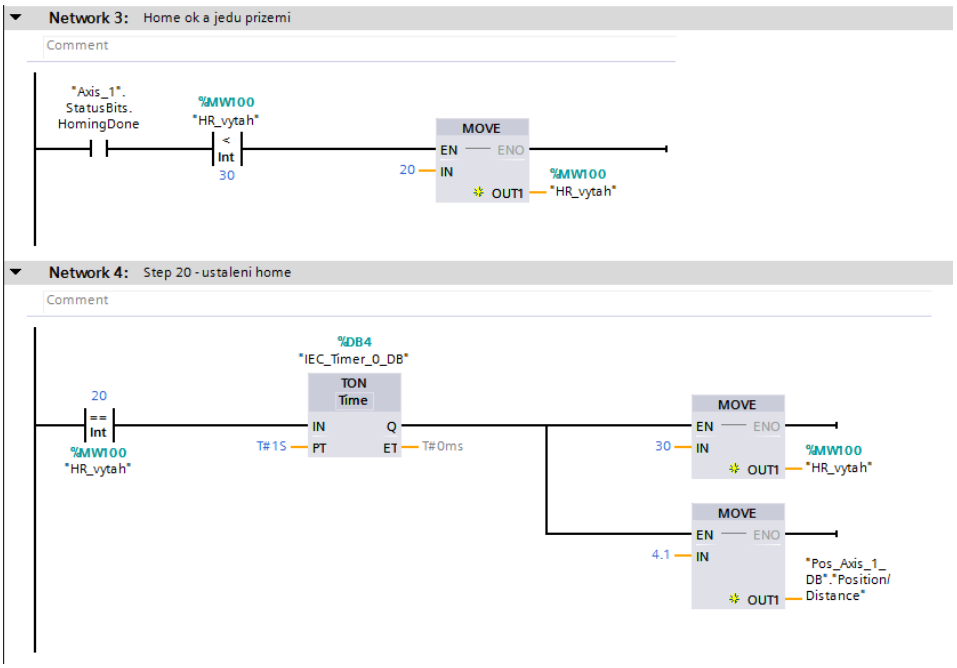
Obr. 8-3 Ošetření podmínek nutných pro roztočení osy

Pro zprovoznění výtahu musí úspěšně proběhnout homing. Po splnění podmínek z Network 2, program přejde do Network 13, kde provede blok Home, jak je zobrazeno na Obr. 8-4. Tedy při prvním stisknutí tlačítka přízemí bude proveden homing.



Obr. 8-4 Homing

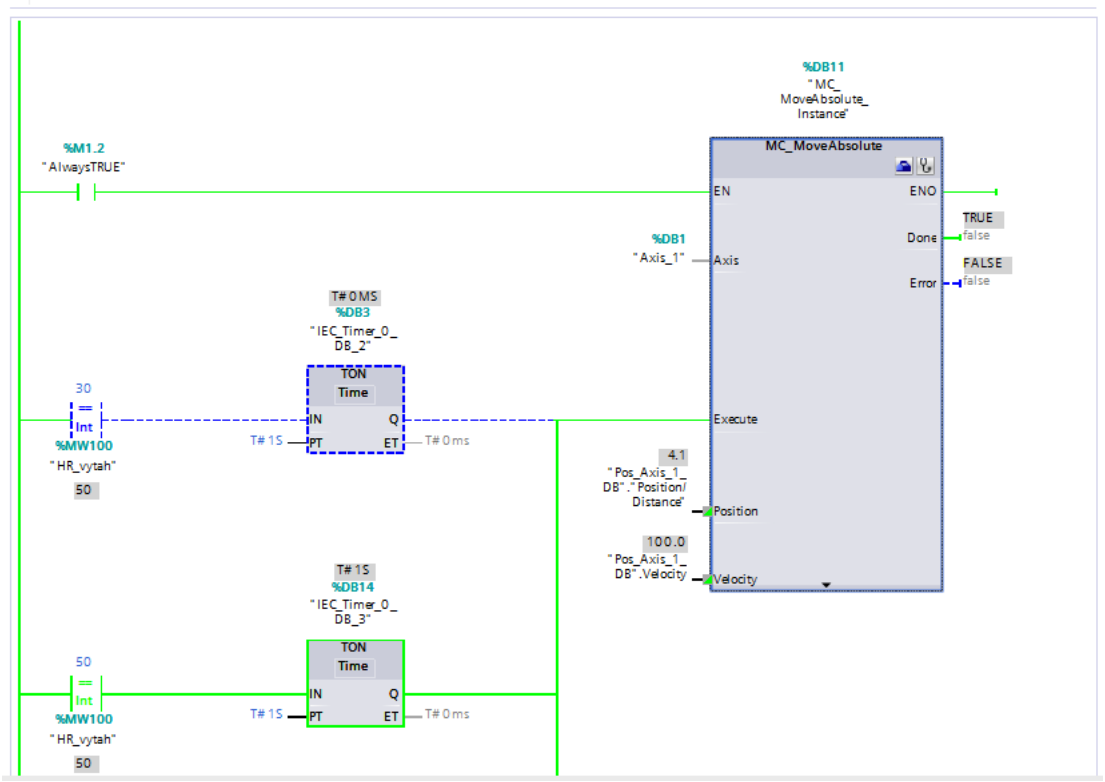
Po úspěšném provedení bloku Home je důležité přesunout výtah do polohy pro přízemí. Jak je vidět na Obr. 8-5, opět jsem využila proměnné HRvýtah a funkce TON Time. Důvodem bylo, aby program počkal 1 vteřinu než se po provedení homing dopraví do polohy 4.1. Jízdu výtahu do dané polohy jsem zajistila přiřazením polohy 4.1 do proměnné Position/Distance.



Obr. 8-5 Příprava výtahu pro řízení

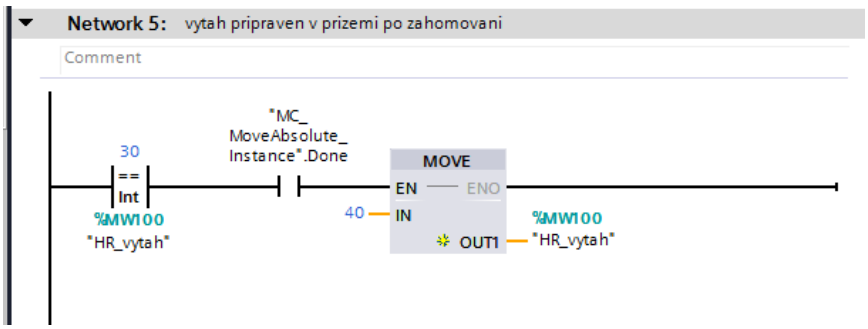
Pokud se hodnota HRvýtah rovná 30, tak se vykoná blok MoveAbsolute. Vstupu Position se přiřadí hodnota 4.1, kterou jsou přidělila do proměnné, která je na tomto vstupu. Na Obr. 8-6 je zobrazena logika přiřazení jednotlivých pater pomocí paralelního zapojení, neboli pomocí logické funkce OR. To znamená, že se vždy provede vykonání pohybu, do jaké polohy záleží však na podmínce, která je k dané hodnotě přiřazená. Důležité bylo přidat zpoždění TON 1 vteřinu

k tlačítku pro každé patro. Takhle se bloku MoveAbsolute nejprve přiřadí poloha odpovídající danému patru a teprve po 1 vteřině se provede pohyb. Například pro hodnotu proměnné HRvýtah rovnou 60 se přiřadí pozice 236.308, kabina počká jednu vteřinu, a výtah pojede do prvního patra, které má tuto pozici.



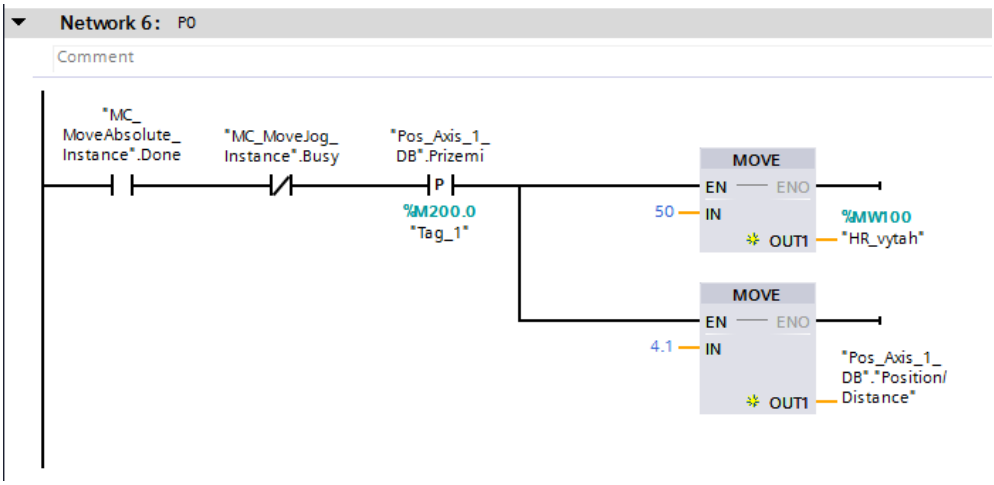
Obr. 8-6 Logika OR pro řízení jízdy do jednotlivých pater

Na Obr. 8-7 je vidět, že po vykonání bloku MoveAbsolute je výtah již připraven k provozu. Pokud je totiž v proměnné HRvýtah uložena hodnota 40, tak vím, že byly splněny všechny podmínky v předchozích Network. Nejprve jsem použila blok Compare HRvýtah roven 40 na začátku Network pro každé patro, ale při testování programu na hardwaru na katedře jsem zjistila, že tento postup nefunguje. Proto jsem jako podmínku stejnou pro všechna patra zvolila MoveAbsolute_Instance.Done. To znamená, že pokud výtah pojede např. do třetího patra, tak příkaz pro jízdu např. do přízemí lze zadat až po dojezdu výtahové kabiny do třetího patra.



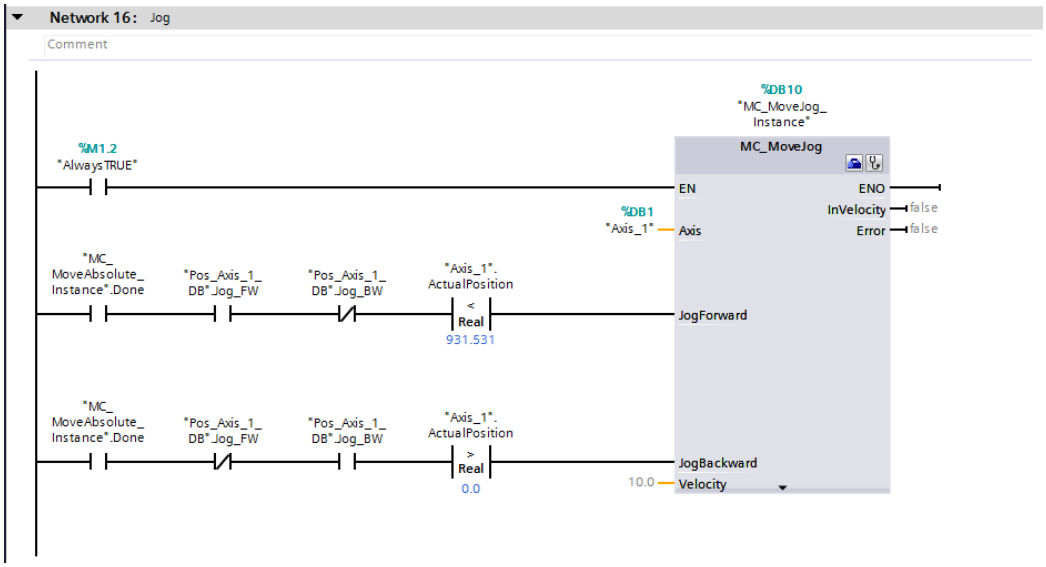
Obr. 8-7 Výtah je připraven k provozu

Jak již bylo výše uvedeno, Network pro jednotlivá patra musí obsahovat podmínku MoveAbsolute_Instance.Done. Dále jsem využila rozpínacího kontaktu, který slouží k zablokování automatického řízení pomocí tlačítek při ručním ovládání. Kontakt pro tlačítko přízemí s písmenem P uvnitř znamená, že reaguje na náběžnou hranu. Tímto druhem kontaktu u tlačítka zamezím problémům vzniklým při přidržení tlačítka.



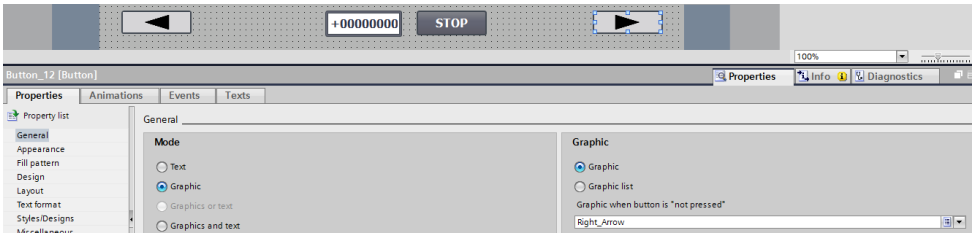
Obr. 8-8 Výtah pojede do přízemí

Pro ruční ovládání výtahu slouží funkce Jog Fw a Jog Bw. I zde musí být splněna podmínka MoveAbsolute_Instance.Done jako u Network pro patra. To znamená, že pokud zavolám výtah pomocí tlačítka např. do druhého patra, tak výtah musí nejprve do druhého patra dojet, a až poté ho mohu ovládat ručně. Důležité je, aby nebyl možný pohyb dopředu a dozadu zároveň. Pro ošetření tohoto problému slouží rozpínací kontakty. Na Obr. 8-9 je vidět, že pokud jede výtah dopředu, tak se rozepne kontakt u JogBw, a nebude možné výtahem jet dozadu. Také je na obrázku vidět opět blok Compare, v tomto případě jsem použila jeho jinou formu. U JogForward jsem blok Compare použila pro vytvoření horního limitu pro pohyb dopředu. Hodnotu tohoto limitu jsem zvolila polohu pro čtvrté patro. Spodní limit funkce JogBackward je roven přízemí.



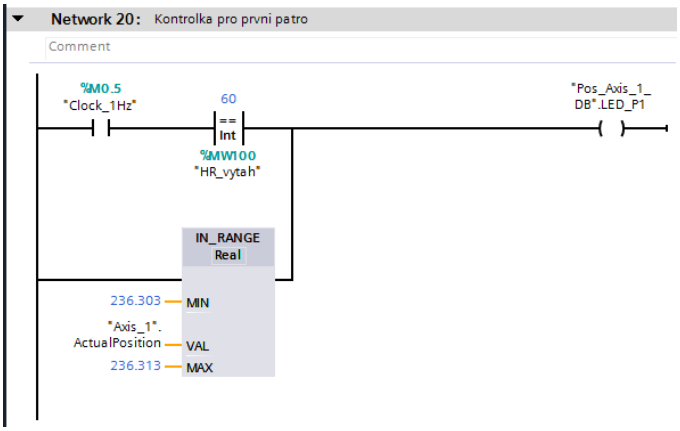
Obr. 8-9 Ruční řízení výtahu pomocí Jog Fw a Jog Bw

Pro ruční řízení jsem vytvořila v HMI panelu vizualizaci pomocí tlačítek, stejně jako pro tlačítka pater, s tím rozdílem, že jsem upravila grafiku tlačítka do vzhledu šipky. Na Obr. 8-10 je vidět, jak snadno lze měnit vzhled tlačítek.



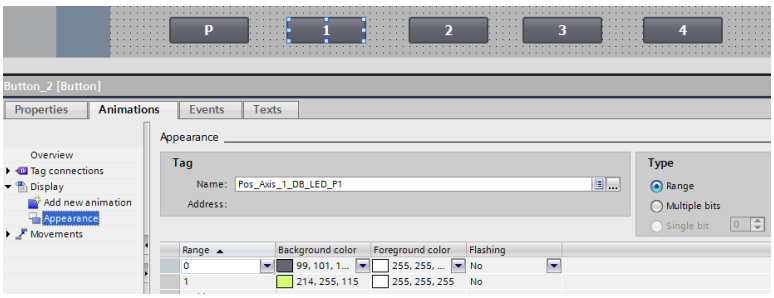
Obr. 8-10 Změna grafiky tlačítka

Pro signalizaci příjezdu výtahu jsem vytvořila proměnné LED P0, P1, P2, P3 a P4 typu Bool v datovém bloku Pos_Axis_1 DB. Frekvenci blikání jsem zvolila pomocí Clock na 1 Hz. Na Obr. 8-11 je zobrazeno, že se hodnota HRvýtah musí rovnat 60, což jsou podmínky proto, aby výtah přijel do prvního patra. Paralelně k těmto podmínkám je zapojen blok IN_Range, který sleduje, zda je aktuální poloha kabiny výtahu v daných mezích.



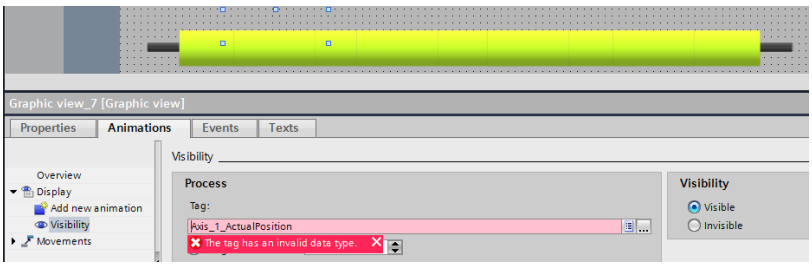
Obr. 8-11 Kontrolka pro první patro

Proměnnou LED_P1 jsem použila pro signalizaci příjezdu výtahu do prvního patra. Na Obr. 8-12 je zobrazeno použití proměnné LED_P1 pro změnu vzhledu tlačítka pro první patro.



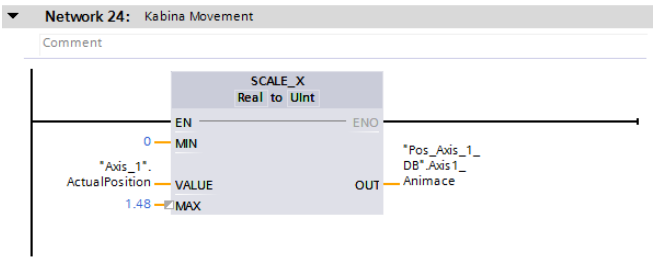
Obr. 8-12 Využití kontrolky v HMI panelu

Část programu je věnována převodu proměnné typu Real na UInt. Ve vizualizaci výtahu jsem se setkala s problémem v animaci kabiny. Zamýšlela jem vytvořit animaci výtahu pomocí proměnné ActualPosition, díky které by se moje vizualizace výtahu hýbala jako skutečný výtah. Ale na Obr. 8-13 je zobrazeno upozornění TIA portálu na fakt, že nelze pro animaci využít proměnnou datového typu Real.



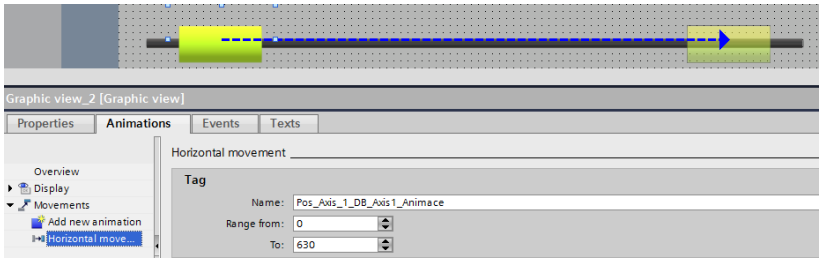
Obr. 8-13 Chybové hlášení špatného datového typu

Vyhovující řešení jsem nalezla v manuálu od společnosti Siemens. Zde [3] je uvedeno, že pro převody proměnné na jiný datový typ se využívají bloky SCALE_ X a NORM_X. Já jsem ve svém programu využila SCALE_X, jak je zřejmé z Obr. 8-14. Vytvořila jsem si proměnnou Axis1_Animace datového typu UInt, a pomocí bloku SCALE_X jsem převedla proměnnou ActualPosition do proměnné Axis1_Animace.



Obr. 8-14 Změna datového typu pomocí SCALE_X

Animaci pro pohyb výtahu lze vytvořit více způsoby. První návrh vizualizace jízdy výtahu jsem vytvořila pomocí Display – Visibility. Tato vizualizace spočívá ve změně viditelnosti kabiny při změně polohy. V tomto druhu vizualizace připomínal pohyb výtahu teleportaci. Při řešení problému s proměnnou ActualPostion v Display – Visibility, jsem nalezla také další způsoby animací ve vzdělávacích videích na webu. Zde jsem se dozvěděla, jak pomocí funkce Movements a proměnné Axis1_Aminace mohu jednoduchým přetažením kabiny vytvořit horizontální pohyb výtahu.

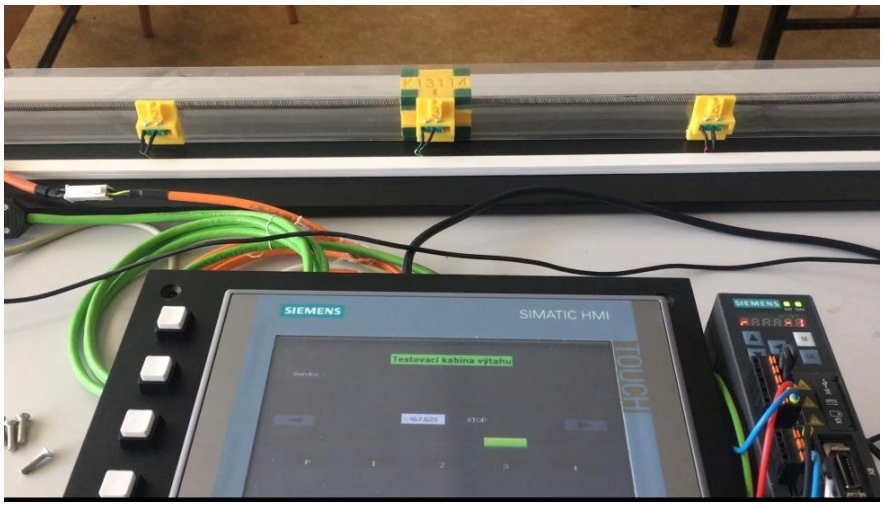


Obr. 8-15 Použití horizontálního pohybu v animaci

Konečná vizualizace mého projektu je na Obr. 8-16 a Obr. 8-17. Na jedné obrazovce je tady umožněno jak automatické ovládání pomocí tlačítek pro patra, tak ruční ovládání pomocí šipek. Pro kontrolu jsem vložila I/O field, který zobrazuje aktuální polohu výtahu. Nakonec je zde STOP tlačítko, kterým lze v případě poruchy výtah okamžitě zastavit.

Obr. 8-16 Vizualizace HMI panelu

V závěrečné fázi jsem program otestovala na modelu výtahu, který je na Katedře elektrických pohonů a trakce. Na Obr. 8-17 je vidět vizualizace na HMI panelu.



Obr. 8-17 Testování programu na soupravě na Katedře elektrických pohonů a trakce

ZÁVĚR

Práce na tomto projektu pro mne byla velkou výzvou, a zároveň příležitostí vyzkoušet si programování PLC. Na začátku mé práce jsem se potýkala s výběrem vhodného hardwaru počítače, který by splňoval požadavky pro vývojové prostředí SIMATIC TIA Portál. Pro úspěšné vypracování mé práce jsem si musela navýšit velikost RAM a SSD disku. Toto vylepšení mi umožnilo propojení mého počítače s hardwarem pro model výtahu. Samotné programovaní bylo velmi přehledné, a do jisté míry mi připomínalo reléovou logiku. Na druhou stranu tu bylo pro mne spoustu neznámých funkcí, které sice v podobě bloků působily přehledně, ale zvládnutí jejich ovládání bylo časově náročné. Poté, co jsem se v prostředí naučila pracovat, jsem postupně začala sestavovat program pro řízení mého modelu výtahu. Na modelu na Katedře elektrických pohonů a trakce jsem zjistila polohu jednotlivých pater tak, že jsem pomocí ručního řízení výtahem přijela do daného patra, a z proměnné ActualPostion jsem si zapsala danou pozici. V TIA Portálu jsem pak pomocí absolutního pozicování vytvořila program pro jízdu výtahu do jednotlivých pater. Program jsem nejprve vytvořila pomocí funkčního bloku. Bohužel jsem při tvorbě programu pro jízdu do konkrétních pater zjistila, že nemohu využít proměnné ActualPostion, jelikož byla ve funkčním bloku definována pouze jako input. Celý program jsem tedy zkopírovala do Funkce výtah, kde jsem ho také upravila. Struktura programu ve funkci se od funkčního bloku příliš neliší. Stejně jako ve funkčním bloku jsou i zde uspořádány příkazy do jednotlivých řádků, na kterých jsou umístěné bloky a kontakty. Na rozdíl od funkčního bloku lze ve funkci snáze vyhledávat. Při návrhu vizualizace HMI panelu jsem mohla být více kreativní, což mě při práci na spíše technickém projektu překvapilo. Modely pro osu a kabinu výtahu jsem navrhla v 3D malování a poté je vložila do HMI panelu. Nejprve jsem se domnívala, že práce na programu pro PLC a vizualizace v HMI panelu jsou naprosto oddělené. Při vytvoření animace jízdy výtahu jsem ale narazila na problém s proměnnou Actual position. Pro animace lze totiž využít proměnné jen datového typu Int a UInt. Musela jsem se tedy vrátit k programu, a vytvořit ve funkci blok Scale_X pro převod proměnné datového typu Real do proměnné datového typu UInt. Díky tomu jsem zjistila jak je práce na programu a vizualizaci provázána, a že ten, kdo vytváří vizualizaci v HMI panelu, musí mít i znalosti programování PLC. Tato práce mi tedy přinesla spoustu znalostí a zkušeností, a určitě se budu v této oblasti dále vzdělávat.

LITERATURA

[1] Ing. Pavel Kobrle, Ph.D. a prof. Ing. Jiří Pavelka, DrSc. *Elektrické pohony a jejich řízení,* 3. přepracované vydání. Praha: České vyssoké učení technické v Praze 2016

[2] Jens Weidauer, Richard Messer *Electrical drives,* Public Erlangen, 2014

[3] SCE Training Curriculum, Siemens AG, 2016

[4] Durry B. The Control Techniques Drives and Controls Handbook 2nd ed., leT, 2009

[5] Historie výtahů ve světě © 2020 [cit. 21.dubna 2021] Dostupné z: https://www.starevytahy.cz/historie/celkova.php

[6] ARCHIMEDES OF SYRACUSE © Copyright 2021 Greece.com [cit. 21. dubna 2021] Dostupné z: https://www.greece.com/info/people/Archimedes_of_Syracuse/

[7] Who invented the elevator? © 2021 A&E Television Networks, LLC. [cit. 9. května 2021] Dostupné z: https://www.history.com/news/who-invented-the-elevator

[8] Elevator Wikipedia ® is a registered trademark of the Wikimedia Foundation, Inc. [cit. 21. dubna 2021] Dostupné z: https://en.wikipedia.org/wiki/Elevator

[9] Thomas Savery British engineer and inventor ©2021 Encyclopædia Britannica, Inc. Dostupné z: https://www.britannica.com/technology/engineering

[10] Elisha Otis Wikipedia ® is a registered trademark of the Wikimedia Foundation, Inc. [cit. 21. dubna 2021] Dostupné z: https://en.wikipedia.org/wiki/Elisha_Otis

[11] Siemens is celebrating the 200th birthday of the company's founder © Siemens 1996 – 2021 . [cit. 21. dubna 2021] Dostupné z: https://press.siemens.com/global/en/feature/siemens-celebrating-200th-birthday-companys-founder

[12] HISTORIE VÝTAHU © Copyright NetDesign, s.r.o. [cit. 21. dubna 2021] Dostupné z: https://www.i-vytahy.cz/cs/sekce/nove-vytahy/historie-vytahu.html

[13] Podklady z předmětu Návrh komponent elektrického pohonu [cit. 2. dubna 2021] Dostupné z: https://moodle.fel.cvut.cz/course/view.php?id=5644

[14] Reléové obvody (c) mylms.cz 2006 – 2021 [cit. 15. listopadu 2020] Dostupné z: https://www.mylms.cz/releove-obvody/

[15] WHAT IS LADDER LOGIC? © 2021 RealPars B.V. All rights reserved. [cit. 15. listopadu 2020] Dostupné z: https://realpars.com/ladder-logic/

[16] Ladder Logic Programming Copyright © 2021 [cit. 15. listopadu 2020] Dostupné z: https://ladderlogicworld.com/ladder-logic-programming/

[17] PLC BASICS: THE ULTIMATE GUIDE IN 2021! © 2017-2021 PLCGurus.NET [cit. 15. listopadu 2020] Dostupné z: https://www.plcgurus.net/plc-basics/

[18] Function Block Diagram (FBD) Programming Tutorial Copyright 2021 PLC Academy [cit. 15. listopadu 2020] Dostupné z: https://www.plcacademy.com/function-block-diagram-programming/

[19] What are Instruction Lists (ILs) for PLC programming? Copyright © 2021 WTWH Media, LLC. All Rights Reserved. [cit. 15. listopadu 2021] Dostupné z: https://www.designworldonline.com/what-are-instruction-lists-ils-for-plc-programming/

[20] What are sequential function charts (SFCs) for PLCs? Copyright © 2021 [cit. 15. listopadu 2020] Dostupné z: https://www.motioncontroltips.com/sequential-function-charts-sfcs-plcs/

[21] Sequential Function Charts for All Copyright © 2005-2021 plcdev.com [cit. 15. listopadu 2020] Dostupné z: http://www.plcdev.com/sequential_function_charts_all

[22] TwinCAT 3 Tutorial: Structured Text [cit. 15. listopadu 2020] Proudly powered by WordPress Dostupné z: http://www.contactandcoil.com/twincat-3-tutorial/structured-text/

[23] SINAMICS V90 and SIMOTICS S-1FL6 Optimized servo drive solution for motion control applications dostupné z: http://www.mcs.com.tr/imagess/sinamics_v90documen_2dcrund_75135.pdf

[24] SINAMICS V: Position Control with SIMATIC S7-1200 TO and SINAMICS V90 PN (S mode) © Siemens AG 2009-2021 [cit. 20. března 2021 Dostupné z: https://support.industry.siemens.com/cs/document/109743917/sinamics-v%3A-position-control-with-simatic-s7-1200-to-and-sinamics-v90-pn-(s-mode)?dti=0&lc=en-US

[25] Totally Integrated Automation Portal siemens.com Global Website © Siemens 1996 – 2021. [cit. 2. dubna 2021] Dostupné z: https://new.siemens.com/global/en/products/automation/industry-software/automation-software/tia-portal.html

PŘÍLOHA A: SEZNAM ZKRATEK

PLC	Programmable Logic Controller
SIMATIC	Siemens and Automatic
TIA Portál	Totally Integrated Automation Portal
HMI	Human-Machine Interface
ČKD	Českomoravská Kolben-Daněk
CPU	Central Processing Unit
LD	Ladder Diagram
FBD	Function Block Diagram
SFC	Sequential Function Charts
IL	Instruction List
ST	Structured Text
DC	Direct Current
USB	Universal Serial Bus
CBA	Component Based Automation
I/O	Input/Output
TO	Technology Object
IP	Internet Protocol
SMTP	Simple Mail Transfer Protocol
MC	Motion Control
CAD	Computer-Aided Design
HRvýtah	Hlavní Řízení výtahu
TON	On Delay Timer
Fw	Forward
Bw	Backward
Pos	Positioning
DB	Data Block
LED	Light-Emitting Diode
3D	Three-Dimensional