ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

FAKULTA STROJNÍ

Ústav přístrojové a řídící techniky

Odbor automatického řízení a inženýrské informatiky

BAKALÁŘSKÁ PRÁCE

Laboratorní úloha CP Lab

2020 Jan Mohelník

Zadání

Prohlášení

Prohlašuji, že jsem předloženou bakalářskou práci zpracoval samostatně a že jsem uvedl v přiloženém seznamu veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací, vydaným ČVUT v Praze 1.7. 2009.

Nemám závažný důvod proti užití tohoto školního díla ve smyslu § 60 Zákon č.121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon). Výsledky této práce mohou být dále využity dle uvážení vedoucího této práce Ing. Pavel Trnka, Ph.D. jako jejího spoluautora. V případě publikace si přeji být uveden jako spoluautor.

Datum……

Podpis……

Poděkování

Chtěl bych tímto poděkovat všem, kteří mi při vytváření této práce pomáhali. Především děkuji své rodině, kteří se mnou měli trpělivost a podporovali mne. Dále bych chtěl zvlášť poděkovat svému dědečkovi, jenž mi vždy byl a je mým životním vzorem v mnoha ohledech. A zajisté chci také poděkovat svému vedoucímu práce Ing. Pavlu Trnkovi, Ph.D. za věnovaný čas a připomínky.

Abstrakt:

Tato práce by měla sloužit jako návod a objasnění některých částí z oblasti automatického řízení. Jelikož je však automatizace širokým pojmem, tak se specificky budu zabývat třemi oblastmi, které jsou: logické řízení, frekvenční vlastnosti systémů a uzavřený regulační obvod. Z těchto tří oblastí také zrealizuji laboratorní úlohy pro předmět Automatické řízení, kde si studenti budou moci sami vyzkoušet veškerou zde probranou teorii. To vše se bude provádět na výukovém systému CP Lab, který nabízí zmenšené části reálného průmyslu, jako jsou dopravník, vrtačka, pec a spoustu dalšího. Vybavení jak hardwarové, tak softwarové odpovídá reálnému průmyslovému řešení, díky čemuž dokážu studentům ukázat, jak přibližně vypadá práce takového průmyslového inženýra a s jakými problémy se potýká v běžné praxi.

Klíčová slova: CP LAB, PLC, FESTO, ET200SP, TIA PORTAL, HORKOVZDUŠNÝ TUNEL, PID

Abstract:

This work should serve as a guide and clarification of some parts of the field of automatic control. However, since automation is a broad concept, I will deal specifically with three areas, which are logic control, frequency characteristics of systems and closed loop control. From these three areas, I will also carry out laboratory tasks for the subject of Automatic Control, where students will be able to try out all the theory discussed here. I will do all this on the CP Lab training system, which offers scaled down parts of the real industry, such as conveyors, drills, furnaces and much more. Both hardware and software equipment correspond to a real industrial solution, thanks to which I can show students what the work of such an industrial engineer looks like and what problems he encounters in everyday practice.

KEYWORDS: CP LAB, PLC, FESTO, ET200SP, TIA PORTAL, HEAT TUNNEL, PID

Obsah

Přehled použitých symbolů a veličin

CP Lab - The Cyber Physical Lab

PLC - Programmable Logic Controller

Master/Slave – model komunikace, kdy jeden člen je nadřazen nad druhým

MES4 – řídící systém od firmy Festo

PT100 – Platinový odporový snímač teploty

RFID – rádio-frekvenční senzor pro čtení a zápis dat

DQ, DI – DQ digitální výstup, DI digitální vstup

AQ, AI – AQ analogový výstup, AI analogový vstup

IODD – soubor popisující objekt (senzor, akční člen..) pro IO Link

IO Link – standardizovaná technologie pro komunikaci se senzory a aktuátory

Profinet – průmyslová komunikační sběrnice (průmyslový ethernet)

Profibus – průmyslová komunikační sběrnice (komunikace skrze RS-485)

WinCC – program pro tvorbu HMI v platformě Tia Portál

Tia Portal – programová platforma od firmy Siemens

HMI – Human Machine Interface

Tag - symbolické jméno adresy v PLC

Kapitola 1

Úvod

Rychleji, levněji, lépe. To jsou nejčastější požadavky dnešního moderního průmyslu. Všechna jeho odvětví se však společně skloňují s jedním slovem a to **„Automatizace“**. Průmyslová automatizace je velice obsáhlá a stále vyvíjející se oblast. Jelikož však s přibývající novější technologií musíme držet krok, je třeba se stále vzdělávat a k tomuto například mohou sloužit výukové systémy CP Lab. Ty zprostředkovávají určitou formu simulace reálné situace (zařízení) v průmyslu, s nimiž se jako inženýr v praxi můžeme setkat.

Z toho důvodu jsem si zrovna zvolil toto téma, neboť výukové systémy CP Lab právě ukazují moderní technologie a jejich využití. Připraví tak člověka co možná nejlépe, než přijde do kontaktu se skutečným průmyslem. Je zde tedy možnost vidět, jak vypadá nejen celková mechanická konstrukce zařízení, což samozřejmě zahrnuje i veškerou senzoriku, řídící jednotku PLC s veškerými přídavnými moduly, měřící karty, jak je vyřešena komunikace mezi všemi těmito částmi, ale i jak je sepsán samotný kód programu, na kterém celý systém běží.

Hlavním úkolem této práce je tedy primárně připravit pro studenty skrze jejich laboratorní cvičení v předmětu **„Automatické řízení“** praktickou ukázku, jak to ve skutečném průmyslu vypadá. Abychom ale navázali i na probíranou látku, tak tato praktická cvičení budou odpovídat třem hlavním okruhům, tedy třem úlohám, jimiž za semestr student projde.

Jako první bude řešena problematika logického řízení, kde budou mít studenti za úkol vymyslet a naprogramovat algoritmus, na kterém by měla patřičná stanice fungovat. Druhým úkolem bude prověřit chování a vlastnosti stanice skrze její frekvenční charakteristiky. Posledním úkolem bude seřídit regulátor, jenž je součástí uzavřeného regulačního obvodu a bude řídit některý z akčních členů stanice.

Stanic CP Lab je několik typů a každá disponuje jinými funkcemi a hardwarovým rozhraním. Tato zařízení již samotná byla dodána a následně sestavena firmou FESTO, která i ke každé sestavě dodala vlastní program do PLC. Díky tomu tak tyto jednotky můžou fungovat jak samostatně, tak v případě koupě celé sady, tvoří celistvou výrobní linku. V naší laboratoři se nacházejí tyto stanice dvě: **CP Lab Heat Tunnel** a **CP Lab Drilling application**. Je důležité si tedy dané stanice dobře prozkoumat a zjistit, která bude pro naše účely nejvhodnější.

Tvorbu těchto praktických cvičení, jinak řečeno laboratorních úloh, jsem se rozhodl vymyslet trošku odlišnou formou, než je u většiny úloh v prostorech automatického řízení zvykem. Jedná se o to, že převážná část cvičení je zde spouštěna přes rozhraní, jež vyžaduje zapnutý počítač a řešili jsme tak danou úlohu přímo v něm. My však máme tu výhodu, že CP Lab stanice jsou vybaveny operátorskými panely, které lze naprogramovat tak, abychom si zde vytvořili ideální pracovní rozhraní pro řešení úlohy a nepotřebovali tak spuštěný počítač s daným softwarem. Tomuto spuštění počítače se však nevyhneme při řešení první úlohy, kdy má student za úkol sám kus programu vytvořit.

Celkově si tak projdeme funkčnost jednotlivých stanic, vytvoříme na jedné z nich tři úlohy a k těmto úlohám pak návody, které budou studenty vést při jejich laboratorním cvičení.

Kapitola 2

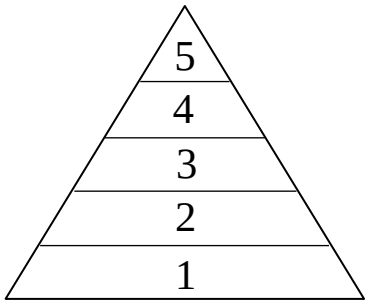
Rozbor samotných stanic

V dnešní době se hodně mluví o takzvaném Průmyslu 4.0, kdy tento název by měl být jakýmsi synonymem moderního průmyslu. Pod zkratkou CP Lab se pak skrývá výukový systém **„The Cyber-Physical Lab“** od firmy Festo, který je poskládán tak, aby právě veškeré jeho prvky splňovaly toto průmyslové řešení 21. století. Jedná se o modulární systém stanic. To znamená, že jednotlivé stanice fungují jak samostatně, tak v různých kombinacích jako celek **„výrobní linka“.**

Hlavní výhody tohoto systému je právě moderně řešená komunikace mezi všemi stanicemi a na nich samotných, což je jedním z hlavních parametrů moderního průmyslu. Různorodost senzorů a akčních členů, jež nám umožňuje mechanicky kombinovat a přestavovat stanice dle našich požadavků a vkusu je zde samozřejmostí.

Jednotlivé stanice nabízí aplikace jako: Vrtání, Kamerová inspekce, Zásobník, Stlačování, Měření, Ohřívání a mnoho dalších.

Tyto všechny aplikace se přidávají jako sekundární, tedy **„aplikační modul“** k modulu společnému **„základnímu**“, jenž slouží jako mechanická komunikace mezi všemi stanicemi. Tato společná základna je posuvný pásový dopravník. O stanicích, které využíváme v laboratoří si povíme ještě níže a více do detailu.



Obr. 2. 1: Pyramida automatizace

Jako vzor pro vysvětlení celkového přínosu této výukové sestavy, bych si vzal klasickou pyramidu automatizace, rozloženou na pět úrovní.

1. První úroveň je základní aplikační, tudíž je to oblast, kam spadají všechny senzory, pohony, ventily a vlastně cokoliv, co na stanicích ovládáme. Jsou to tedy převážně takzvané **„polní instrumentace“**.
2. Druhá úroveň je řídící. To je centrum, kde se vykonává program v PLC. Nachází se v našem propojovacím modulu s pohyblivým pásem. Je možno volit mezi více typy řídících jednotek, kde v našem případě řídíme modul skrze ET200SP. Avšak pozor, v některých případech ho používáme jen jako takzvaný **„smart slave“** a hlavní řídící PLC je ještě přidáno nad ním v aplikačním modulu.
3. Třetí úroveň se nazývá dohlížející (dozorčí), tuto funkci nám v našem případě plní ovládací dotykové panely, které jsou u každého základního modulu zvlášť.
4. Čtvrtá úroveň je takzvaná plánovací (MES). K tomuto společnost FESTO vyvinula svůj vlastní MES4 software, který má v sobě právě všechny tyto prvky, jak pro plánování výroby, tak vytváření databáze a kontroly veškerých linek.
5. Poslední pátá úroveň je manažerská, která je už spíše o ekonomickém zvažování výroby z hlediska zisku, což je ale v této době hlavní prioritou podniku.

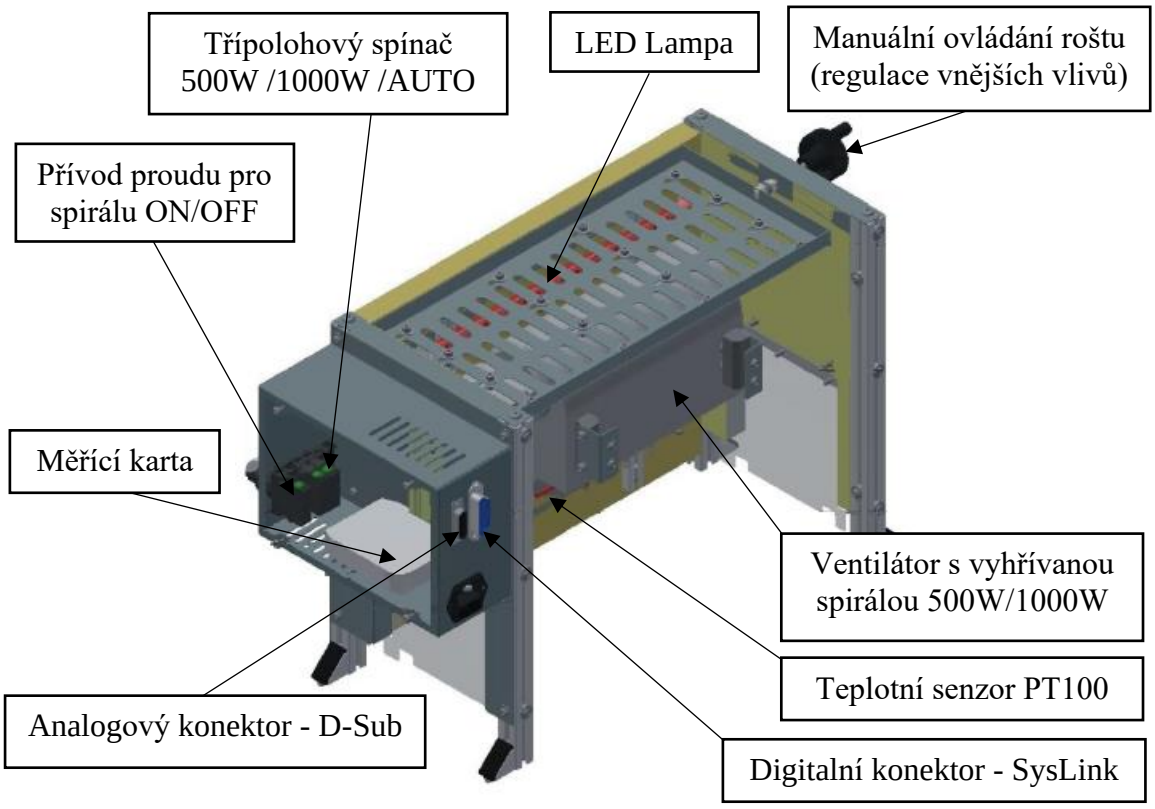
2.1 Horkovzdušný tunel

CP Lab Heat Tunnel (Horkovzdušný tunel) – Tato stanice má za úkol simulovat průmyslové řešení sušení nátěru (barvy) při průjezdu tunelem. Jako akční člen zde slouží ventilátor se zabudovanou výhřevnou spirálou, kterou lze regulovat na výkonu mezi dvěma hodnotami 500W a 1000W. Toto lze provést buďto fyzicky za pomocí třípolohového spínače (500W/1000W/AUTO), nebo přepnout tímto spínačem do režimu AUTO a nastavovat výkon skrze PLC.

Pro měření teploty zde používáme odporový teploměr PT100. Ten skrze měřící kartu zpracovává změřenou teplotu a posílá své data dál do PLC ve formě napětí v rozsahu 0 až 10V.

Máme zde taky možnost simulace rušení procesu vnějšími vlivy. Tuto chybu lze nasimulovat fyzicky, a to za pomocí kovového roštu s dírami, jenž je přidělán na peci a ovládán skrze pohybový šroub přes plastové kolečko. Tím dokážeme na roštu dané díry otevírat, nebo je nechat úplně uzavřené a ovlivňovat tak teplotu procesu sušení.

Jako ukazatele zapnuté pece, tedy probíhajícího procesu schnutí, zde máme červenou LED lampu, která nám symbolizuje svou červenou barvou rozžhavenou pec. Senzor nám dovoluje měřit až do teplot okolo 200°C. Jelikož se ale jedná o školící stanici, tak z důvodu bezpečnosti skutečná maximální dosažitelná teplota, po kterou můžeme pec rozehřát je 75~80 °C. Tato hranice teploty je dána hardwarově pomocí dvou bezpečnostních bimetalových spínačů uvnitř tunelu.



Obr. 2. 2: Horkovzdušný tunel [1]

2.1.1 Senzorika a jiné

Teplotní senzor PT100 – Neboli RTD (Resistance Temperature Detector) je typ senzoru, který měří teplotu pomocí změny elektrického odporu v kovu „**což je v našem případě platina(Pt)“,** způsobenou změnou teploty okolí. Číslo ve značce senzoru pouze značí, že při 0°C je odpor senzoru 100Ohmů.

Rozsah pracovní teploty: (-50 až 200) °C
Typ: 4 vodičový
Citlivost senzoru: 0,385 Ohm/°C
Toleranční skupina: B

Obr. 2. 3: Teplotní senzor PT100 [2]

Bimetalový spínač ESKA 36TXE21 – Jedná se o bezpečnostní prvek, který nám hlídá za pomocí technologie bimetalu (rozdílné roztažnosti dvou různých kovů) teplotu v peci. Kdy při dosažení hraniční hodnoty 80 °C rozepne kontakty přivádějící elektrický proud do topné spirály. Z důvodu bezpečnosti, zde máme dva spínače, a to pro případ, že by jeden z nich selhal při rozepnutí obvodu.

Teplota pro otevření: 80 °C (± 5 °C)
Zavírací teplota: 65 °C
Životnost: 100 000 cyklů

Obr. 2. 4: Bimetalový spínač ESKA [2]

Měřící karta pro PT100 – Jedná se o převodník pro teplotní senzory. Převádí hodnoty senzorů do teplotně-lineárního výstupu, a to ve formě proudu 4-20mA, nebo napětí 0-10V, s kterým již PLC umí pracovat.

Rozsah teplot: -200°C až 850°C
Podporované technologie: 2,3,4 vodičové

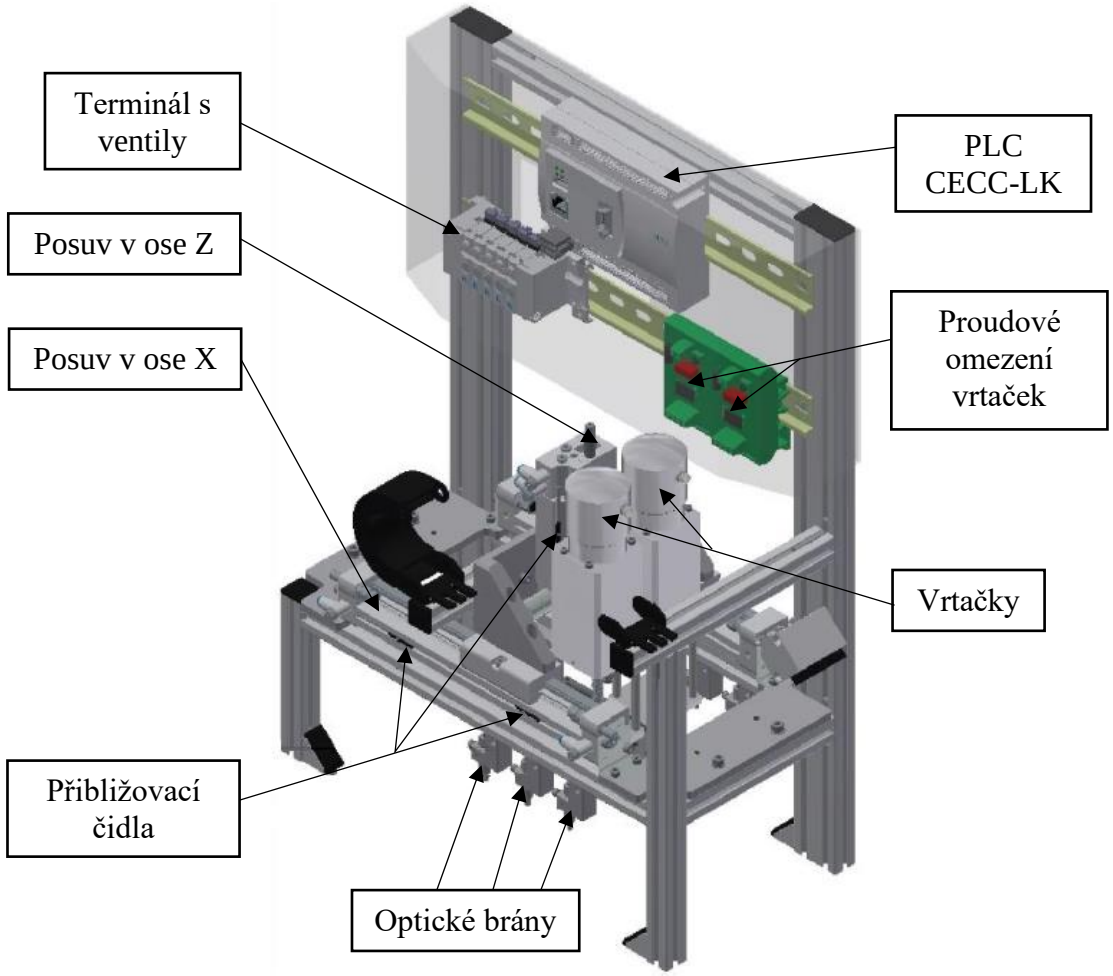
Obr. 2. 5: Převodník pro PT100 [2]

2.1.2 Řídící program

Naprogramovaný cyklus v PLC funguje v následných sekvencích. Pec na začátku čeká na příjezd vozíku s polotovarem, ten se zastaví na pozici, kde jej zachytí stoper. Tato pozice je dána rozmístěním polohových senzorů podél pásu. Poté se změří teplota pomocí PT100 a zapne se žhavení spirály s větráky. Čekáme, než je dosažena požadovaná hodnota teploty, kterou jsme si zvolili na operačním panelu, nebo jsme nechali původní nastavení pro uschnutí barvy. Jakmile je tato teplota dosažena, začne běžet časovač, při kterém pec drží konstantní teplotu pomocí dvoupolohové regulace a po uplynutí času vypne spirálu s větráky, zajede stoper a pustí vozík dál k následující stanici.

2.2 Vrtačka

CP Lab Drilling CPS application (Vrtačka) – Stanice slouží jako ukázka kooperace polohování s akčními členy, v našem případě vrtačkou. Ty se zde nacházejí dvě. Jsou poháněny stejnosměrnými motory, jejichž otáčky jsou samostatně regulovány přes proudový omezovač. Jelikož se jedná o úlohu, kde se silně využívá pneumatika, tedy proces, kde stlačený vzduch měníme na práci, tak se zde nachází i rozvodný terminál pro ventily. Tímto způsobem jsou řízeny posuvy vrtaček do os Z a X. Každá dráha posuvu libovolné osy je na obou koncích chráněna koncovým pneumatickým magneto-rezistivním snímačem, někdy také nazývaným **„přibližovací čidlo“.** Ty nám prozradí situaci, kdy se dostane jezdec posuvu do krajní meze a program tak zastaví jeho pohyb. Jelikož se jedná o pneumatiku, tak na koncích drah máme mechanické brzdy pro případ nárazu, jsou to vlastně malé plastové dorazy na pružinkách, které chrání konstrukci.



Obr. 2. 6: Vrtací stanice [3]

Co se týče pozicování pro vrtání, tak se jedná o kooperaci polohových senzorů na pásovém dopravníku a tří optických bran, jež nám díky svému umístění, které přesně odpovídá rozložení polotvaru na paletě poví, v jakém je náš výrobek stádiu. První brána zprava na obrázku nám ukazuje, zda je polotovar (jeho spodní část) na paletě správně umístěn. Prostřední optická brána nám říká, jestli má polotovar víko (pokud by měl, tak vrtaní není možno) a poslední optická brána nám ukazuje, jestli se zde vůbec nějaký polotovar nachází.

2.2.1 Senzorika a jiné

Polohový detektor SME-10 NO – senzor přiblížení fungující na principu magnetické-jazýčkovém měření. Jedná se tedy o dva feromagnetické jazýčky, které se propojí při přiblížení magnetického pole, které nám vytvoří malý magnet vložený do jezdce posuvu. Pomocí nich snímáme koncovou polohu pro posuv jezdce.

Typ snímače: Normálně otevřený
Napájecí napětí 10-30V

Obr. 2. 7: Přibližovací čidlo SME-10 [4]

Fotoelektrický senzor SOEG-L-Q30-P-A-S-2L – Jedná se o optickou bránu, která zaznamenává přerušení paprsku a přes technologii optických vláken vysílá informaci do PLC. Máme zde použité celkem tři, a to pro snímání polotovaru na vozíku.

Pracující rozsah: 0-120mm
Pracovní napětí: 10-30V
Barva laseru: Červený

Obr. 2. 8: Optická brána SOEG [4]

PLC CECC-LK – V případě této stanice zde máme vlastní řídící PLC, které kontroluje veškerý proces vrtačky a skrze komunikaci i řídí PLC dopravního pásu. Jedná se tedy o typickou ukázku **„master/slave“** řízení, kde máme hlavní PLC s druhým podpůrným PLC, které však slouží jen jako určitá sběrnice dat a vykonavatelem akčních zásahů, kdežto hlavní PLC CECC-LK je ve **„velíně“** a kontroluje veškeré procesy.

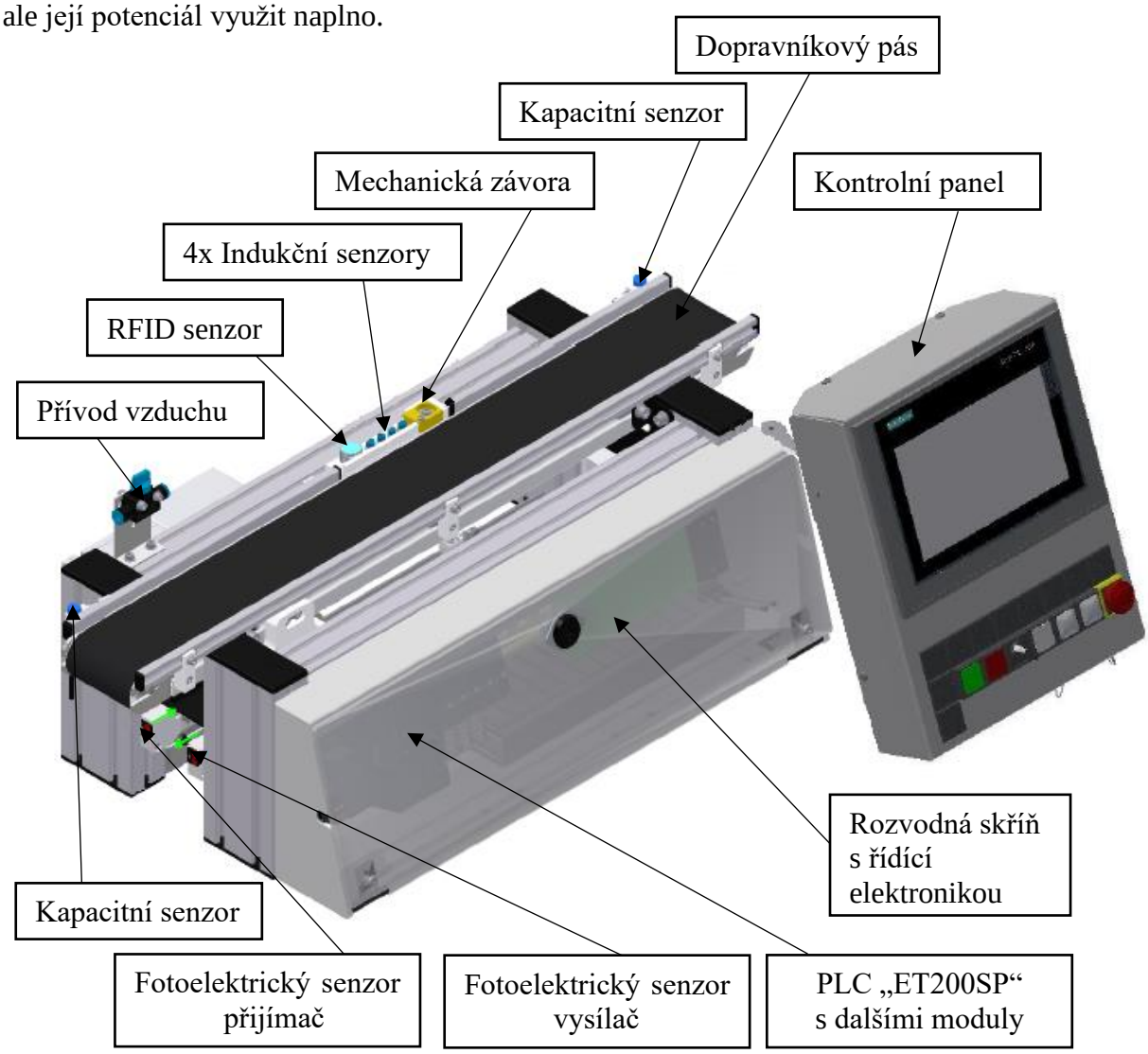
Obr. 2. 9: PLC CECC-LK [4]

2.2.2 Řídící program

Aplikace funguje tak, že nejprve musí být rozpoznáno indukčním čidlem pásu, že přijíždí vozík. Ten dojede až k optickým bránám a ty začnou prověřovat, v jakém je vozík stavu. Nastává kontrola přítomnosti polotovaru přes první optickou bránu, pomocí druhé optické brány kontrolujeme, zda se na vozíku nachází horní kryt a zda je správně umístěn. Další optická brána zase zkontroluje, jestli je polotovar celkově na paletě dobře umístěn. Pokud je spodní část krytu přítomná a správně umístěná, najede vrtačka do první pozice a vyvrtá první dvě díry na začátku vozíku. Poté se vrtačka posune o kousek v ose X a vrtačka znovu vyvrtá další dvě díry. Veškeré vrtání je však pouze **„na oko“**, vrtáky se totiž pouze těsně přiblíží k otvorům, ale nezajedou do nich. Po této činnosti se vrtačka vrátí do základní pozice a vozík opouští stanici. Pokud se však nacházela na spodním víčku jeho vrchní část, veškerý vrtací a polohovací proces se neděje a polotovar jen projede celou stanicí do další, aniž by se s ním něco stalo. Vrták je také z bezpečnostních důvodů zhotoven z plastu, aby nedošlo k nebezpečí vzniku úrazu studenta.

2.3 Dopravník

CP Lab Pallet Transfer Line (Dopravníkový pás) slouží jako určitý základní modul pro všechny ostatní aplikační moduly (např. pec, vrtačka, kamerová kontrola a mnoho dalších). Všechny tyto modulární sestavy byly postaveny tak, aby se daly jednoduše kombinovat a spojit pouze za pomocí šroubů. Veškerý rozvod vzduchu a kabeláže tak na sebe jednoduše dosedá. Samotná stanice sice jde použít pouze jako dopravník, není pak ale její potenciál využit naplno.



Obr. 2. 10: Dopravníkový pás [7]

Každá dopravníková stanice je vybavena na svých stranách fotoelektrickými senzory, které zajišťují celkové propojení s více stanicemi seriově za sebou.

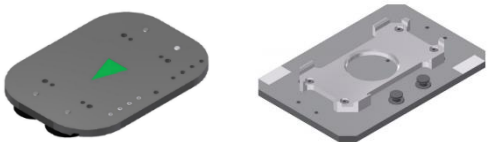
Máme zde velké zastoupení indukčních a kapacitních proximitních senzoru, sloužících pro sledováni pozice vozíku na páse a RFID senzor, který zpracovává data, jež vozík se sebou přenáší v datové minci. Pás je v našem případě poháněn stejnosměrným motorem, jehož otáčky jsou snímány inkrementálním enkóderem.

Nejdůležitější část je zde rozvodná skříň, která v sobě ukrývá veškerou řídící elektroniku. Ta kontroluje nejen dopravník, ale i veškeré připojené moduly, které se přes komunikaci a sběrnice dostanou až do PLC, jež pak vykonává nahraný program.

Vozík

K dopravnímu pásu byly dodány tyto speciální vozíky, které svými rozměry přesně odpovídají šířce dopravního pásu a ostatním aplikačním modulům této výukové sestavy.

Na povrchu černé plastové základny vozíku, můžeme vidět čtyři malé šrouby, které jsou z vodivého materiálu (kovu) a rozmístěny tak, aby byly zachyceny indukčními senzory podél pásu. Z tohoto důvodu je na něm nakreslena zelená šipka pro správný směr pohybu, jelikož jsou tyto senzory jen na jedné straně pásu. Na spodní straně vozíku je umístěna RFID paměťová karta ve tvaru mince, do níž je zapisováno, nebo je z ní čteno, pomocí patřičného senzoru. Ke každému vozíku je pak vyrobena kovová paleta, která slouží k bezpečnému uložení polotovaru.



Obr. 2. 11: Vozík a kovová paleta [5]

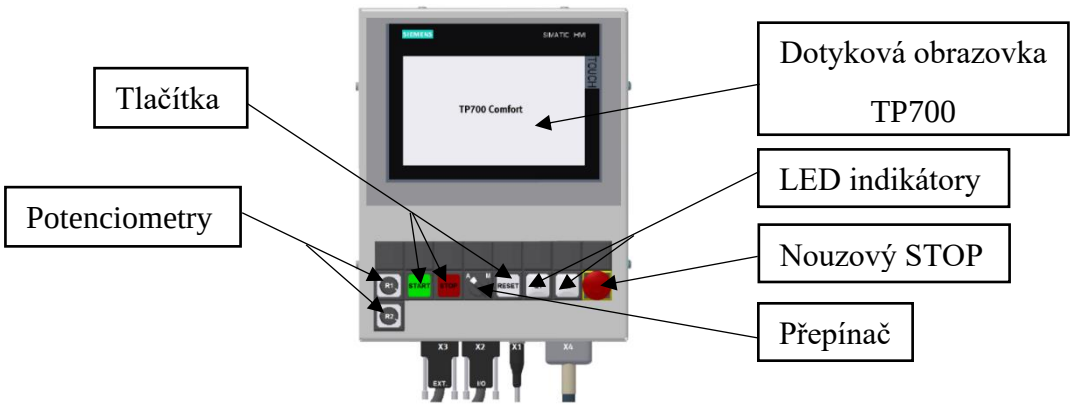
Polotovar

Polotovar je složen ze tří částí. Spodní část slouží jako základna a svým tvarem odpovídá úchytkám na kovové paletě. Do základny je možno vložit tištěný spoj s konektory pro trubičkové pojistky (tato úloha s tištěným spojem a pojistkami však spadá do jiné stanice) a na to je možno ještě vložit horní kryt. Aplikace vrtání, jak jsme si vysvětlili výše se koná tedy pouze do spodní části krytu, který jak můžeme vidět na obrázku, již má předvrtané čtyři otvory.

Obr. 2. 12: Spodní kryt, tištěný spoj, horní kryt [5]

Ovládací panel

Každý dopravníkový pás pak má svůj vlastní ovládací panel. Ten je složen z dotykové obrazovky s externě připojenými tlačítky a led žárovkami. Obrazovka má své vnitřní rozhraní upravené podle toho, k jaké aplikaci je modul připojen. V našem případě tedy spojení dopravníkového pásu s vyhřívaným tunelem, nebo vrtačkou. Obchodní název obrazovky je TP700 Comfort od firmy Siemens. Tato zkratka znamená, že se jedná o 7palcový dotykový displej s rozšířenými možnosti, co se týče rozhraní programovacího nástroje WinCC (řada Comfort). Zkratka: TP700 (T-Touch, P-Panel, 700-7palců).



Obr. 2. 13: Ovládací panel s TP700 [7]

2.3.1 Senzorika a jiné

RFID RF210R – senzor pro čtení a zápis dat do datových mincí, které na své spodní straně přenášejí vozíky. Do této paměti lze zapisovat veškeré informace o tom, co vozík přenáší, nebo například k jakým procesům je určen.

Obr. 2. 14: TW-R16-B128 paměť a RFID Senzor [5]

Indukční proximitní snímač Festo SIEN – slouží pro detekci kovových předmětů v krátké vzdálenosti až 2,5 mm od senzoru. Typ výstupu tranzistor PNP-NO s napájením senzoru 30V a výstupním proudem 200mA. Máme je vcelku čtyři a polohují nám přesné umístění vozíku podél pásu.

Obr. 2. 15: Indukční senzor SIEN [6]

Kapacitní proximitní snímač – slouží pro detekci předmětů libovolného materiálu v krátké vzdálenosti až 5 mm od senzoru. Typ výstupu tranzistor PNP-NO s napájením senzoru 30V a výstupním proudem 200mA. Jsou zde dva a hlídají nám koncové pozice stanice na pásovém dopravníku.

Obr. 2. 16: Kapacitní senzor [6]

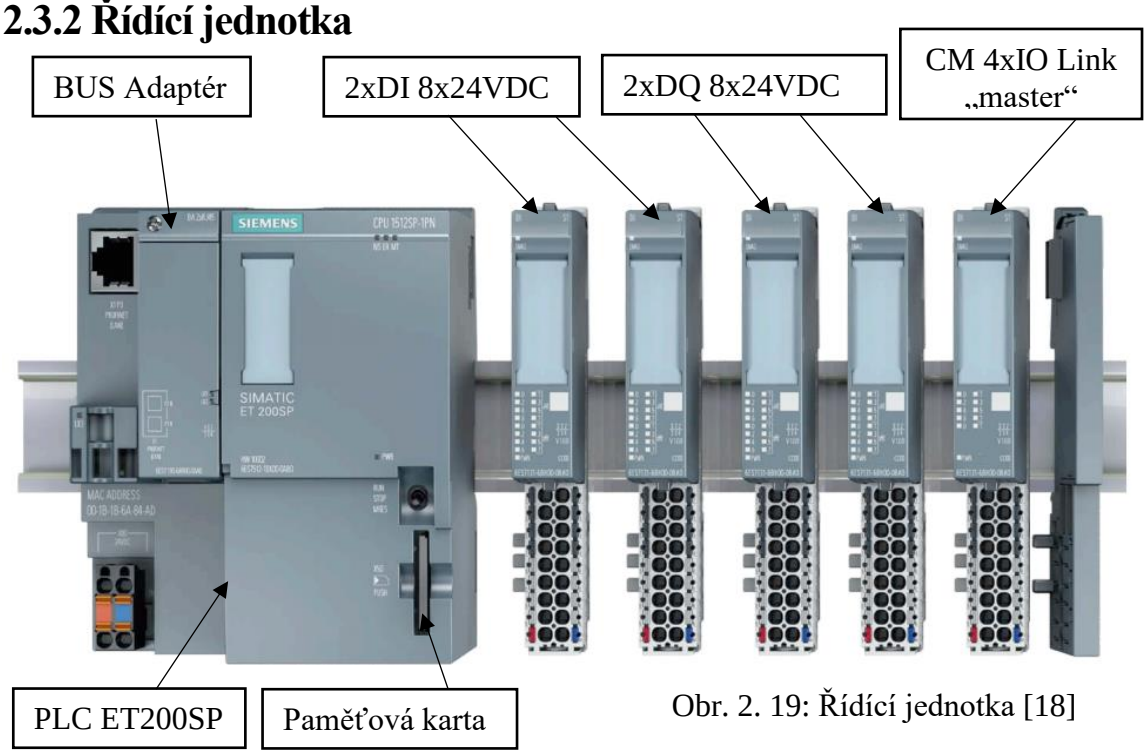
Mechanická závora AEVUZ-16-5-P-A - slouží pro zastavení pohybu vozíku na pásu pomocí pístní tyče, kterou pod tlakem vysune (jedná se o jednostranné vysouvání). Poloha tyče je snímaná proximitním senzorem a pracovní tlak pro vysunutí se pohybuje mezi 1,3 až 10 bary.

Obr. 2. 17: Mechanická závora [6]

Fotoelekrický přijímač/vysílač SOEG-S-Q30-S-L- slouží pro komunikaci mezi stanicemi, jež jsou k sobě mechanicky připojeny v sérii. Senzory se nacházejí na každé straně dopravníkové stanice, a to vždy v páru vysílač/přijímač.

Obr. 2. 18: Fotoelektrický přijímač/vysílač [6]

2.3.2 Řídící jednotka



Obr. 2. 19: Řídící jednotka [18]

PLC: ET200SP / CPU1512SP F-1PN

- Modul ET200SP s CPU1512 F-1PN a integrovanou pracovní pamětí 1MByte, z toho důvodu musí být přítomna paměťová karta, na kterou se nahrává program.
- Tento modul byl vytvořen pro řízení vzdáleného I/O zařízení a slouží jako **„chytrý“** programovatelný **„slave“**. Z dálky je pak kontrolován z řídící místnosti přes **„master“** PLC např. S7-1500. V našem případě je však toto PLC použito jako samotný **„master“.**
- Jedná se tedy primárně o úsporu kabeláže, abychom tak nemuseli veškeré senzory a akční členy připojovat přímo do **„master“** PLC, ale právě použili toto vzdálené I/O, které nám bude skrze komunikaci posílat všechna potřebná data a to pouze přes jeden kabel.
- Pro komunikaci zde máme 3 porty, z toho jeden PROFINET a dva BUS adaptéry.
- Modul lze rozšiřovat o různé signálové karty. [18]

Modul (karta) pro digitální vstupy: DI / 8x24VDC

- jedná se o modul pro příjímání logických informací ze senzorů.
- napájení 24V, maximální počet vstupů 8.
- logické úrovně: -30 až +5V pro log „0“
  +11 až +30V pro log „1“

Modul (karta) pro digitální výstupy: DQ / 8x24VDC 0,5A

- modul pro nastavování výstupů.
- napájení 24V, maximální počet výstupů 8, maximální odběr jednoho výstupu 0,5A.
- typ sepnutí výstupu PNP tranzistor.

CM / 4x IO-Link ST

- modul pro komunikaci a přenos dat mezi více zařízeními (Master/slave).
- napájení 24V.
- počet vstupů 4.
- je na něj připojen RFID senzor a FESTO IO Link D.

2.4 Volba mezi stanicemi

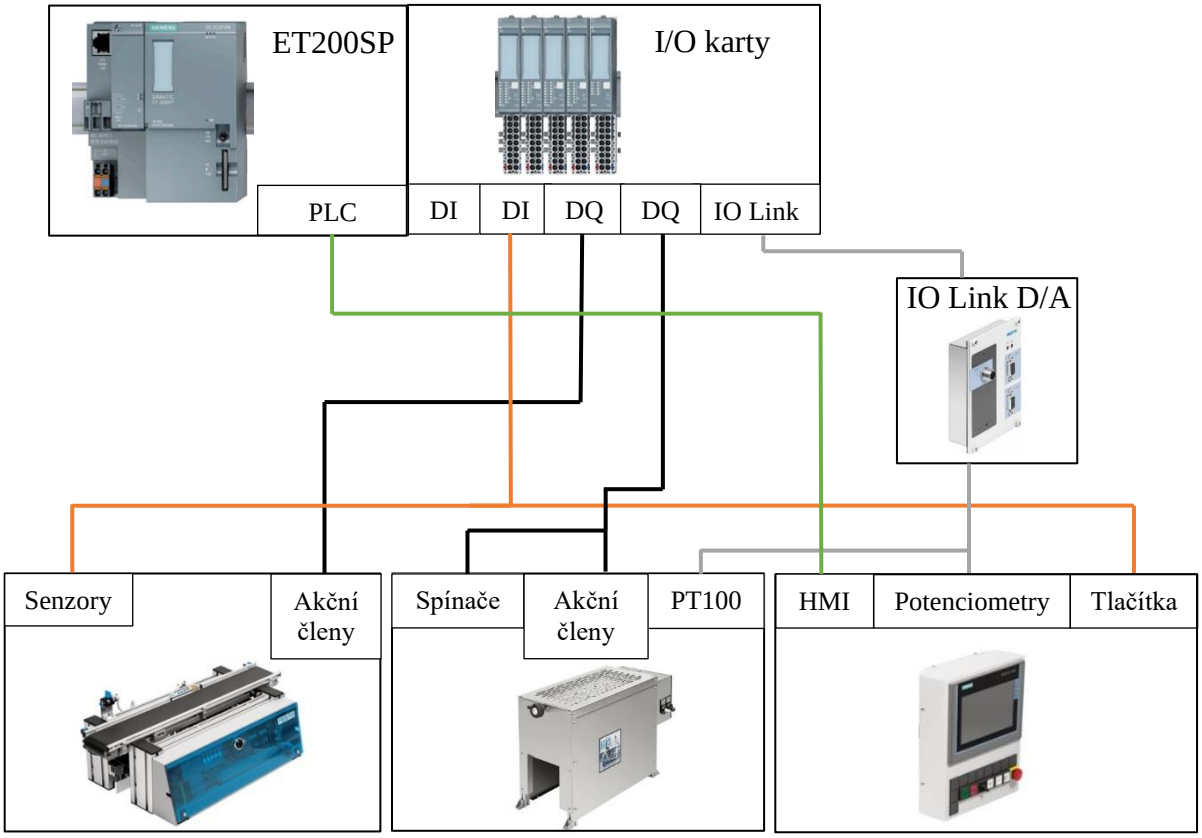
Porovnejme tedy vše, co jsme doposud zjistili. Horkovzdušný tunel nám nabízí možnou regulaci teploty až do ~80°C. Má sice pouze dva digitální výstupy (pro sepnutí spirály a volbu výkonu), ale zato nabízí jeden analogový vstup s připojeným senzorem teploty PT100. Vše je zde řízeno přes externí PLC ET200SP, které je umístěno na základním modulu. Navíc můžeme mechanickým způsobem ovlivňovat vnější vlivy soustavy přes konstrukci samotného tunelu.

Vrtací stanice na druhou stranu nabízí vrtání do čtyř předvrtaných děr v polotovaru a posuvy vrtacích hlavic ve směru dvou os. Tyto pohony pro vrtací hlavice však bohužel nejde nijak kvalitně řídit a regulovat, jelikož zde není žádná zpětná vazba o rychlosti otáček. Stanice má pouze digitální vstupy a výstupy, které jsou ještě navíc připojeny k internímu PLC CECC-LK. Jelikož je však základní modul s pohyblivým pásem řízen přes jiné PLC, vzniká nám zde problém, že musíme každé PLC programovat jiným programem, a navíc řešit jejich společnou komunikaci. Když si to tedy shrneme, tak nemožnost jakéhokoliv lepšího řízení otáček motorů, rozdílné programování dvou typů PLC a žádná analogová zpětná vazba systému nám v tomto uskupení přináší víc záporů, než plusů. Pokud bychom však **„předrátovali“** zapojenou senzoriku do našeho ET200SP a pořídili senzor pro měření otáček, situace by mohla být jiná.

Samotný pásový dopravník nám nabízí 24V DC motor, jehož otáčky však také zpětnovazebně řídit nemůžeme. Máme zde alespoň zpětnou vazbu o tom, do jaké strany se točí. Je to veliká škoda, jelikož je vybaven právě inkrementálním enkóderem, bohužel naše PLC nedisponuje ničím, co by dokázalo tyto pulzy z enkóderu zachytávat, a i kdybychom to dokázali, tak rozlišení samotného enkóderu není příliš dobré, rotační kotouč má totiž pouhých 8 pozic. Museli bychom tedy dokoupit kartu s čítačem a pořídit enkóder s vyšším rozlišením. Máme zde však kapacitní senzory na obou koncích pásu, čtyři indukční pro polohování vozíku a RFID pro práci s daty vozíku. Samotný ovládací panel pak má spínače, přepínače, ledky a dva analogové potenciometry. Stanice si zde vše řídí přes své PLC ET200SP.

Po důkladném zvažování jsem se tedy rozhodl pro horkovzdušný tunel v kombinaci s dopravníkovým pásem. U tunelu můžeme sice brát za nevýhodu jeho nízký počet vstupů a výstupů, tedy proměnných, s kterými by studenti mohli pracovat. Na rozdíl od vrtačky však vlastní potřebný analogový vstup, který vyžadujeme u dvou typů úloh spojitého řízení, a navíc má svůj řídící program na stejném PLC jako základní modul s pásovým dopravníkem. To pro nás znamená, že dokážeme programovat linku jako celek v jednom programátorském prostředí.

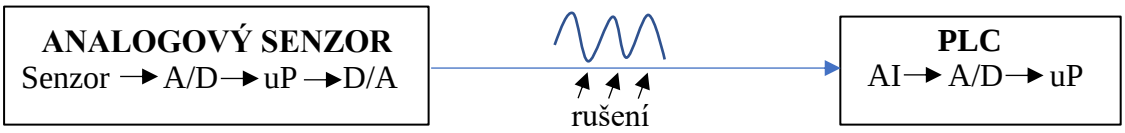
2.5 Komunikace mezi zařízeními



Obr. 2. 20: Komunikace

Komunikace mezi senzorikou, akčními členy a PLC je zde řešena tak, aby byly ukázány dva nejvíce typické přístupy v praxi. Většina zařízení je zde jednoduše připojena skrze sběrnice do karet „DQ, DI“ u PLC ET200SP.

Způsob připojení senzorů, akčních členů do karet „DI, AI, DQ, AQ“



Obr. 2. 21: Připojení analogu skrze karty

(V senzoru se analogová hodnota převede do digitální formy, poté ji v mikroprocesoru zesílíme a převede zpět do analogu, který pak putuje do PLC)

Obr. 2. 22: Připojení digitálu skrze karty

(Porovnává úroveň napětí na senzoru a pomocí komparátoru pak překlápí do log.1, nebo log.0 a následně tuto informaci posílá do PLC).

Metodika pro analogové a digitální výstupy je obdobná, jediný rozdíl je, že hodnoty posíláme z PLC do akčních členů.

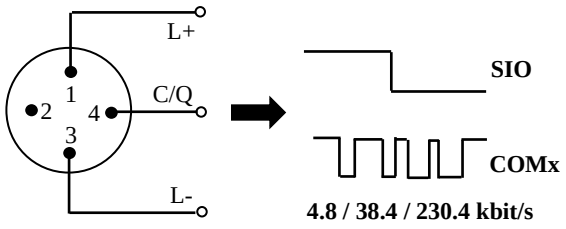
Nevýhodou tohoto způsobu připojování je však větší spotřeba karet v rozvaděči (což se silně odráží na ceně), také rušení přenášeného analogového signálu, které řešíme stíněnými kabely (zase vyšší cena) a problematické řešení komunikace mezi zařízeními jiných společností. Samozřejmě jde vzít i v potaz složitější **„drátování“** na kartách a motorových startérech. Z historického důvodu a jednoduchosti, je to však stále silně používaný způsob připojování.

Způsob komunikace skrze IO Link

Zbytek zařízení u naší sestavy (převážně analogy) jsou však připojeny modernějším způsobem skrze IO Link, který nahrazuje analogový přenos dat mezi zařízením a PLC. Je sice omezen vzdálenosti kabelu 20m a používá převážně nestíněných vedení, ale dokáže posílat jak digitální, tak analogový stav senzoru skrze pulzní modulaci signálu, tedy způsobem velice dobře odolným vůči rušení. Tato komunikace mezi senzorem a masterem bývá často tří, nebo čtyř vodičová.

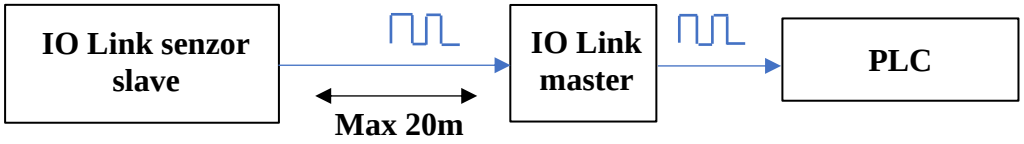
Pin	Značení	Popis
1	L+	24V
2	I/Q	Nepřipojen, DI, nebo DQ
3	L-	0V
4	Q	„Logický signál TRUE/FALSE“ (**SIO**)
	C	„Datový signál“ (**COM1,COM2,COM3**)

Tab. 2. 1: IO Link - čtyřvodič [13]



Obr. 2. 23: Čtyř vodičová komunikace [13]

Možná nevýhoda je v tom, že použité senzory, motorové startéry, musí být uzpůsobeny tomu, aby s IO Linkem komunikovaly. Jedná se o to, že každé zařízení, které je přes IO Link připojeno, má v tomto **„ IO Link masteru“** k sobě přiřazen takzvaný IODD soubor, který v sobě uchovává detailní informace a parametry příslušného zařízení. Takže starší technologie, které tuto knihovnu nemají, není možno použít.



Obr. 2. 24: Připojení senzoru skrze IO Link

IO Link je komunikační protokol, který dokáže pracovat mezi senzory, zařízeními a PLC od různých výrobců a sjednotit tak jejich komunikaci. Řídící zařízení se nazývá **„IO Link master“** a jeho úkolem je zpracovávat data z **„IO Link slave“**, tedy podřízených členů.

Funguje to na principu **Point-to-Point** komunikace mezi zařízeními. Pro vysvětlení této komunikace si lze představit telefonní hovor, kde to, co říká jeden člověk na jedné straně telefonu, může být slyšeno pouze na konci druhém.

S PLC poté **„master“** komunikuje například skrze Profinet. Profibus, nebo Modbus. IO-Link master nám tedy zařizuje komunikaci mezi IO-Link zařízeními (senzory, aktuátory) a PLC. Fyzicky pak může být tento **„master“** zapojen v rozvaděči jako karta vedle PLC (což je náš případ), nebo jsou také průmyslová řešení, které se instaluje poblíž zařízení. [13]

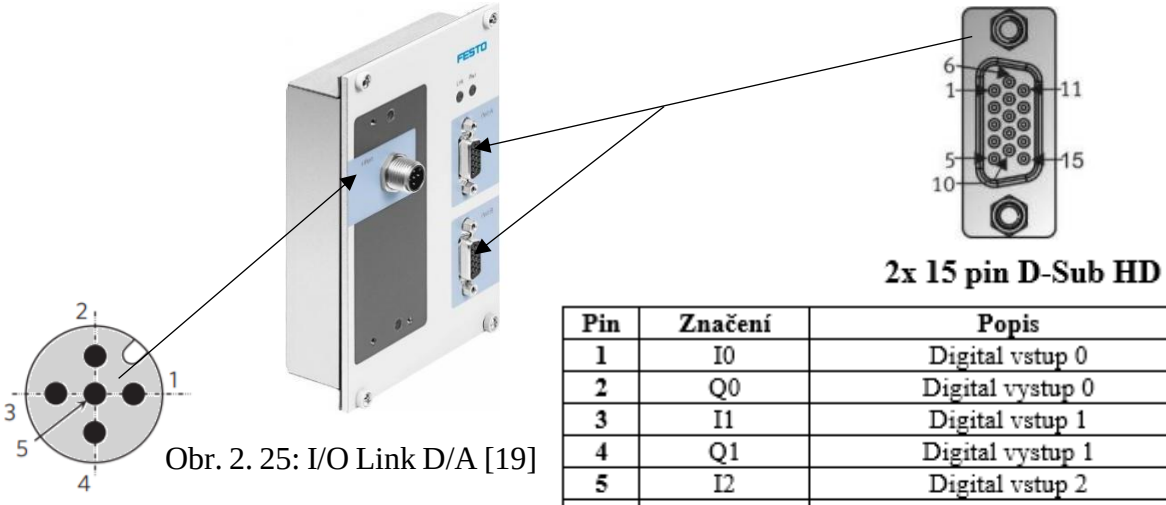
IO Link master pracuje v čtyřech režimech: DI, DQ, DEACTIVATED a IO-Link.

1. DI „digital inputs“ mód znamená, že zpracovává vstupní hodnoty jako digitální např. proximitní senzory.
2. DQ „digital outputs“ znamená, že se umí chovat jako logický výstup.
3. IO-Link, to je mód, kde si posílá balík dat s připojeným zařízením. Tento balík přenáší cyklicky procesní hodnoty, jako je například rychlost, teplota, časová perioda. Ale na zavolání dokáže i přenášet informace o zařízení, jako jsou jeho parametry, sériové číslo a diagnostiku zařízení. Tato data se pak dají nejen číst, ale mohou být i přepsána.
4. Deactivated mód nám říká, které vstupy/výstupy nebyly použity.

Výhodou IO Linku v průmyslu je, že šetří místo v rozvaděči, zamezuje vzniklému rušení na dlouhém vedení analogových senzorů (tak že IO Link zařízení se často nachází v blízkosti daného zařízení, kde jsou veškeré senzory přivedeny do něj, on je zpracuje a v digitální formě pošle do nadřízeného členu, vyhneme se tak dlouhé cestě analogového signálu od zařízení až k PLC). Velikou výhodou je také to, že na tento IO Link můžeme přistupovat z více PLC. Důležitá je samozřejmě cena, která je nižší, než bychom utratili za potřebné karty. V naší sestavě tedy máme IO-Link master **„CM 4xIO-Link“** určený pro PLC ET200SP od firmy Siemens, na nějž je přiveden RFID senzor a IO-Link D/A sběrnice od firmy FESTO, kde jsou pak přivedeny analogové/digitální signály, jako jsou potenciometry na ovládacím panelu a měření teploty. Tento IO-Link D/A pak posílá svá data do CM 4xIO Link masteru, který je přenáší do našeho PLC ET200SP. [13]

IO-Link D/A „slave“ rozhraní pro připojení zařízení

Zařízení slouží jako sběrnice pro senzory, akční členy. S IO-Link **„masterem“** je připojen skrze 5-pin M12 konektor a jednotlivé zařízení jsou zde přivedeny přes 2x 15-pin D-Sub konektor



Obr. 2. 25: I/O Link D/A [19]

Pin	Značení	Popis
1	24VB	+24V vstupy, logika
2	24VA	+24V výstupy, zátěž
3	GND B	0V vstupy, logika
4	IO-Link	Signál
5	GND A	0V výstupy, zátěž

Tab. 2. 2: IO Link M12-5pin [19]

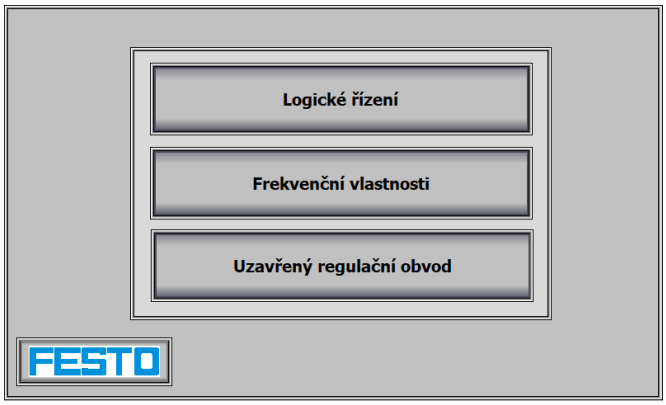
Pin	Značení	Popis
1	I0	Digital vstup 0
2	Q0	Digital vystup 0
3	I1	Digital vstup 1
4	Q1	Digital vystup 1
5	I2	Digital vstup 2
6	Q2	Digital vystup 2
7	I3	Digital vstup 3
8	Q3	Digital vystup 3
9	AI0	Analog vstup 0 (0 ...10 V), 12bit
10	AI1	Analog vstup 1 (0…10 V), 12bit
11	AQ0	Analog vystup 0 (0…10 V), 12bit
12	VCC výstup	+24V zátěž
13	VCC vstup	+24V vstupy, logika
14	GND výstup	0V zátěž
15	GND vstup	0V vstupy, logika

Tab. 2. 3: IO Link D-Sub HD-15pin [19]

Kapitola 3

Typy laboratorních úloh

Při výuce automatizace v laboratořích přistupujeme ke třem základním typům úloh: Logické řízení, Frekvenční vlastnosti a Uzavřený regulační obvod



Obr. 3. 1: Hlavní menu

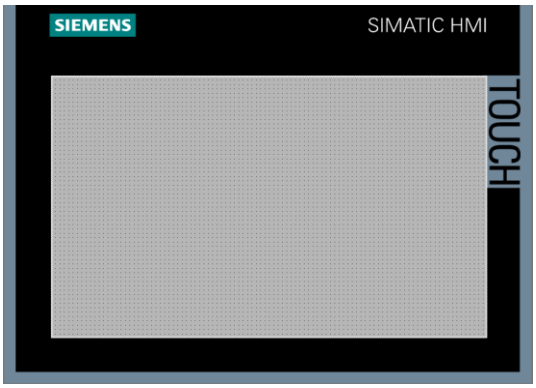
Tyto úlohy si vysvětlíme níže více do detailu a ukážeme si různé přístupy při jejich řešení. Abychom však předvedli co možná nejrealističtější ukázku z praxe, tak pro řešení veškerých úloh bude využito v dnešní době velmi oblíbené a používané platformy pro průmyslovou automatizaci, TIA Portál od firmy Siemens. Rozhodnutí, k jakému programu se přiklonit, je často dáno právě volbou hardwaru a jelikož používáme řídící jednotku ET200SP, tak byla naše volba dopředu jasná. Firma Siemens všeobecně v dnešní době silně vládne trhu s automatizací, proto i já sám vidím jako veliké plus studenty s tímto programem více seznámit.

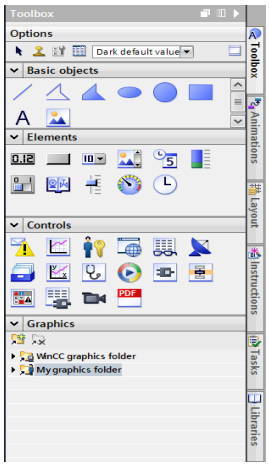
Co se týče první úlohy, tak má snaha je vytvořit zadání jako reálnou objednávku na realizaci automatizace linky, v našem případě horkovzdušného tunelu. Po určitém zvažování vidím jako nejlepší cestu pro studenta skrze programovací jazyk LAD, jenž mi přijde nejvíce intuitivní a v němž budu také část úloh vypracovávat. Možnost volby však zůstává na studentu samém a může zvolit jakýkoliv jiný podporovaný jazyk. Jelikož je tento software opravdu profesionálně užívaný v praxi, tak se jeho cena pohybuje ve vysokých částkách, tudíž zde není moc možností, aby si jej student někde volně stáhnul a často se tak s programováním setkává prvně buďto ve škole, nebo až v samotné praxi. Z tohoto důvodu je třeba první úlohu logického řízení připravit tak, aby byla přehledná. Přijít totiž na laboratorní cvičení a seznámit se s úplně novým vývojovým prostředím není vždy až tak snadné. U druhé a třetí úlohy, tak zde již studenty tak úplně samotné pracovat nenecháme a prostředí jim předpřipravíme o něco více. Jelikož se také jedná o úlohy, kde se dost pracuje s charakteristikami chování soustavy, tak jsem uviděl jako velikou příležitost využít právě ovládací panel, jímž je stanice vybavena a pomocí softwaru pro jeho programování WinCC, jenž je součástí TIA Portalu, jeho rozhraní naprogramovat. Každá úloha tak má svoje vlastní grafické rozhraní na panelu a v samotném programu.

Jelikož firma FESTO dodává tyto stanice spolu se svým vlastním programem, tak jsem se rozhodl jej tedy zachovat a pouze tuto část oddělit, aby studenti mohli trénovat své dovednosti s PLC a přitom zde byla možnost pro vyučující (pomocí panelu), přepnout zpátky do původního programu skrze tlačítko „FESTO“ a mít tak hotový ukázkový program.

3.1 Tvorba HMI pro laboratorní úlohy

K programování obrazovek budeme používat program Simatic WinCC pro HMI, jenž je součástí TIA Portálu. Jedná se o SCADA systém, který slouží k monitorování a hlídání procesů v průmyslu. Představíme si zde však jen základní prvky, s kterými se při tvorbě obrazovky setkáme. Jako první si zvolíme v programu naši sedmipalcovou (dotykovou) obrazovku TP700. Základem při programování je takzvaný **„screen“,** tedy obrazovka (šedé pole) na které lze pracovat. Dá se říct, že existují tři typy obrazovek: základní screeny, pop-up **„vyskakovací“** screeny a slide-in **„vyjížděcí“** screeny. Každá obrazovka se pak dle situace může hodit pro něco jiného. Na tyto obrazovky se dají z našeho **„Toolboxu“** přidávat různé prvky, ale můžeme i naprogramovat jejich chování, které vykonají v PLC při samotném zobrazení.





Obr. 3. 2: Rozhraní tvorby HMI s Toolboxem

Basic objects - Je zde možnost vložení základních objektů, jako jsou čára, obdélník, kruh, elipsa, či text. S těmito objekty se pak dá pracovat různě. Buď je použijeme čistě pro vzhled obrazovky, a tedy modelujeme naše grafické rozhraní panelu, nebo je můžeme použít v kombinaci s **„tagy“**. Objekty pak například mění barvu, polohu, viditelnost, nebo se rozblikají, a to v závislosti na naprogramování v PLC. Kromě těchto objektů zde jde vložit i obrázky a ikony.

Elements - Jedná se o funkční prvky obrazovky (tlačítka, přepínače, pole pro sledování a vkládání hodnot a další). Tyto prvky mají stejnou možnost animací jako basic objects, tedy měnit barvu, viditelnost a jiné. Oproti objektům jim však můžeme nastavit takzvaný **„event“.** To je událost, při které provedou určitý příkaz v PLC/HMI. Například můžeme zvolit element **„Tlačítko“** a přiřadit mu událost, že když jej někdo stiskne, nastaví nám určitý bit v PLC. Nebo můžeme vložit element pole, po jehož stisku vkládáme číselnou hodnotu.

Controls – Jež patří komplexnější funkce a prvky, jenž lze v HMI použít, jako jsou například možnosti zobrazení alarmů, přehrání videa, pdf, diagnostika zařízení, administrace, nebo vykreslování grafů.

WinCC nabízí spousty dalších funkcí a prvků. Pracujeme zde nejen s **„tagy“** ale i text/grafik listy. Můžeme vytvářet technologické objekty, což jsou vlastně kombinace objektů a elementů v jednom. Lze vyvolávat menší obrazovky a kombinovat je tak s hlavními, nebo nechat jednotlivé obrazovky vyjíždět ze stran. Spousty těchto věcí v našich laboratorních úlohách používám, bohužel však zde není úplně prostor pro vysvětlení naprogramování každé, a tak se tedy budeme spíše v této práci zabývat funkčností daných obrazovek a jejich aplikací v laboratorní úloze. Tedy veškeré obrazovky, které zde uvidíte, jsou vytvořeny mnou a jejich funkčnost naleznete v příloze.

3.2 Úloha: Logické řízení

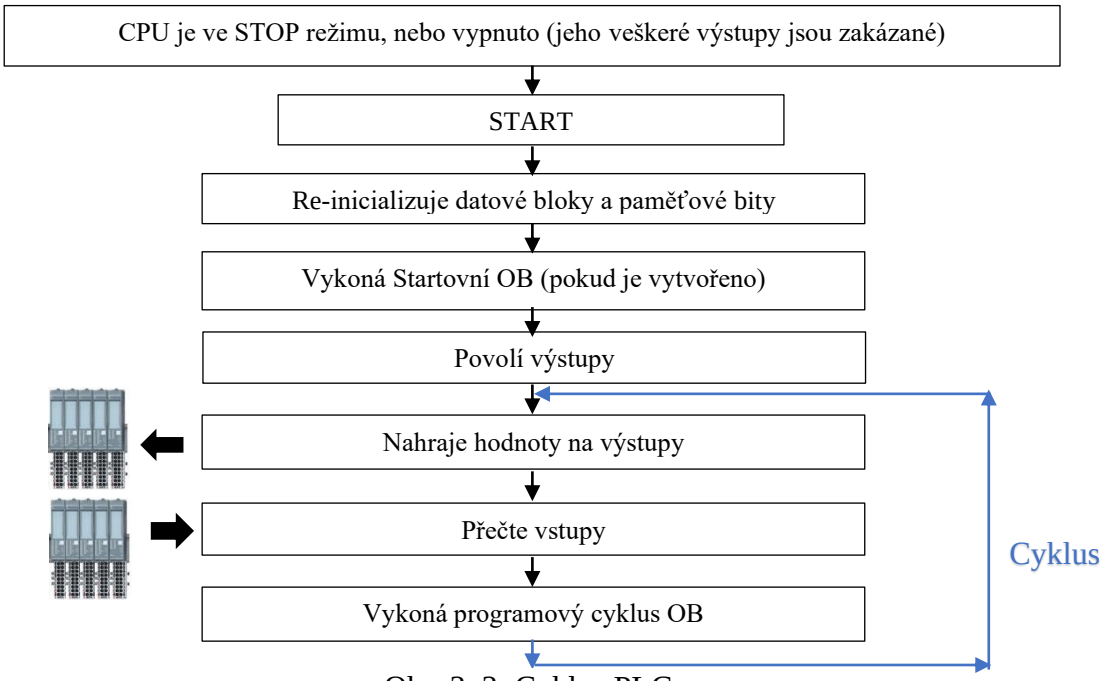
Jedná se o základní kámen veškeré průmyslové automatizace. Již ze slova **„řízení“** můžeme vyvodit, že se bavíme o určité formě kontroly daného procesu. V tomto případě se zabýváme řízením logickým, což je vlastně vyhodnocování pomocí dvouhodnotové veličiny **„log.0 nebo log.1“.** Prvním takovým předpokladem pro dobré navrhnutí řízení je znalost soustavy.

Doporučit postup, či metodiku při programování je obtížné. Existují sice metody s pravdivostními tabulkami, nebo karnaughovou mapou. Ty jsou však účinné jen v případech, kdy řízená soustava není příliš komplikovaná a neobsahuje hodně proměnných, což je stav, který v praxi téměř nenastává. Z mého pohledu bych tedy doporučil, když se dostanete k nějakému komplexnějšímu zařízení si takzvaně **„poodstoupit“** a například si na papír načrtnou, nebo napsat určitý logický postup, jak mají jednotlivé kroky po sobě následovat a poté to zkusil naprogramovat v jazyku, jenž nejlépe ovládám.

3.2.1 Teoretický rozbor úlohy

Budeme programovat logickou úlohu, kde se pokusíme co nejlépe vyřešit automatizaci horkovzdušného tunelu, který je řízen přes PLC ET200SP. Toto PLC pak námi napsaný program bude přehrávat opakovaně v **„cyklech“,** dokud jej nevypneme. [20]

Základní cyklus PLC

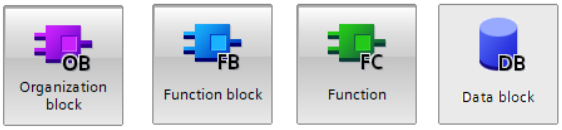


Obr. 3. 3: Cyklus PLC

K jeho naprogramování slouží patřičné softwarové rozhraní, které umí s tímto modelem pracovat. Každá větší firma, jež se touto problematikou zabývá, má ke svým zařízením i vlastní software, tudíž je velice důležité při koupi hardwaru dobře zvažovat, jelikož komunikace mezi zařízeními od jiných společností bývá často doprovázeno s obtížnějším řešením daného problému. Dobrou ukázkou je právě stanice **„CP Lab Drilling“**, která je řízena právě dvěma PLC, kde se každé programuje jiným programem, a navíc ještě musíme řešit komunikaci mezi nimi. Pro naši stanici nám postačí platforma TIA Portál, jenž byl vyvinut speciálně pro moduly od firmy Siemens. Jedná se o rozhraní, které obsahuje všechny potřebné programy. My budeme pracovat pouze s dvěma, a to STEP7 pro programování PLC a WinCC pro naprogramování řídících panelů.

3.2.1.1 Základní principy programování

Samotné programování se provádí v takzvaných blocích, které jsou systematicky spouštěny za sebou v závislosti na jejich důležitosti. Máme zde několik typů bloků:



Obr. 3. 4: Programovací bloky

Organizační blok (OB) je takovým základem komunikace mezi operačním systémem a uživatelským programem. Těchto bloků máme vícero druhů. Například cyklický, který je neustále volán v nekonečném cyklu, nebo startup, to je blok, který je zavolán pouze na startu PLC. Cyklické přerušení je velmi zajímavý blok a sám ho v této úloze využívám, jelikož nám přeruší chod cyklického bloku, vykoná svou funkci a nechá cyklický pokračovat tam kde skončil, to se děje v námi nastavených časových intervalech. Typů tohoto bloku je zde veliké množství a každý z těchto OB má svou číselnou prioritu a podle ní jsou v pořadí volány.

Funkce (FC) tu používáme pro určitou aplikaci, kterou budeme po programu vyžadovat. Tento blok má tu výhodu, že jde napsat tak, abychom jej mohli volat dle libosti a přiřazovat mu pokaždé jiné parametry. Jednoduše řečeno ho používáme pro část programu, která je potřeba volat vícekrát za cyklus.

Funkční blok (FB) má úplně stejné vlastnosti jako Funkce, ale na rozdíl od ní má k sobě připojen vlastní datový blok, který nám uchovává hodnoty při vykonávání daného programu. Využíváme jej tedy tam, kde potřebujeme uchovat určité hodnoty paměti procesu.

Datový blok (DB) je vlastně pouze globální datová oblast, kterou lze použít pro uchování dat ze všech bloků.

Jednotlivé bloky se dají také chránit před vstupem neautorizované osoby a to tak, že buďto dovolíme uživateli do bloku pouze nahlédnout a nic neměnit, nebo mu zakážeme přístup úplně.

Datové typy

Jedná se o formát dat, ve kterém se zapisují hodnoty do paměti. Mají tedy svou maximální velikost a specifickou hodnotu. Vypsal jsem zde pouze ty nejtypičtější, jelikož jich existuje v samotném PLC více druhů [20].

Datový typ	Délka v bitech	Rozsah a číselný formát (MIN/MAX)
BOOL (bit)	1	TRUE/FALSE
BYTE (Byte)	8	B#16#0 až B#16#FF
WORD (Word)	16	W#16#0 až W#16#FFFF
DWORD (Double word)	32	W#16#0000_0000 až W#16#FFFF_FFFF
INT (Integer)	16	-32768 až 32767
DINT (Double Integer)	32	-2147483648 až 2147483647
REAL (Floating-point number)	32	Horní +/-3.402823e+38, Spodní +/-1.175495e-38

Tab. 3. 1: Datové typy PLC

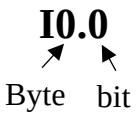
Hardwarové adresování

K vstupům a výstupům z PLC **„I,Q“** je pak možno přiřadit specifický datový typ, který rozhodne o chování a struktuře na dané adrese. Lze tak například adresovat čistě jeden bit výstupu skrze BOOL **„I0.0“**, nebo celý BYTE **„IB0“.**

Merker paměti **„M“** jsou spíše historická pozůstalost z doby, kdy v PLC nebyly datové bloky a vše se ukládalo skrze tyto paměti. [20]

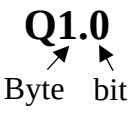
Vstupu:

Jednotlivé adresy jsou přidělovány vstupům po fyzickém připojení k PLC.
IB0 = I0.0, I0.1, I0.2, I0.3, I0.4, I0.5, I0.6, I0.7
IW0 = IB0, IB1 = I0.0, I0.1, …, I0.7, I1.0, I1.1, …, I1.7
ID0 = IW0, IW1 = IB0, IB1, IB2, IB4 = I0.0, …, I3.7



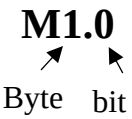
Výstupu:

Jednotlivé adresy jsou přidělovány výstupům po fyzickém připojení k PLC.
QB0 = Q0.0, Q0.1, Q0.2, Q0.3, Q0.4, Q0.5, Q0.6, Q0.7
QW0 = QB0, QB1 = Q0.0, Q0.1, …, Q0.7, Q1.0, Q1.1, …, Q1.7
QD0 = QW0, QW1 = QB0, QB1, QB2, QB4 = Q0.0, …, Q3.7



Paměti:

Lze použít takzvané systémové „memory bity“ pro ukládání dat.
MB0 = M0.0, M0.1, M0.2, M0.3, M0.4, M0.5, M0.6, M0.7
MW0 = MB0, MB1 = M0.0, M0.1, …, M0.7, M1.0, M1.1, …, M1.7
MD0 = MW0, MW1 = MB0, MB1, MB2, MB4 = M0.0, …, M3.7



Tagy v PLC

Jedná se vlastně o přiřazení unikátního názvu k hardwarové adrese v PLC. Pomocí těchto názvů pak lze přistupovat k jednotlivým fyzickým vstupům a výstupům nejen v PLC programu, ale i v HMI (které pak má svůj další vlastní tag list, jenž je navázaný na tento v PLC). Zde v tabulce jsou vypsány tagy, které budou studenti používat při svém programování.

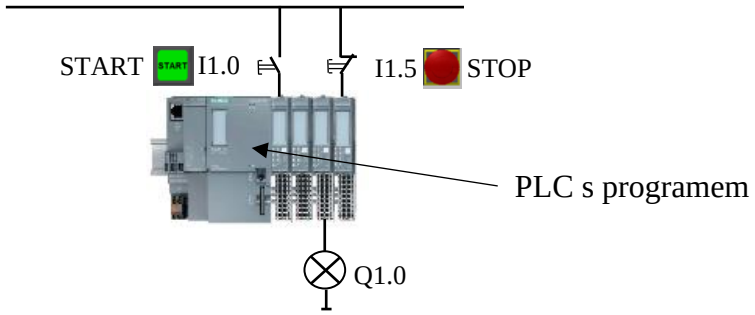
Name	Data type	Address
START	Bool	%I1.0
STOP	Bool	%I1.1
MANU/AUTO	Bool	%I1.2
RESET	Bool	%I1.3
POZICE1	Bool	%I1.4
BEZPEČNOSTNÍ_STOP	Bool	%I1.5
ZAČÁTEK_DOPRAVNÍKU	Bool	%I1.6
KONEC_DOPRAVNÍKU	Bool	%I1.7
POZICE2	Bool	%I42.1
POZICE3	Bool	%I42.2
POZICE4	Bool	%I42.3
STOPER_DOLE	Bool	%I42.7
TeplotaPece	Word	%IW43
Potenciometr1	Byte	%IB45
Potenciometr2	Byte	%IB48
LED_Start	Bool	%Q1.0
LED_Reset	Bool	%Q1.1
LED_1	Bool	%Q1.2
LED_2	Bool	%Q1.3
Pás_Doprava	Bool	%Q1.4
Pás_Doleva	Bool	%Q1.5
Pás_Zpomal	Bool	%Q1.6
Vysun_Stopper	Bool	%Q1.7

Obr. 3. 5: Tag list

3.2.1.2 Volba programovacího jazyka

Předvedeme si názorný příklad rozdílnosti při volbě programovacího jazyka na jednoduché úloze sepnutí LED žárovky horkovzdušného tunelu. Úkol je tedy takový, že pomocí tlačítka START žárovku rozsvítíme, pokud je ale stisknut bezpečnostní STOP, tak žárovka sepnout nepůjde.

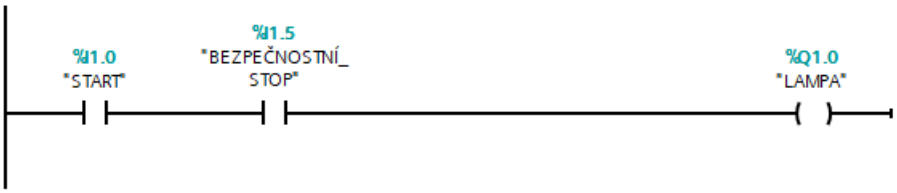
Mechanické zapojení:



Obr. 3. 6: Ukázka zapojení

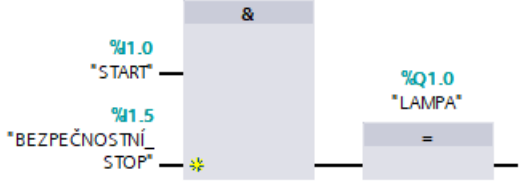
Programové řešení:

Ladder Logic (LAD) – jedná se o grafický programovací jazyk, kde místo textu při programování používáme takzvané symboly. Tento jazyk původem vznikl z důvodu, že většina techniků měla znalosti v elektrických schématech a jednotlivé symboly jsou tak tomu uzpůsobeny. Celá hierarchie programování je vlastně založena na principu reléových schémat. Primárním důvodem použití toho jazyka je tedy přehlednost (jak pro programátora, tak i pro servisní techniky) a jednoduchá čitelnost funkce programu. Na obrázku zde máme v sérii dva spínací kontakty s přiřazenými adresami tlačítek a značku výstupu s adresou připojené lampy.



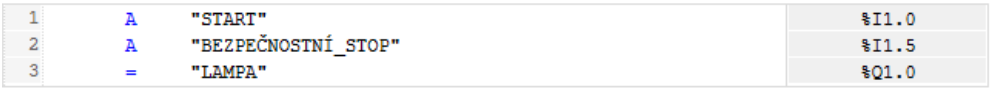
Obr. 3. 7: Ukázka programování LAD

Function Block Diagram (FBD) – jelikož při programování PLC primárně pracujeme tak, že si programujeme vlastní, či volíme systémové funkční bloky (časovače, PID, ...), tak je tento jazyk skvělá volba, jak přehledně tyto bloky volat a přiřazovat jim parametry. Princip zde tedy funguje jako volání funkčních bloků, či funkcí se vstupy a výstupy. Tyto bloky se dají vzájemně kombinovat a vytvořit tak další blok, nebo je jen můžeme vzájemně propojovat. Na obrázku vidíme, že voláme funkci AND a funkci výstupu, které jednoduše propojíme a přiřadíme jim patřičné tagy.



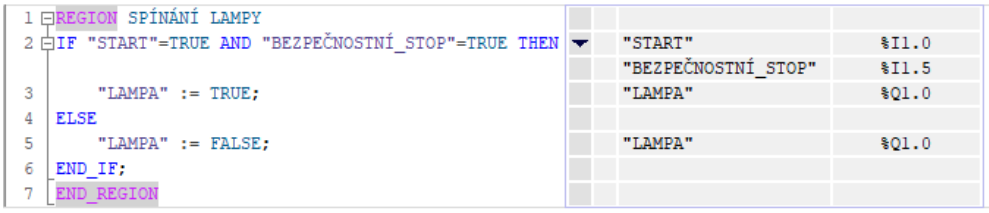
Obr. 3. 8: Ukázka programování FBD

Statement List (STL) – je textový programovací jazyk, který je velice podobný strojovému kódu. Pohybování v programu se pak provádí takzvanými skoky **„JUMP“** a lze zde také volat funkční bloky. Jakožto programovací jazyk je o něco obtížnější a není tak dobře čitelný oproti ostatním metodám. Z historického hlediska se řadí mezi nejstarší způsoby programování PLC. Stále se s ním však jde setkat a používá se, i když už se od tohoto jazyka v poslední době ustupuje.



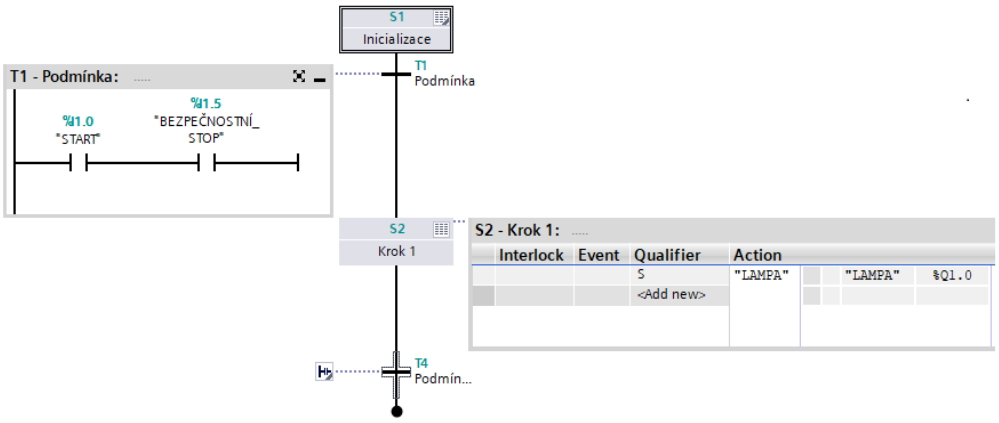
Obr. 3. 9: Ukázka programování STL

Structured Control Language (SCL) – moderní textový programovací jazyk. Oproti STL používá podmínky IF, ELSE, THEN, smyčky FOR, WHILE a tzv. CASE stavy. Tyto funkce jsou velice důležité, když pracujeme a počítáme s velikým množstvím dat. Jako strukturování programu používá SCL takzvané „Regiony“.



Obr. 3. 10: Ukázka programování SCL

Graph – grafické programování ve formě sekvenceru. V každém kroku tak popíšeme, co se má stát a poté přidáme podmínku tohoto kroku (napsanou např. v LAD). V praxi né tak příliš využívané, ale pro určité situace může být výhodné.



Obr. 3. 11: Ukázka programování GRAPH

Dá se říct, že s jakýmkoliv jazykem dokážeme naprogramovat tu samou funkci, je ale otázka, jaký způsob je pro nás nejpříjemnější. Co se týče přehlednosti, tak bych určitě doporučil LAD jazyk, jelikož se dá velice dobře pozorovat v jednotlivých reléových schématech funkčnost programu. Co se týče pracování s daty tak SCL. Velice jednoduchá metoda pomocí základních znalostí programování, které vám ušetří spoustu starostí, například při práci s nepřímou adresací dat v LAD. Hlavní výhoda je, že LAD, SCL a STL, se dají používat a kombinovat v jednom bloku naráz. Co se týče FBD a GRAPH, tak pro jejich použití musíme vytvořit bloky s patřičným jazykem zvlášť.

3.2.2 Programové rozhraní úlohy v TIA Portálu a HMI

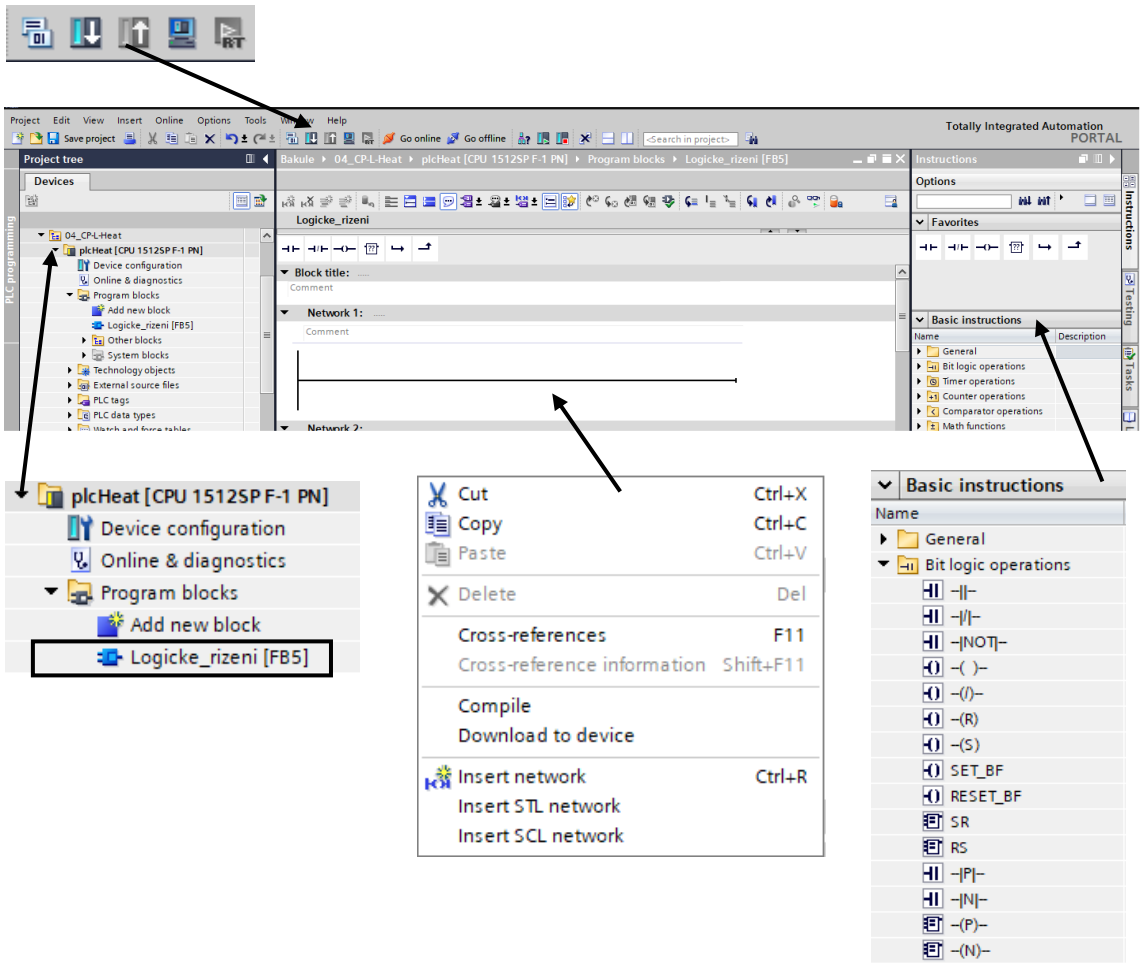
Prostředí tohoto softwaru je vcelku obsáhlé a pustit se do popisu všech funkcí a prvků by zabralo rozsahem na další bakalářskou práci. Proto jsem se rozhodl popsat jen části, které využijí studenti při svém programování. Ostatní věci si jde samozřejmě vyhledat na internetu, kde jsou volně dostupné návody od autorů programu.

Pro programování jsem vytvořil funkční blok **„Logicke_rizeni [FB5]“**, v kterém budou moct studenti tvořit svůj algoritmus horkovzdušného tunelu. Tento funkční blok se nachází v **„plcHeat“**, což je naše ET200SP.

V kategorii **„Program blocks“,** po rozkliknutí našeho **„Logicke_rizeni [FB5]“**, se nám otevře vnitřní struktura tohoto bloku s networky. Tyto networky jsou vlastně řádky (sekce) programu, které PLC postupně v pořadí vykonává. Základní nastavený programovací jazyk je LAD. Když však klikneme pravým tlačítkem do pole networku, otevře se nám možnost přidání **„insert“** dalšího networku a to buď v původním jazyku LAD, nebo STL, či SCL. Pro použití jazyku FBD, či GRAPH bychom museli založit nový funkční blok, jelikož se tyto dva nedají kombinovat s ostatními.

V pravé části obrazovky se nachází pole **„Basic instructions“**, kde je seznam veškerých možných funkcí, které lze při programování použít (např. spínače, negace, cívky, časovače...). Tyto funkce přidáme do našeho networku jednoduchým přetažením pomocí levého tlačítka, k nim už pak jen dopíšeme potřebné adresy.

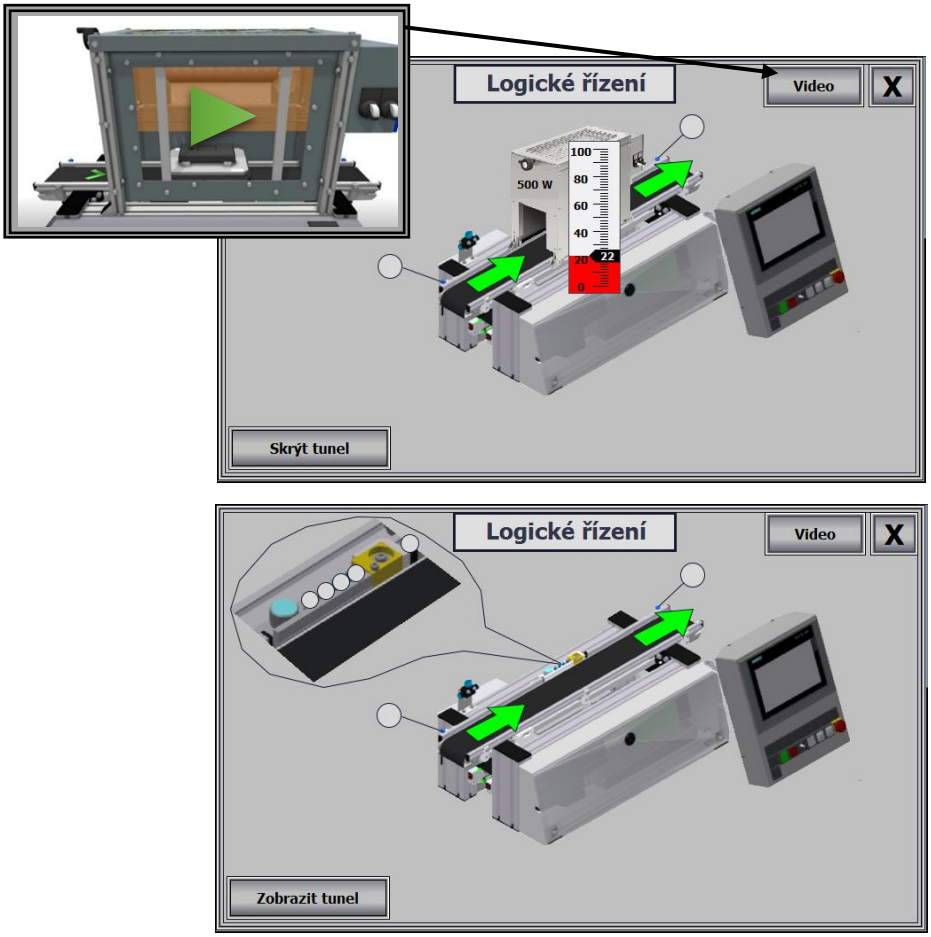
Nahrání programu do PLC pak probíhá přes tlačítka na horní liště, kde klikneme na obrázek se šipkou dolů „“ a vše po úspěšné kompilaci **„kontrole“** potvrdíme. Pokud se nám vyskytne chyba v programu, musíme ji opravit a následně se pokusit o znovu nahrání do PLC.



Obr. 3. 12: Programovací prostředí PLC

Pro úlohu pak máme připravenou obrazovku s modelem tunelu a na něm vyznačenými čidly. Jednotlivé puntíky nám zobrazují, či je čidlo **„senzor“** v log.1 (zelené „ ●“), nebo v log.0 (šedé „ ●“). Jako ukázku principu fungující naprogramované stanice jsem zde přidal tlačítko **„Video“**, které spustí na obrazovce animaci funkční linky, kterou jsem stáhl od výrobců této sestavy.

Směr otáčení dopravního pásu je znázorněn blikajícími zelenými šipkami ve směru jízdy „ ⇦ , ⇨ “. V levém dolním rohu pak máme možnost zobrazení, nebo skrytí aplikační stanice horkovzdušného tunelu. Na něm je ukázána aktuální teplota a sepnutý výkon tunelu.

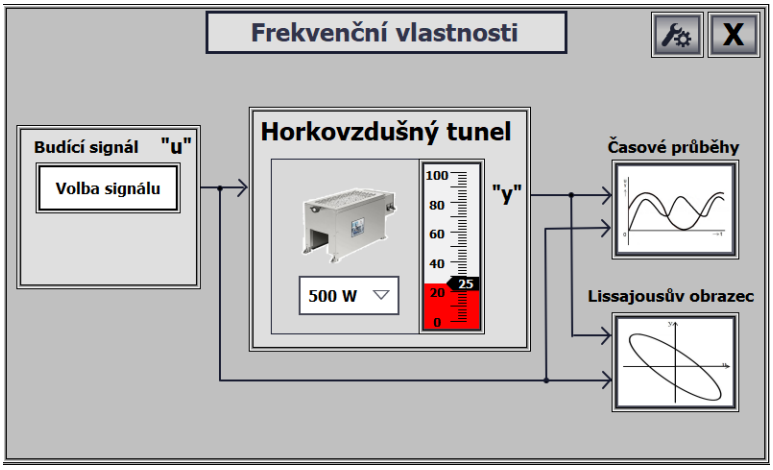


Obr. 3. 13: Grafické rozhraní úlohy: Logické řízení

Programování úlohy ve funkčním bloku **„Logicke_rizeni [FB5]“** jsem zvolil záměrně, a to z bezpečnostních důvodů. Pokud totiž studenti nahrají svůj program do PLC a bude napsán v FB, tak dokážeme řídit jeho spuštění. To by se v případě programování do organizačního bloku nepodařilo.

Jedná se o to, že funkční bloky nemohou fungovat, pokud nejsou zavolány ve výše postaveném bloku. Toto volání my omezíme tím, že je nutno mít spuštěno grafické rozhraní na HMI pro **„Logické řízení“**. Pokud není tato obrazovka otevřena na panelu, blok není volán a naprogramovaná úloha nemůže fungovat. Udělal jsem to z důvodu, že když by byl nahrán nějaký program do PLC a nikdo ho pak nesmazal, mohl by narušovat svým chováním ostatní úlohy, nebo v nejhorším případě poškodit stanici. Je to vlastně i určitý způsob **„bezpečnostního stopu“**, kde při vypnutí obrazovky přerušíme chod programu. Kvůli bezpečnosti ještě navíc uzavřeme veškeré ostatní bloky a funkce, tak aby zde studenti nemohli nic přepisovat a narušit tak funkčnost zařízení.

3.3 Úloha: Frekvenční vlastnosti



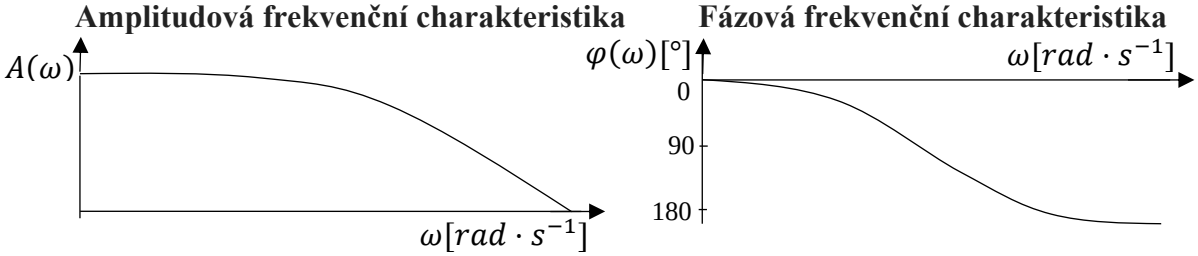
Obr. 3. 14: Grafické rozhraní úlohy: Frekvenční vlastnosti

3.3.1 Teoretický rozbor úlohy

Tato úloha se zabývá metodami pro analýzu chování neznámých soustav, tedy v našem případě **„horkovzdušného modelu“.** To provádíme tak, že za pomocí budícího harmonického signálu **u =A*sin(ωt)** působíme na soustavu a sledujeme její chování skrze vynucený výstup **y =A*sin(ωt + φ)**. Vynucené kmity bývají opožděny za budícím signálem o fázový posuv **„φ = ω · tφ“** s úhlovou frekvencí **„ω = $\frac{2\cdot\pi}{T_p}$“.** [14]

Obr. 3. 15: Buzení soustavy signálem

Jednotlivé změny fázového posunutí mezi signály a jejich amplitudový poměr znázorňujeme do takzvaných **„Bodeho frekvenčních charakteristik“**, které mají svou amplitudovou a fázovou část.



Obr. 3. 16: Bodeho frekvenční charakteristiky

Vztahy pro amplitudový poměr $A(\omega)$:

$$A(\omega) = |G(j\omega)| = \frac{y_A}{u_A} \qquad (3.1)$$

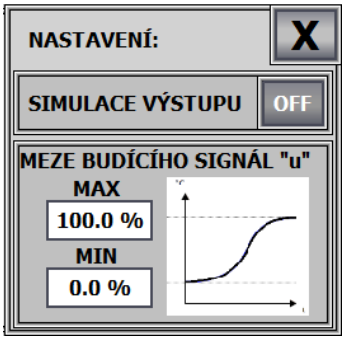
Vztahy pro fázový posuv $\varphi(\omega)$:

$$\varphi(\omega) = \mathrm{argG(j\omega)} \qquad (3.2)$$

(důležité je brát v potaz, že naše soustava je buzena nízkými frekvencemi a tomu také bude odpovídat né příliš vysoké hodnoty „**ω**“)

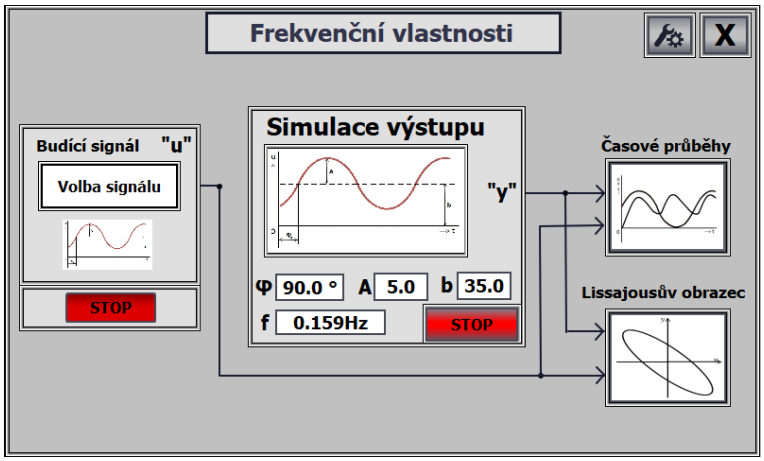
Možnosti nastavení úlohy

Po rozkliknutí ikony nastavení ( ), která se nachází v pravém horním rohu obrazovky, se nám otevře malá obrazovka s možností omezení hodnoty budícího signálu z generátoru a druhou možností nastavení je zapnutí/vypnutí **„Simulace výstupu“**.



Obr. 3. 17: Nastavení úlohy

Simulace výstupu je vlastně určitý **„bypass“** tedy objezd měřené soustavy. Při zapnutí této funkce kompletně odpojíme horkovzdušný tunel a pomocí navolených parametrů harmonického simulátoru budeme generovat výstupní signál. Pomocí této funkce jsme pak schopni otestovat teoretické znalosti a prověřit chování křivek. Využíval jsem funkce simulátoru při generování křivek a obrazců, kde jsem tak ukázal chování různých parametrů rovnic a uviděl jsem to jako takový zajímavý prvek, který by se mohl některým hloubavějším studentům zalíbit. Proto jsem jej sem přidal. Původně zde byla i možnost přepnutí mezi sin, cos, ale rozhodl jsem se tuto možnost odebrat, jelikož si cosinus dokážeme lehce nastavit přes fázový posuv, jenž se zde zadává ve stupních.



Obr. 3. 18: Simulátor výstupu

Kód simulátoru je napsán v SCL a to z toho důvodu, že chci ukázat porovnání v způsobů programování jiným jazykem. Obdobný harmonický signál totiž generuje i náš **„Generátor signálů“,** v němž jsem však použil jazyk LAD.

```
IF #SimulatorVystupu=TRUE  AND #SimulatorVystupuStart = true THEN
    #Vystup_SimVystupu := (#SimVystupu_A*SIN_REAL(2*3.141593*#SimVystupu_f*#Cas
                          + (#SimVystupu_fi0*0.01745329))+#SimVystupu_b);
ELSE
    #Vystup_SimVystupu := 0.0;
    #SimulatorVystupuStart := FALSE;
END_IF;
```

3.3.2 Lissajousův obrazec

Jedné se o jednu z metod pro prověření vlastností soustavy a vytvoření **„Bodeho charakteristik“**. Lissajousova křivka **„obrazec“** je vytvořená dvěma harmonickými signály, promítanými na osách x, y, kde:

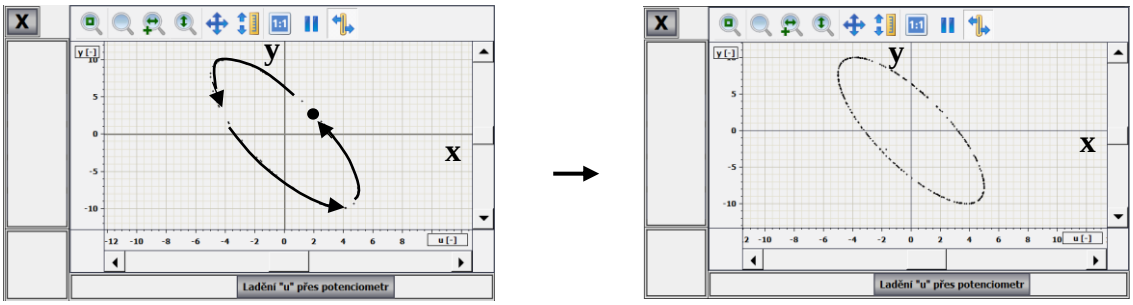
$$x = A_1 \cdot \sin(\omega_1 t + \varphi_1) + b_1 \quad (3.3)$$

$$y = A_2 \cdot \sin(\omega_2 t + \varphi_2) + b_2 \quad (3.4)$$

(sinus muže být nahrazen cosinem)

Co se týče laboratorní úlohy pro studenty, tak na ose **„x“** je zobrazován **„u“** budící harmonický signál z generátoru, který si sami nadefinujeme a na ose **„y“** je zobrazena hodnota teploty soustavy, která je právě vynucená měnícími se požadavky budícího signálu a jejíž chováni je pro nás neznámé.

Vykreslování křivky funguje skrze funkci „**f(x) (TrendView)**“, kde nám **jeden bod jezdí po vypočtených souřadnicích křivky „x, y“** a přibližně co 150 ms zaznamenává do grafu černým puntíkem svou dráhu. Po uplynutí určité doby tedy dostáváme z těchto bodů celkový graf.



Obr. 3. 19: Princip vykreslování f(x) TrendView

Vlivy parametrů rovnic na tvar Lissajousovy křivky

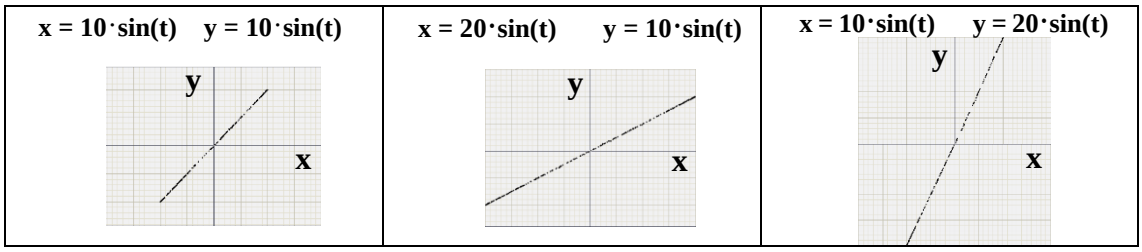
Pomocí generátoru signálu a simulátoru výstupu, jsem vygeneroval křivky různých tvarů, abych poukázal, jaký vliv mají určité parametry rovnice na její tvar. U každého obrazce jsou napsány použité rovnice, s kterými jsem tuto křivku tvořil. [12, 14]

V laboratorním měření, které pak budou provádět studenti, bude známá právě jen rovnice **„x“**, jejíž parametry zadají v generátoru signálu a budou tak zjišťovat chování neznámé **„y“** rovnice, která popisuje soustavu.

Pozor! Tyto mnou volené rovnice slouží pouze jako ukázka fungování lissajousových obrazců. Reálná rovnice soustavy „**y“**, kterou budou studenti hledat má totožnou frekvenci s budícím signálem a velice malý fázový posuv. V reálné situaci je tedy spíše stav **„x = sin(t)“** a **„y = sin(t + φ)“**, kde **φ** nabývá malých hodnot.

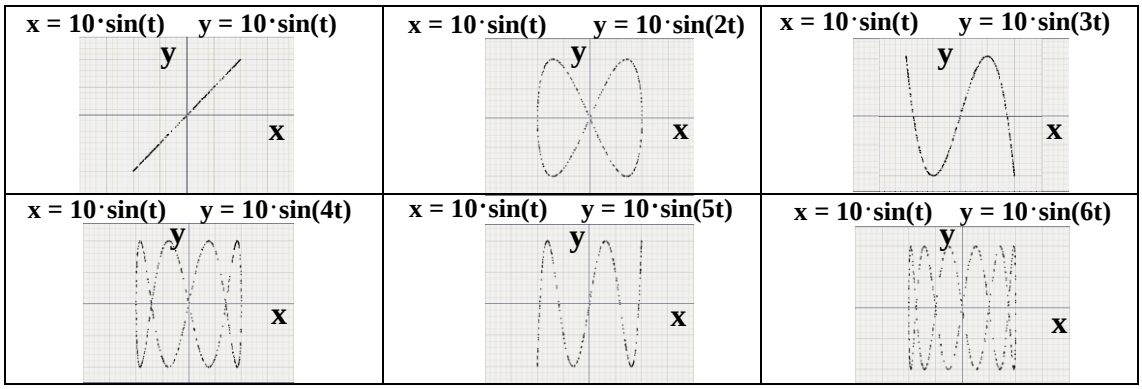
Změny amplitudy „A_1, A_2“

Vidíme jak při změně amplitudy **„A“** měníme šířku a výšku křivky.



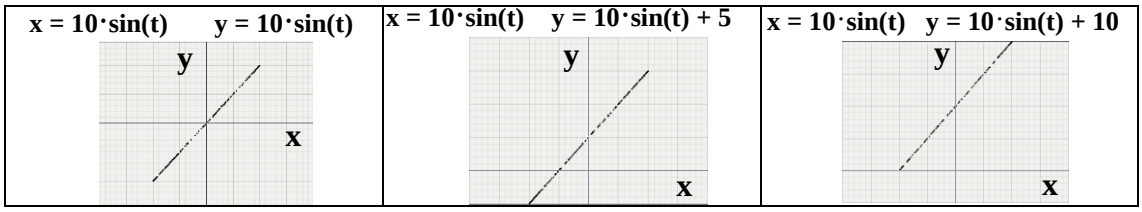
Obr. 3. 20: Lissajousova křivka při změně amplitudy

Vliv změny periody oscilace přes „ω_1, ω_2“
Při změně **„ω“**, tedy nastavení určité frekvence, měním počet kmitů (vrcholů) na ose y/x.



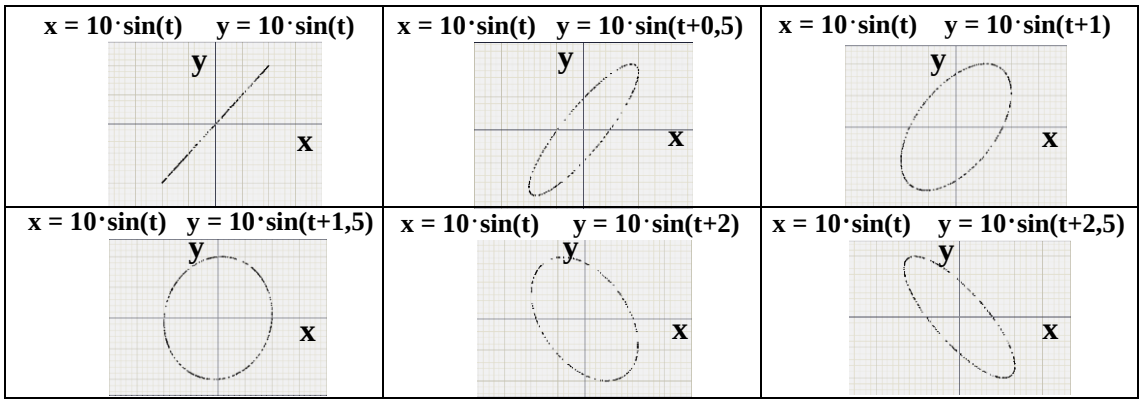
Obr. 3. 21: Lissajousova křivka při změně frekvence

Vliv změny offsetu „b_1,b_2“
Tento proměnný parametr nám zařizuje horizontální, nebo vertikální posuv po ose v závislosti, na které ose tuto hodnotu přičítáme.



Obr. 3. 22: Lissajousova křivka při změně offsetu

Vliv změny fázového posuvu „φ_1, φ_2“
Přidáváme pomocí **„φ“** (v radiánech, kde 1 rad ≈ 57,3°) fázový posuv a tím křivku nakláníme.

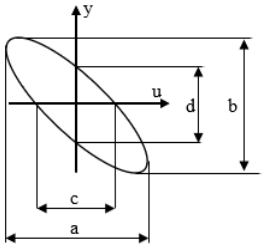


Obr. 3. 23: Lissajousova křivka při změně fázového posuvu

Vztahy pro výpočet

Amplitudový poměr
$$\boldsymbol{A(\omega) = \frac{b}{a}} \quad (3.5)$$

Fázový posuv
$$\boldsymbol{\varphi(\omega) = \arcsin\left(\frac{d}{b}\right) = \arcsin\left(\frac{c}{a}\right)} \quad (3.6)$$



Obr. 3. 24: Parametry Lissajousova obrazce

3.3.3 Časové průběhy

Další z metod, jak prověřit vlastnosti soustavy a vytvořit **„Bodeho charakteristiku“**. Sledujeme zde dva harmonické signály v závislosti na čase **„t“**.

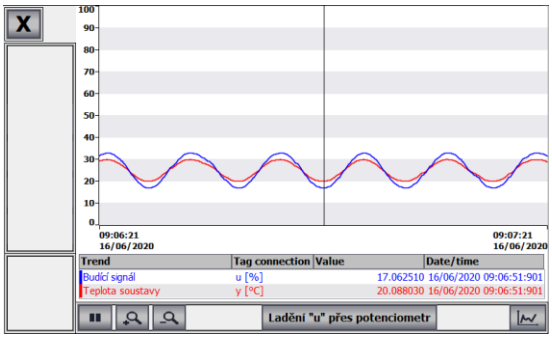
$$x = A_1 \cdot \sin(w_1 t + \varphi_1) + b_1 \quad (3.7)$$

$$y = A_2 \cdot \sin(w_2 t + \varphi_2) + b_2 \quad (3.8)$$

(sinus muže být nahrazen cosinem)

Stejně jako u předchozí metody, tak na ose **„x“** je zobrazován **„u“** budící harmonický signál z generátoru a na ose **„y“** je zobrazena hodnota simulovaného výstupu soustavy. Rozdíl oproti předchozí metodě je tedy pouze v tom, že tyto rovnice zobrazujeme každou zvlášť v časové ose.

Vykreslování křivky funguje skrze funkci „**TrendView**“, kde se nám v závislosti na přibývajícím čase (který řídí PLC) zobrazují co 100ms nové hodnoty v grafu ve formě bodů, které jsou interpolovány **„spojeny“** s předchozími, aby tak křivka vypadala spojitě a hladce. Můžeme si povšimnout určité nepřesnosti ve vykreslování **„zubů“** na křivce. To je způsobeno vcelku nízkou vzorkovací frekvencí, o čemž si povíme níže. Máme zde možnost pomocí posuvné osy měřit hodnoty jednotlivých křivek a změřené hodnoty se nám pak zobrazují společně s časem ve spodní tabulce grafu. Další z možností je zde pak pomocí potenciometru na panelu ovlivňovat **„ladit“** parametry budícího signálu.



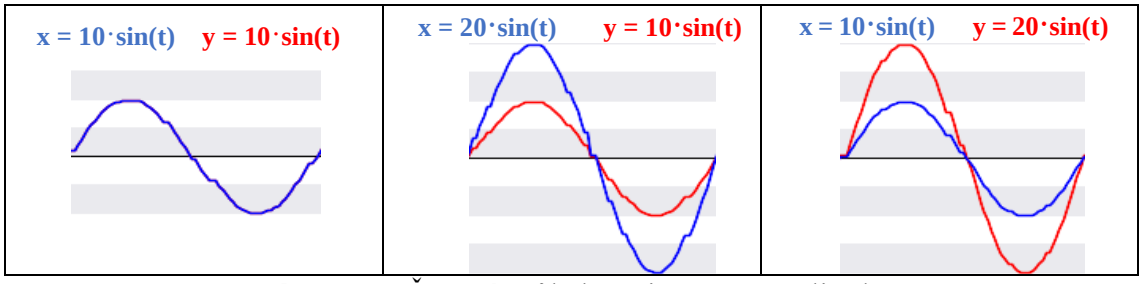
Obr. 3. 25: Princip vykreslování TrendView

Vlivy parametrů rovnic na tvar časových průběhů

Stejně jako u předchozí metody jsem zde vygeneroval křivky, abych ukázal, jaký vliv mají určité parametry rovnice na její tvar a záměrně jsem také zvolil stejné rovnice jako u **„lissajousových obrazců“** pro porovnání těchto metod. U každého obrazce jsou napsány použité rovnice, s kterými jsem tuto křivku tvořil.

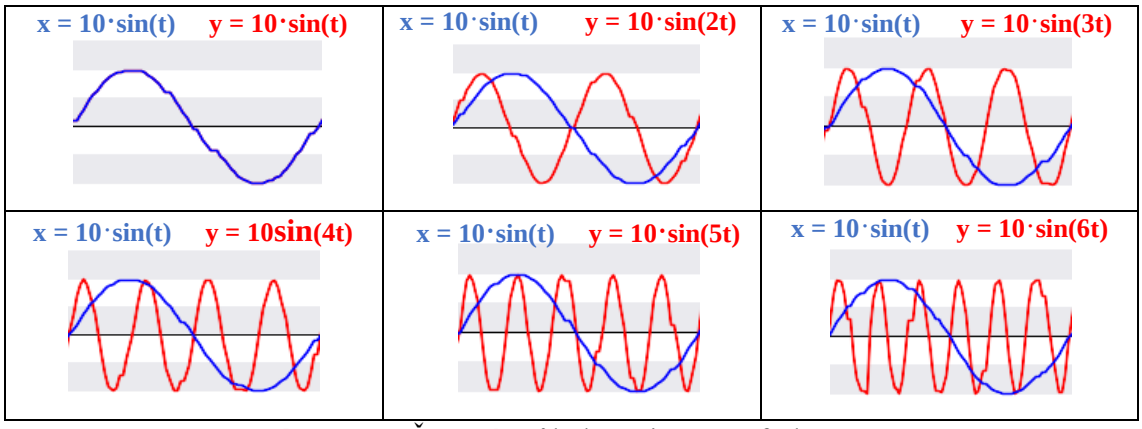
Změny amplitudy „A_1, A_2“

Při změně amplitudy **„A“** ovlivňujeme maximum a minimum probíhající křivky.



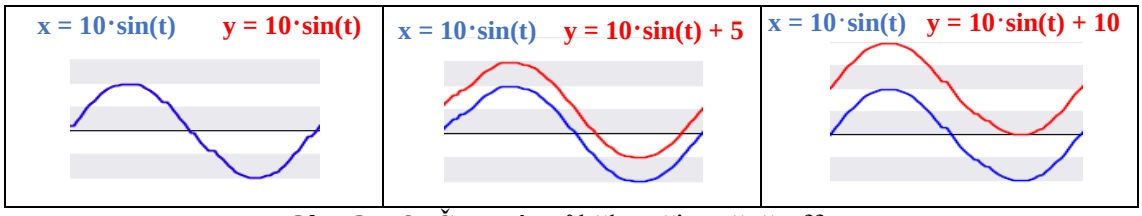
Obr. 3. 26: Časové průběhy při změně amplitudy

Vliv změny periody oscilace skrze **„ω_1, ω_2“**
S přibývající frekvencí a tedy rostoucím „ω“, se nám zužuje perioda signálu.



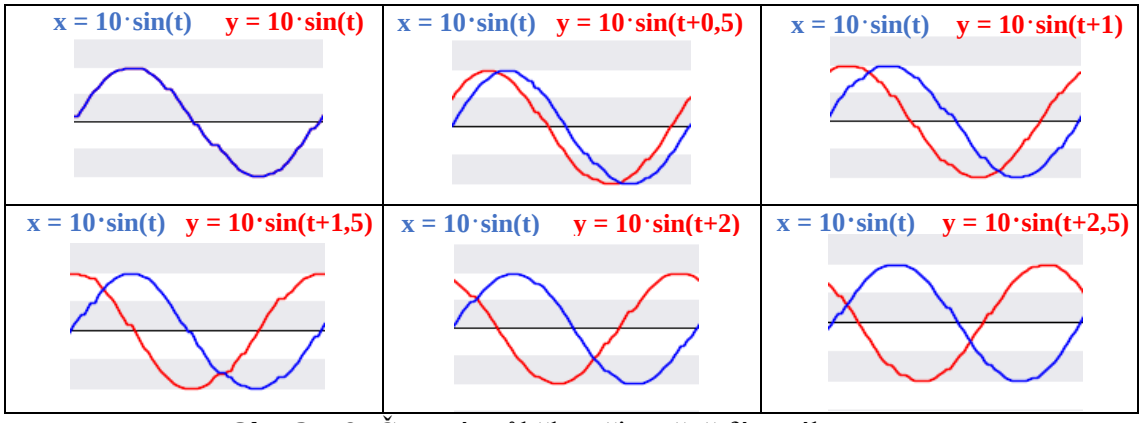
Obr. 3. 27: Časové průběhy při změně frekvence

Vliv změny offsetu „b_1,b_2“
Tento proměnný parametr nám zařizuje možnost posuvu křivky tak, že určíme hodnotu kolem, které bude oscilovat.



Obr. 3. 28: Časové průběhy při změně offset

Vliv změny fázového posuvu „φ_1, φ_2“
Pomocí „φ“, které zde zadáváme v radiánech, zařizujeme fázový posuv mezi křivkami.



Obr. 3. 29: Časové průběhy při změně fázového posuvu

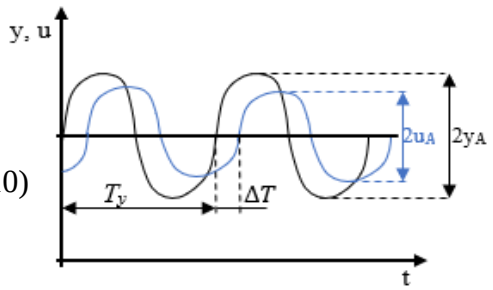
Vztahy pro výpočet:
Amplitudový poměr

$$A(\omega) = |G(j\omega)| = \frac{y_A}{u_A} \quad (3.9)$$

$$A(\omega)_{dB} = |G(j\omega)|_{dB} = 20 \cdot log_{10}|G(j\omega)| \quad (3.10)$$

Fázový posuv

$$\varphi = -\frac{\Delta T}{T} 360° \quad (3.11)$$



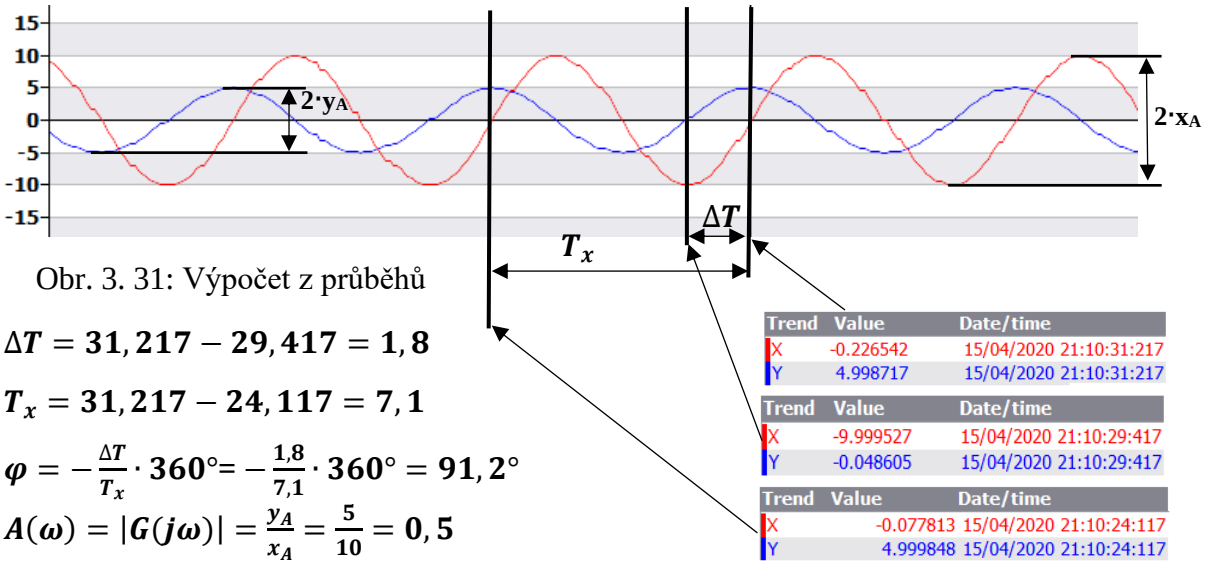
Obr. 3. 30: Parametry časových průběhů

3.3.4 Srovnání a aplikace metod

Na generátoru funkcí jsem si zvolil pro osu **„x“** sinusový signál o amplitudě 10 a pro osu **„y“** volím možnost simulace výstupu, na který pustíme sinus o amplitudě 5. Takto nějak bude vypadat reálné měření, jež budou studenti provádět, s rozdílem, že při malých frekvencích nedochází k tak velkému fázovému posunu, jako jsem si zde sám zvolil (90°).

Časové průběhy:

x = 10·sin(t), y = 5·sin(t+1,57)



Obr. 3. 31: Výpočet z průběhů

$$\Delta T = 31,217 - 29,417 = 1,8$$

$$T_x = 31,217 - 24,117 = 7,1$$

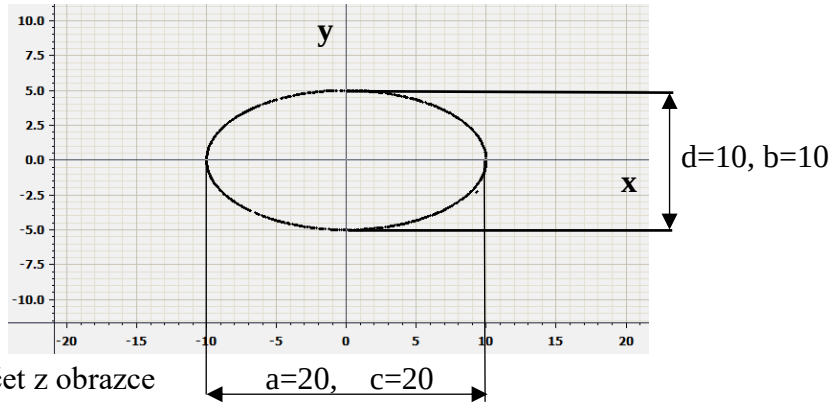
$$\varphi = -\frac{\Delta T}{T_x} \cdot 360° = -\frac{1,8}{7,1} \cdot 360° = 91,2°$$

$$A(\omega) = |G(j\omega)| = \frac{y_A}{x_A} = \frac{5}{10} = 0,5$$

U měření je důležité si dávat pozor na správné umístění posuvné měřící osy.

Lissajousův obrazec:

x = 10·sin(t), y = 5·sin(t + 1,57)



Obr. 3. 32: Výpočet z obrazce

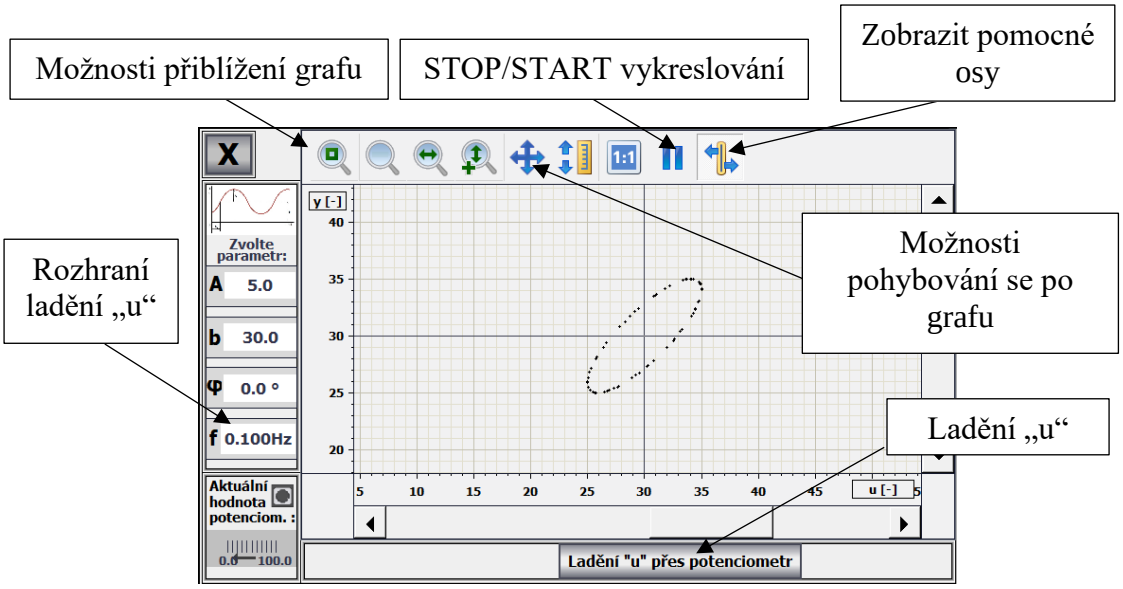
$$A(\omega) = \frac{b}{a} = \frac{10}{20} = 0,5$$

$$\varphi(\omega) = \arcsin\left(\frac{d}{b}\right) = \arcsin\left(\frac{c}{a}\right) = \arcsin\left(\frac{20}{20}\right) = 90°$$

Výsledek v obou případech vyšel správně a odpovídá našemu simulovanému výstupu. Při počítání posuvu z časových průběhů však vidíme, že se lišíme o 1°. To může být způsobeno buďto nekvalitním odečtením, nebo špatným nastavením polohy měřící osy. Také je zde určitý problém, co se týče kvality zobrazování křivky, jelikož vzorkovací frekvence je bohužel vcelku omezená. Každopádně bych měření nazval úspěšným. Oba dva způsoby se prokázaly jako funkční a je už tedy na studentovi, po kterém sáhne.

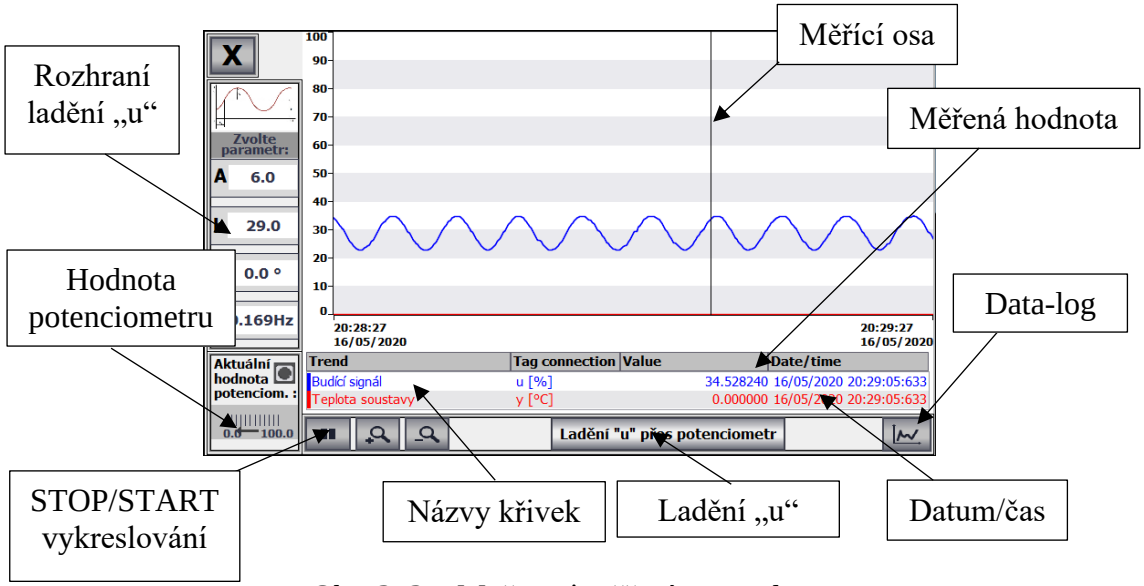
3.3.5 Možnosti zobrazení a měření charakteristik

Měření přes **f(x) Trendview** využíváme jen u úlohy s frekvenčními vlastnostmi. Zobrazení hodnoty vykresleného bodu jde skrze přidržení místa na obrazovce, musíme však mít vypnuty ostatní prvky pro přiblížení, nebo s pomocnou osou a jednoduchým odečtením **„od oka“**. Počet záznamových bodů je teoreticky nekonečný, tedy body stále přibývají s časem. Jedinou nevýhodou zde je, že pokud vypneme toto okno s měřením, tak se graf smaže a budeme muset znovu čekat na jeho vykreslení při dalším spuštění.



Obr. 3. 33: Možnosti měření s f(x) Trendview

Měření přes funkci **Trendview** využíváme v druhé i třetí úloze. Máme funkci pro měření probíhajících hodnot křivek **„tabulka s měřící osou“**. Jen je třeba upozornit, že záznam není nekonečný a dokáže zobrazit jen přibližně 100 sekund průběhu. Tedy od sté sekundy se začne starší záznam přemazávat novým. Pro zobrazení celého průběhu měření zde máme funkci **Data-log**, která do sebe ukládá až dvě hodiny probíhaného měření.



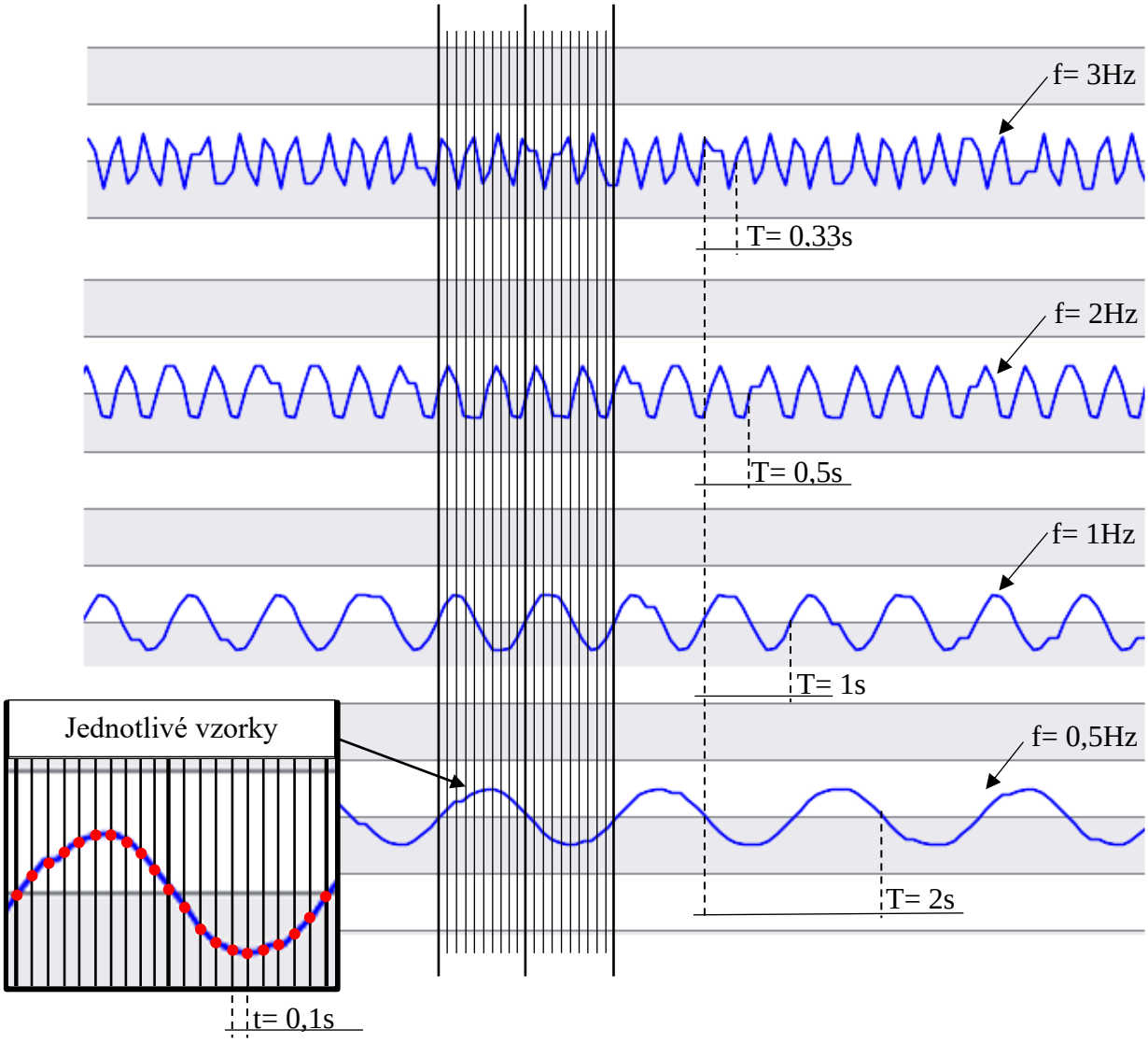
Obr. 3. 34: Možnosti měření s Trendview

Měřené hodnoty a čas v tabulce odpovídají místu, kde křivky **protínáme pomocí měřící osy**. Datum a čas je zde napsán ve formátu: D/M/ROK HOD/MIN/SEK/MS

3.3.6 Problémy se vzorkováním

Jinak řečeno, problém s nekvalitním zobrazováním signálu. Jako názornou ukázku jsem si zvolil generátor budícího signálu s volbou harmonické funkce sinus. Tento signál bude mít ve všech případech stejné nastavení. Jediným prvkem, který budeme měnit bude jeho frekvence **„f“.**

Kde je tedy skryt kámen úrazu? Je schován v jednom velice známém problému, co se zobrazování signálu týče a souvisí se **vzorkovací frekvencí**. Tato frekvence nám určuje, jak často můžeme zaznamenat aktuální hodnotu signálu za určitou časovou dobu. Co se týče zobrazování našich signálu na operátorském panelu HMI, tak nejlepší vzorkovací perioda, kterou můžeme zvolit je **100ms = 0,1s**. To znamená, že vkládáme hodnoty z PLC generátoru do HMI grafu rychlostí desetiny sekundy (zobrazeno červenými puntíky níže). To nám odpovídá vzorkovací frekvenci **10Hz**. S touto frekvencí se již bohužel nedá hýbat a není šance ji v PLC jakýmkoliv způsobem zvýšit.



Obr. 3. 35: Ukázka vzorkování signálu o různých frekvencích v HMI

Na grafu můžeme vidět, že u nízkých frekvencí nenastává žádný problém. Při zvolené frekvenci sinusoidy **f=0,5Hz**, máme signál o délce periody (**T=2s**), vykreslen pomocí **dvaceti vzorků**. To bohatě postačuje pro kvalitní zobrazení.

Problém však nastává, když budeme frekvenci signálu zvyšovat, a tedy tak nepřímou úměrou velikost periody zužovat. Pro **f=1Hz** již odpovídá tvar signálu na periodě pouze **deseti vzorkům**, signál buzen frekvencí **f=2Hz** je už tvořen pouhými **pěti vzorky**. Tímto bych chtěl ukázat důvod, proč je dobré nepřesáhnout tuto frekvenci 2Hz, pokud budeme pracovat s HMI. Jelikož mi to přijde jako hraniční hodnota, co se týče ještě určité obstojné kvality zobrazení signálu.

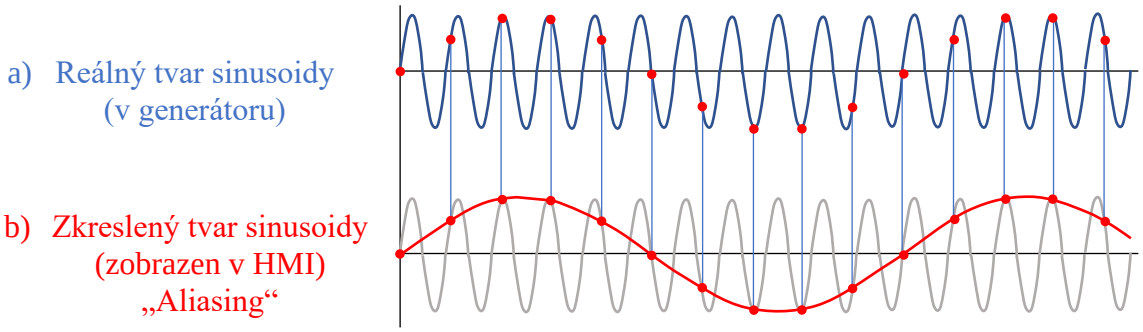
Toto chování je pokryto teorií a pravidly, které je třeba si vysvětlit. Hlavní kritérium při volbě frekvence je **Shannonův teorém** (někdy nazývaný Nyquistův-Shannonův, nebo Shannonův-Nyquistův-Kotělnikovův teorém). Ten nám říká, že pro kvalitní rekonstrukci spojitého signálu potřebujeme minimálně dvojnásobnou vzorkovací frekvenci, než je frekvence signálu snímaného.

$$f_{vzorkovací} \geq 2\, f_{signálu} \tag{3.12}$$

Nedodržením této podmínky pak riskujeme, že budeme zobrazovat zkreslený, a tedy nereálný signál tzv. **aliasing**. Tento jev je nebezpečný v tom, že hodnoty, co vidíme zobrazené na grafu, vlastně vůbec neodpovídají realitě.

Problém je tedy v zachycení celé periody signálu co možná nejpřesněji, když totiž máme příliš malou vzorkovací frekvenci oproti frekvenci signálu, tak zachytíme jen pár bodů z ní a zrealizujeme nesmysl!

Na obrázku můžeme vidět vzorkování signálu **„a)“** pomocí nízké vzorkovací frekvence, která určitě neodpovídá podmínce výše uvedené, jelikož na periodu signálu připadá přibližně **jeden vzorek**. Z toho nám pak vzniká tvar signálu **„b)“**, který se vygeneruje z těchto náhodně zachycených bodů.



Obr. 3. 36: Zobrazení chybného vzorkování

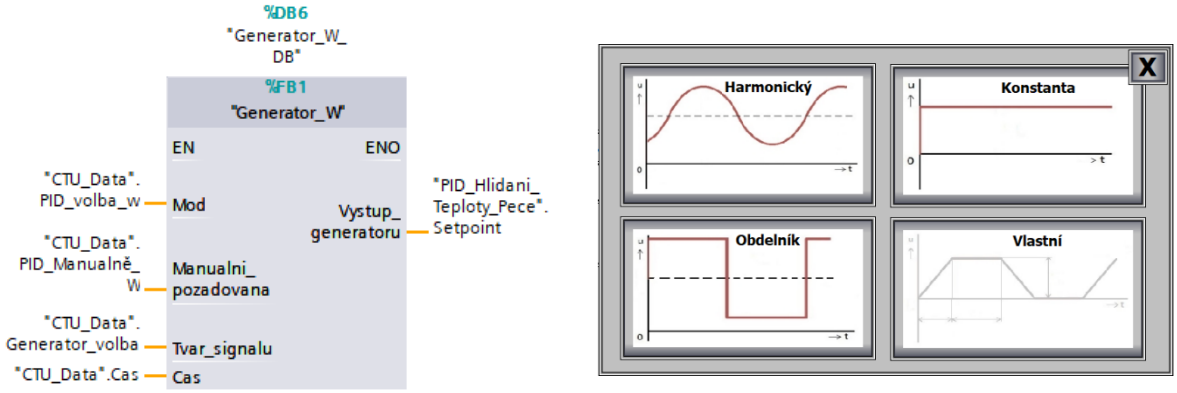
Pojďme však zpátky k našemu měření. Přesto, že je podmínka vzorkování dodržena, tak signál při frekvenci **3Hz**, což odpovídá podmínce poloviční hodnoty oproti vzorkování **10Hz**, tak je tvar signálu stále velice zkreslený. Jedná se o to, že samotné PLC není zkonstruováno k těmto účelům a nemůžeme tedy očekávat, že bude zvládat stejné činnosti jako osciloskop. Ten má pro tyto případy různé metody vzorkování a filtry, s kterými i při frekvenci, která se blíží podmínce tohoto teorému, dokáže vykreslit hezký graf. Pokud bychom chtěli my něco takového dokázat, tak je zapotřebí pořídit nějaký měřící hardware navíc.

Co se týče našeho měření, tak při regulaci teploty horkovzdušného modelu příliš vysokou frekvenci budícího signálu nepotřebujeme, jedná se totiž o velmi pomalou reakci soustavy na žádanou hodnotu, kterou tímto signálem nastavujeme. Příliš vysoké frekvence by naše soustava nezvládala zpracovat.

3.3.7 Generátor signálu

V druhé a třetí laboratorní úloze potřebujeme generovat určitý signál. V případě frekvenčních vlastností to bude **„signál budící“** a pro uzavřený regulační obvod zase **„žádaná hodnota“**. Z tohoto důvodu jsem se rozhodl vytvořit generátor, který bude mít na výběr ze čtyř plně nastavitelných signálů.

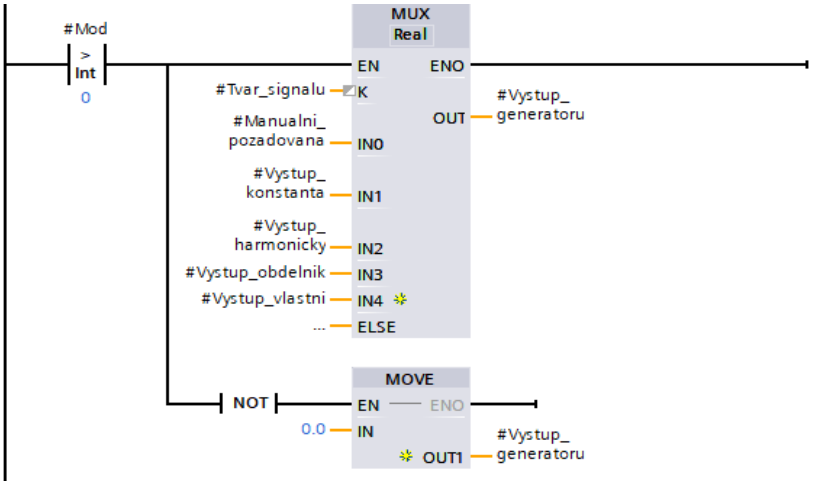
Jedná se vlastně o funkční blok FB1, jenž je volán v hlavním organizačním bloku OB1. Jeho vnitřní struktura je napsaná v LAD a SCL jazyce. Pro tyto jazyky jsem se rozhodl z toho důvodu, že přehlednost v programování je velice dobrá. A proč LAD namísto STL? Kód by sice byl kratší, avšak pro člověka s menšími zkušenostmi v programování méně čitelný. Po stisknutí tlačítka **„Volba signálu“** se nám otevře vyskakovací okno s čtyřmi možnostmi (Konstanta, Obdélník, Harmonický a Vlastní).



Obr. 3. 37: Generátor signálu: Funkční blok a jeho grafické rozhraní

Blok má celkem čtyři vstupy (Mod, Manualni_pozadovana, Tvar_signalu, Cas) a jeden výstup. Tyto vstupní hodnoty zde posíláme z globálního datového bloku (CTU_Data). Samotná funkce, jelikož se jedná o funkční blok, tak má samozřejmě také svůj vlastní datový blok (Generator_W_DB), kde se uchovávají patřičné parametry generátoru a hodnoty výpočtů.

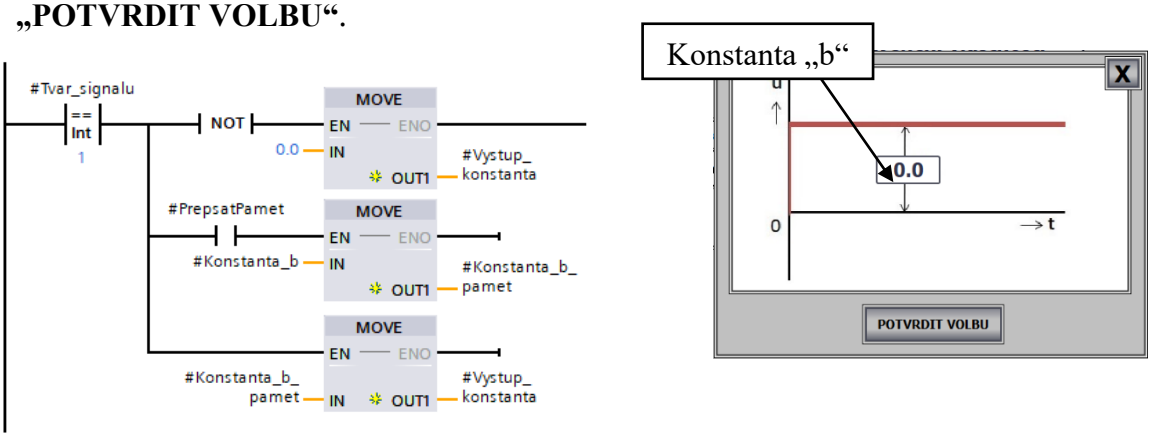
První network, neboli řádek v programu slouží pro volbu signálu. Dle hodnoty **„Mod“** rozhodneme, v jakém režimu má generátor fungovat (0-Neaktivní, 1-Manuální režim, 2-Generátor). Pokud je zadán manuální režim, tak na výstup přivedeme číselně zadanou hodnotu z obrazovky **„Vlastní hodnota“**. Pokud zvolíme generátor, tak se čeká na volbu **„Tvar signálu“** skrze obrázky (0-Manuální, 1-Konstanta, 2-Sinus, 3-Obdelník, 4-Vlastní), jenž poté připne multiplexor MUX na výstup. Multiplexor funguje tak, že podle hodnoty **„K“** posílá patřičný IN na OUT (např. jestli K=1 tak IN1 =OUT)



Obr. 3. 38: Generátor signálu: Volba skrze multiplexor

3.3.7.1 Konstanta

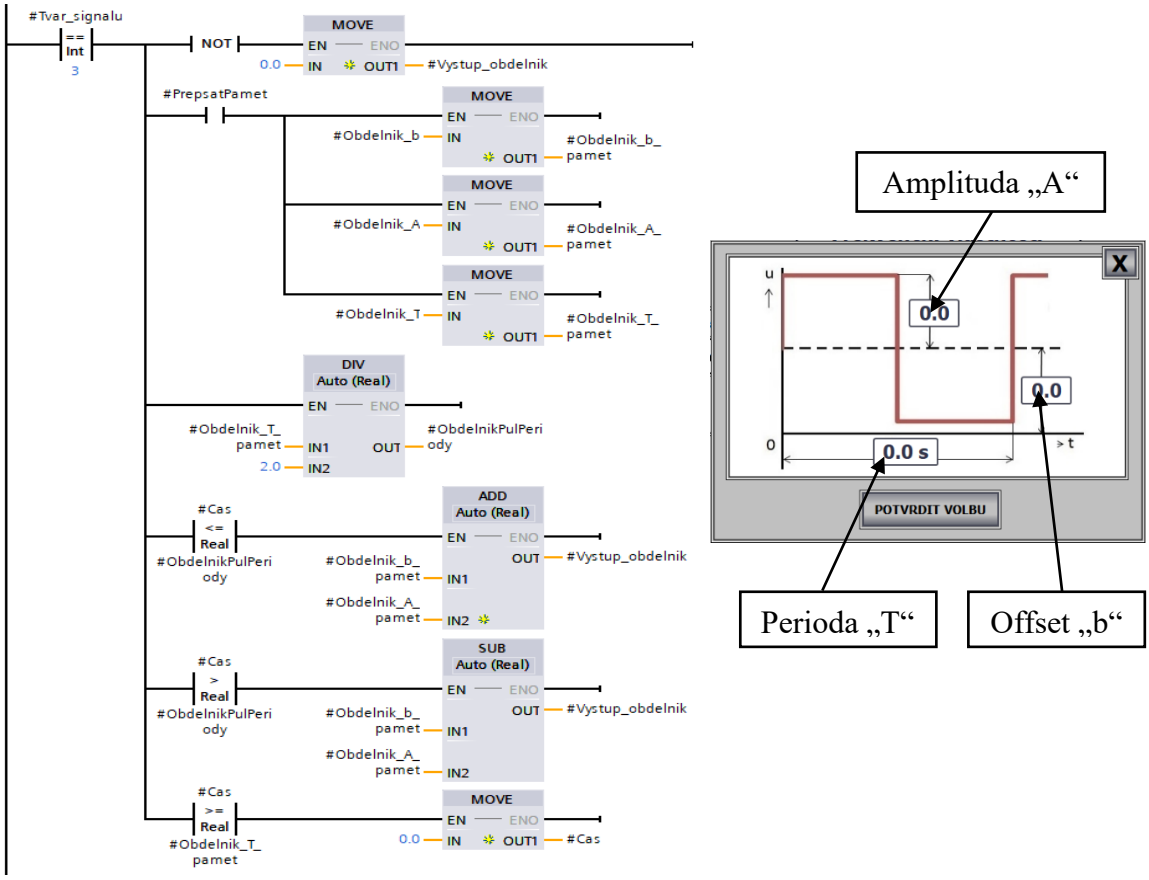
Pro hodnotu proměnné **„Tvar signálu = 1“** se dostaneme do networku s konstantním signálem. Jedná se o nejjednodušší druh signálu, pouze zde přečteme zadanou hodnotu v HMI, která se však nahraje na výstup generátoru až po stlačení tlačítka **„POTVRDIT VOLBU“**.



Obr. 3. 39: Konstanta: Network a jeho grafické rozhraní

3.3.7.2 Obdélník

Pro hodnotu proměnné **„Tvar signálu = 3“** jdeme do networku s obdélníkovým tvarem signálu. Ten je tvořen třemi volenými parametry v HMI (Amplitudou, Offsetem a Periodou). Generátor poté co dostane **„POTVRDIT VOLBU“**, periodu rozpůlí a v první polovině hodnotu amplitudy k offsetu přičítá a v druhé polovině zase odečítá, přičemž na konci periody časovač vynuluje a jede znovu. Pracujeme přitom s časem **„Time“**, který je brán z globálního datového bloku **„CTU_Data“.**



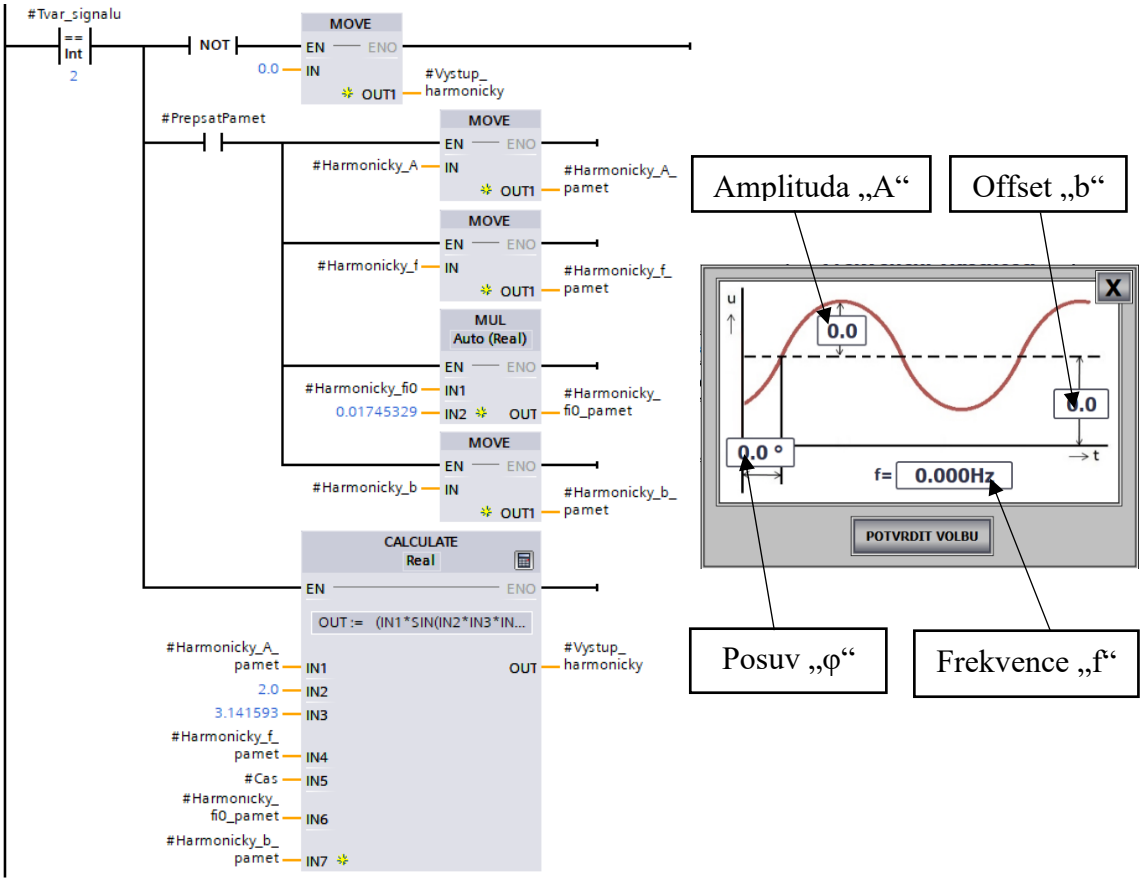
Obr. 3. 40: Obdélník: Network a jeho grafické rozhraní

3.3.7.3 Harmonický

Pro hodnotu proměnné **„Tvar signálu = 2“** aktivujeme network s harmonickým signálem. Zde máme sinus, který generujeme přes matematický blok **„Calculate“**, do kterého lze vkládat libovolnou rovnici poskládanou ze vstupů na tomto bloku. Obdobný program, akorát napsaný v jiném jazyce jsme mohli vidět u **„simulátoru výstupu“**.

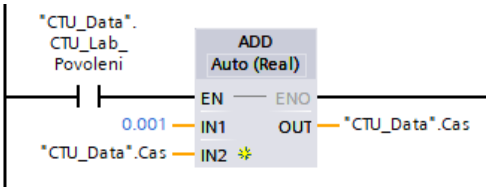
Použitá rovnice: **OUT = (IN1*SIN(IN2*IN3*IN4*IN5 + IN6) + IN7)**

Což v překladu s našimi parametry, které jsou Amplituda, Frekvence, Posuv a Offset, tvoří rovnici: **Y = A*sin(2*π*f*t + f0) + b**
Veškeré parametry jsou zde vloženy skrze HMI a následným **„POTVRDIT VOLBU“** nahrány do generátoru. Fázový posuv je zde přepočten, aby šel zadávat ve stupních.



Obr. 3. 41: Harmonický: Network a jeho grafické rozhraní

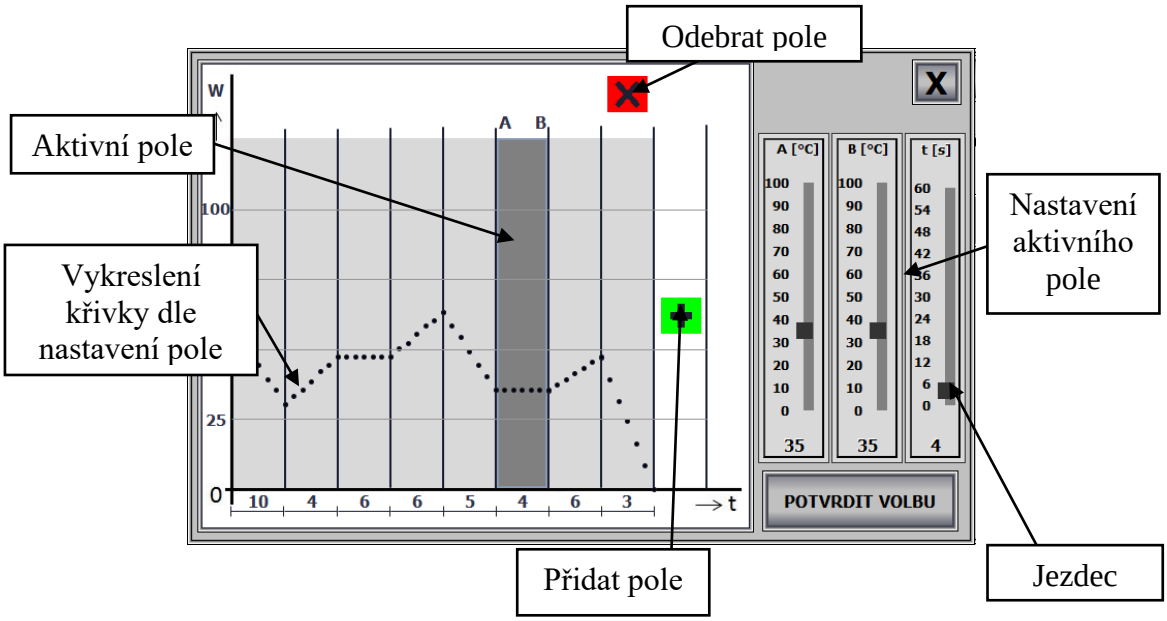
Jedinou výjimkou je zde čas **„t“**, ten si generujeme zvlášť v cyklicky volaném OB a jehož hodnotu ukládáme do globálního datového bloku **„CTU_Data“**. Blok samotný voláme každou milisekundu, přičemž pokaždé přičítá hodnotu 0,001 a vytváří tak čas s přesností na tři desetinná místa. Tento čas pak využíváme ve všech signálech.



Obr. 3. 42: Generátor času

3.3.7.4 Vlastní

Pro hodnotu proměnné **„Tvar signálu = 4“** přeskočíme do networku s vlastním tvarem signálu. Generátor může mít celkově až devět nastavitelných polí, kde každé pole má maximální periodu trvání 60ti sekund. Tvar signálu je zde tvořen pomocí dvou posuvných jezdců **„A, B“**, které mají maximální nastavení 100°C). Periodu **„t“** také nastavujeme pomocí jezdce, ale je zde pouze v číselné reprezentaci a na tvar křivky signálu v tomto editoru nemá žádný vliv. Jak toto vykreslování křivky v editoru funguje si vysvětlíme níže, teď ale přejděme k funkcionalitě generátoru.



Obr. 3. 43: Vlastní: Grafické rozhraní editoru

Po přidání pole se nám otevře možnost nastavování hodnot aktivního pole pomocí jezdců. Pokaždé tedy nastavujeme pouze aktivní pole **(tmavě zvýrazněné).** Hned po nastavení polohy jezdce se parametry uloží a vykreslovací funkce pomocí bodů vyznačí naši volbu křivky.

Mezi těmito poli lze libovolně přepínat pomocí dotyku na obrazovce, přičemž se nám každé nastavení jednotlivých polí ukládá do datového bloku generátoru, kde máme připraveno datové rozhraní ve formě **Array[0..9]** tvořené z takzvaných UDT datových struktur. Zde máme strukturu **„Pole“** a v ní své tři nastavitelné parametry. Funguje to tak, že každé pole má svou číselnou hodnotu. Jakmile je aktivní například první pole, tak se nastaví **„Aktivni_pole=1“** a hodnoty jezdců se nám začnou ukládat do **„VlastniHodnoty[1]“** pole. Při každém přepnutí mezi poli se nám tak přiřadí patřičná datová oblast. Pole nula je pouze inicializační a nepoužívá se.

```
IF #ZmenaPole = FALSE THEN
    #VlastniHodnoty[#Aktivni_pole]:= #HodnotyJezdcu;
ELSE
    #HodnotyJezdcu := #VlastniHodnoty[#Aktivni_pole];
END_IF;

IF #Tvar_signalu = 4 AND #PrepsatPamet = TRUE THEN
    #VlastniHodnotyPamet := #VlastniHodnoty;
    #SumaPoliPamet := #Suma_poli;
END_IF;
#VypocetniHodnota := #VlastniHodnotyPamet[#Pole];
```

VlastniHodnoty	Array[0..9] of "Pole"
VlastniHodnoty[0]	"Pole"
Hodnota_teploty_A	DInt
Hodnota_teploty_B	DInt
Hodnota_casu_t	DInt
VlastniHodnoty[1]	"Pole"

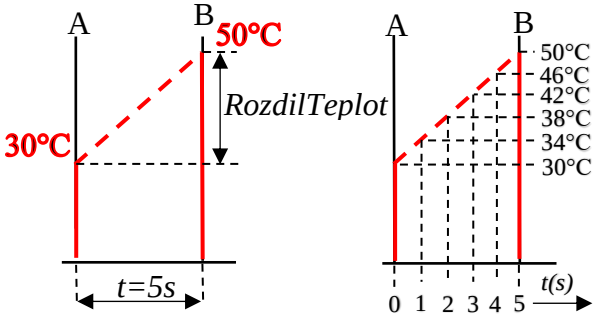
Obr. 3. 44: Vlastní: Network pro ukládání dat

Aby se nám však při každém přepnutí mezi aktivními poli nesmazaly hodnoty nastavení jezdce, jelikož jezdec vždy ukazuje pouze svou nastavenou hodnotu a nikoliv nastavení pole (na kterém se nachází). Poradil jsem si tak, že dle čísla aktivního pole nahrajeme hodnotu z patřičného **Array** do nastavení jezdce.

Pokud už tedy máme vytvarovaný signál dle naší představy, tak pomocí tlačítka **„POTVRDIT VOLBU“** veškerá data přeneseme do další **Array[0..9]** paměti o stejné velikosti s názvem **„VlastniHodnotyPamet“**, která slouží jako základna pro veškeré výpočty. Do tohoto pole se hodnoty nahrají jen vždy po potvrzení, tudíž jakákoliv změna parametrů v editoru bez výsledného potvrzení, nemá žádný vliv na probíhající generovaný signál. Tuto funkci jsem nastavil u všech generátorů signálů, aby tak nedocházelo k chybám z nešikovnosti při manipulaci s HMI.

Jakmile tedy má generátor potřebná data, začne je zpracovávat. Jako první krok si převede parametry **„A, B“** do REAL hodnot. Jezdec nám totiž dává celočíselnou hodnotu INT a pro náš případ se přesněji pracuje s čísly v reálných hodnotách.

Provedeme výpočet rozdílu maximálních hodnot jezdců s teplotami **„RozdilTeplot“** v absolutní hodnotě, tento rozdíl podělíme zvolenou periodou a vznikne nám tak velikost přírůstku **„HodnotaRustu“** za jednu sekundu.



Růst po 1s (pro periodu 5s):

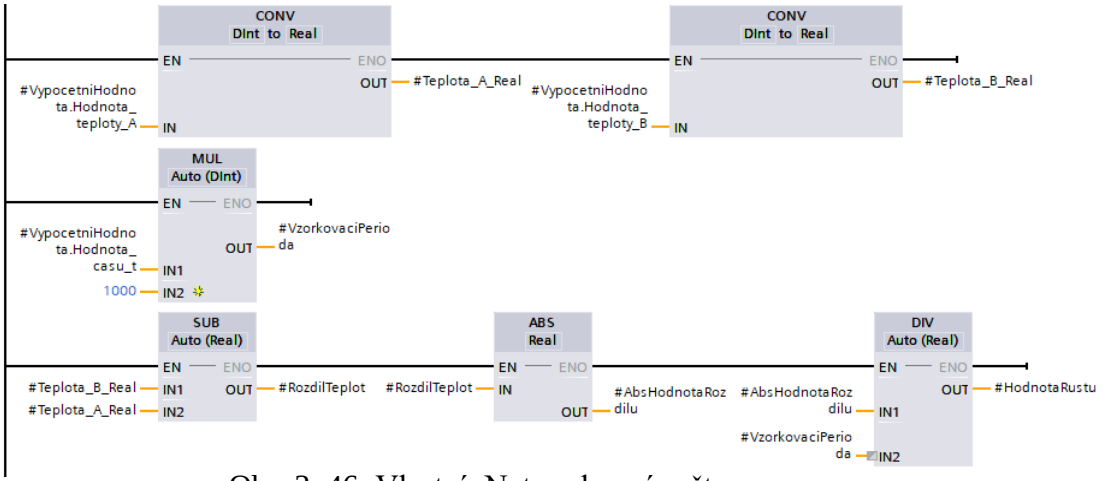
$$HodnotaRustu = \frac{RozdilTeplot}{t} = \frac{|B-A|}{t} = \\ = \frac{|50-30|}{5} = \mathbf{4°C}$$

Růst po 1ms (pro periodu 5s):

$$HodnotaRustu = \frac{RozdilTeplot}{t} = \frac{|B-A|}{t} = \\ = \frac{|50-30|}{5*1000} = \mathbf{0,004°C}$$

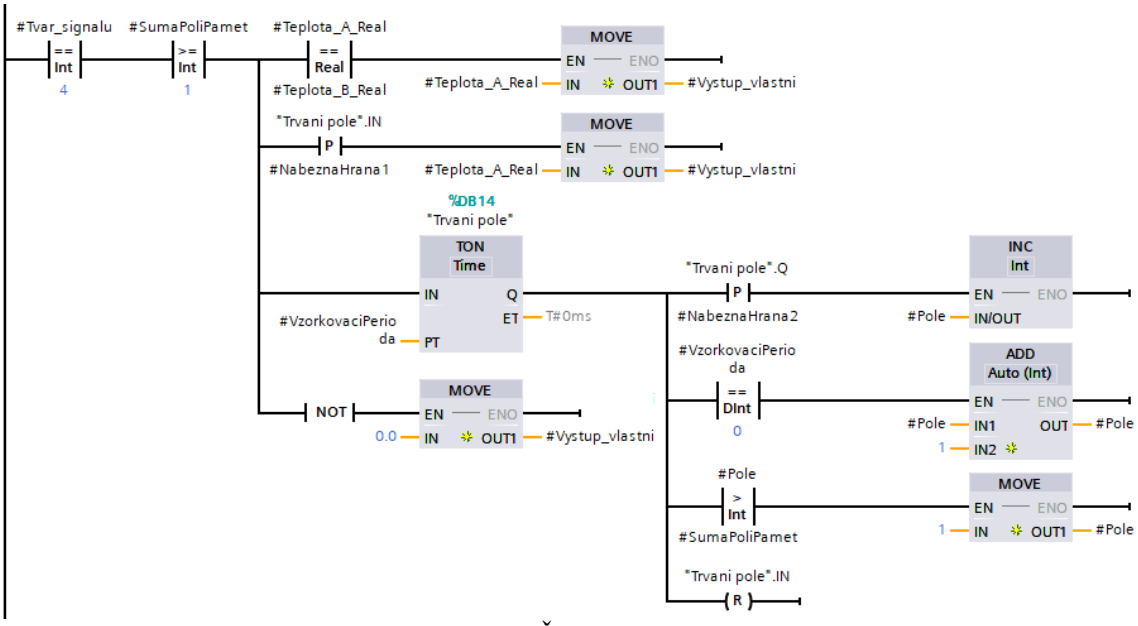
Obr. 3. 45: Vlastní: Princip výpočtu

Pokud by naše vzorkovací schopnost byla 1Hz, tak je to ideální stav a pokračujeme dál. Jenomže možnosti zobrazování jsou zde lepší a to takové, kde na HMI lze dosáhnout až 10Hz a v rozhraní Trace dokonce 1000Hz přesnosti. Z tohoto důvodu přepočteme dané časy na nejvyšší možnou rychlost vzorkování a roznásobíme tak hodnotu periody 1000x, abychom dostali vzorek pro každou 1ms.



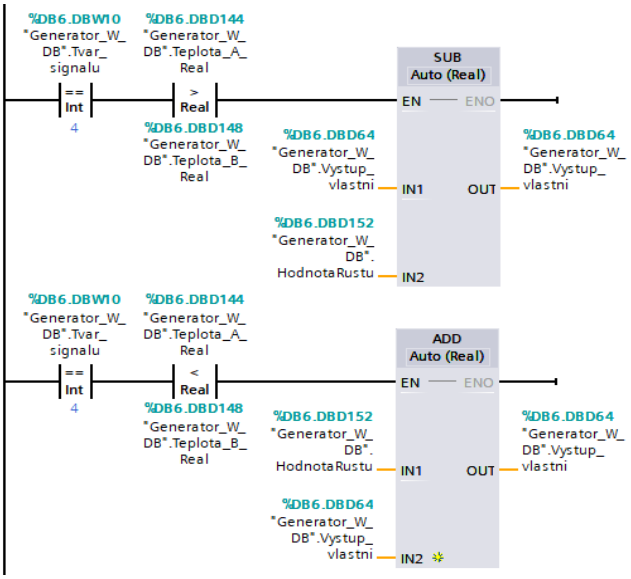
Obr. 3. 46: Vlastní: Network s výpočty

Hlavní funkcionalita vlastního signálu je napsaná tedy tak, že po **„POTVRZENÍ VOLBY“** v editoru a vypočtení potřebných přírůstků **„HodnotaRustu“** přejdeme ke generování výstupu. Pracujeme zde s balíky dat nastavení každého pole, přičemž s patřičným balíkem pracujeme tak dlouho, jak je dáno hodnotou periody patřičného pole. Toto trvání nám zde hlídá časovač TON **„Trvání pole“**, který po příchodu logické **„1“** na vstup začne počítat dle nastavené PT hodnoty v milisekundách. Po uplynutí času pak nastaví výstup časovače, čímž přejdeme k dalšímu balíku dat pole a tento koloběh se nám znova opakuje. Během tohoto ubíhání času mohou nastat na výstupu generátoru tři děje. Když se hodnoty jezdců **„A, B“** rovnají a na výstup je poslána hodnota A.



Obr. 3. 47: Vlastní: Řídící network

Pokud se však hodnoty **„A, B“** pole nerovnají, tak přejdeme do **cyklicky volaného OB**. Zde se rozhodujeme na základě porovnání nastavených teplot A, B a to zda budeme stoupat, či klesat o vypočtený růst **„HodnotaRustu“** a to s rychlostí volaného bloku, tedy co jednu milisekundu. Jelikož se jedná o organizační blok, tak veškeré data zde musíme přivést z našeho datového bloku generátoru.

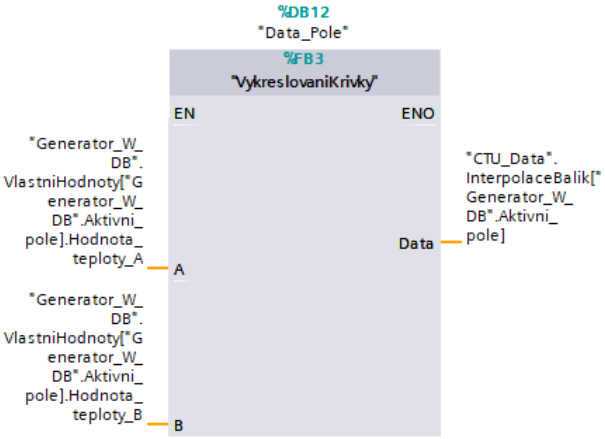


Obr. 3. 48: Network pro růst/pokles křivky

Vykreslování křivky vlastní volby

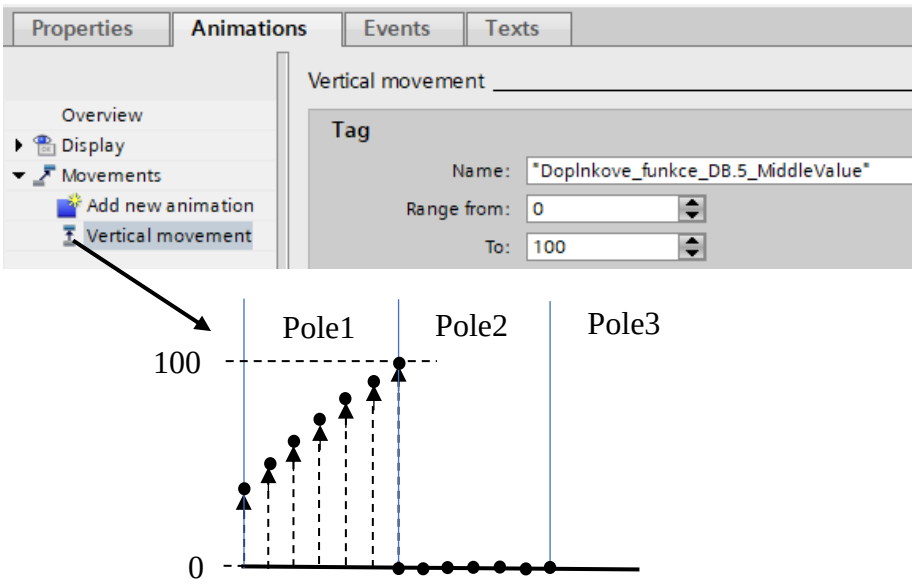
Jako první nápad bylo, že uživatel pouze zadal číselně parametry křivky (tedy tři hodnoty pro každé pole) a poté by se mu až daná křivka vykreslila přes **„TrendView“**. Mělo to však nevýhodu v tom, že právě vytvořený signál bychom rovnou pustili do soustavy, a to vlastně jen s naší představou tvaru křivky v hlavě. Tuto možnost jsem tedy zavrhl a vytvořil funkční blok FB3, který nám naše zadané parametry křivky graficky promítne ještě předtím, než je aplikujeme na soustavu. Funkcionalita tohoto bloku je docela jednoduchá, ale rozsah počtu použitých matematických funkcí a různých převodů a přenosů dat je vcelku rozsáhlý. Proto jí zde popíši pouze slovně.

Tento blok funguje tedy následovně. Podle aktivního pole v editoru křivky, nám přiřadí jeho hodnoty do naší funkce. Ta první udělá rozdíl **„B-A“**, čímž získáme výšku křivky. Tuto výšku pak rovnoměrně rozdělíme mezi 7 částí, přičemž dvě krajní hodnoty již máme zadány skrze počáteční **„A, B“.** Tyto hodnoty jsou pak uchovávány v našem globálním datovém bloku **„CTU_Data“,** takže v případě vypnutí editoru nastavení křivek neztratíme.



Obr. 3. 49: Blok pro vykreslování v editoru

V každém poli HMI tak máme k dispozici pro vykreslení křivky celkem 7 bodů. Využíváme k tomu animace ve formě vertikálního posuvu objektu. Zde posíláme **„tag“** hodnotu, podle které se posouvá náš bod v rozsahu 0 až 100. Tímto způsobem vykreslíme všechny křivky nastavených polí.



Obr. 3. 50: Princip vykreslování

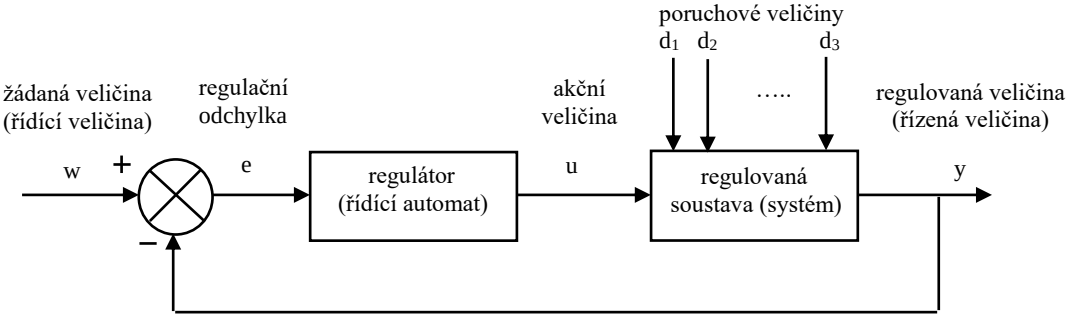
3.4 Úloha: Uzavřený regulační obvod

Jedním ze základních znalostí, jakožto studenta automatického řízení je nastavování a rozpoznání různých typů regulátorů. A o tomto taky tato úloha bude. Největší zastoupení v praxi má PI a PID regulátor, který se právě vyznačuje svou léty ověřenou praxí a širokou univerzálností pro různé odvětví průmyslu. Podle studií a průzkumů je však množství správně nastavených regulátorů jako šafránu.

Professor Aidan O'Dwyer z Dublinského institutu technologie uvádí ve svém díle „Handbook of PI and PID Controller Tunning Rules“ [21], že přibližně 30% regulačních obvodů pracuje v manuálním režimu a 65% pracuje v automatickém režimu, bohužel né však vždy s dobře nastavenými parametry, což je způsobeno často neznalostí, ale spíše opomíjením důležitosti této problematiky a přístupem k věci stylem „hlavně že to nějak funguje“. [8, 10, 14]

3.4.1 Teoretický rozbor úlohy

Nedostačuje-li nám řízení logické (dvouhodnotové), je nutné zavést řízení spojité. To je rozdílné tím, že jeho hodnoty již nejsou stálé, ale proměňují se v závislosti na čase. Také nejsou pouze binární a nabývají většího počtu hodnot. V teorii se tak můžeme bavit o řízení zpětnovazebním, jenž nám ve zpětné smyčce vrací reakci soustavy, ale je zde také možnost řízení soustavy bez zpětné vazby. Pro naši úlohu využijeme řízení obvodu se zpětnou vazbou, který znázorňuje obrázek uzavřeného regulačního obvodu níže.



Obr. 3. 51: Uzavřený regulační obvod

Nejdůležitější prvky tohoto obvodu jsou:

Žádaná veličina (W), to je hodnota, kterou požadujeme po systému, aby nám ji poskytl na výstupu.

Regulační odchylka (e), často také nazývaná „chybová“. Ta nám ukazuje, jak se výstup liší od požadované hodnoty. Veliký vliv na tuto chybu, mají právě poruchové veličiny d.

$$e = w - y \tag{3.13}$$

Regulovaná veličina (y) je výstupní hodnota ze soustavy

Regulovaná soustava je pak náš kontrolovaný systém (např. část výrobní linky), kterou se snažíme korigovat přes **regulátor** (např. PID).

Akční veličina (u) uzpůsobuje tedy svou velikost, dle patřičných požadavků řízení a je jakýmsi sprostředkovatelem požadavku řízení po soustavě.

Celkově nám tedy jde o kvalitní nastavení regulátoru tak, aby zvládal řídit chování naší soustavy s měnícími se požadavky řízení, a to s co možná nejmenší regulační odchylkou.

3.4.2 Volba regulátoru

Jako prvotní krok je důležité zvolit správný regulátor. S tím pak budeme pracovat a pokoušet se jej správně parametrizovat pomocí různých metod.

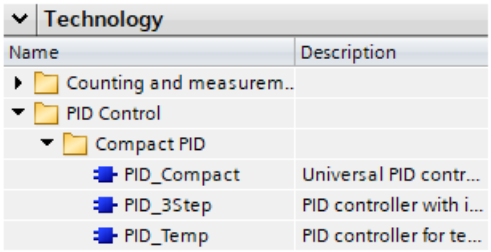
Jak už jsme si říkali, tak s největším zastoupením v praxi se dá považovat PID regulátor. Sice se již nepoužívá jako v dřívějších letech ve formě hardwaru, tedy určitého zařízení, ale dnes se vyskytuje spíše ve formě programové a to tak, že je jeho algoritmus naprogramován v PLC.

Další velice často používaný regulátoru je typu ON/OFF. Jedná se o primitivnější metodu dvoupolohové regulace, avšak často velmi dostačující (např. pro ohřev vody v bojleru, napouštění a vypouštění nádrží). Jedná se o to, že ON/OFF regulátor vypíná akční člen až po dosažení, často i překonání požadované hodnoty. To správně nastavený PID regulátor **„v ideálním případě“** neudělá a k požadované hodnotě se pouze přiblíží bez překmitu (v některých případech je však překmit žádoucí).

Třetí formu regulace v naší úloze je možnost formou manuálního nastavení. Je zde v zastoupení pomocí PID regulátoru v manuálním režimu, jehož výstup je pak čistě ovládán pomocí potenciometru na ovládacím panelu

3.4.3 PID

V této verzi programu máme k dispozici tři typy funkčních bloků, jež plní funkci PID regulátoru.



Obr. 3. 52: PID regulátory v TIA Portal

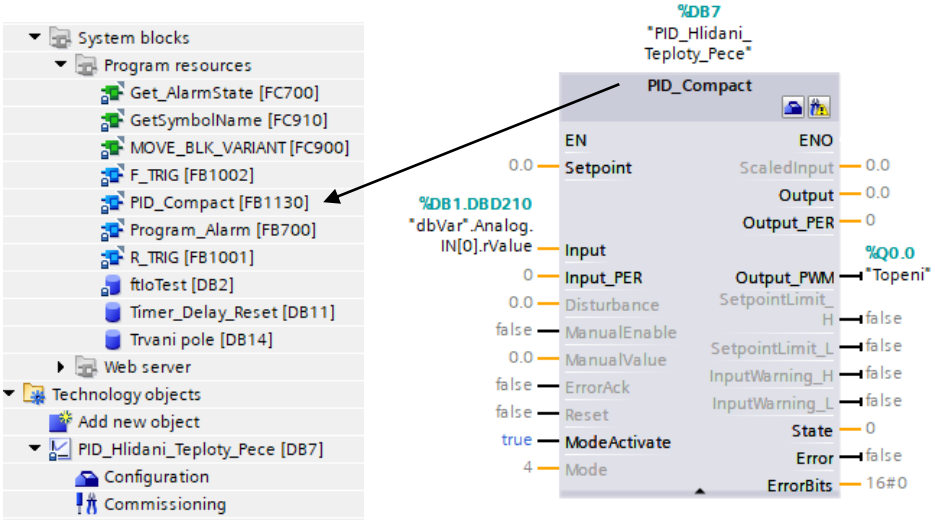
PID_Compact V2 – Je regulátor PID s integrovaným laděním, jenž se hodí pro řízení akčních členů s proporcionálním působením. Má jak možnost logického řízení, tak formou analogu, a i pulzně šířkové modulace.

PID_3Step V2 – Jedná se o třístavový regulátor PID s vlastním laděním, hodící se pro ventily a akční členy, které mají chování integračního rázu. Jeho výhodou je, že dokáže spínat až dva logické členy, například ventily. K těmto ventilům zde má i zabudovanou určitou logiku, která pracuje se zpětnou vazbou z těchto členů, tedy „otevřený/zavřený“. Bohužel však nemá možnost pulzně šířkové modulace.

PID_Temp – Je regulátor PID s integrovaným laděním parametrů pro teplotní procesy. Slouží čistě k aplikacím, které se zabývají chladícími a ohřívacími procesy. Dá se říct, že se funkčně rovná bloku PID_Compact s jediným rozdílem a to, že má v sobě tyto bloky dva a dokáže tak regulovat dvě soustavy naráz.

PID_Compact V2

Rozhodl jsem pro tento případ vybrat funkční blok FB1130 (PID_Compact V2), jelikož se jedná o systém, kde je zapotřebí regulovat pouze jeden člen soustavy. V případě, že bychom měli v naší soustavě více akčních členů, které by potřebovaly řídit (např. chlazení), zvolil bych PID_Temp.



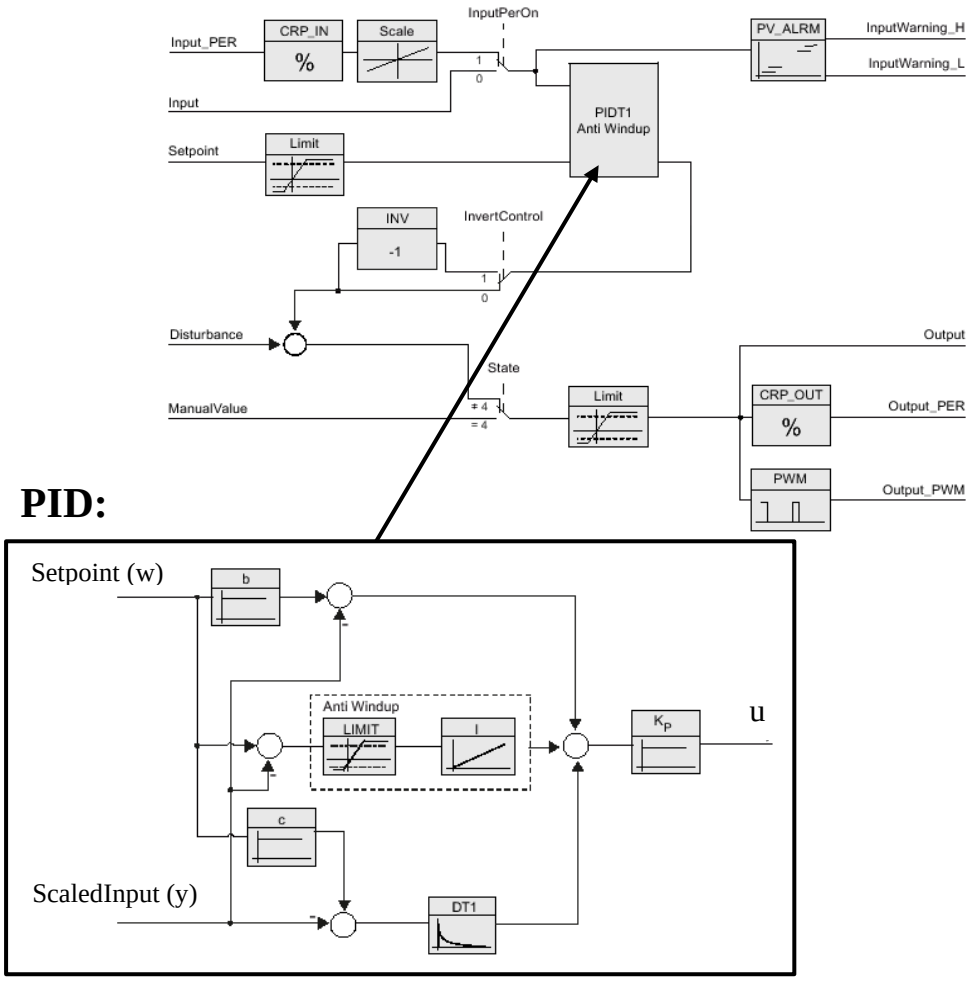
Obr. 3. 53: PID Compact blok

V tabulce níže jsou rozepsány vstupní a výstupní parametry tohoto bloku. Je samozřejmě složen z vícero parametrů, ale k nim samotným se již přistupuje interně, přes datový blok.

Parametr	Datový typ	Popis vstupních parametrů PID_Compact
Setpoint	REAL	Žádaná hodnota v automatickém modu
Input	REAL	Hodnota sledující proces (např. senzor teploty), bez škálování
Input_PER	INT	Hodnota sledující proces (např. senzor teploty), přiřazená jako analogová hodnota, která je ještě v bloku upravena
Disturbance	REAL	Hodnota rušení
ManualEnable	BOOL	Povolovací bit pro manuální režim
ManualValue	REAL	Hodnota, která se nastavuje v manuálním režimu na výstup
ErrorAck	BOOL	Potvrzení chyby/erroru
Reset	BOOL	Restart regulátoru
ModeActive	BOOL	PID se rozhoduje v jakém je modu, dle vnitřního nastavení.
Mode	INT	0=Neakt,1=Předladit,2=Doladit,3=Auto,4=Manu,5=Chyba
Parametr	**Datový typ**	**Popis výstupních parametrů PID_Compact**
ScaledInput	REAL	Škálovaná hodnota vstupu
Output	REAL	Výstup je ve formátu REAL (%)
Output_PER	INT	Výstup je ve formátu analogu (0– 27648)
Output_PWM	BOOL	Pulzně šířková modulace, výstup ve formátu „TRUE/FALSE“
SetpointLimit_H	BOOL	Varování, že byl dosažen maximální limit žádané hodnoty
SetpointLimit_L	BOOL	Varování, že jsme pod minimálním limitem žádané hodnoty
InputWarning_H	BOOL	Varování, že vstup dosáhl maximální hodnoty
InputWarning_L	BOOL	Varování, že vstup dosáhl minimální hodnoty
State	INT	PID ukazatel aktivního modu
Error	BOOL	Příchozí error (chyba)
ErrorBits	DWORD	Bitová zpráva popisující vzniklou chybu

Tab. 3. 2: PID Compact vstupy/výstupy

Blokový diagram PID_Compact V2:



Obr. 3. 54: Blokové schéma funkce PID regulátoru [Tia Portal Help "F1"]

Funkční blok má spousty bezpečnostních prvků, jakož jsou nastavitelné limity žádané hodnoty, výstupu, vstupního senzoru. Poté různé alarmy, které se sepnou při dosažení těchto limitních hodnot. Máme zde i vstupní hodnotu pro vnější rušení, inverze výstupu a další. Samotný PID pak obsahuje své nastavitelné parametry a jejich váhy. Na blokovém diagramu pak vidíme dva druhy výstupu z regulátoru:

Output_PER – Ten řídí soustavu pomocí analogové veličiny 0 až 27648 (0-100%)

Output_PWM – Zde je řízena soustava přes logický TRUE/FALSE, který je spínán v určitých intervalech a skrze tento interval tak dokážeme procentuálně rozložit výstup mezi 0-100%. Jinak řečeno se jedná o pulzně šířkovou modulaci „PWM".

Výstupní hodnota **Output** pouze slouží jako ukazatel, na kolik % je výstup nastaven.

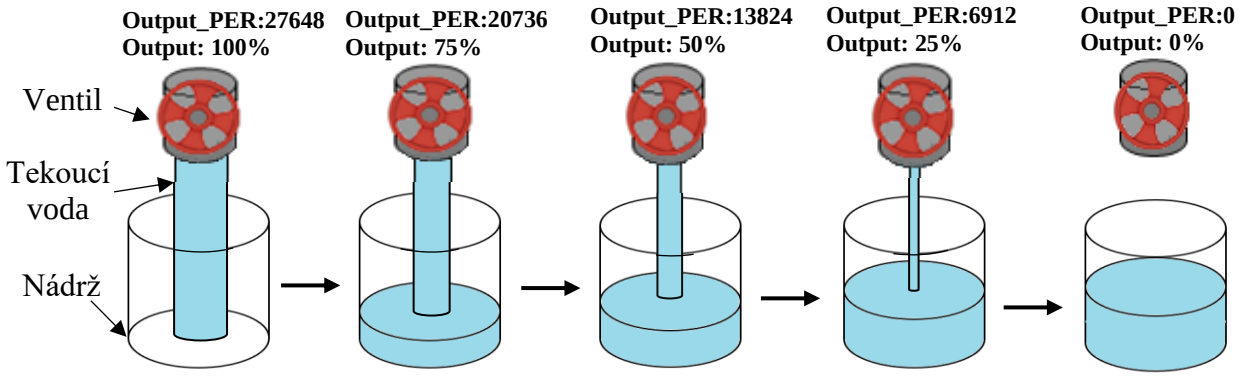
3.4.3.1 Analogové vs PWM řízení

Dva typické způsoby řízení akčního členu skrze regulátor. Jaký však zvolit? Tato volba velmi často nebývá na nás, jelikož pro změnu řízení by bylo potřeba přestavět, či přikoupit potřebnou elektroniku pohonu.

I když jsou však tyto způsoby řízení velice odlišné, tak jejich výsledky, co se týče rychlosti provedení a kvality regulace bývají prakticky totožné.

Analogové řízení

Je založeno na tom, že řízenou veličinou (proud 4..20mA, napětí 0..10V) řídíme akční člen. Hodnota této veličiny je nastavena přes digitální hodnotu v číselném rozsahu 0 až 27648. Je pak jednoduše skrze tuto hodnotu nastavováno, na kolik procent se nám například otevře ventil, roztopí spirála, nebo roztočí motor. Jako ukázku, zde na obrázku máme nádrž, jež napouštíme vodou pomocí řízeného ventilu. Pokud tedy nastavíme analogovou hodnotu výstupu na 27648, bude to znamenat, že se nám ventil otevře na maximální hodnotu a v případě hodnoty 0 se úplně uzavře.



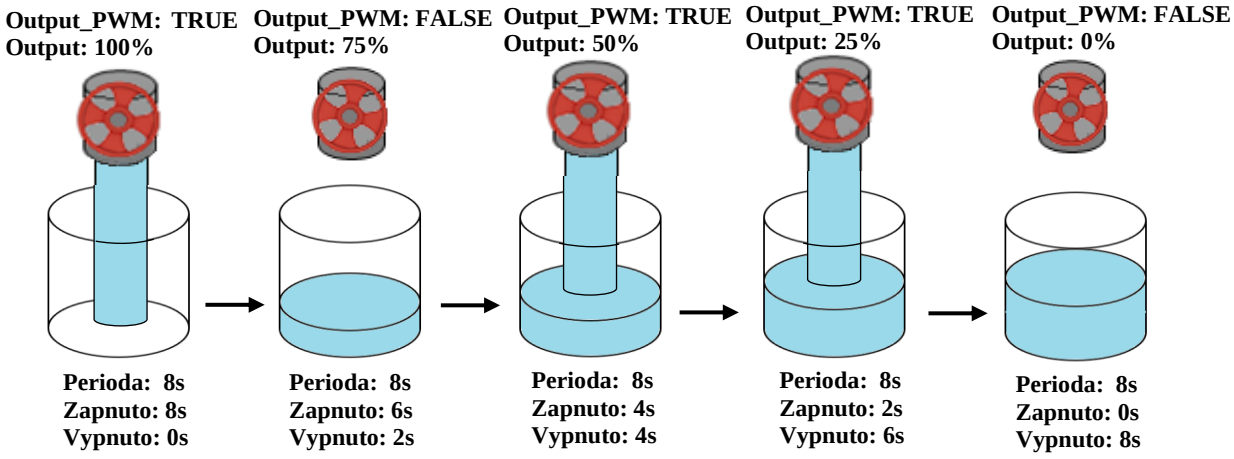
Obr. 3. 55: Analogové řízení napouštění nádrží

Ventil s přívodem vody se tedy zpočátku úplně otevře a s postupem času se uzavírá (zužuje se proud vody), dokud není dosažena požadovaná hodnota hladiny nádrže.

PWM řízení

Oproti analogu zde pokaždé pracujeme s maximálním nastavením (otevřením) výstupu na 100%, tedy hodnotou TRUE. Abychom však dokázali určitým způsobem řídit tento výstup, používáme takzvanou pulzně šířkovou modulaci **„PWM“.**

PWM pracuje v určitých časových cyklech **„periodách“.** Tuto periodu si rozdělí na dvě části pomocí procentuálního nastavení, které na výstupu požadujeme. To znamená, že pokud například máme pulzní periodu 8 sekund a chceme nastavit výstup na 75% výkon, PWM nám na 6 sekund podrží TRUE a na 2 sekundy FALSE. Tímto způsobem dokážeme řídit akční členy, které lze nastavit pouze na TRUE/FALSE (zapnuto/vypnuto)



Obr. 3. 56: PWM řízení napouštění nádrží

Vidíme zde, že ventily jsou pokaždé otevřeny (zapnuty) na maximální hodnotu, avšak pouze v určitých časových intervalech.

Když si to tedy shrneme, tak analogové řízení má výhodu v tom, že dokáže spojitě regulovat akční člen pomocí číselné hodnoty v rozsahu 0 až 27648. To u PWM nejde, jelikož stále skáčeme mezi stavy zapnuto/vypnuto. Toto přeskakování však nemusí být pro každé zařízení příliš přívětivé a může to vést k jeho poškození. Z tohoto důvodu je v nastavení regulátoru možnost minimálního PWM ON/OFF času, kterým dokážeme zamezit tomuto rychlému přepínání. Pokud však nastavíme tyto časy příliš vysoké, potlačíme tím celkovou funkčnost PWM a regulace zkolabuje.

Horkovzdušný tunel má pouze logický bit Q0.0 pro sepnutí a vypnutí spirály, takže je naše volba výstupu regulátoru skrze PWM předem jasná.

3.4.3.2 Vliv parametrů PID regulátoru na jeho řízení

Rovnice výstupu PID regulátoru :

Tato rovnice nám popisuje chování výstupu PID regulátoru s ohledem na všechny jeho nastavitelné parametry, které jsme mohli vidět v blokovém diagramu.

$$\boldsymbol{u = K_P \cdot \left[(b \cdot w - y) + \frac{1}{T_i \cdot s}(w - y) + \frac{T_D \cdot s}{a \cdot T_D \cdot s + 1}(c \cdot w - y)\right]} \tag{3.14}$$

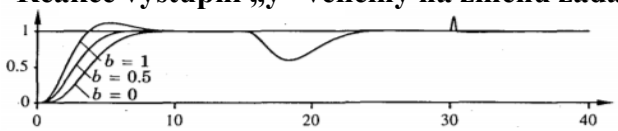
(**K_P**…proporční zisk, **T_i**…integrační časová konstanta, **T_D**…derivační časová konstanta, **a**…derivační zpoždění, **c**…váha derivačního členu , **b**…váha proporčního členu, **u**…akční veličina, **w**…žádaná veličina, **y**…regulovaná veličina, **s** …Laplaceův operátor)

Váhy řídících členů „b, c“

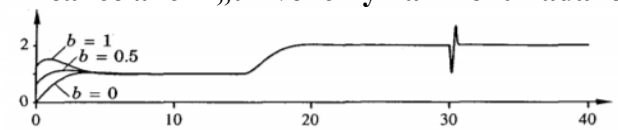
Tyto váhy **„b, c“** používáme pro zjemnění reakce regulátoru na změnu žádané hodnoty. Jak už jsme si pověděli v úvodu, tak hlavní odpověď systému na regulátor je vlastně dán regulační odchylkou (chybou) **„e“.** Ta nám ukazuje rozdíl mezi žádanou veličinou a výstupem. Celá soustava je tedy řízena touto chybou a takové řízení se pak nazývá **„systémové řízení se zpětnou chybou“.**

Více pokročilé systémy již ale pracují s žádanou hodnotou a výstupem odděleně. Využívají takzvané **„váhy“** řídících členů regulátoru. Takové řízení je pak dáno upravenou rovnicí, kterou se také řídí náš PID regulátor, kde **„(b·w - y)“** je chybou proporční části, **„(c·w - y)“** je chybou derivační části a **„(w - y)“** je chybou integrační části a tedy vlastně i regulační odchylkou systému. Regulátor se změněnými hodnotami vah **„b, c“** nebude na vliv rušení a šumu reagovat s nějakou výraznou změnou oproti klasickému PID, který tyto **tlumící** členy nemá. Tyto členy se pak zadávají v rozsahu 0 až 1, kdy v případě utlumování reakce na změnu žádané veličiny, můžeme přitvrdit v hlavních parametrech regulátoru **T_i, T_d, K_P** a nerozhodit si tak celkovou regulaci. [11]

Reakce výstupní „y“ veličiny na změnu žádané



Reakce akční „u“ veličiny na změnu žádané

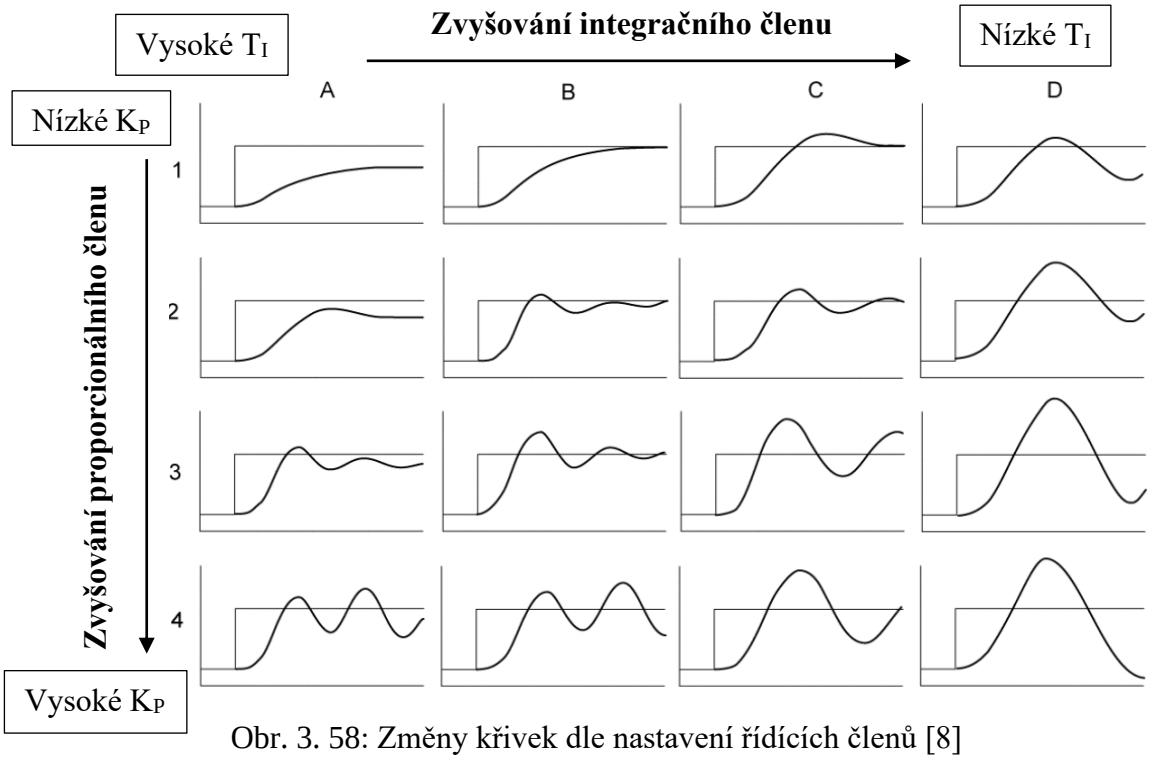


Obr. 3. 57: Změny křivek při aplikaci vah členů [11]

Zde vidíme, že při **„b=0“** je rychlost růstu, a tedy i překmit při náhlé změně nejmenší. Parametr **„c“** obvykle nastavujeme roven 0, abychom se vyhnuli velkým přechodovým změnám, při změně žádané hodnoty. Jeho využití bývá v tu chvíli, kdy máme za sebou například více regulátorů v kaskádovém zapojení.

Základní řídící členy regulátoru „K_P, T_i, T_D“

Pro zvládnutí metod seřizování je dobré vědět chování jednotlivých parametrů regulátoru, s kterými budeme experimentovat. Proto zde volím obrázek níže, na kterém si graficky ukážeme chování jednotlivých částí regulátoru:



Obr. 3. 58: Změny křivek dle nastavení řídících členů [8]

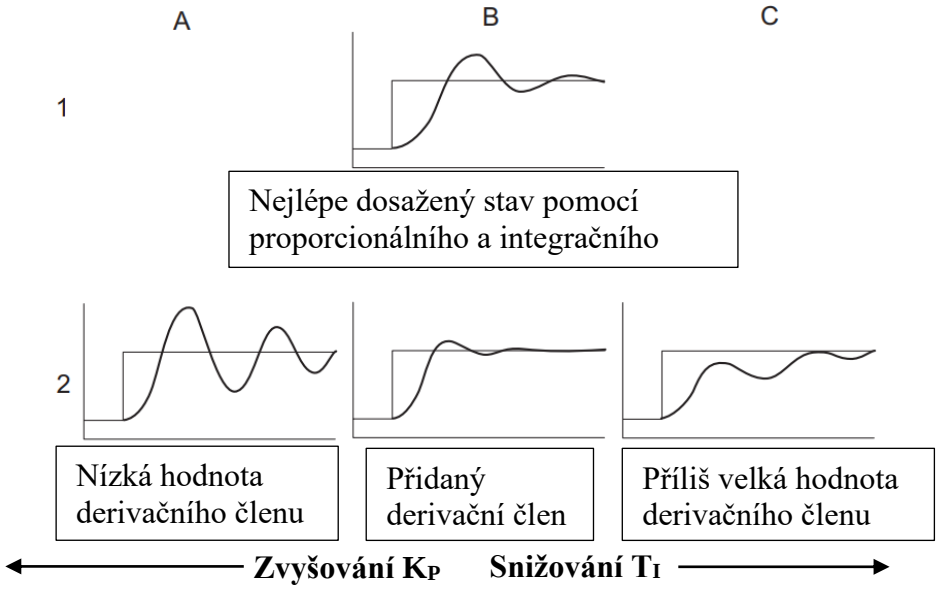
V levém horním poli A1 můžeme vidět, jak by měl vypadat počáteční stav nastavování regulátoru neznámé soustavy, tudíž nízká hodnota K_P a pokud možno s nulovými integračními a derivačními členy. Jakmile začneme zvyšovat pouze hodnotu proporcionálního členu K_P (levý sloupec), tak vidíme, že se začíná objevovat místy překročení požadované hodnoty a ve snaze ji dosáhnout, nám začne výstupní hodnota oscilovat viz. pole A4.

Nyní bychom se na to mohli podívat z jiného úhlu a to tak, že budeme pouze pracovat s integračním členem. Pohybujeme se tedy jen v horním řádku, pouze pomocí hodnoty integračního členu, tedy snižováním integrační časové konstanty T_I a necháme proporcionální člen K_P na nízké hodnotě. Počáteční chování samotného integrátoru je snímání chyby ustáleného stavu a snahy o její snížení na nulu pomalým náběhem křivky, což vidíme v poli B1. Při ještě větším snižování časové konstanty vidíme, že křivka začíná růst více ostře a způsobuje nám překmit, jako například na poli D1. Lze tak tedy říct, že při příliš vysoké hodnotě integračního členu naši soustavu spíše destabilizujeme.

Je třeba tedy balancovat a hledat dobrou souhru konstant. Tady jsme mohli vidět, že není dobré mít ani příliš vysoké K_P a ani příliš nízké T_I, jelikož obě tyto cesty vedou k oscilaci, neboli ztrátě stability. Na zbylých obrázcích je pak vidět, že při správném nastavování tedy nikdy nevede cesta přes zvyšování obou členů, ale je to vždy o kompromisu něco za něco. Pokud tedy zvyšujeme proporcionální složku a roste nám tak možná celková destabilizace systému, musíme ubrat na integrační složce, abychom systém znovu vrátili do mezí stability. [8, 10]

Také je důležité nezapomenout na vnější poruchové vlivy, které musíme vzít v úvahu a nehnat se tak za dokonale stabilním systémem. Když bychom například náš systém posunuli z pole C2 do B3, tak vidíme, že nám vznikne o trochu větší oscilace, ale v případě změny vnějšího vlivu by tento systém mnohem lépe reagoval.

Určitě jste si ale všimli, že stále řešíme pouze integrační a proporcionální složku, kdežto PID má ještě složku třetí a to derivační. Ta se právě nastavuje vždy až ve chvíli, kdy nalezneme vhodný balanc mezi proporcionální a integrační složkou, jako obrázek níže v poli B1.



Obr. 3. 59: Vliv derivačního členu [8]

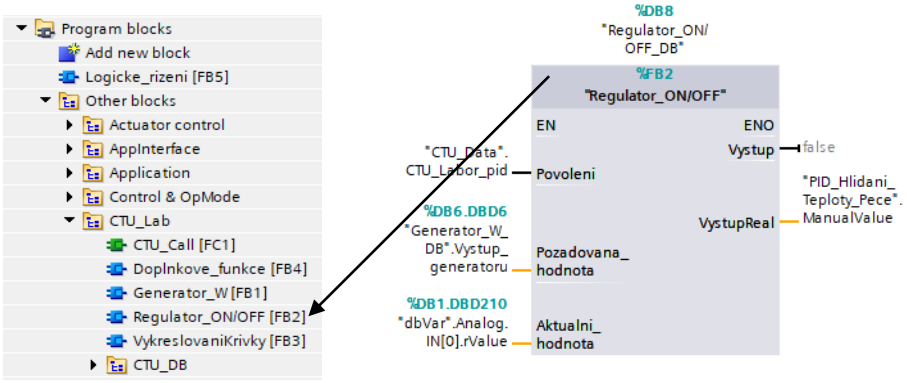
Jak vidíme zde, derivační člen může být pozitivním přínosem do soustavy a zajistit tak lepší stabilitu systému, ale musíme brát v potaz, že tak přinášíme další parametr do soustavy, přičemž jeho nesprávným zvolením můžeme soustavu rozhodit úplně.

Velikou výhodou k použití derivačního členu je, že můžeme po jeho přidání zvyšovat proporcionální a integrační člen, které by za normálních okolností měly spíše destabilizační efekt a snižuje také maximální odchylku.

Toto, co jsme si doposud popsali bylo však spíše teoretické rozebrání, které bylo třeba ukázat, aby měl student určitý nadhled o chování parametrů PID regulátoru. [8, 10]

3.4.4 Dvoupolohová regulace ON/OFF

Pro dvoupolohovou regulaci (jinak řečeno metodu relé) si potřebujeme naprogramovat funkční blok, jenž nám bude plnit funkcí tohoto regulátoru. Vytvořil jsem tedy FB2, který je volán v hlavním organizačním bloku OB1. Tento blok má tři vstupní a dva výstupní parametry. Jako povolovací bit **„Povoleni“** zde slouží informace z HMI, že je aktivována laboratorní úloha pro Uzavřený regulační obvod. Hodnotu **„Pozadovane“** regulátor získává z výstupu generátoru žádané hodnoty a stav regulované veličiny **„Aktualni_hodnota“** zde přivádí senzor teploty. Funkci výstupů si vysvětlíme níže. [11]



Obr. 3. 60: Funkční blok pro dvoupolohovou regulaci

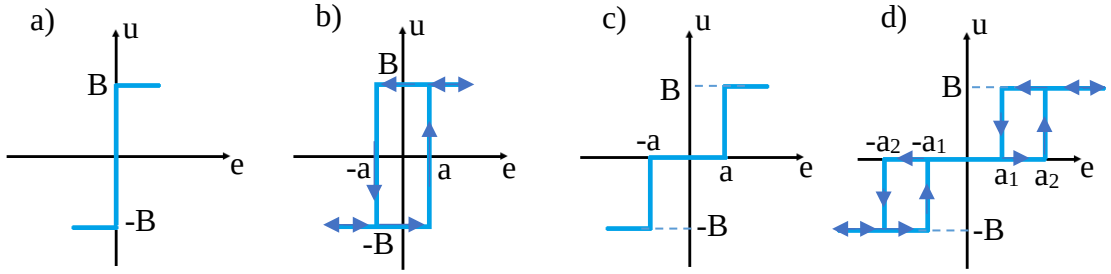
Typy RELE

a) Dvoupolohové relé - pro hodnotu e=0 není definován, jedná se o ideální, jednoznačnou linearitu, která nemá paměť.
u = B pro e > 0
u = -B pro e < 0

b) Dvoupolohové relé s hysterezí - když však přidáme prvek hystereze, začne se chovat nejednoznačně a má paměť. To znamená, že pro sepnutí/vypnutí je důležitý prvek minulosti, kde se před překročením meze hodnota nacházela(šipky)
u = B pro e > a s minulostí u > 0
pro e > - a s minulostí u < 0
u = -B pro e < a s minulostí u > 0
pro e < - a s minulostí u < 0
Toto relé bude naší volbou pro ON/OFF regulátor

c) Třípolohové relé
d) Třípolohové relé s hysterezí

Symetričnost a nesymetričnost relé rozeznáme podle chování akčního zásahu. Zda je tedy hodnota nad osou a pod osou symetrické kolem nuly, či ne.



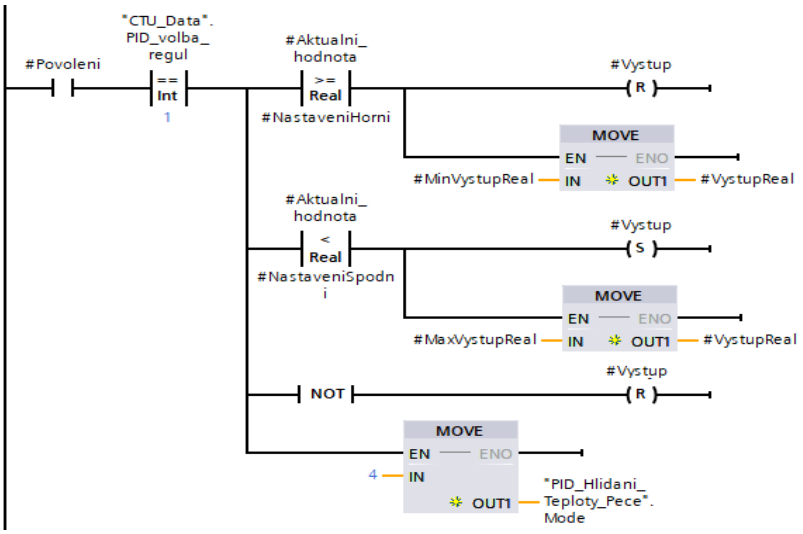
Obr. 3. 61: Typy použitelných relé pro regulaci [11]

Funkce je tedy napsána tak, že pokud je v úloze zvolena možnost ON/OFF regulátoru, začne blok porovnávat žádanou hodnotu **„w“** se stavem regulované veličiny **„y“**. Pokud se teplota tunelu nachází pod hranicí požadované hodnoty, přijde povel **„zapnout“**. V případě, když teplota přesahuje požadovanou, tak se naopak spirála **„vypne“.**

Kdybychom takhle chtěli regulovat a řešit jen sepnutí a vypnutí při nastavení 0% nebo 100% výstupu, nebyl by zde žádný problém. Použili bychom **„Vystup“** našeho bloku, kde jen připneme bit **Q0.0** pro spínání spirály a ten by tak v případě 100% byl celou dobu zapnutý a v případě 0% zase vypnutý.

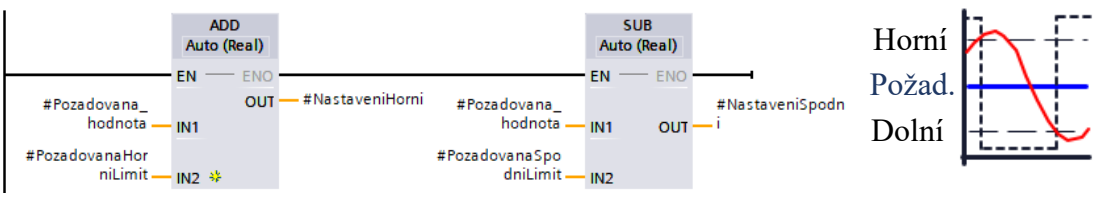
Naše regulace je však mnohem sofistikovanější a chceme zde mít možnost přecházet mezi stavem zapnuto/vypnuto s nastavením výstupu například 20% a 80%. To ale znamená, že při příchozím požadavku **„zapnout“** a nastavení 80% výstupu, nemůže být celou dobu spirála sepnutá, jakožto bylo u 100% nastavení. Z tohoto důvodu jsem vymýšlel různá řešení a našel jedno, skrze náš funkční blok PID_Compact.

Využijeme samotný PID regulátor. Ten díky typu naší úlohy používáme jako regulátor s PWM výstupem, což se nám pro tento případ perfektně hodí. Funguje to tedy na principu, že když zvolíme dvoupolohovou regulaci ON/OFF, automaticky se nám zapne PID regulátor v manuálním režimu. Ten se v tomto režimu chová tak, že cokoliv mu přijde na vstupní hodnotu **„ManualValue“**, překlopí na svůj výstup v takové formě, v jaké je nastaven (v našem případě tedy PWM). V praxi to tedy znamená, že naše dvoupolohová regulace bude do PID posílat informaci, na kolik procent se má výstup nastavit a náš PID podle toho bude korigovat spínání bitu pece Q0.0 stejným způsobem, jako to dělá on sám v automatickém režimu. Jediný rozdíl zde bude v tom, že my rozhodujeme trvání délky PWM periody výstupu při určitém procentuálním nastavení.



Obr. 3. 62: Network plnící funkci dvoupolohové regulace

Tuto časovou délku rozhodujeme tím, jak velkou hysterezi požadované hodnoty zvolíme. Je zde možnost nastavení horní i spodní meze zvlášť. Tedy hysterezi, což je vlastně jen určité posunutí hranice vypnutí a sepnutí, můžeme nastavit i nesymetricky. To znamená, že při dosažení **horní meze vypne** a při dosažení **spodní meze zapne** výstup.



Obr. 3. 63: Network pro hysterezi požadované hodnoty

3.4.5 Metody seřizování regulátorů

Představíme si zde několik v praxi používaných metod pro správné nastavení regulátoru. V reálném prostředí je spousty faktorů, které nám kvalitní regulaci komplikují. Jeden z hlavních problémů je nepříliš dobrá znalost dynamiky procesu, která se v průběhu času mění. Poté tu máme rušení všemi možnými vnějšími vlivy. Často také pracujeme na systému, který je už více komplexnější, nebo je ve stále trvajícím procesu výroby a neumožňuje nám tak příliš velké zásahy, co se týče jakéhokoliv testování, či experimentování. Každá metoda tak má své plusy a mínusy, proto je důležité správně volit, co se zrovna pro náš případ bude hodit nejlépe.

3.4.5.1 Experimentální metoda „pokus – omyl“

Jedna ze základních metod, z které i často vychází ostatní. Originální přistup k této metodě popsal [Wade Harold L. 2004] a do formy určité zjednodušené kuchařky ji převedli [Miluše Vítečková a Antonín Víteček, 2011]. [8, 9]

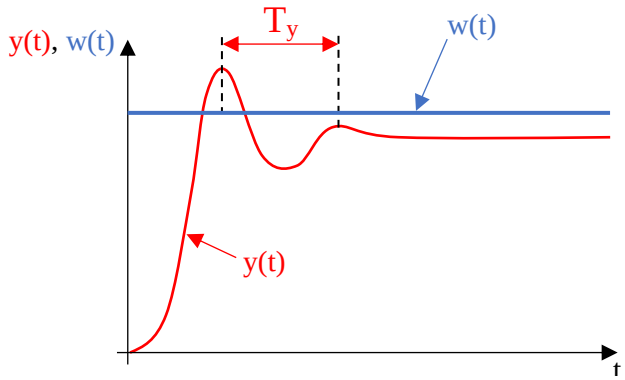
Doporučený postup:

1. Zkontrolovat celý obvod a funkčnost všech částí soustavy kvůli bezpečnosti, abychom při experimentování nijak nepoškodili ostatní zařízení.

2. Nastavíme požadovanou hodnotu w(t) tak, že v manuálním režimu regulátoru výstup je přibližně roven požadované hodnotě $w(t) \approx y(t)$ a vyřadíme integrační, derivační složku ($T_D=0$, $T_I=\infty$). Hodnotu zesílení K_P ponecháme na nízké hodnotě a přepneme do automatu.

3. Hodnotu zesílení K_P pomalu zvyšujeme, dokud se na výstupním signálu y(t), při zadání nové požadované hodnoty w(t) (tedy skoková změna), nastane kmitavý průběh. Pokud se nám objeví offset, není třeba se obávat, ten se doladí později přes integrační složku.

4. Poté zesílení K_P snížíme na 0,75- 0,9 (tudíž na 75 -90% ze své aktuální hodnoty) a začneme pomalu snižovat hodnotu integrační časové konstanty T_I po dobu, než odstraníme regulační odchylku. Doporučená hodnota je $T_I=0{,}67T_y$.



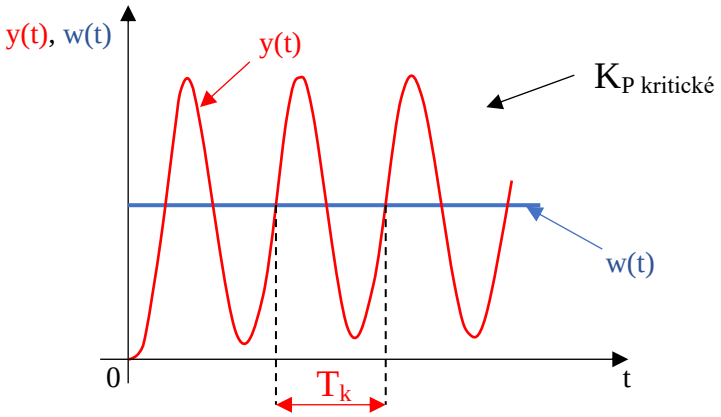
Obr. 3. 64: Tvar křivky při metodě "pokus-omyl"

5. Je možno ještě doladit náš výstup y(t) pomocí K_P.

6. Samozřejmě je tu možnost přidat derivační složku a to tak, že počáteční hodnota derivační časové konstanty bude $0{,}1T_I$. Pokud se neprojeví škodlivé jevy, šumy, tak můžeme zvýšit derivační složku až na $0{,}25T_I$, kde poté ale musíme zvýšit K_P přibližně o 25% a časovou integrační konstantu snížit o 33%.

3.4.5.2 Experimentální metoda kritických parametrů

Tato metoda vychází z podobného přístupu jako předchozí „Pokus - omyl“. Nazýváme ji Zieglerova-Nicholsova metoda kritických parametrů. Znovu tedy vyloučíme derivační s integračním členem (T_D=0, T_I=∞) a pomalu zvyšujeme zesílení regulátoru K_P, než se dostaneme do oblasti **„kritické“** oscilace výstupu. [9]



Obr. 3. 65: Tvar křivky při metodě "kritických parametrů"

Poté odečteme kritickou periodu **T_K** z našeho kmitavého průběhu a zapamatujeme si hodnotu kritického zesílení **K_{Pk}**. Zbylé hodnoty dopočteme z přiložené tabulky.

Regulátor	K_P	T_I	T_D
P	$0{,}5 \cdot K_{Pk}$	-	-
PI	$0{,}45 \cdot K_{Pk}$	$\frac{T_k}{1{,}2} \approx 0{,}83 \cdot T_k$	-
PID	$0{,}6 \cdot K_{Pk}$	$0{,}5 \cdot T_k$	$0{,}125 \cdot T_k$

Tab. 3. 3: Metoda kritických parametrů [9]

Tato metoda je vhodná pro soustavy, o kterých nic nevíme a pracujeme s reálným regulátorem. Hlavní nevýhoda spočívá v tom, že musíme soustavu přivést na mez stability (vznik oscilace), což nám málokterá soustava dovolí, a navíc je i tímto velice nebezpečná pro naše zařízení.

Doporučený postup:

1. a 2. Tyto kroky jsou stejné jako u předchozí metody „pokus-omyl“.

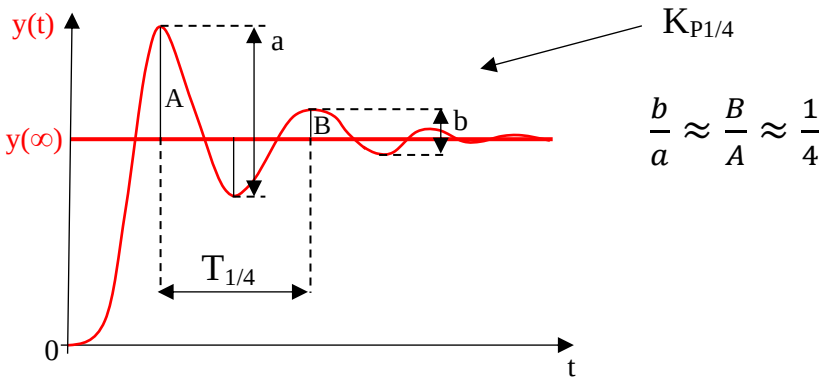
3. Hodnotu zesílení proporcionálního členu K_P pomalu zvyšujeme, až nám při změně žádané hodnoty w(t) začne výstup oscilovat s kmity o amplitudách stejné hodnoty, což bude kritický stav stability.

4. Z tohoto průběhu odečteme hodnotu kritické periody T_K a zapamatujeme si nastavení kritického zesílení K_{Pk}.

5. Dle tabulky dosadíme ostatní hodnoty pro časovou hodnotu integračního a derivačního členu.

3.4.5.3 Metoda čtvrtinového tlumení

Tato metoda vychází ze Zieglerovy-Nicholsovy, avšak má oproti jí velikou výhodu, a to v tom, že nepotřebujeme rozkmitat regulační obvod, tedy ohrozit soustavu mezí stability. To nám umožňuje pracovat s obvody, které se chovají spíše lineárně. [9]



Obr. 3. 66: Tvar křivky při metodě "čtvrtinového tlumení"

Regulátor	K_P	T_I	T_D
P	$K_{P1/4}$	-	-
PI	$0{,}9K_{P1/4}$	$T_{1/4}$	-
PID	$1{,}2 \cdot K_{P1/4}$	$0{,}6 \cdot T_{1/4}$	$0{,}15 \cdot T_{1/4}$

Tab. 3. 4: Metoda čtvrtinového tlumení [9]

Doporučený postup:

1. a 2. Tyto kroky jsou stejné jako u předchozí metody „pokus-omyl“

3. Postupně zvyšujeme zesílení K_P, dokud při skokové změně požadované hodnoty w(t) nedostaneme takovou charakteristiku, kde podíl dvou po sobě následujících amplitud bude roven ¼ (tedy útlum =4)

4. Poté odečteme periodu kmitu $T_{1/4}$ a zapamatujeme si nastavenou hodnotu zesílení $K_{P1/4}$.

5. Nakonec dopočteme a dosadíme zbylé hodnoty regulátoru z tabulky.

3.4.5.4 Metoda překmitu

Metoda navržená pro analogový PI regulátor a proporcionální soustavy. [9]

Doporučený postup:

1. a 2. Tyto kroky jsou stejné jako u předchozí metody „pokus-omyl“

3. Zesílení regulátoru K_P zvedáme, dokud nám naše soustava při skokové změně požadované hodnoty $w(t) = w_0$ nevykáže lehký relativní překmit κ, který by se měl nacházet hodnotou mezi 0,1 až 0,6.

4. Z přechodové charakteristiky tedy odečteme čas $\mathbf{t_m}$, za který bylo dosaženo první maximum $\mathbf{y_m}$, a dopočteme:

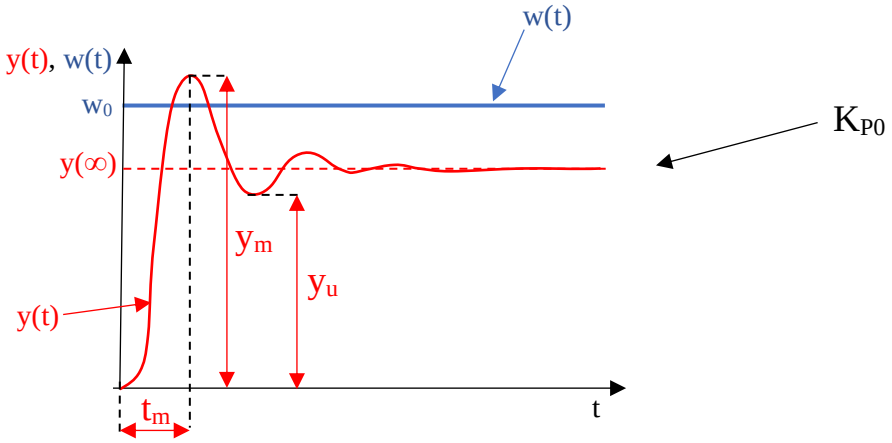
Relativní překmit: $$\boldsymbol{\kappa = \frac{y_m - y(\infty)}{y(\infty)}} \quad (3.15)$$

Pokud nelze z grafu odečíst $\boldsymbol{y(\infty)}$ kvůli krátkého průběhu , pak tuto hodnotu získáme z:

$$\boldsymbol{y(\infty) = 0,45(y_m + y_u)} \quad (3.16)$$

Zesílení otevřeného regulačního obvodu:

$$\boldsymbol{K_{PO} = \frac{b'}{1-b'}} \text{ kde } \boldsymbol{b' = \frac{y(\infty)}{w_0}} \quad (3.17)$$



Obr. 3. 67: Tvar křivky při metodě "překmitu"

5. Dopočteme pomocné hodnoty: $A' = 1{,}152 \cdot \kappa^2 - 1{,}607 \cdot \kappa + 1$ (3.18)

z čehož určíme hodnoty **parametrů regulátoru:**

$$\boldsymbol{K_P^*} = K_{PO} \cdot \frac{A'}{F} \quad (3.19)$$

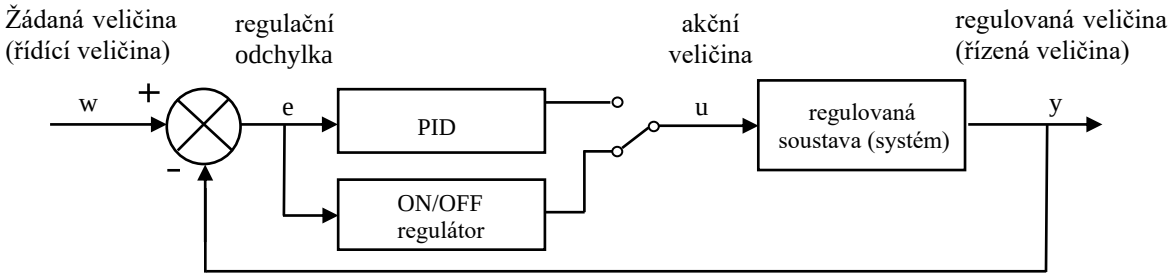
$$\boldsymbol{T_I^*} = \min\left(0{,}86 \cdot A' \cdot t_m \cdot \frac{b'}{1-b'}; 2{,}44 \cdot t_m \cdot F\right) \quad (3.20)$$

kde F parametr volíme dle situace:

F<1 pro rychlé, ale méně robusní
F=1 pro rychlé a robusní
F>1 pomalejší, ale více robusní

3.4.5.5 Metoda relé Åström-Hägglund

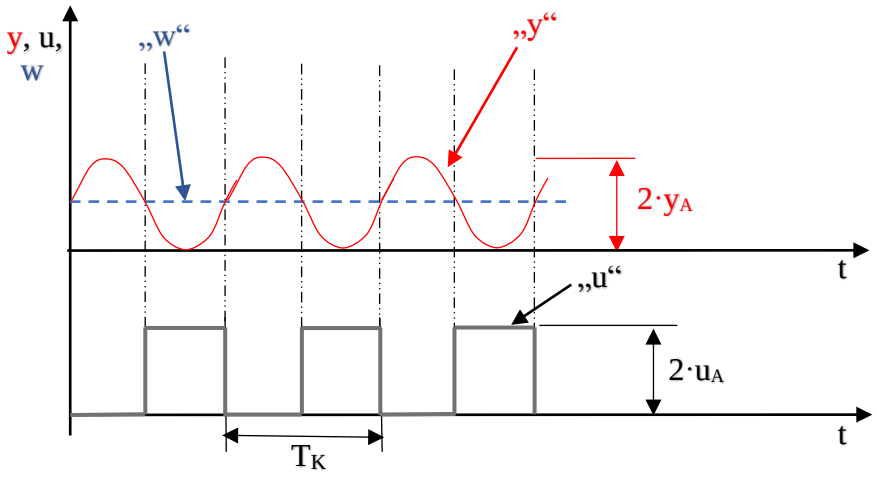
Jedná se o experimentální zjištění kritické hodnoty zesílení K_{Pk} s kritickou periodou T_k skrze dvoupolohový regulátor se symetrickým relé (bez hystereze) s amplitudou u_A. Tato metoda je určitým vylepšením Ziegler-Nicholsovy metody, jelikož zde pracujeme s oscilacemi, které však dokážeme udržet v bezpečných hodnotách, díky limitnímu nastavení ON/OFF regulátoru. Tato metoda se velice často užívá v praxi při takzvaném Auto-tuningu. [11, 16]



Obr. 3. 68: Uzavřený regulační obvod s dvěma regulátory

Doporučený postup:

1. Uveďte systém do klidu a připněte soustavu na ON/OFF regulátor (relé).
2. Nastavte konstantní požadovanou hodnotu někde v bezpečné oblasti.
3. Nechte vygenerovat aspoň tři periody a změřte na jedné z nich velikosti amplitud akční veličiny u_A, regulované veličiny y_A a kritickou periodu T_K.



Obr. 3. 69: Chování křivky při metodě " Åström-Hägglund "

4. Dopočtěte parametry pro PID regulátor z tabulky Ziegler-Nicholsovy metody kritických parametrů.

$$K_{Pk} = \frac{4 \cdot u_A}{\pi \cdot y_A} \quad (3.21)$$

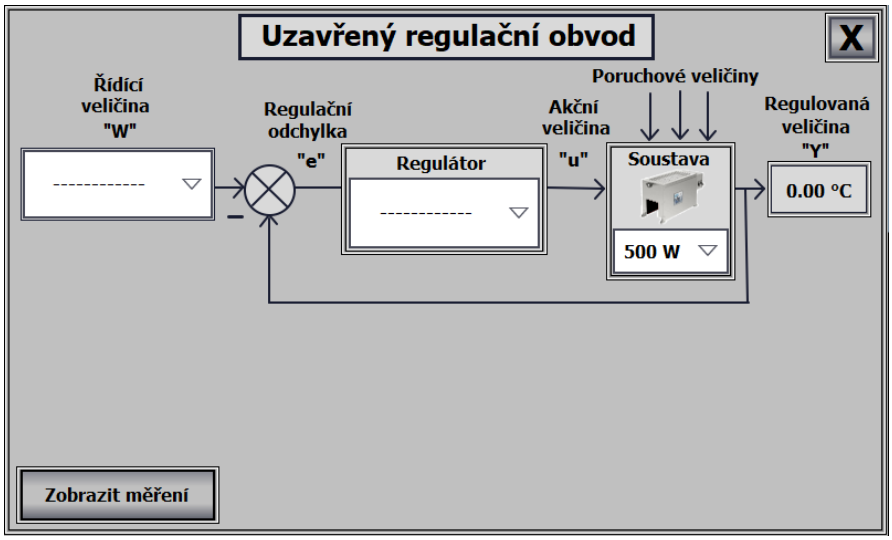
Regulátor	$K_{Pk} = r_0$	$T_I = r_0/r_I$	$T_D = r_D/r_0$
P	0,5 K_{Pk}	-	-
PI	0,45 K_{Pk}	0,85 T_K	-
PID	0,6 K_{Pk}	0,5 T_K	0,125 T_K

Tab. 3. 5: Metoda Åström-Hägglund [9]

5. Přepněte soustavu na PID regulátor a dosaďte vypočtené parametry.

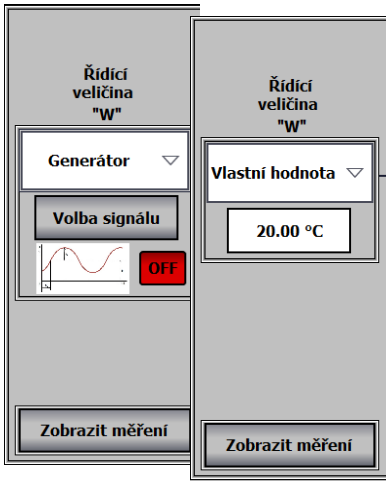
3.4.6 Grafické rozhraní v HMI

Pokud zvolíme na hlavní obrazovce tlačítko **„Uzavřený regulační obvod“**, otevře se nám okno s nákresem obvodu, jenž má představovat měřenou úlohu s horkovzdušným tunelem. Soustavu zde symbolizuje ikonka pece, která v případě sepnuté spirály zčervená. Regulovaná veličina **„Y“** nám pomocí PT100 ukazuje aktuální stav teploty v tunelu. Výkon topné spirály lze nastavit, avšak pouze mezi dvěma hodnotami. S tímto výkonem se už bohužel nedá nijak nadále pracovat. Poruchové veličiny přicházející do soustavy formou vnějších vlivů zde samozřejmě také mají své místo, stačí například otevřít okno v místnosti, nebo u naší pece lze vnější poruchu simulovat, a to ovládáním roštu (horní stěny krytu pece), který však na sobě nemá žádnou zpětnou vazbu pro systém, a tedy co se týče například velikosti pootevření krytu, nevíme nic.



Obr. 3. 70: Grafické rozhraní úlohy: Uzavřený regulační obvod

Vidíme zde možnosti několika nastavitelných prvků, které student může při své snaze o nejlepší parametrizaci regulátoru použít. Jako první zleva máme možnost volby řídící veličiny, kterou lze nastavovat jako **„Vlastní hodnotu“**, tedy číselně vloženou hodnotu (v našem případě požadovanou teplotu tunelu), nebo jí vytvořit pomocí **„Generátoru“,** o němž víme z předchozí kapitoly, že si můžeme vygenerovat harmonický, obdélník, konstantu, nebo čistě svůj vlastní signál tvarovaný dle sebe.



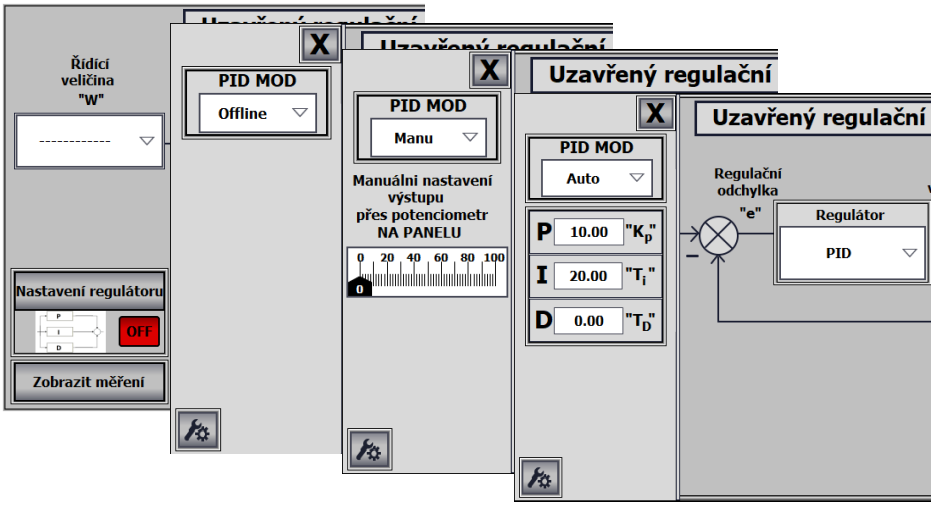
Obr. 3. 71: Volba řídící veličiny

Další nastavitelná část je volba regulátoru. Zde máme volbu mezi dvěma nejtypičtějšími zástupci tohoto prvku a to **„PID regulátor**“, který lze samozřejmě použít jako PI, PD…, v závislosti na tom, jaké složky budeme používat, nebo možnost primitivnější, avšak v praxi stále používané dvoupolohové **„ON/OFF regulace“.**

PID regulátor je možno přepínat mezi třemi módy. V offline módu je regulátor vypnutý a dostaneme se tak pouze do jeho vnitřního nastavení ( ).

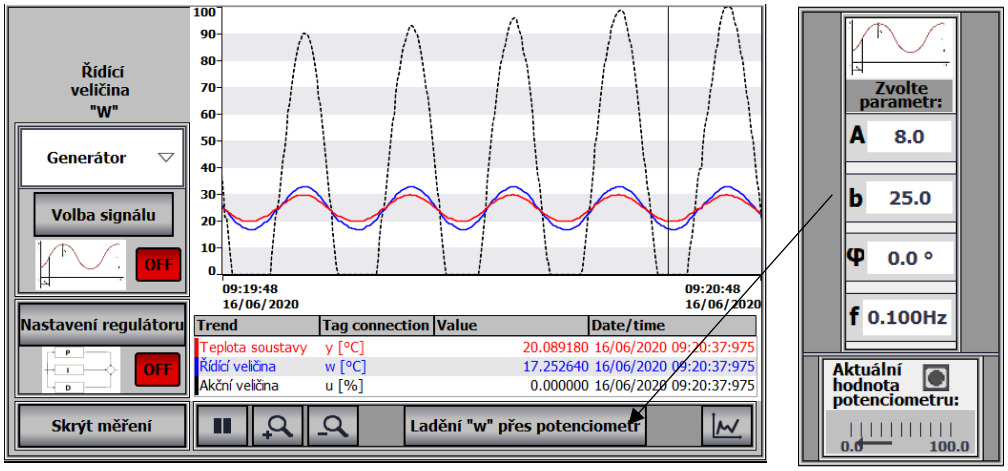
V manuálním módu již můžeme nastavovat naše řídící prvky (proporcionální, derivační, integrační), ale jejich vliv na probíhající regulaci je nulový. Manuální režim je totiž čistě o ovládaném výstupu pomocí potenciometru na ovládacím panelu. Spousty metod nastavení regulátoru totiž vyžadují, aby před nastavováním parametrů bylo prvně docíleno manuálně **w(t) ≈ y(t).**

Jakmile se však přepneme do automatického módu, tak regulátor začne pracovat se zvolenými parametry a žádanou hodnotou, ve snaze sám zregulovat chování systému s nastavenými prvky.



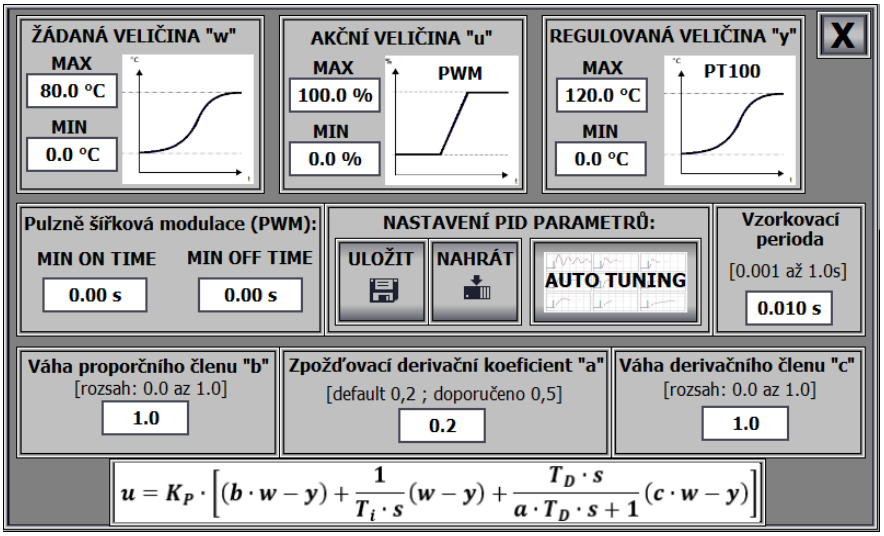
Obr. 3. 72: Jednotlivé módy PID regulátoru

Během tohoto procesu jsem úlohu postavil tak, abychom ve stejném čase mohli sledovat průběh charakteristiky regulace, skrze tlačítko **„Zobrazit měření“** a zároveň nám byla umožněna změna žádané hodnoty, nebo jakéhokoliv parametru regulátoru, aniž by měření bylo přerušeno. Lze tak krásně pozorovat chování soustavy na tyto změny. Avšak při změně samotného regulátoru, např. nahrazením PID za Dvoupolohovou regulaci se měření z bezpečnostních důvodů přeruší.



Obr. 3. 73: Měření probíhající regulace

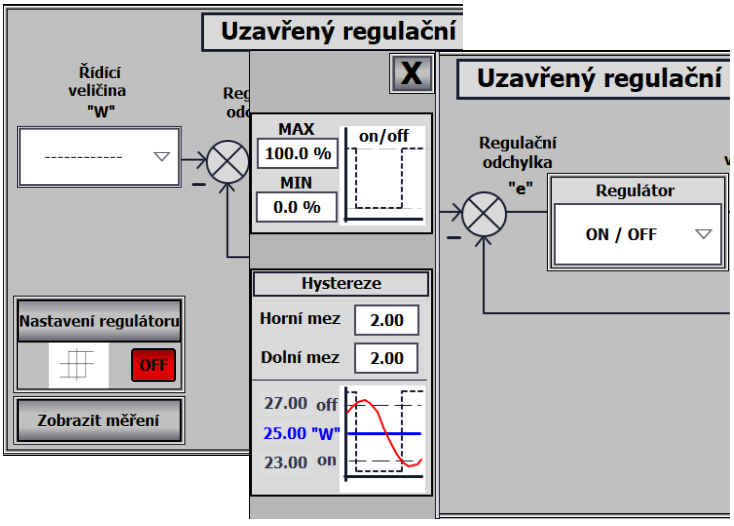
Abych udělal onu úlohu co možná nejvíce optimalizovatelnou, přidal jsem zde přístup k celkovému nastavení ( ) funkčního bloku PID_CompactV2. K tomuto nastavení se dostaneme přeš ikonku klíče v levém dolním rohu. Jsou zde pak omezení hlavních vstupních a výstupních parametrů regulátoru, tak také přístup k parametrům jako jsou takzvané váhy členů. Pro přehlednost, aby šlo vidět, jakým způsobem tyto parametry ovlivňují finální výstup z regulátoru, je zde přiložena i rovnice chování tohoto PID_CompactV2 bloku. Přidal jsem také možnost **„Auto-tuningu“** a nahrání/uložení parametrů ze zálohy v PLC.



Obr. 3. 74: Nastavení PID regulátoru

U dvou-polohové regulace zde máme možnost nastavení účinnosti výstupu, jenž bude pec používat při on/off regulaci, tedy zda při sepnutí pojede na plných 100%, nebo při vypnutí skutečně klesne až na 0%.

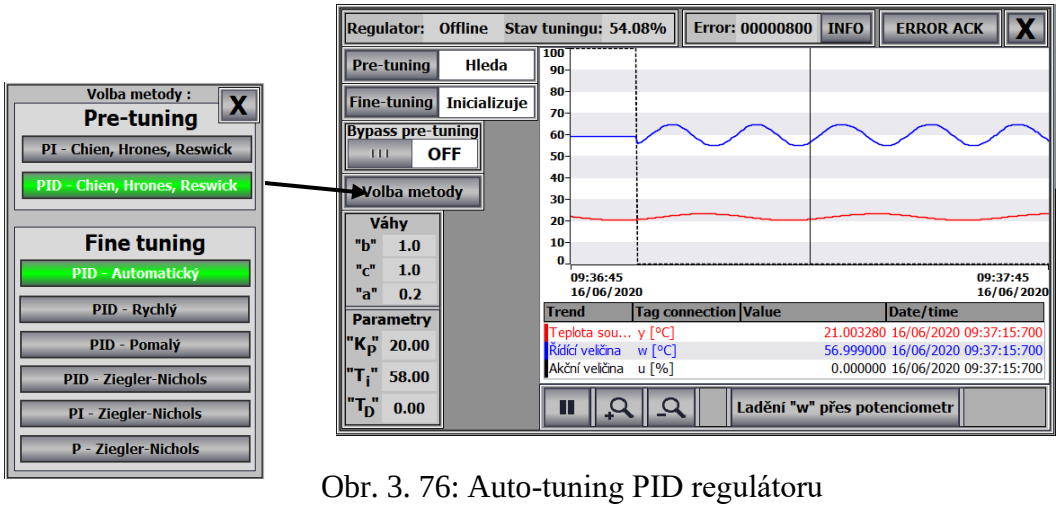
Hystereze požadované teploty, také jeden z důležitých prvků nastavení u tohoto způsobu regulace, díky ní posuneme hranici sepnutí/vypnutí spirály a ušetříme jí tak na životnosti, jelikož tato metoda bývá k hardwaru velice nepřívětivá. To je způsobeno tím, že časté vypínání a zapínání u určitých zařízení může způsobit jejich brzký konec.



Obr. 3. 75: Nastavení ON/OFF regulátoru

3.4.7 Auto-tuning

Tento PID regulátor má také v možnostech svůj vlastní **auto-tuning**, který se skládá ze dvou částí, **Pre-tuning** a **Fine-tuning**. Jedná se vlastně o zjednodušení práce pro uživatele/programátora, jak co nejrychleji a nejefektivněji nastavit potřebné parametry pro fungování regulátoru. Tento způsob vychází z obdobných metod, jako jsme si uvedli v kapitole výše. Jediný rozdíl je v tom, že za nás veškeré výpočty a nastavování provádí PLC a nám tak stačí pouze zmáčknout jedno tlačítko. Máme zde však možnost výběru, a to určit metodu, kterou necháme program provádět. V laboratorní úloze tento postup samozřejmě používat nebudeme, ale viděl jsem to jako dobrou ukázku průmyslového řešení parametrizace regulátoru v praxi, takže jsem pro zvídavé studenty tuto možnost také přidal.



Obr. 3. 76: Auto-tuning PID regulátoru

Doporučený postup při auto-tuningu:

V horní liště vidíme políčko **„Error“**, pokud tedy nastane nějaká chyba, regulátor ji ohlásí patřičným číslem a podle tohoto čísla zjistíme, v jakém místě nastala chyba. Seznam se všemi možnými chybami **„čísly“,** je ukryt pod tlačítkem **„INFO“.** Jakmile tuto chybu opravíme, musíme potvrdit její vyřešení přes tlačítko **„ERROR ACK“** a pak se teprve můžeme pustit do dalšího tuningu.

Tuning se spouští přes tlačítka **„Pre-tuning/Fine-tuning“,** ke kterým jsou přiřazena bílá okénka s informací, v jakém stavu se daný režim nachází, nebo zda jej například nelze použít. V horní liště je pak v procentech zobrazen stav tuningu a právě probíhající mód regulátoru. Jako nastavitelné prvky zde máme volbu metody výpočtu. Jednotlivé nastavené parametry a váhy jsou zde zobrazovány čistě pro informaci.

Pre-tuning – podmínkou je, že požadovaná hodnota musí být vzdálená od aktuální regulované hodnoty (ideální stav je při startu regulace se skokovou změnou žádané veličiny). Doporučuji zapínat z offline stavu regulátoru. Tento tuning probíhá ve dvou fázích:

1. Fáze – nastaví výstup na 0%, tedy offline
2. Fáze – nastaví výstup na 100%, tedy max

Fine-tuning – podmínkou je, že požadovaná hodnota musí být přibližně stejná jako regulovaná veličina. Jedná se o finální **„dolaďování“** parametrů a lze aplikovat jedině po úspěšném **pre-tuningu**. Je zde však možnost pomocí **„bypass“** obejít **pre-tuning** a přejít rovnou na **fine-tuning**, tento postup se však nedoporučuje.

Kapitola 4
Tvorba návodu
4.1 Horkovzdušný tunel – logické řízení
Popis úlohy:

Jedná se o horkovzdušný tunel, který je připojen k pásovému dopravníku s ovládacím panelem. Jako akční členy zde slouží dopravníkový pás a topné spirály, jenž lze přepínat mezi výkony 500W a 1000W. Tyto spirály jsou navíc ještě ofukovány větráky, pro lepší rozvod tepla v peci. Jako simulace vnějších vlivů zde slouží kovový rošt, který můžeme otevírat a uzavírat pomocí plastového kolečka na vrchní části tunelu. V ovládacím panelu je pak připraveno rozhraní pro laboratorní úlohu.

Obr. 4. 1: Horkovzdušný tunel [1]

Úkol:

Naprogramujte funkci programu v libovolném jazyce (LAD, STL, SCL), aby fungovala následným způsobem:

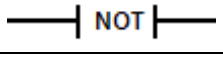
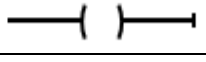
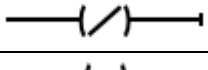
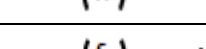
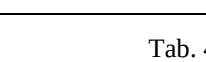
1. Horkovzdušný tunel čeká, až se na vstupním kapacitním senzoru objeví vozík. Jakmile je tato podmínka splněna, tak se rozsvítí tlačítko START. Po stisku tlačítka START se rozjede pás směrem VPRAVO a tlačítko zhasne.

2. Jakmile vozík dorazí na pozici v peci, tak se vysune stopper a zapne pec. Znovu se pak rozsvítí tlačítko START a po jeho stisknutí sjede stopper dolů, vypne se pec a vozík pokračuje na konec pásu, kde po dosažení koncového senzoru pohyb pásu vypne.

3. Pokud je kdykoliv během chodu stisknut nouzový stop, vše se zastaví a po deaktivaci nouzového stopu znovu rozjede.

4.BONUS: Pokaždé když má svítit tlačítko START, rozblikejte ho s frekvencí 0,5Hz.

Instrukční list:

	***Spínací kontakt***
	***Rozpínací kontakt***
	***Negace***
	***Výstup (uvede výstup do log.1)***
	***Negovaný výstup***
	***RESET Výstup (uvede výstup do trvalé log.0)***
	***SET Výstup (uvede výstup do trvalé log.1)***

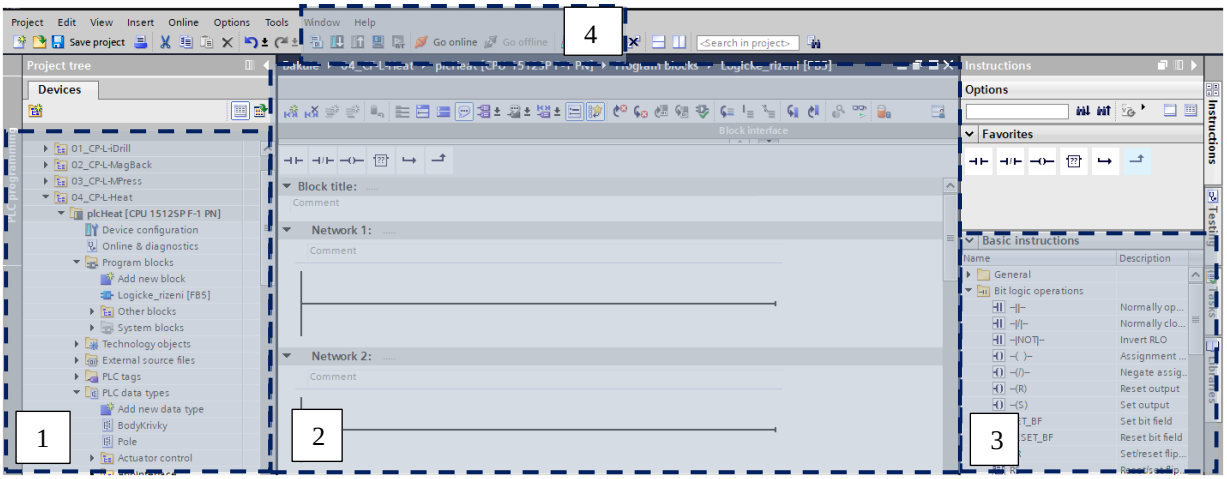
Tab. 4. 1: Instrukční list

IO LIST:

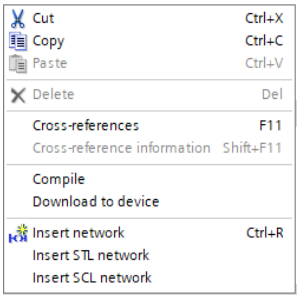
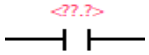
Adresy výstupů	Tag	Popis funkce
Q0.0	Topeni	***Topná spirála s větrákem***
Q1.0	LED_Start	***Led dioda pod tlačítkem***
Q1.1	LED_Reset	***Led dioda pod tlačítkem***
Q1.2	LED_1	***Led dioda***
Q1.3	LED_2	***Led dioda***
Q1.4	Pás_Doprava	***Pohyb dopravníkového pásu - doprava***
Q1.5	Pás_Doleva	***Pohyb dopravníkového pásu - doleva***
Q1.6	Pás_Zpomal	***Pohyb dopravníkového pásu – zpomalit (společně se směrem pohybu dopravníku)***
Q1.7	Vysun_Stopper	***Vysunou stoper pro zastavení vozíku***
Adresy vstupů	**Tag**	**Popis funkce**
I1.0	START	***Tlačítko spínací*** *Log.0= Nestlačeno, Log.1= Stlačeno*
I1.1	STOP	***Tlačítko rozpínací*** *Log.0= Stlačeno, Log.1= Nestlačeno*
I1.2	MANU/AUTO	***Přepínač*** *Log.0=Manu, Log.1= Auto*
I1.3	RESET	***Tlačítko spínací*** *Log.0= Nestlačeno, Log.1= Stlačeno*
I1.4	POZICE	***Indukční senzor polohy na dopravníku*** *Log.0= Neobsazeno, Log.1= Obsazeno*
I1.5	BEZPEČNOSTNÍ_STOP	***Rozpínací tlačítko*** *Log.0= Stlačeno, Log.1= Nestlačeno*
I1.6	ZAČÁTEK_DOPRAVNÍKU	***Kapacitní senzor na začátku dopravníku*** *Log.0= Neobsazeno, Log.1= Obsazeno*
I1.7	KONEC_DOPRAVNÍKU	***Kapacitní senzor na konci dopravníku*** *Log.0= Neobsazeno, Log.1= Obsazeno*
I42.7	STOPER_DOLE	***Pozice vysunutí stoperu*** *Log.0= Je vysunut, Log.1= Není vysunut*
Adresy merkerů	**Tag**	**Popis funkce**
M0.7	xF0_5Hz	***Bit spínaný frekvencí 0.5Hz***
M20.0	Teplota_30°C	***Hlášení dosažení teploty 30°C***
M20.1	Teplota_40°C	***Hlášení dosažení teploty 40°C***
M20.2	Teplota_50°C	***Hlášení dosažení teploty 50°C***
M20.3	Teplota_60°C	***Hlášení dosažení teploty 60°C***
M20.4	Teplota_70°C	***Hlášení dosažení teploty 70°C***

Tab. 4. 2: IO List

Návod na práci v TIA Portálu:

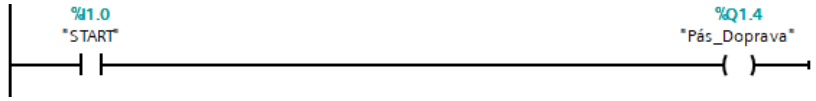


Obr. 4. 2: Programové rozhraní TIA Portal

1. Projektový strom	- zde je třeba najit složku „ 04_CP-L-Heat “ - v této složce najdeme PLC „ plcHeat [CPU 1512SP F-1 PN]“ - v Program blocks pak otevřít funkční blok „ Logicke_rizeni [FB5] “
2. Rozhraní FB5	- v tomto prostředí budeme programovat naši stanici v tzv. **networcích** - je zde možnost programovat v jazycích LAD, SCL, STL. - základní jazyk je LAD, ale pokud klikneme pravým tlačítkem do bílého pole, otevře se možnost přidání dalšího networku a to i s jiným jazykem 
3. Základní instrukce	- seznam instrukcí, které používáme při programování - tyto instrukce jednoduše přetáhnete levým tlačítkem do vašeho rozhraní FB5  - k těmto instrukcím je pak nutno na pozici „<??.?>“ doplnit adresu z **IO listu (např I1.0)** -seznam instrukcí najdete na úvodní straně návodu
4. Ovládací panel	- nahrání programu do PLC „ “ - kontrola chyb programu „ “ - předtím než se však program nahraje, tak proběhne automaticky kompilace a pokud PLC najde nějako chybu ve vašem programu, upozorní vás na ni a program do PLC nenahraje.

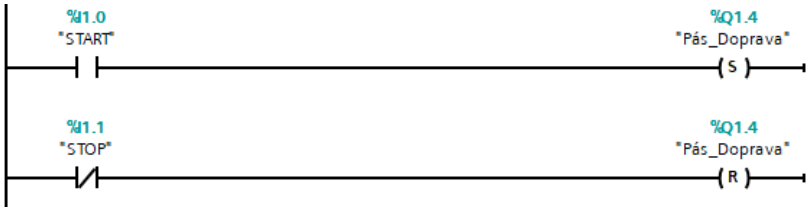
Vzory naprogramovaných networků v LAD:

Příklad: Pokud stiskneme tlačítko START, pás se rozjede doprava a pojede, dokud toto tlačítko budeme držet.



Obr. 4. 3: Vzor programu v jazyce LAD

Příklad: Pokud stiskneme tlačítko START, pás se rozjede doprava a pojede do té doby, dokud nestiskneme STOP.



Obr. 4. 4: Vzor programu v jazyce LAD

(POZOR: pokud nasetujeme „S“ výstup, pro vypnutí musíme bit restartovat „R“)

Network musí být vždy zakončen výstupem ! (např. ——( )——)

DŮLEŽÍTE:

PŘI TESTOVÁNÍ PROGRAMU JE NUTNO MÍT OTEVŘENOU OBRAZOVKU „LOGICKÉ ŘÍZENÍ“, JINAK VÁM PROGRAM NEPŮJDE !!!

Obr. 4. 5: Tlačítko pro spuštění obrazovky úlohy

Literatura:

HOFREITER, Milan. *Základy automatického řízení*. V Praze: České vysoké učení technické, 2012. ISBN 978-80-01-05007-1.

HOFREITER, Milan. *Základy automatického řízení - příklady*. 4. přepracované vydání. V Praze: České vysoké učení technické, 2016. ISBN 978-80-01-05899-2.

4.2 Horkovzdušný tunel – frekvenční vlastnosti

Popis úlohy:

Jedná se o horkovzdušný tunel, který je připojen k pásovému dopravníku s ovládacím panelem. Jako akční členy zde slouží dopravníkový pás a topné spirály, jenž lze přepínat mezi výkony 500W a 1000W. Tyto spirály jsou navíc ještě ofukovány větráky, pro lepší rozvod tepla v peci. Jako simulace vnějších vlivů zde slouží kovový rošt, který můžeme otevírat a uzavírat pomocí plastového kolečka na vrchní části tunelu. V ovládacím panelu je pak připraveno rozhraní pro laboratorní úlohu.

Obr. 4. 6: Horkovzdušný tunel [1]

Úkol:

Změřte alespoň 3 body amplitudové a fázové frekvenční charakteristiky této soustavy pomocí níže popsaného experimentu a vyznačte je v klasických souřadnicích (logaritmické souřadnice z důvodu nízkých frekvencí nepoužívat).

Teoretický základ pro vypracování úlohy:

Časové průběhy:

Vztahy pro stanovení amplitudového poměru:

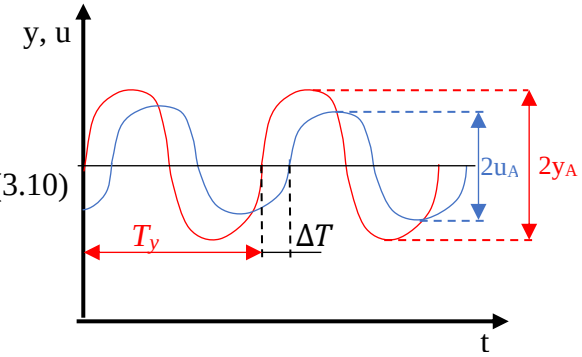
$$A(\omega) = |G(j\omega)| = \frac{y_A}{u_A} \quad (3.9)$$

$$A(\omega)_{dB} = |G(j\omega)|_{dB} = 20 \cdot log_{10}|G(j\omega)| \quad (3.10)$$

Vztahy pro stanovení fázového posuvu:

$$\varphi = -\frac{\Delta T}{T} 360° \quad (3.11)$$

$(T = T_y = T_u)$



Obr. 4. 7: Parametry časových průběhů

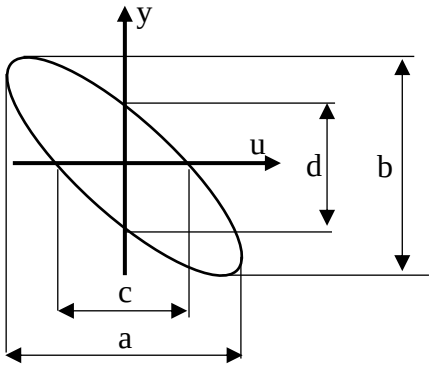
Lissajousův obrazec:

Vztahy pro stanovení amplitudového poměru:

$$A(\omega) = \frac{b}{a} \quad (3.5)$$

Vztahy pro stanovení fázového posuvu:

$$\varphi(\omega) = \arcsin\left(\frac{d}{b}\right) = \arcsin\left(\frac{c}{a}\right) \quad (3.6)$$



Obr. 4. 8: Parametry Lissajousova obrazce

(umístění φ do správného kvadrantu je nutno stanovit úvahou, převážně $\varphi < 0°$)

Doporučený postup experimentu:

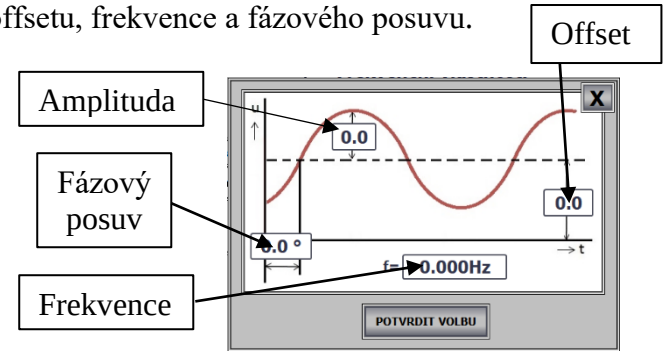
1. Na dotykovém panelu klikněte na tlačítko **„FREKVENČNÍ VLASTNOSTI“.** Otevře se vám obrazovka se schématem soustavy, která je buzena signálem a zaznamenávána dvěma typy grafů.

2. Klikněte na bílé okno **„Volba signálu“** a zvolte možnost **„Harmonický“**. Zde nastavte parametry amplitudy, offsetu, frekvence a fázového posuvu.

Doporučené rozsahy hodnot

Amplituda: 0 až 20
Frekvence: 0.001 až 1 Hz
Offset: 25 až 50
Fázový posuv: 0

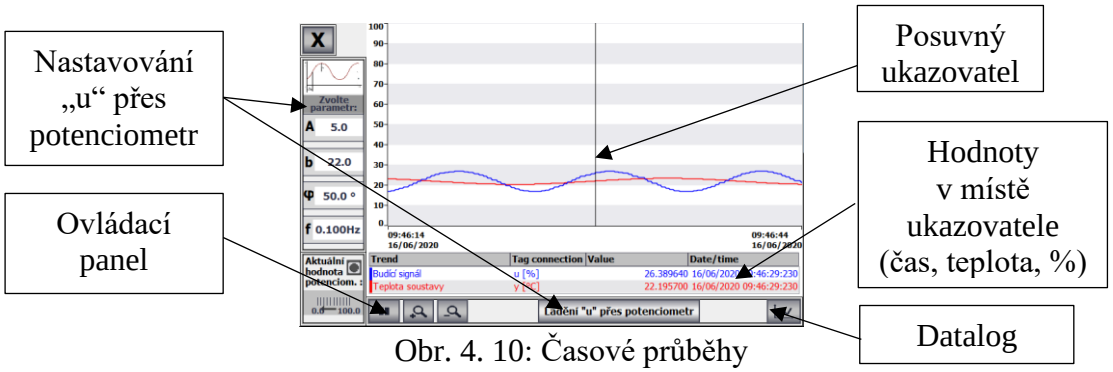
POZOR:
Maximální nastavení „u“je 100%!



Obr. 4. 9: Harmonický signál

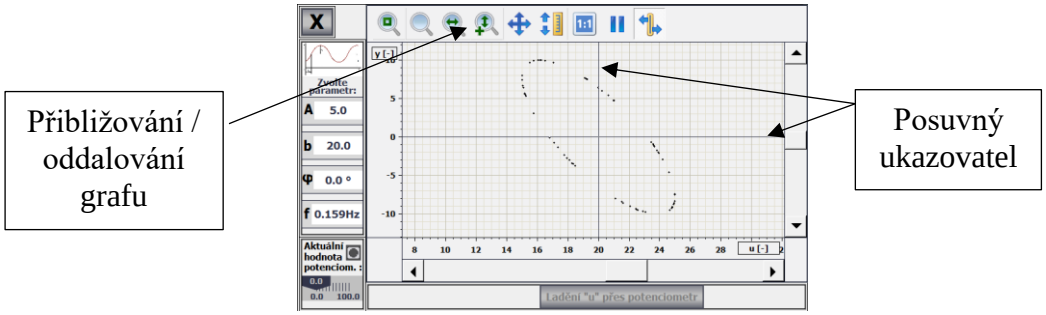
3. Poté tlačítkem **„Potvrdit volbu“** pustíte budící signál do soustavy a můžete začít pracovat s grafy.

4. Zvolte tlačítko **„Časové průběhy“** a zobrazí se vám graf s hodnotami. Zde naměřte hodnoty pro vztahy zadané v úkolu ($\mathbf{y_A}$, $\mathbf{u_A}$, ΔT)



Obr. 4. 10: Časové průběhy

5. Zvolte tlačítko **„Lissajousův obrazec“** a začne se vám vykreslovat křivka pomocí bodů. Zde naměřte hodnoty pro vztahy zadané v úkolu (**a, b, c, d)**



Obr. 4. 11: Lissajousův obrazec

Literatura:

HOFREITER, Milan. *Základy automatického řízení*. V Praze: České vysoké učení technické, 2012. ISBN 978-80-01-05007-1.

HOFREITER, Milan. *Základy automatického řízení - příklady*. 4. přepracované vydání. V Praze: České vysoké učení technické, 2016. ISBN 978-80-01-05899-2.

4.3 Horkovzdušný tunel – uzavřený regulační obvod

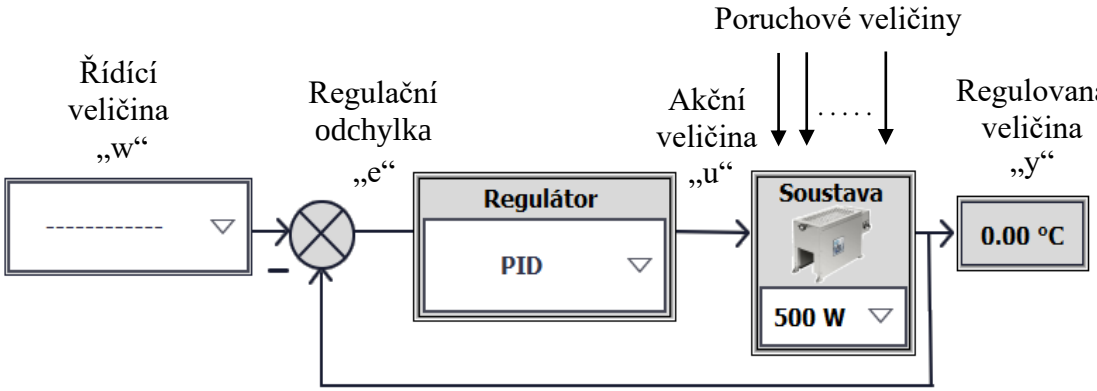
Popis úlohy:

Jedná se o horkovzdušný tunel, který je připojen k pásovému dopravníku s ovládacím panelem. Jako akční členy zde slouží dopravníkový pás a topné spirály, jenž lze přepínat mezi výkony 500W a 1000W. Tyto spirály jsou navíc ještě ofukovány větráky, pro lepší rozvod tepla v peci. Jako simulace vnějších vlivů zde slouží kovový rošt, který můžeme otevírat a uzavírat pomocí plastového kolečka na vrchní části tunelu. V ovládacím panelu je pak připraveno rozhraní pro laboratorní úlohu.

Obr. 4. 12: Horkovzdušný tunel [1]

Úkol:

Před sebou máte uzavřený regulační obvod, kde řízená **„Soustava“** je horkovzdušný tunel. Ve zpětné vazbě pak dostáváme informaci od senzoru teploty PT100 o teplotě soustavy **„y“**.



Obr. 4. 13: Uzavřený regulační obvod

Vaším úkolem je:

1. Nastavit PID regulátor pomocí „**metody relé Åström-Hägglund**“, aby zvládal řídit teplotu soustavy dle požadavků řízení „w“ a docílit tak ideálního případu, kdy $w(t) \approx y(t)$.

2. Vypracovat tabulku s veškerými naměřenými hodnotami a výpočty.

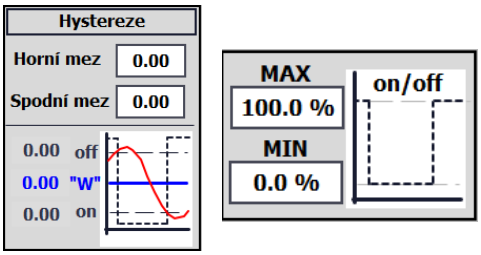
3. Otestovat funkčnost nastaveného PID regulátoru, pomocí generátoru signálu řídící veličiny „w“.

4. V závěru porovnat jednotlivé metody (PID, ON/OFF) a zhodnotit, kde je jaký regulátor vhodné použít a kde ne.

Doporučený postup:

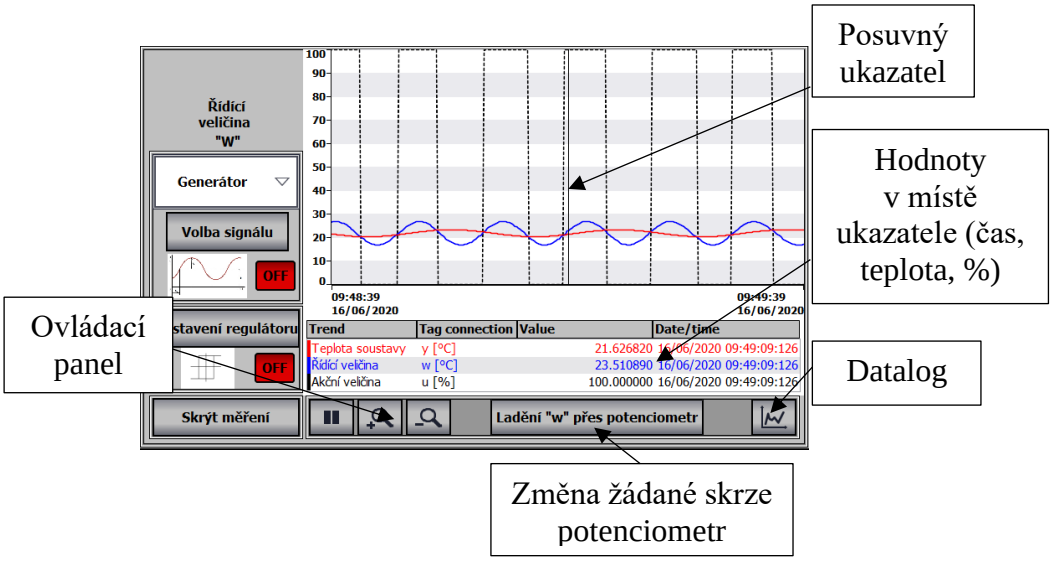
1. Na dotykovém panelu klikněte na tlačítko **„UZAVŘENÝ REGULAČNÍ OBVOD“.** Otevře se vám obrazovka se schématem.

2. Klikněte na bílé okno **„Regulátor“** a zvolte možnost ON/OFF. Na obrazovce se vám potom hned objeví tlačítko možnosti nastavení regulátoru. Toto nastavení otevřete a nastavte parametry v bílých oknech ON/OFF regulátoru na:



Obr. 4. 14: Nastavení dvoupolohové regulace

3. Poté klikněte na přepínač **„Zobrazit měření“**. Zobrazí se vám okno, na kterém se budou vykreslovat charakteristiky a zde budete provádět svá měření. K měření využívejte posuvného ukazatele, který vždy vypisuje dole v tabulce hodnoty všech křivek v místě, kde je tento ukazatel protnut. Měření si kdykoliv můžete stopnou a znovu spustit, nebo přiblížit/oddálit, pomocí ovládacího panelu pod tabulkou s měřenými hodnotami.
(Pozn. formát času v tabulce: HOD : MIN : SEC : MSEC)



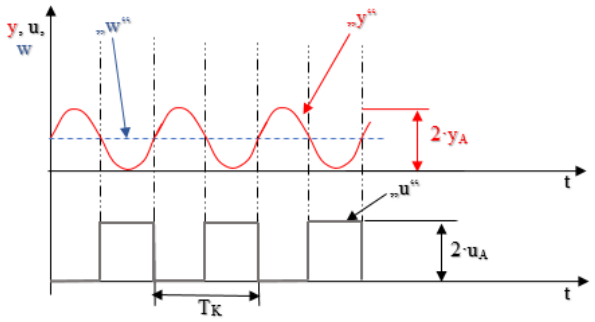
Obr. 4. 15: Probíhající měření regulace soustavy

(hodnoty z datalogu se ukládají do excelu na ploše PC)

4. Okno s měřením si nechte otevřené a v možnostech **„Řídící veličina W“** zvolte **„Vlastní hodnota“,** kde zadejte požadovanou teplotu 40° C. V tu chvíli se Vám spustí regulátor a běh soustavy.

5. Z charakteristik, které mohou vypadat podobně jako na obrázku, odměřte potřebné hodnoty amplitud pro výpočet K_{Pk} a kritickou periodu T_K.

$$K_{Pk} = \frac{4 \cdot u_A}{\pi \cdot y_A} \quad (3.21)$$



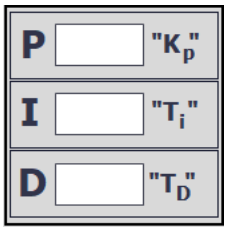
Obr. 4. 16: Křivky při ON/OFF regulaci

6. Naměřené a vypočtené hodnoty pak použijte pro zjištění PID parametrů za pomocí Ziegler-Nicholsovy tabulky:

Regulátor	$K_{Pk} = r_0$	$T_I = r_0/r_I$	$T_D = r_D/r_0$
P	0,5 K_{Pk}	-	-
PI	0,45 K_{Pk}	0,85T_K	-
PID	0,6 K_{Pk}	0,5T_K	0,125T_K

Tab. 4. 3: Metoda kritických parametrů [9]

7. Stiskněte tlačítko **„Skrýt měření“** a vyměňte ON/OFF regulátor za PID. Zde otevřete jeho nastavení a zvolte mód „Auto“ tedy automatický režim. V něm nastavte vaše vypočtené parametry regulátoru.



Obr. 4. 17: Parametry PID regulátoru

8. Pro zobrazení charakteristik znovu stiskněte **„Zobrazit měření“**. Nyní můžete pozorovat, jak kvalitně jste nastavili regulátor. Zkuste například nastavit skokovou změnu na žádané hodnotě. Tam kde teď máte 40°C nastavte 50°C, nebo si zvolte vlastní řídící veličinu **„w“** z generátoru.

Literatura:

HOFREITER, Milan. *Základy automatického řízení*. V Praze: České vysoké učení technické, 2012. ISBN 978-80-01-05007-1.

HOFREITER, Milan. *Základy automatického řízení - příklady*. 4. přepracované vydání. V Praze: České vysoké učení technické, 2016. ISBN 978-80-01-05899-2.

Závěr

Hlavním cílem této práce bylo zhotovit tři laboratorní úlohy pro předmět Automatické řízení, a to za pomocí jedné z výukových stanic CP Lab.

Na začátku jsme si tedy prošli všechny možné volby stanic s jejich veškerými výhodami a nevýhodami, které nám pro potřebné úlohy mohly poskytnout. Rozebrali jsme tak nejen jejich konstrukci, ale i komunikaci, chování a navrhli jejich možné vylepšení do budoucna. Ve výsledku vyšlo, že nejlepší volbou pro nás bude modul horkovzdušného tunelu v kombinaci s dopravníkem.

V další části práce jsme si pak úlohy rozdělili jako podkapitoly. U každé podkapitoly jako první probíráme patřičnou teorii, ke které se tato úloha váže. Poté vždy přicházíme s určitými postupy, které může student použít při řešení a úskalí jenž s touto úlohou souvisí. První úloha je postavena na samostatném programování studenta a další dvě jsou zaměřeny spíše na probíranou teorii a její aplikování v praxi.

Tvorba úloh je pak složena ze dvou částí. První z nich je program samotný, tedy mnou vytvořené funkční bloky a celkový algoritmus fungování úlohy v PLC. V této práci však neukazuji veškeré funkční bloky, funkce a části programu, které jsem vytvořil a jsou nezbytné pro správné fungování. Chtěl jsem se totiž více zaměřit na úlohu samotnou, a tak si vždy ukazujeme pouze bloky, co souvisí s nějakou specifickou částí úlohy. Celý kód je pak samozřejmě k dispozici v příloze. Druhou částí je pak grafické rozhraní v ovládacím panelu, jenž je také pro každou úlohu specificky vytvořeno a napojeno na program v PLC. Samotný program studenti vidět nemohou, tedy pokud si nepřečtou tuto bakalářskou práci, ale HMI rozhraní již uvidí celé. Zde jsem se snažil o co nejefektivnější a přehledné zobrazení jednotlivých tlačítek a interaktivních polí, aby tak student zbytečně nepátral po obrazovce a bylo vše srozumitelně na svém místě.

Přidal jsem zde několik prvků různých nastavení, omezení, simulátor výstupu, či auto-tuning, které ale k samotnému splnění úlohy nejsou potřeba, avšak pro určité zvídavé jedince mohou být tyto prvky zajímavé a pomoci jim pochopit dané téma více do hloubky.

V poslední části práce jsou pak vytvořeny návody pro jednotlivá laboratorní cvičení, kde jsou zadané úkoly a postupy pro správné řešení dané úlohy. Úlohy jsou postaveny tak, aby se časově vešly do délky laboratorního cvičení a nebyly tedy příliš náročné, jelikož spousta studentů se setkává s touto oblastí, tedy automatizací poprvé.

Mé finální hodnocení je tedy takové, že mohu prohlásit laboratorní úlohy za funkční a připravené k provozu v laboratoři. Úlohy jsem simuloval, jak na svém počítači, tak ve skutečném PLC v laboratoři. Je tu však jeden problém, který se týká samotného ET200SP. Sám o sobě totiž nemá příliš dobrou výpočetní sílu, a tak často nezvládá například provozovat PID regulátor při příliš rychlých změnách požadované hodnoty. Tento problém jsem objevil právě, až po porovnání s chodem programu na počítači a PLC, kde výpočetní výkon samotného ET200SP nemá proti simulaci, která probíhá na počítači šanci. Jako zajímavý námět pro další práci tak vidím otestovat, jak by podobné úlohy obstály na PLC S7-1500 s novějším a výkonnějším procesorem.

Seznamy

Seznam použité literatury:

[1] Heat Tunnel manual. In: *Festo didactic - CP Factory/Lab - Application modules* [online]. Denkendorf: Festo Didactic SE, c2019 [cit. 2019-02]. Dostupné z: https://ip.festo-didactic.com/InfoPortal/CPFactoryLab/data/download.php?model=CP-AM-HEAT&file=manual&lang=en

[2] Heat Tunnel data sheets. In: *Festo didactic - CP Factory/Lab - Application modules* [online]. Denkendorf: Festo Didactic SE, c2019 [cit. 2019-02]. Dostupné z: https://ip.festo-didactic.com/InfoPortal/CPFactoryLab/data/download.php?model=CP-AM-HEAT&file=datasheets&lang=en

[3] iDrilling manual. In: *Festo didactic - CP Factory/Lab - Application modules* [online]. Denkendorf: Festo Didactic SE, c2019 [cit. 2019-02]. Dostupné z: https://ip.festo-didactic.com/InfoPortal/CPFactoryLab/data/download.php?model=CP-AM-iDRILL&file=manual&lang=en

[4] iDrilling data sheets. In: *Festo didactic - CP Factory/Lab - Application modules* [online]. Denkendorf: Festo Didactic SE, c2019 [cit. 2019-02]. Dostupné z: https://ip.festo-didactic.com/InfoPortal/CPFactoryLab/data/download.php?model=CP-AM-iDRILL&file=datasheets&lang=en

[5] CP Lab Conveyor manual. In: *Festo didactic - CP Factory/Lab - Base modules* [online]. Denkendorf: Festo Didactic SE, c2019 [cit. 2019-11]. Dostupné z: https://ip.festo-didactic.com/InfoPortal/CPFactoryLab/data/download.php?model=CP-L-CONV&file=manual&lang=en

[6] CP Lab Conveyor data sheets. In: *Festo didactic - CP Factory/Lab - Base modules* [online]. Denkendorf: Festo Didactic SE, c2019 [cit. 2019-11]. Dostupné z: https://ip.festo-didactic.com/InfoPortal/CPFactoryLab/data/download.php?model=CP-L-CONV&file=datasheets&lang=en

[7] CP Lab Conveyor circuit diagrams. In: *Festo didactic - CP Factory/Lab - Base modules* [online]. Denkendorf: Festo Didactic SE, c2018 [cit. 2018-26-11]. Dostupné z: https://ip.festo-didactic.com/InfoPortal/CPFactoryLab/data/download.php?model=CP-L-CONV&file=circuit-diagram&lang=en

[8] WADE, Harold L. & Harold L. WADE. Basic and advanced regulatory control: system design and application. 2nd ed. Research Triangle Park, NC: ISA-The Instrumentation, Systems, and Automation Society, c2004. ISBN 1-55617-873-5.

[9] VÍTEČKOVÁ, Miluše a Antonín VÍTEČEK. Vybrané metody seřizování regulátorů. Ostrava: Vysoká škola báňská - Technická univerzita, 2011. ISBN 978-80-248-2503-8.

[10] Vilanova, R. & Visioli, A. Pid Control in the Third Millennium; Lessons Learned and New Approaches (Advances in Industrial Control). Springer Verlag London Ltd., 2012. ISBN 978-1-4471-2424-5

[11] ÅSTRÖM, Karl J., Tore HÄGGLUND a Karl J. ÅSTRÖM. *PID controllers*. 2nd ed. Research Triangle Park, N.C.: International Society for Measurement and Control, c1995. ISBN 1-55617-516-7.

[12] *Croline Teti,* Introduction to Lissajous Figures. In: *Academia.edu* [online]. University fo California, c2018 [cit. 2018]. Dostupné z: https://www.academia.edu/37261286/Introduction_to_Lissajous_Figures._Exploring_Variable_Effects_in_a_System_of_Parametric_Equations

[13] Jeff DeAngelis, IO Link Handbook. In: *Maximintegrated.com* [online]. Maxim Integrated, United States c2017 [cit. 2017-08]. Dostupné z: https://pdfserv.maximintegrated.com/en/sg/IO-Link-Handbook.pdf

[14] HOFREITER, Milan. *Základy automatického řízení*. V Praze: České vysoké učení technické, 2012. ISBN 978-80-01-05007-1.

[15] HOFREITER, Milan. *Základy automatického řízení - příklady*. 4. přepracované vydání. V Praze: České vysoké učení technické, 2016. ISBN 978-80-01-05899-2.

[16] Vance Vandorfen, Ph.D., P.E., Relay Method Automates PID. In: *controleng.com* [online]. c2009 [cit. 2009-09]. Dostupné z: https://www.controleng.com/articles/relay-method-automates-pid-loop-tuning/

[17] prof. Dr. Ing. Petr Novák, Průmyslové řídicí systémy. V Ostravě: Vysoká škola báňská – Technická univerzita Ostrava, 2013. ISBN 978-80-248-3032-2.

[18] SIMATIC ET200SP. In: https://new.siemens.com/ [online]. Fest AG c2018 [cit. 2018-08]. Dostupné z: https://new.siemens.com/global/en/products/automation/systems/industrial/io-systems/et-200sp.html

[19] IO -Link DA interface. In: https://www.festo-didactic.com/ [online]. Festo Didactic c2014 [cit. 2014-12]. Dostupné z: https://www.festo-didactic.com/ov3/media/customers/1100/8038559_deenesfr_v2.0_lp8038560_io_link_da_interface_brief_description_print.pdf

[20] Programming guideline for S7-1500. In: https://support.industry.siemens.com / [online]. Siemens c2017 [cit. 2017-03]. Dostupné z: https://support.industry.siemens.com/cs/document/90885040/programming-guideline-for-s7-1200-s7-1500?dti=0&lc=en-CZ

[21] O´DWYER, Aidan. Handbook of PI and PID Controller Tuning Rules. Dublin Institute of Technology, Ireland. 2009. ISBN 978-1-84816-242-6.

Seznam obrázků:

Seznam tabulek:

Seznam příloh:

Elektronické přílohy (na CD)

BakalářskáPráce – Soubor s programem pro TIA15.1 (.zap15_1)

Datové bloky - Použité datové bloky v programu (.pdf)

Funkční bloky – Použité funkční bloky v programu (.pdf)

Funkce - Použité funkce v programu (.pdf)

OB1 – Main – Hlavní organizační blok v programu (.pdf)

OB30 – Cyklické přerušení – Cyklický blok v programu (.pdf)

Obrazovky – Vytvořené obrazovky v programu (.pdf)

Seznam použitého softwaru:

TIA Portal V15.1

SIMATIC STEP 7 V15.1

SIMATIC WinCC V15.1