



# VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

## FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

## ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

DEPARTMENT OF CONTROL AND INSTRUMENTATION

## HMI PRO BECKHOFF TWINCAT 3

HMI FOR BECKHOFF TWINCAT 3

### SEMESTRÁLNÍ PRÁCE

SEMESTRAL THESIS

### AUTOR PRÁCE

AUTHOR

Bc. Ondřej Balga

### VEDOUCÍ PRÁCE

SUPERVISOR

Ing. Jan Pásek, CSc.

BRNO 2018

# Diplomová práce

magisterský navazující studijní obor **Kybernetika, automatizace a měření**

Ústav automatizace a měřicí techniky

**Student:** Bc. Ondřej Balga

**ID:** 158631

**Ročník:** 2

**Akademický rok:** 2017/18

**NÁZEV TÉMATU:**

## HMI pro BECKHOFF TWINCAT 3

### POKYNY PRO VYPRACOVÁNÍ:

Naprogramovat HMI aplikaci pro TwinCAT verze 3.0. se základními potřebnými objekty s možností nasazení na vhodnou technologii zařízení ADS. Cíle práce jsou:

1. Vytvoření komunikace na ADS zařízení pro HMI.
2. Vytvoření standardních objektů pro HMI - průmyslové objekty, alarmy, trendy, zařízení.
3. Vytvoření spravovatelného přihlášení pro uživatele s možností práv.
4. Vytvoření záznamového deníku (journal, žurnál) o provedených úkonech z daného HMI.
5. Vytvoření ukázkového řešení HMI pro ŘS TwinCAT 3.

### DOPORUČENÁ LITERATURA:

- [1] Beckhoff Information System [online]. Německo: Beckhoff, 2017 [cit. 2017-04-21]. Dostupné z: <https://infosys.beckhoff.com/>
- [2] TS1800 | TwinCAT PLC HMI - BECKHOFF New Automation Technology, online: [https://www.beckhoff.com/english.asp?twincat/twincat\\_plc\\_hmi.htm](https://www.beckhoff.com/english.asp?twincat/twincat_plc_hmi.htm)
- [3] WMS in Dynamics AX. Dostupné z: <http://www.uxceclipse.com/production-order-receiving-advanced-wms-dynamics-ax/>
- [4] Report as finish with a route card journal in AX 2012. Dostupné z: <https://timsaxblog.wordpress.com/2015/10/12/report-as-finish-with-a-route-card-journal-in-ax-2012/>

**Termín zadání:** 5. 2. 2018

**Termín odevzdání:** 14. 5. 2018

**Vedoucí práce:** Ing. Jan Pásek, CSc.



**doc. Ing. Václav Jirsík, CSc.**  
předseda oborové rady

### UPOZORNĚNÍ:

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



## **Abstrakt**

Tato práce se věnuje vytvoření HMI v programovacím jazyce C# pro BECKHOFF TWINCAT 3. V teoretické části se práce zaměřuje na popsání .NET frameworku, WPF, MVVM, Prism a softwarových prostředků, které jsou v práci použity. Následující kapitola se poté věnuje tvorbě vývojového prostředí a runtime. Jsou zde popsány všechny funkcionality a možnosti vytvořené aplikace. Použitý vytvořeného vývojového prostředí a runtime je demonstrováno na ukázkovém řešení.

## **Klíčová slova**

HMI, SCADA, TWINCAT 3, C#, .NET, PC, ADS, WPF, XAML, PLC

## **Abstract**

This thesis deals with the creation of HMI in the C # programming language for BECKHOFF TWINCAT 3. In the theoretical part, the thesis focuses on description of .NET framework, WPF, MVVM, Prism and software tools used in the thesis. The following chapter focuses on creating a development environment and a runtime. All functionalities and capabilities of the created application are described. The created development environment and runtime are used and demonstrated on the sample solution.

## **Keywords**

HMI, SCADA, TWINCAT 3, C#, .NET, PC, ADS, WPF, XAML, PLC

### **Bibliografická citace:**

BALGA, O. HMI pro BECKHOFF TWINCAT 3. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2018. 67 s. Vedoucí diplomové práce Ing. Jan Pásek, CSc.

## **Prohlášení**

„Prohlašuji, že svou diplomovou práci na téma „HMI pro BECKHOFF TWINCAT 3“ jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

V Brně dne: **11. května 2018**

.....  
podpis autora

## **Poděkování**

Děkuji vedoucímu diplomové práce Ing. Janu Páskovi, CSc. za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé diplomové práce.

V Brně dne: **11. května 2018**

.....  
podpis autora

# Obsah

1.	Úvod.....	13
2.	Teoretická část .....	14
2.1	Framework .NET.....	14
2.1.1	Jazyk C#.....	16
2.1.2	WinForms (WF).....	16
2.1.3	eXtensible Application Markup Language (XAML).....	17
2.1.4	Windows Presentation Foundation (WPF) .....	17
2.2	Model-View-ViewModel (MVVM) .....	18
2.3	Prism .....	19
2.4	SQLite .....	20
2.5	eXtensible Markup Language (XML).....	21
3.	Softwarové prostředky .....	22
3.1	Microsoft Visual Studio .....	22
3.2	Beckhoff.....	23
3.3	TwinCAT .....	24
3.3.1	Strukturový text .....	26
3.3.2	Jazyk kontaktních schémat .....	27
3.3.3	Seznam instrukcí.....	27
3.3.4	Funkční bloky .....	28
3.3.5	Metoda sekvenčních funkčních grafů .....	28
4.	Tvorba vývojového prostředí a runtime .....	29
4.1	Postup vytváření projektu .....	29
5.	Funkce vývojového prostředí a runtime.....	33
5.1	Struktura vývojového prostředí.....	33
5.1.1	Sdílená služba ADS .....	35
5.1.2	Sdílená služba SQLite.....	37
5.2	Design vývojového prostředí .....	37
5.2.1	Panel nástrojů.....	41
5.2.1.1	PageButton .....	41
5.2.1.2	Zařízení.....	42

5.2.1.3	Alarmy .....	42
5.2.1.4	Žurnál .....	43
5.2.1.5	Label .....	44
5.2.1.6	Button .....	45
5.2.1.7	Ledka, Čerpadlo, Ventil, PipeI, PipeL, PipeT a Míchadlo .....	45
5.2.1.8	Parametr .....	46
5.2.1.9	Trend .....	47
5.2.2	Pracovní plocha .....	47
5.2.3	Průzkumník řešení .....	48
5.2.3.1	Uživatelé .....	49
5.2.3.2	Alarmy .....	49
5.2.3.3	Obrazovky .....	50
5.2.3.4	Zařízení .....	51
5.2.3.5	Design .....	51
5.2.3.6	Menu .....	52
5.2.4	Vlastnosti objektu .....	52
5.3	Struktura Runtime .....	52
5.4	Design Runtime HMI .....	53
5.5	Instalační balíček HMI .....	54
6.	Tvorba ukázkového řešení .....	55
6.1	Tvorba programu PLC .....	55
6.1.1	Ventil .....	55
6.1.2	Čerpadlo .....	56
6.1.3	Tank .....	56
6.1.4	Poloautomatický režim .....	57
6.1.5	Automatický režim .....	58
6.2	Tvorba HMI .....	59
7.	Závěr .....	64

# Seznam symbolů a zkratek

## Zkratky:

HMI	...	Human Machine Interface technologií
PLC	...	Programmable Logic Controller
FCL	...	Framework Class Library
CLR	...	Common Language Runtime
WPF	...	Windows Presentation Foundation
WF	...	WinForms
WCF	...	Windows Communication Foundation
CIL	...	Common Intermediate Language
UI	...	User Interface
XAML	...	eXtensible Application Markup Language
XML	...	eXtensible Markup Language
MVVM	...	Model-View-ViewModel
ADS	...	The Automation Device Specification
TwinCAT	...	The Windows Control Automation Technology
ST	...	Structured Text
LD	...	Ladder
SFC	...	Sequential Function Chart

## Seznam obrázků

Obr. 1 Framework .NET [1] .....	15
Obr. 2 Převod do jazyka CIL [2] .....	15
Obr. 3 Ukázka kódu v jazyce C# .....	16
Obr. 4 Ukázka WinForms aplikace [3] .....	17
Obr. 5 Ukázka XAML .....	17
Obr. 6 Ukázka WPF aplikace [4] .....	18
Obr. 7 Model-View-ViewModel [6] .....	19
Obr. 8 Prism – kompozitní aplikace [7] .....	20
Obr. 9 Ukázka použití SQLite v jazyce C# .....	21
Obr. 10 Ukázka XML .....	21
Obr. 11 Microsoft Visual Studio .....	22
Obr. 12 Beckhoff sortiment [8] .....	23
Obr. 13 Základní struktura software TwinCAT 3 [9] .....	24
Obr. 14 Způsob použití mnohójádrových aplikací [11] .....	25
Obr. 15 ADS komunikace [12] .....	26
Obr. 16 Strukturní text .....	26
Obr. 17 Jazyk kontaktních schémat .....	27
Obr. 18 Seznam instrukcí .....	27
Obr. 19 Funkční bloky .....	28
Obr. 20 Metoda sekvenčních funkčních grafů .....	28
Obr. 21 Vytvořený Bootstrap pro vývojové prostředí .....	30
Obr. 22 Ukázka tvoby jednoho z Modulů .....	30
Obr. 23 Rozhraní sdílené služby ADS .....	31
Obr. 24 Rozhraní sdílené služby SQLite .....	31
Obr. 25 Vytvořený bootstrap pro runtime .....	32
Obr. 26 Ukázka vytváření instalačního souboru .....	32
Obr. 27 Struktura vývojového prostředí .....	34
Obr. 28 Ukázka agregátoru událostí pro Workspace a Property moduly .....	35
Obr. 29 Datový tok mezi HMI a ADS zařízením .....	35
Obr. 30 Navázání komunikace s ADS zařízením .....	36

Obr. 31 Vzhled vývojového prostředí.....	38
Obr. 32 File menu .....	39
Obr. 33 Editační menu .....	40
Obr. 34 Ukázka projektového menu .....	40
Obr. 35 Ukázka objektů v panelu nástrojů .....	41
Obr. 36 PageButton.....	42
Obr. 37 Zařízení.....	42
Obr. 38 Alarmy .....	43
Obr. 39 Vlastnosti žurnálu .....	43
Obr. 40 Uživatelský žurnál .....	43
Obr. 41 Procesní žurnál .....	44
Obr. 42 Label .....	44
Obr. 43 Button .....	45
Obr. 44 Ledka, Čerpadlo, Ventil, PipeI, PipeL a PipeT .....	46
Obr. 45 Parametr.....	46
Obr. 46 Trend.....	47
Obr. 47 Ukázka objektu na pracovní ploše.....	48
Obr. 48 Průzkumník řešení .....	48
Obr. 49 Uživatelé.....	49
Obr. 50 Alarmy .....	50
Obr. 51 Ukázka XML uložené obrazovky.....	50
Obr. 52 Ukázka ukládání zařízení.....	51
Obr. 53 Ukázka lišty alarmů .....	51
Obr. 54 Design .....	52
Obr. 55 Struktura Runtime.....	53
Obr. 56 Ukázka načteného projektu v RUNTIME .....	54
Obr. 57 Funkční blok ventilu .....	56
Obr. 58 Funkční blok čerpadla .....	56
Obr. 59 Funkční blok Tanku.....	57
Obr. 60 Vývojový diagram automatického režimu .....	58
Obr. 61 Design HMI .....	59
Obr. 62 Menu .....	60

Obr. 63 Vytvořené alarmy .....	60
Obr. 64 Technologie ukázkového řešení .....	61
Obr. 65 Parametry ukázkového řešení .....	62
Obr. 66 Obrazovka trendů .....	62
Obr. 67 Ukázka procesního žurnálu automatického režimu .....	63
Obr. 68 Diagnostika .....	63

# 1. ÚVOD

HMI (Human Machine Interface) tvoří rozhraní mezi zařízeními (systémem, robotem) a člověkem (obsluhou). Dříve bylo toto rozhraní označováno jako vizualizace a ovládání stroje. HMI nabízí objekty pro zobrazení a předání informace o stavu zařízení obsluze a zároveň poskytuje možnost k zadávání parametrů a ovládání stroje.

V době nástupu řídicích systémů v roce 1969/1970 byla vizualizace přímo na stroji problematická a cenově nákladná. Z tohoto důvodu se používali monochromatické fluorescenční nebo LCD displeje doplněné o klávesnici. Pouze pro náročné a obsáhlé vizualizace byli používány samostatné sestavy PC a sériová komunikace dat z PLC.

Dnešní ovládací panely už nelze srovnávat s prvními prostředky vizualizace. Technologický pokrok zlepšil výkon i cenovou dostupnost. HMI není už jen přídatkem k PLC, ale tvoří jeden celek. Současné HMI nabízí různé prvky, animace, objekty, grafy, archivaci hodnot, receptury, zobrazování provozních a chybových hlášení, řízení oprávnění přístupu, paralelní provoz více panelů, vzdálenou správu a další.

Možnosti HMI, účel i provedení se liší nejen podle množství předávaných informací a potřeb ovládání, ale i podle účelu zařízení a ergonomiky. Prakticky by měla být vizualizace a ovládání řešeno podle konkrétní aplikace a užití, avšak šablonovitě a standardizovaně.

Diplomová práce se věnuje tvorbě vývojového prostředí a runtime HMI. Cílem je vytvoření prostředí, kde si uživatel bude moci jednoduše, a přitom rozsáhle vytvořit HMI pro zařízení s TwinCAT 3. Vytvořenou vizualizaci může uživatel spustit pomocí jednoho tlačítka nebo při startu systému v podobě runtime.

Práce se zabývá tvorbou HMI pro TwinCAT 3 v programovacím jazyce C#. V následujících kapitolách jsou popsány softwarové prostředky použité pro tvorbu. Následuje popis vytvořeného vývojového prostředí a jeho runtime. V závěru jsou poté shrnuty dosažené výsledky a popsána možnost dalšího rozvoje.

## 2. TEORETICKÁ ČÁST

V následující části budou vysvětleny všechny potřebné pojmy a teorie použité v diplomové práci. Největší část teoretické části zaujímá .NET framework, díky kterému jsem poté celé vývojové prostředí a runtime vytvořil. Teoretická část se věnuje popsání a vysvětlení jazyků C#, XML a XAML. Je zde popsán starší grafický subsystém WF a také jeho nástupce WPF, který je dále v práci používán. Je zde popsán také návrhový vzor MVVM, který je taktéž v práci použit. V teorii je také popsána kompozitní aplikace pomocí Prism. Kapitola je zakončena popisem souborového databázového systému SQLite.

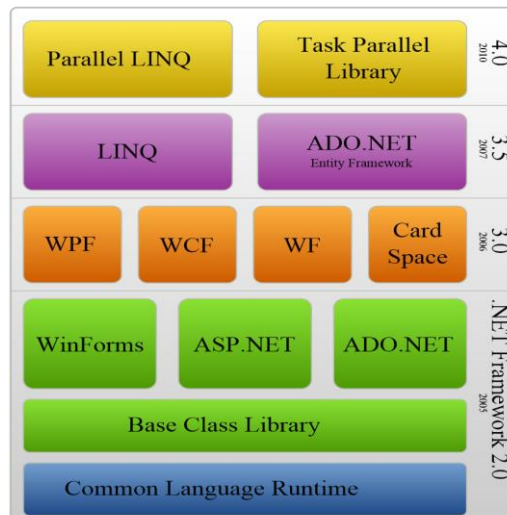
### 2.1 Framework .NET

Framework je softwarová struktura, která slouží jako podpora při programování a vývoji a organizaci jiných softwarových projektů. Může obsahovat podpůrné programy, knihovny API, podporu pro návrhové vzory nebo doporučené postupy při vývoji. Ve zkratce tedy můžeme říci, že obsahuje strukturu, kterou bychom bez něj museli složitě implementovat. Díky jeho použití se poté můžeme věnovat hlavní řešené problematice.

.NET Framework je softwarový framework vyvinutý společností Microsoft. Obsahuje velkou knihovnu s názvem Framework Class Library (FCL) a poskytuje jazykovou interoperabilitu v několika programovacích jazycích. Programy napsané pro rozhraní .NET Framework se spouštějí v softwarovém prostředí s názvem Common Language Runtime (CLR), virtuálním strojem aplikace, který poskytuje služby, jako je zabezpečení, správa paměti a správa výjimek. FCL poskytuje uživatelské rozhraní, přístup k datům, připojení k databázi, kryptografii, vývoj webových aplikací, numerické algoritmy a síťovou komunikaci.

Framework je určen pro většinu nových aplikací vytvořených pro platformu Windows. Společnost Microsoft také vytváří integrované vývojové prostředí, které je z velké části určeno pro .NET software s názvem Visual Studio. .NET Framework vedlo k vytvoření rodiny platform .NET zaměřených na mobilní výpočetní systémy, vestavěná zařízení, alternativní operační systémy a plug-iny webového prohlížeče.

.NET Framework v dnešní době obsahuje WinForms, ASP.NET, ADO.NET, WPF, WCF a Entity Framework. Platforma nepředepisuje a nespecifikuje použití programovacího jazyka, ale vždy se kód přeloží do mezi-jazyka CIL.

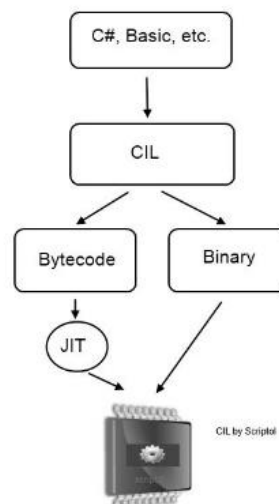


The .NET Framework Stack

**Obr. 1 Framework .NET [1]**

Common Intermediate Language (CIL), dříve označovaný jako Microsoft Intermediate Language (MSIL), je byte-kód a jazyk platformy .NET, do kterého je kompilován zdrojový kód napsaný v jazycích vysoké úrovně. CIL patří mezi objektově orientované jazyky založené na zásobníku a jsou prováděny virtuálním strojem.

Programovací jazyk C# je vysokoúrovňový objektově orientovaný programovací jazyk vyvinutý firmou Microsoft zároveň s platformou .NET framework. Jazyk je založený na jazycích C++ a Java. Programovací jazyk C# podporuje jazykovou interoperabilitu, což znamená, že má přístup ke kódu napsanému v jakémkoliv jazyce kompatibilní s .NET a může také dědit jejich třídy.



**Obr. 2 Převod do jazyka CIL [2]**

Hodnotové typy lze rozdělit na 3 části:

- Primitivní datový typ – jde o celočíselné a reálné primitivní datové typy (Char, Byte, Integer, Float, Double, Decimail)

- Struktury – jde o uživatelsky definované datové typy, které jsou podobné třídám, avšak neumí dědit ani být děděné a jsou alokované v zásobníku na sobě.
- Výčtové typy (Enum) – V podstatě se jedná o pojmenované konstanty názvem seznamu enumerator.

Referenční datové typy, na rozdíl od hodnotových datových typů, neuchovávají samotnou hodnotu, ale pouze odkaz na místo v paměti, kde je požadovaná hodnota uložena.

### 2.1.1 Jazyk C#

C# je multi-paradigmatický, imperativní, deklarativní, funkcionální, generický a objektově orientovaný jazyk. Jazyk C# byl vytvářený tak, aby byl jednoduchý a moderní jazyk pro všeobecné použití. Jazyk C# podporuje nativní komponentové programování. Obsahuje pouze jednoduchou dědičnost s možností násobené implementace. Jazyk C# zajišťuje typovou bezpečnost. Správa paměti je automatická. Podporuje zpracování chyb pomocí výjimek. C# podporuje struktury, přetížení operátorů, a předprocesory. Prostřednictvím C# se dá jednoduše volat funkce Windows API a přistupovat k COM komponentům.

C# nabízí pro framework .NET 4.5 a vyšší možnost používání asynchronního programování. Jeho využití je velice důležité pro zrychlení chodu aplikace zvláště při využívání komunikací s jiným zařízením a mohlo by dojít k určitému časovému zpoždění. Aplikace by poté musela čekat a byla blokována, dokud se daný proces nedokončí.

Použití asynchronních metod je obzvláště vhodné pro aplikace s uživatelským rozhraním, díky nimž nedojde k zablokování aplikace. Uživatelské rozhraní poté nepřestane reagovat a uživatel má neustálou možnost aplikaci ovládat.

```

using ObjectLibrary;
using System.Windows;

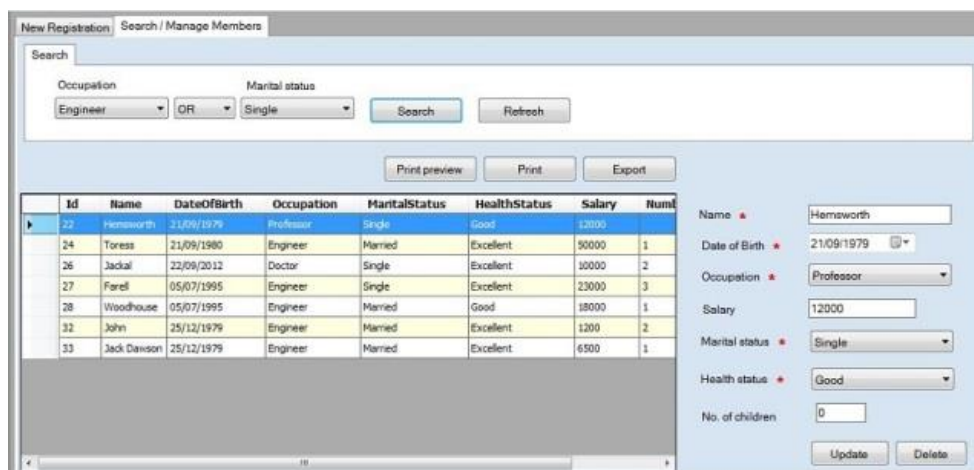
namespace DP_BeckhoffTwincat3
{
    /// <summary>
    /// Interaction logic for App.xaml
    /// </summary>
    public partial class App : Application
    {
        protected override void OnStartup(StartupEventArgs e)
        {
            base.OnStartup(e);
            Bootstrapper bootstrapper = new Bootstrapper();
            bootstrapper.Run();
        }
    }
}

```

Obr. 3 Ukázka kódu v jazyce C#

### 2.1.2 WinForms (WF)

WinForms (WF) je knihovna tříd pro tvorbu grafického rozhraní (GUI). Obsahuje různé komponenty, které může uživatel používat. Bohužel UI je velice jednoduché a omezené. Ovládací prvky jsou umístovány s absolutní pozicí, což může být pro aplikace s různým rozlišením dost nepraktické. Velikost ovládacích prvků nelze automaticky přizpůsobovat velikosti okna, aniž by nebylo zapotřebí napsat velké množství kódu. V současnosti se WF stále používá, protože je velmi jednoduchý a existuje velké množství aplikací, které jsou v něm napsané.



Obr. 4 Ukázka WinForms aplikace [3]

### 2.1.3 eXtensible Application Markup Language (XAML)

eXtensible Application Markup Language (XAML) je všestranný, deklarativní jazyk založený na XML, který se používá pro deklarování objektových grafů, které jsou instancovány při běhu. XAML používají vývojáři WPF, aby deklarovali uspořádání uživatelského rozhraní (UI) a prostředky používané v tomto uživatelském rozhraní. V programování WPF není nutné používat XAML. Cokoliv, co lze provést v XAML, lze provést v kódu. Avšak použití XAML dělá mnoho scénářů vývoje UI mnohem jednodušší a rychlejší.

```

<Grid Background="{Binding BackColor, Converter={StaticResource ColorToBrush}}">
  <Ivc:CartesianChart Grid.Row="1" x:Name="LiveChart" AnimationsSpeed="0:0:0.5" Hoverable="False" DataTooltip="{x:Null}" Series="{Binding SeriesCollection}" LegendLocation="Bottom">
    <Ivc:CartesianChart.ChartLegend>
      <Ivc:DefaultLegend BulletSize="20" Background="Red"/>
    </Ivc:CartesianChart.ChartLegend>
    <Ivc:CartesianChart.AxisX>
      <Ivc:Axis LabelFormatter="{Binding DateTimeFormatter}"
        MaxValue="{Binding AxisMax}"
        MinValue="{Binding AxisMin}"
        Unit="{Binding AxisUnit}"
        <Ivc:Axis.Separator>
          <Ivc:Separator Step="{Binding AxisStep}" />
        </Ivc:Axis.Separator>
      </Ivc:Axis>
    </Ivc:CartesianChart.AxisX>
  </Ivc:CartesianChart>

```

Obr. 5 Ukázka XAML

XAML dokument se skládá s elementů, které je možné do sebe různě vnořovat. Struktura je tedy stromová. Každý dokument obsahuje právě jeden kořenový element, do kterého jsou ostatní podelementy vnořeny. Vkládání elementů do sebe se říká nesting

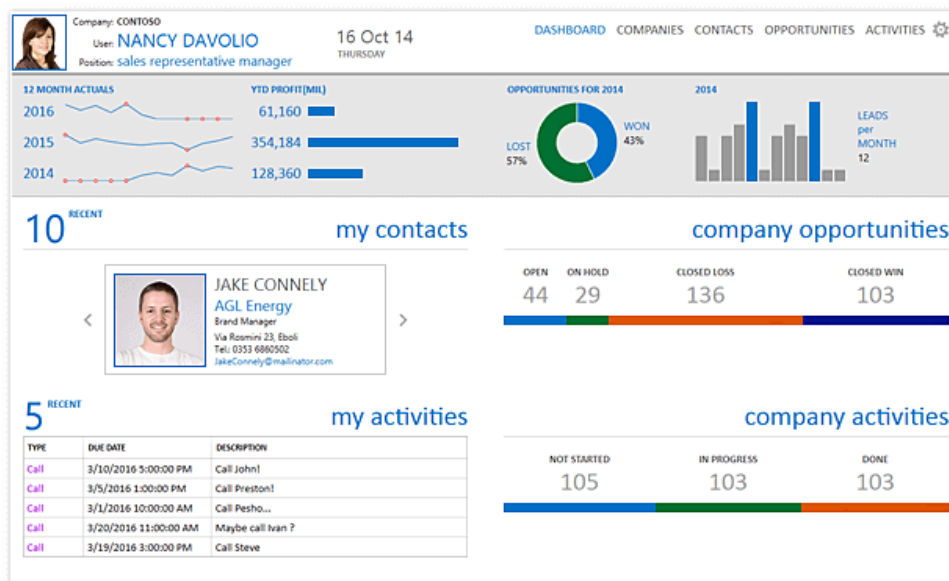
Elementy zapisujeme pomocí lomených závorek, které vždy začínají názvem element. Každý element musí být vždy ukončen. Element může být ukončen buď jako párový nebo nepárový. Nepárový element je ukončen lomítkem před uzavírací lomenou závorkou. Párový element poté obsahuje dva stejně pojmenované element na stejné úrovni, kde druhý stejnojmenný element má za lomenou závorkou lomítko. Do párových element se vkládá obsah element.

### 2.1.4 Windows Presentation Foundation (WPF)

Windows Presentation Foundation (WPF) je grafický subsystem společnosti Microsoft pro vykreslování uživatelského rozhraní. WPF, dříve známý jako „Avalon“, byl vydán

jako součást .NET Framework 3.0 v roce 2006. pro tvorbu aplikací, který je součástí .NET frameworku od verze 3.0 firmy Microsoft. WPF používá DirectX. WPF poskytuje konzistentní programovací model pro vytváření aplikací a odděluje uživatelské rozhraní (UI) od programové logiky.

WPF používá jazyk XAML pro definování a propojení různých prvků rozhraní. Aplikace WPF lze nasadit jako samostatné desktopové programy nebo hostované jako vložený objekt na webových stránkách. WPF runtime knihovny jsou součástí všech verzí systému Microsoft Windows od Windows Vista a Windows Server 2008.



Obr. 6 Ukázka WPF aplikace [4]

Hlavní myšlenkou WPF je skládání jednotlivých elementárních grafických prvků dohromady. Je možné si to představit na objektu Button, které je ve WPF možné rozložit na elementy Button a TextBlock. Tato vlastnost nabízí nové možnosti tvorby uživatelského rozhraní. WPF je založené na vektorové grafice, díky které jednoduše reaguje na změnu rozlišení obrazovky a DPI.

WPF vzniklo na základě tlaku na graficky bohatší aplikace, kde už starší WinForms (WF) narážel na své limity. Jedním z hlavních důvodů je rozšíření podpory o mobilní telefony a tablety, kde má WF slabou podporu DPI. WF ukládá absolutní pozici každého elementu, což je velice limitující. Proto se WPF inspirovalo jazykem HTML, který využívá kaskádové style CSS a zavedl definici podle XAML. Poslední nevýhodou WF oproti WPF je tou, že využívá starší grafické rozhraní Windows (GDI), které je pomalejší a má značné omezení. Zatímco WPF využívá Direct3D pro akcelerovanou grafiku a tím jsou WPF aplikace rychlejší a méně zatěžují CPU.

WPF používá data binding pro provázání dat v C# a objektů v XAML. Díky tomu jsou zobrazovaná data v objektech okamžitě aktualizována při jejich změně.

## 2.2 Model-View-ViewModel (MVVM)

Model-View-ViewModel (MVVM) je návrhový vzor pro WPF aplikace. Používá se pro oddělení logiky aplikace od uživatelského rozhraní. Jeho použitím dochází ke snížení

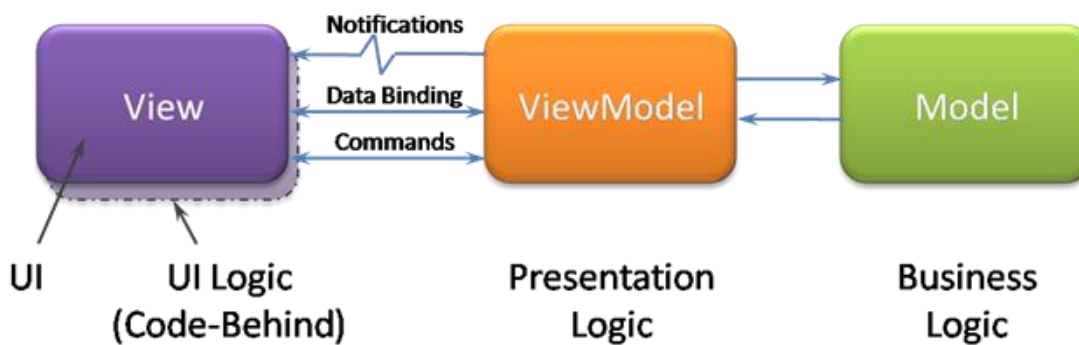
množství kódu, větší přehlednosti a případné změny nejsou noční můrou. MVVM odděluje data, stav aplikace a uživatelské rozhraní. WPF bylo pro jeho použití navrženo a využívá se v něm binding a příkaz jako náhrada za uživatelské rozhraní řízené událostmi.

Hlavní cílem MVVM jen vytvořit třídu, která drží stav aplikace. Tato třída se nazývá ViewModel a uživatelské rozhraní s ní sdílí údaje pomocí databindingu. MVVM se skládá z těchto vrstev: Model, View a ViewModel. [5]

Model popisuje data, se kterými aplikace pracuje. Například při použití Code-First Entity Frameworku, který mapuje tabulky databáze, je modelem. Různé reference na webové služby, třídy, které Visual Studio vygeneruje, tak jsou taktéž modely. Model však nesmí o stavu ovládacího prvku nic vědět.

View představuje uživatelské prostředí v jazyce XAML, přičemž nezáleží, jestli se jedná o ovládací prvek, stránku nebo aplikaci. View bývá taktéž označován jako UI nebo prezenční forma.

ViewModel je nejdůležitější třídou, protože poskytuje všechna svá data pro View. ViewModel poskytuje svá data v takových strukturách, které vyvolají události při jejich změně. Díky tomu může View zobrazit změnu dat okamžitě.



Obr. 7 Model-View-ViewModel [6]

## 2.3 Prism

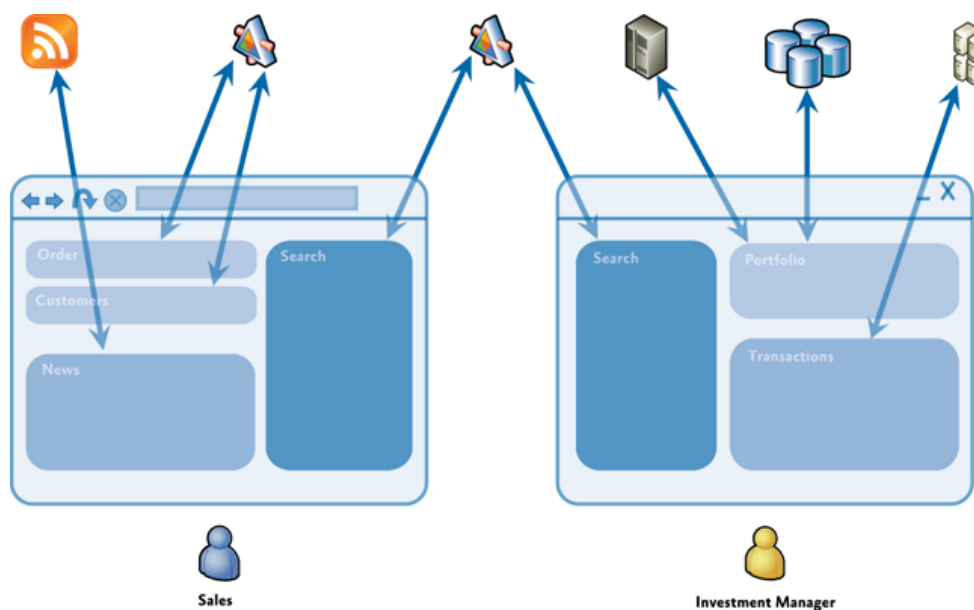
Kompozitní přístup řeší Prism pomocí rozdělení aplikace na řadu diskretních, volně spojených, částečně nezávislých komponent, které lze snadno integrovat do „Shell“ aplikace a vytvořit soudržné řešení. Tímto způsobem vytvořené aplikace jsou známé jako kompozitní aplikace.

Kompozitní aplikace umožňují modulům individuální vývoj, testování a nasazení modulů různými způsoby. Prism umožňuje modulům snadnější úpravu a rozšíření o nové funkce, čímž umožňuje snadnější rozšíření a udržování aplikace.

Prism aplikace nabízejí společnou skořápku složenou z UI komponent, které se skládají z různých modulů reagující volným způsobem. Podporuje opakované použití a čisté oddělení mezi horizontálními možnostmi aplikace, jako je přihlašování a autentifikace, a vertikálními funkcemi, jako jsou obchodní funkce, které jsou specifické pro aplikaci. Díky tomu je snadnější správa závislostí a interakcí mezi komponentami aplikace.

Pomáhá k oddělení rolí při práci na projektu, protože umožňuje různých jedinců nebo skupinám soustředit se na konkrétní úkol nebo funkci podle jejich odbornosti a zaměření. Hlavně poskytuje čistější oddělení mezi uživatelským rozhraním a obchodní logikou aplikace, díky tomu se může návrhář soustředit na vytvoření bohatšího UI.

Kompozitní aplikace jsou vhodné pro řadu klientských aplikací pro vytváření bohatého zážitku koncového uživatele na rozdílných systémech back-end. Kompozitní aplikace je také užitečná, pokud existují nezávisle se vyvíjející komponenty v uživatelském rozhraní, které se navzájem silně integrují a jsou často udržovány samostatnými týmy.



Obr. 8 Prism – kompozitní aplikace [7]

## 2.4 SQLite

SQLite je vestavěný relační databázový engine SQL, který začal svou cestu v roce 2000. Na rozdíl od většiny jiných databází SQL, SQLite nemá samostatný serverový proces. SQLite čte a zapisuje přímo do běžných diskových souborů. Kompletní databáze SQL s více tabulkami, indexy, spouště a pohledy je obsažena v jednom disku. Formát souboru databáze je multi-platformový - můžete libovolně kopírovat databázi mezi 32bitovými a 64bitovými systémy nebo mezi architekturami typu big-endian a little-endian. Díky těmto funkcím je SQLite populární volbou formátu aplikačního souboru.

SQLite je velmi pečlivě testován před každým vydáním a má velmi dobrou pověst. Většina zdrojového kódu je věnována výhradně testování a ověřování. Samozřejmě i přes všechny nejrůznější testování a ověřování obsahuje SQLite stále chyby. Na rozdíl však od ostatních projektů je SQLite otevřený a upřímný o všech chybách, které jsou sepsány v seznamu chyb.

```

using (sqliteConnection = new SQLiteConnection(connectionString))
{
    sqliteConnection.Open();

    string sql = "create table Users (id INTEGER PRIMARY KEY AUTOINCREMENT, Name varchar(50), Password varchar(100), UserLevel int)";
    SQLiteCommand command = new SQLiteCommand(sql, sqliteConnection);
    command.ExecuteNonQuery();

    sql = "insert into Users (Name, Password, UserLevel) values " +
        "('Operator', '111', 1)," +
        "('Technical', '222', 2)," +
        "('Technologist', '333', 3)," +
        "('Admin', '999', 4)";
    command = new SQLiteCommand(sql, sqliteConnection);
    command.ExecuteNonQuery();
}

```

Obr. 9 Ukázka použití SQLite v jazyce C#

## 2.5 eXtensible Markup Language (XML)

eXtensible Markup Language (XML) je rozšiřitelný značkovací jazyk, který byl vyvinut standardizovaným konsorciem W3C. Ve skutečnosti je XML metajazyk, v jehož rámci jde vytvářet vlastní jazyky. Jazyk umožňuje jednoduché vytváření konkrétních značkovacích jazyků pro různé účely a data. Nejčastěji se používá pro serializování dat.

XML je zajímavý tím, že neobsahuje žádné konkrétní značky. Díky tomu je možné si definovat strukturu dle vlastních potřeb. XML neobsahuje formu zobrazení, a proto s ním můžou dvě aplikace pracovat úplně odlišně.

V dnešní době se XML využívá pro snadnou výměnu dat a komunikaci nezávislou na konkrétní aplikaci či platformě. XML nabízí oproti jiným formátům následující výhody: nezávislost, poměrně malou velikost, standardizaci a jednoduchý převod na jiné formáty.

```

<?xml version="1.0" encoding="utf-8"?>
<Root>
  <Workspace>
    <Type>ObjectLibrary.WorkSpace, ObjectLibrary, Version=1.0.0.0, Culture=neutral, PublicKeyToken=null</Type>
    <ScreenName>Technologie</ScreenName>
    <Width>1705</Width>
    <Height>1015</Height>
    <BackColor>#FFC0C0</BackColor>
    <Border_Color>#FF0000</Border_Color>
  </Workspace>
  <Objects>
    <Object>
      <Type>ObjectLibrary.Tank, ObjectLibrary, Version=1.0.0.0, Culture=neutral, PublicKeyToken=null</Type>
      <ID>8bf4d400-9711-4b43-886c-718bb3fb17b5</ID>
      <IsGroup>>false</IsGroup>
      <ParentID>00000000-0000-0000-0000-000000000000</ParentID>
      <Properties>
        <Width>172</Width>
        <Height>346.9999999999998</Height>
        <x>191</x>
        <y>173.15999999999998</y>
        <z>24</z>
        <Rotation>0</Rotation>
        <BackColor>#FFFFFF</BackColor>
        <Device />
        <ReadVariable></ReadVariable>
        <RunColors />
      </Properties>
    </Object>

```

Obr. 10 Ukázka XML

## 3. SOFTWAROVÉ PROSTŘEDKY

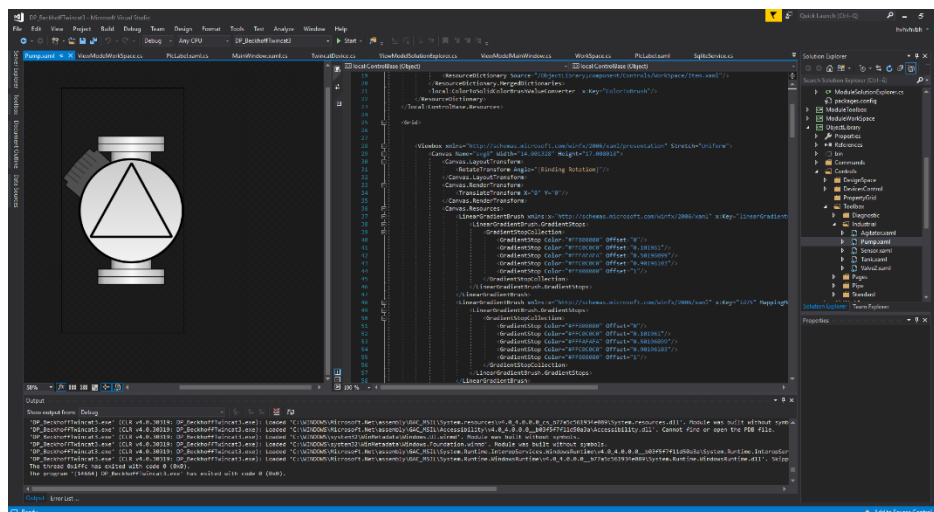
Aplikaci HMI jsem vytvořil pomocí software Visual Studio 2017 Community od firmy Microsoft. V tomto programu jsem vytvořil celé vývojové prostředí a runtime HMI. Pro simulaci ADS zařízení byl použit software TwinCAT 3, který je taktéž integrován do Microsoft Visual Studia jako další jazyk. TwinCAT 3 je od společnosti Beckhoff, který je zdarma ke stažení na stránkách tvůrce a umožňuje testování zařízení na sedm dní zdarma. Tuto dobu lze libovolnou dobu prodlužovat.

### 3.1 Microsoft Visual Studio

Microsoft Visual Studio je integrované vývojové prostředí od společnosti Microsoft. Používá se k vývoji počítačových programů, webových stránek, webových aplikací, webových služeb a mobilních aplikací. Visual Studio podporuje 36 různých programovacích jazyků a umožňuje editoru a debuggeru kódů podporovat téměř libovolný programovací jazyk za předpokladu, že existuje specifická jazyková služba. Vestavěné jazyky obsahují jazyky C, C++, Visual Basic .NET, C #, F #, JavaScript, TypeScript, XML, XSLT, HTML a CSS.

Visual studio obsahuje editor kódu podporující IntelliSense a refraktorování. Integrovaný debugger pracuje jak na úrovni kódu, tak na úrovni stroje. Další vestavěné nástroje zahrnují designer formulářů pro tvorbu aplikací GUI, designer webu, tříd a databázových schémat. Je možné přidávat rozšíření, což vylepšuje funkčnost téměř na každé úrovni.

Visual studio podporuje jazyky prostřednictvím jazykových služeb, což umožňuje přidat jakýkoliv programovací jazyk. Podpora programovacích jazyků je přidána balíčkem zvaným Language Service.



Obr. 11 Microsoft Visual Studio

Visual studio umožňuje vývojářům psát rozšíření. Rozšíření je možné psát ve formě makra, balíčků a rozšíření. Makra reprezentují opakující se úkoly a akce, které vývojáři mohou nahrávat, ukládat, přehrávat a distribuovat. Makra, nicméně, mohou být použita k implementaci nových příkazů nebo vytváření oken nástrojů. Rozšíření (add-ins)

poskytují přístup do objektového modelu Visual Studio a mohou být propojeny s IDE nástroji. Rozšíření mohou být použita pro implementaci nové funkčnosti a přidání nových oken nástrojů. Balíčky jsou vytvořeny pomocí SDK, který poskytuje nejvyšší úroveň rozšiřitelnosti.

## 3.2 Beckhoff

Řídicí systémy Beckhoff se zásadně odlišují od ostatních řídicích systémů. Jedná se o řídicí systémy fungující na platformě průmyslového PC. Kromě řídicího run-time systému na těchto průmyslových PC fungují i operační systémy Windows CE, které vytváří komplexní a univerzální systém. Další hlavní vlastností je přesně definované nastavení délky cyklu řídicího systému. Díky těmto vlastnostem jsou řídicí systémy Beckhoff velmi vhodné pro náročné aplikace v oblasti servopohonů a přesné měření.



Obr. 12 Beckhoff sortiment [8]

Řídicí systém Beckhoff běží na průmyslovém PC, které softwarově vytváří PLC. Firma nabízí mnoho variant od výkonově nejmenších embedded PC, přes kompaktní PC, panelová PC až po výkonné průmyslové servery s více jádrovými procesory Intel Xeon. Nabídka obsahuje také velmi široký sortiment I/O modulů. Beckhoff má vlastní komunikační sběrnici EtherCAT pro real-time aplikace. Je možné řídicí systém rozšířit pomocí komunikačních modulů o další průmyslové sběrnice. Beckhoff také poskytuje hardware pro motion aplikace: servomotory, pohony, krokové a lineární motory.

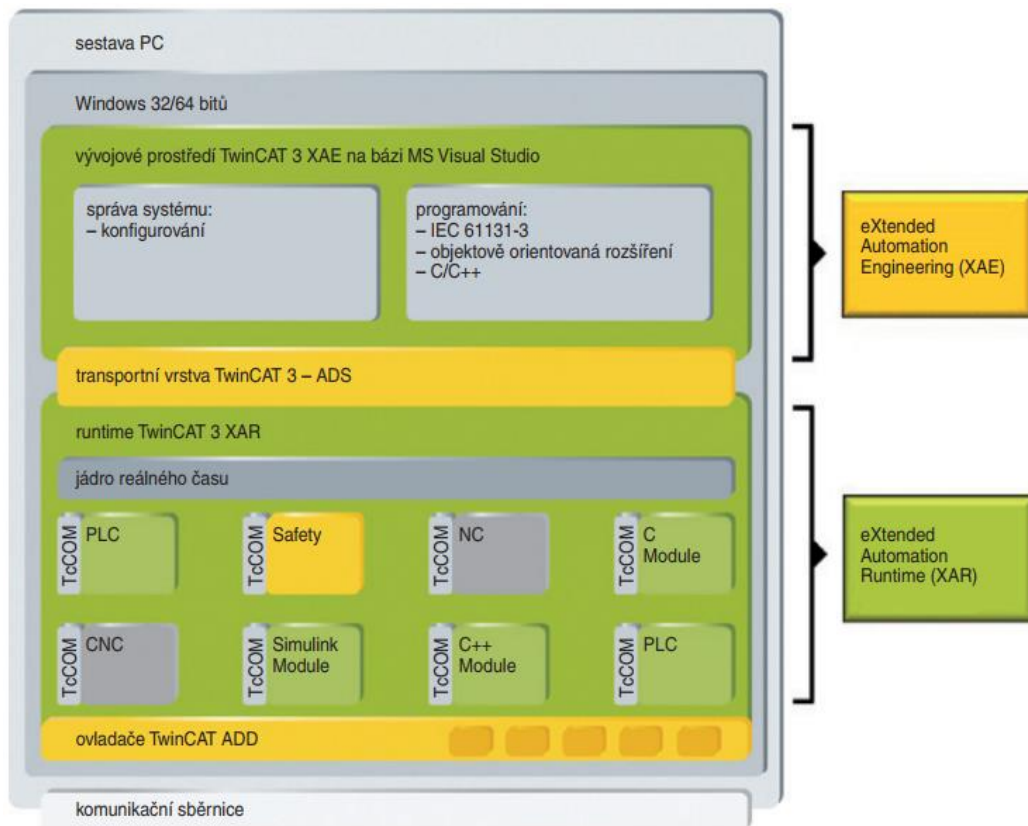
Velmi silnou stránkou Beckhoffu je software. Vývojové prostředí i run-time běžící na průmyslových PC se nazývá TwinCAT. V současnosti jsou podporovány dvě verze – starší TwinCAT 2 a aktuální TwinCAT 3.

Ve vývojovém prostředí TwinCAT 3 je možné programovat všemi jazyky podle normy pro programování PLC IEC 61131-3. Uživatel má navíc možnost vytvářet realtimové PLC aplikace v C++ a UML. Komunikace je zajištěna pomocí komunikační vrstvy ADS (Automation Device Specification).

### 3.3 TwinCAT

Firma Beckhoff představila v roce 1986 software TwinCAT (The Windows Control Automation Technology) jako nástroj pro tvorbu automatizačních systémů na základě PC. Jedná se tedy o vývojový nástroj, který umožňuje změnit z téměř jakékoliv výpočetní techniky na bázi PC na řídicí systém v reálném čase. Obsahuje 4 prostředí pro běh programů (runtime) pro PLC, NC PTP (NC Point-to-Point, řízení pohybu os z bodu do bodu), NCi (NC interpolation, řízení pohybu os s použitím interpolace), plnohodnotné řízení CNC nebo řízení robotů (kinematická transformace). [10]

Struktura TwinCAT 3 je tvořena těmito základními částmi: vývojovým prostředím (eXtended Automation Engineering – XAE) a runtime (eXtended Automation Runtime – XAR), které jsou propojeny transportní vrstvou ADS.



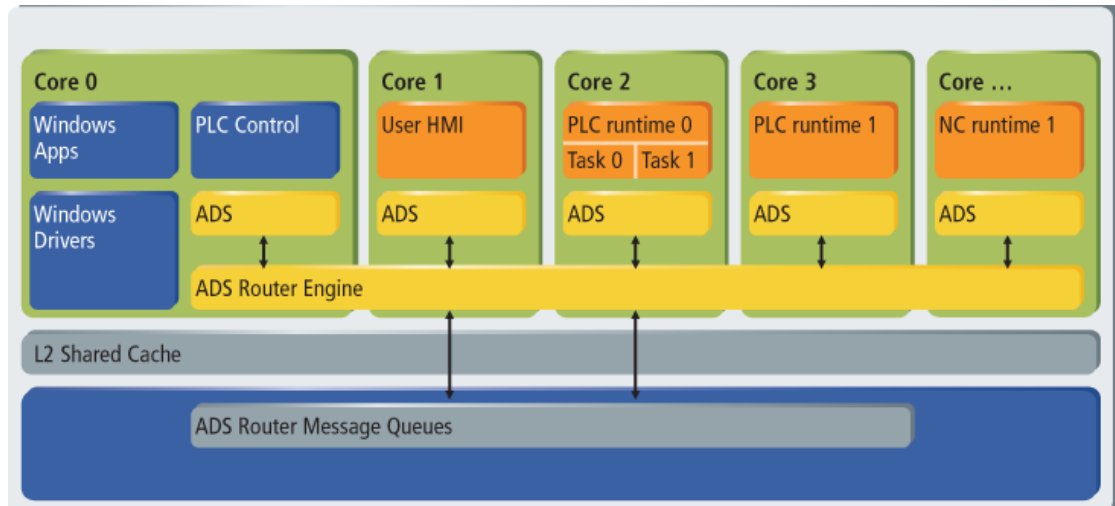
Obr. 13 Základní struktura software TwinCAT 3 [9]

Vývojové prostředí TwinCAT 3 má dvě základní části: správcovský modul pro konfigurování hardwaru a samotné programovací prostředí. [10]

Správcovský modul slouží k nastavení parametrů. Je možné v něm nastavit vlastnosti PLC, parametry I/O modulů, vlastnosti programovaných úloh v C/C++, úloh řízení polohy a pohybu (Motion Control), CNC úloh, bezpečnostních (safety) úloh a průmyslových sběrnic.

Programovací prostředí podporuje objektové programování PLC podle normy IEC 61131-3. Objektové programování přináší efektivitu a flexibilitu při tvorbě aplikací. PLC kód lze zapsat podle normy jedním z těchto jazyků: strukturový text (ST), seznam instrukcí (IL), funkčních bloků (FBD), jazyk kontaktních schémat (LD) a metoda sekvenčních funkčních grafů (SFC).

Současný vývoj v oblasti výpočetní techniky, který nabízí CPU s více a více jádry, umožňuje distribuci úkolů napříč různými jádry. Vývojové prostředí TwinCAT 3 se tímto řídí a může být použit k distribuci funkčních jednotek, jako je HMI, PLC runtime nebo MC do vyhrazených jader. Pro každé jádro používané vývojovým prostředím lze nastavit maximální zatížení i základní čas a tím i možné doby cyklu odděleně.



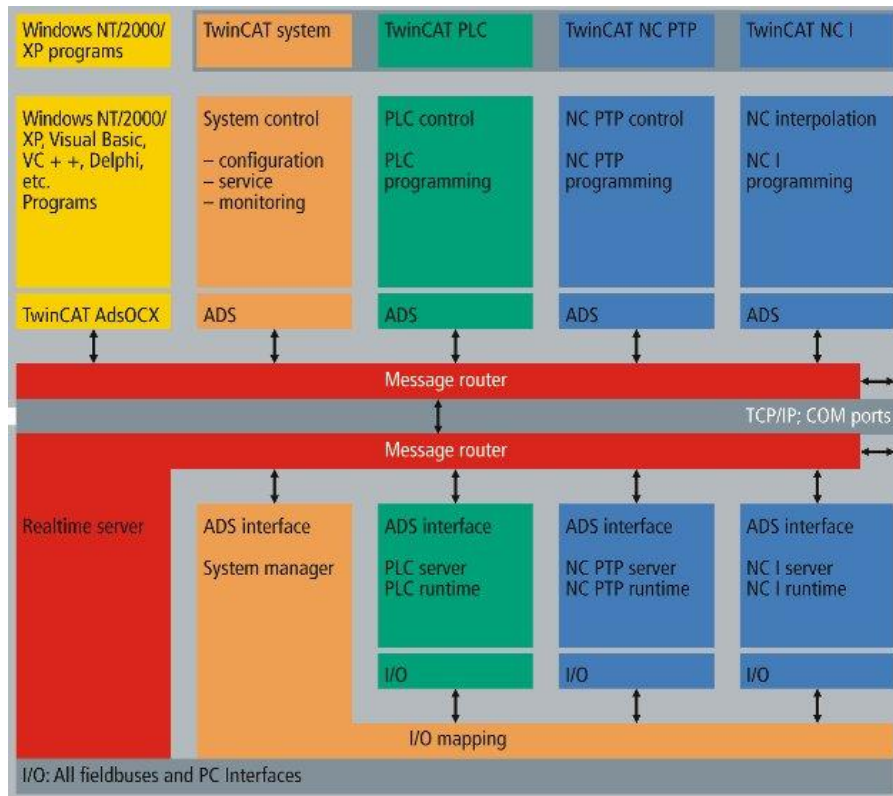
**Obr. 14 Způsob použití mnohójádrových aplikací [11]**

TwinCAT 3 je možné si zdarma stáhnout na stránkách firmy Beckhoff, avšak rozšiřující knihovny jsou licencovány za peníze. Zákazník však má možnost si rozšíření vyzkoušet na týdenní zkušební dobu, kterou je možné opětovně prodlužovat. Díky tomu je možné si vyzkoušet dané rozšíření a poté si jej až zakoupit. Licence se poté zakupují na dané IPC nebo PC.

Protokol ADS je transportní vrstva v systému Beckhoff TwinCAT. Byl vyvinut pro výměnu dat mezi různými softwarovými moduly (například komunikací mezi NC a moduly). Protokol nabízí svobodu používání dalších nástrojů pro komunikaci s jiným zařízením nebo počítačem. To znamená, že v síťovém systému jsou všechny data přístupná z libovolného požadovaného bodu.

Protokol ADS běží na vrcholu protokolů TCP / IP nebo UDP / IP. Umožňuje uživateli v systému Beckhoff používat téměř jakoukoliv spojovací trasu pro komunikaci se všemi připojenými zařízeními a pro jejich parametrizaci. Mimo systému Beckhoff je k dispozici řada metod pro výměnu dat s dalšími softwarovými nástroji. Po ADS lze díky dostupnému API komunikovat s aplikacemi vytvořenými např. v .NET, JAVA, C/C++ či v dalších jazycích.

ADS využívá k identifikaci zařízení `AdsAmsNetID` a `AdsPortNr`. Každé ADS zařízení jsou jednoznačně identifikována pevným portem. `AdsAmsNetID` je rozšíření adresy TCP/IP a identifikuje směrovač zpráv TwinCAT.



Obr. 15 ADS komunikace [12]

### 3.3.1 Strukturový text

Textový jazyk ST (Structured Text) je výkonný vyšší programovací jazyk, který má kořeny v jazycích Pascal a C. Syntaxe jazyka je dána povolenými výrazy a příkazy. Je definováno deset typů příkazů (přiřazení, vyvolání funkce, návrat, výběr apod.). Příkazy jsou odděleny středníkem a může jich být více na jednom řádku. Jazyk ST je vhodným nástrojem pro definování komplexních funkčních bloků, které pak mohou být použity v libovolném programovacím jazyku.

```

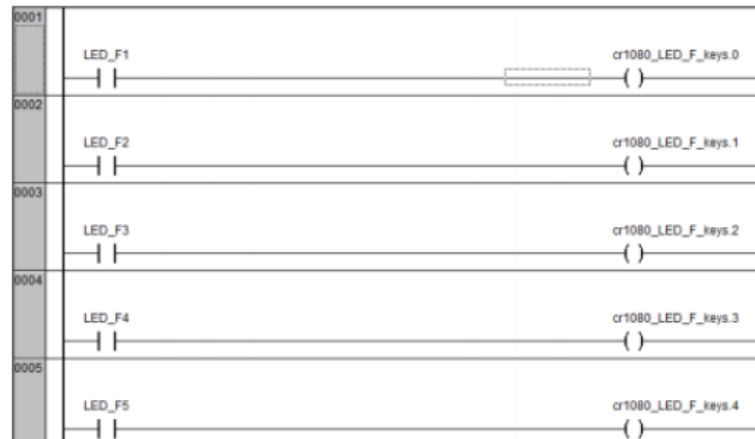
1 FUNCTION_BLOCK Pump
2 VAR_INPUT
3   bEnable:BOOL;
4   bGoAut:BOOL;
5   bGoDal: BOOL;
6   Mode: BOOL;
7   diRm:BOOL;
8   bQuit:BOOL;
9   diFa:BOOL;
10 END_VAR
11 VAR_OUTPUT
12   doRm:BOOL;
13   bEvent: BOOL;
14 END_VAR
15 VAR
16   END_VAR
17
18 doRm := bEnable AND NOT diFa AND ((bGoDal AND NOT Mode) OR bGoAut AND Mode);
19
20 IF diFa THEN
21   bEvent := TRUE;
22 END_IF
23 IF bQuit THEN
24   bEvent := FALSE;
25 END_IF

```

Obr. 16 Strukturový text

### 3.3.2 Jazyk kontaktních schémat

Grafický jazyk LD je založen na grafické reprezentaci reléové logiky. Program je zapsán sítí propojených grafických prvků. Síť v jazyku LD je zleva i zprava ohraničena svislými čarami, které se nazývají levá a pravá napájecí sběrnice. Mezi nimi je tzv. příčka, která může být rozvětvena. Každý úsek příčky, vodorovný nebo svislý, může být ve stavu „zapnuto“ nebo „vypnuto“. Do příček mohou být včleněny kontakty (spínací, rozpínací apod.), cívky (cívka, negovaná cívka apod.) a dále funkce a funkční bloky.



Obr. 17 Jazyk kontaktních schémat

### 3.3.3 Seznam instrukcí

IL (Instruction List) patří do skupiny textových jazyků. Bývá označován také jako jazyk pokynů (povelů), seznam instrukcí poněkud připomíná assembler. Programová organizační jednotka je složena ze sekvence instrukcí, z nichž každá 7 začíná na novém řádku, může obsahovat také komentář. Pomocí modifikátorů se vyjadřují negace, podmíněnost a nepodmíněnost instrukce skoků, volání a návratů a priorit.

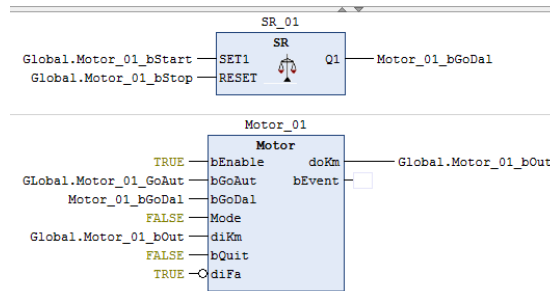
```
FUB (FB-IL)
0001 FUNCTION_BLOCK FUB
0002 VAR_INPUT
0003     PAR1:INT;
0004     PAR2:INT;
0005 END_VAR
0006 VAR_OUTPUT
0007     MULERG:INT;
0008     VERGL:BOOL;

0001 LD     PAR1
0002 MUL    PAR2
0003 ST     MULERG
0004
0005 LD     PAR1
0006 EQ     PAR2
0007 ST     VERGL
```

Obr. 18 Seznam instrukcí

### 3.3.4 Funkční bloky

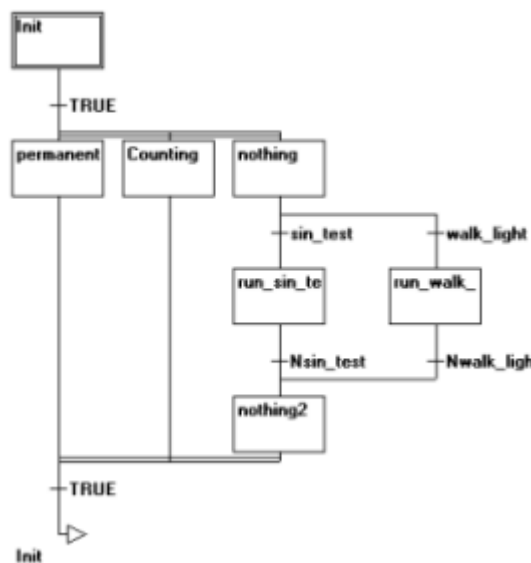
Soubor vzájemně provázaných grafických bloků podobně jako v elektronických obvodových diagramech. Používají se zde standardní funkční bloky pro vyjádření logických funkcí a také čítače, časovače, komunikační bloky a podle potřeby i vlastní bloky.



Obr. 19 Funkční bloky

### 3.3.5 Metoda sekvenčních funkčních grafů

SFC (Sequential Function Chart) popisuje sekvenční chování řídicího programu. Je odvozen ze symboliky Petriho sítě. SFC umožňuje rozložit úlohu řízení na zvládnutelné části a zachovat přitom přehled o chování celku. Sekvenční funkční diagram se skládá z kroků a přechodů. Každý krok reprezentuje stav řízeného systému a má k sobě přiřazen blok akcí. Přechod je spojen s podmínkami, které musí být splněny, aby mohl být deaktivován krok, který přechodu předchází, a naopak aktivován krok, který následuje. Každý prvek, tzn. přechod i blok akcí, může být naprogramován v libovolném jazyku definovaném v normě, včetně vlastního SFC. Jazyk umožňuje i větvení programu se spojením alternativních větví a paralelní souběh více větví s jejich následnou synchronizací.



Obr. 20 Metoda sekvenčních funkčních grafů

## 4. TVORBA VÝVOJOVÉHO PROSTŘEDÍ A RUNTIME

Vývojové prostředí a runtime pro HMI jsem vytvořil v programovacím jazyce C# pomocí frameworku WPF. Aplikace jsem programoval dle návrhového vzoru pomocí vývojového prostředí Visual Studio 2017. Pro lepší testování a vyvíjení aplikace jsem použil kompozitní přístup pomocí Prism. Díky tomu jsem mohl jednotlivé moduly vyvíjet nezávisle na sobě, testovat a poté spojit do výsledné aplikace.

Jako další velice důležitou knihovnou je knihovna od společnosti Beckhoff, díky které jsem mohl využívat funkce a metody. Jednalo se převážně o připojení k ADS zařízení, diagnostice jeho stavu, čtení hodnot ze zařízení, zápis hodnot do zařízení. Je důležité také říci, že tato knihovna vyžaduje nainstalovaný TwinCAT na stanici, kde je používána, protože jej využívá ke komunikaci se zařízením. Dále je dobré zmínit, že je důležité vytvořit takzvanou „routu“ mezi TwinCATem v zařízení a na stanici.

SQLite jsem použil pro uchovávání informací o uživateli. Ve vývojovém prostředí jsou uživatelé vytvořeny do tabulky v SQLite a poté čteny při přihlašování v runtime. Seznamy alarmů jsou udržovány pomocí tohoto řešení. Do databáze jsou také archivovány oba typy žurnálu.

Pro ukládání obrazovek a jejich objektů, zařízení a designu aplikace jsem se rozhodl použít formát XML. Hlavním důvodem použití tohoto formátu je jeho jednoduché a rychlé použití. Při vytváření aplikace je navíc přínosný z důvodu rychlé kontroly správnosti ukládání přímo v souboru. Výhodou je také i to, že přechod z XML například na databázový systém je velice jednoduché, protože stačí jen vytvořit strukturu v databázi a ukládat do ní příslušná data.

### 4.1 Postup vytváření projektu

Nejprve jsem nainstaloval Microsoft Visual Studio 2017, pomocí kterého jsem vytvořil řešení s názvem „DP\_BeckhoffTwincat3“. V tomto řešení jsem vybral, že chci pro projekt použít grafický subsystém WPF, protože tento projekt bude sloužit jako vývojové prostředí pro HMI. Jelikož jsem se rozhodl použít Prism, tak bylo vytvořený projekt potřeba hned upravit pro jeho správnou funkci.

Nejprve jsem si stáhl pomocí „Manage Nuget Packages“ potřebné knihovny pro Prism, bez kterých by jej nebylo možné používat. Poté jsem vytvořil třídu s názvem „Bootstrap“, která dědí patřičné vlastnosti ze třídy UnityBootstrap, která je obsažena v knihovně Prism. V této třídě jsem nainicializoval hlavní okno, které se má spustit po zapnutí aplikace, použité moduly a další použité typy. Poté jsem upravil automaticky vygenerovaný soubor App.xaml, aby nespouštěl přímo hlavní okno aplikace, ale aby spustil vytvořil třídu Bootstrap, která se o vše ostatní postará.

```

namespace DP_BeckhoffTwincat3
{
    public class Bootstrapper : UnityBootstrapper
    {
        protected override DependencyObject CreateShell()
        {
            return Container.Resolve<MainWindow>();
        }

        protected override void InitializeShell()
        {
            Application.Current.MainWindow.Show();
        }

        protected override void ConfigureModuleCatalog()
        {
            ModuleCatalog catalog = (ModuleCatalog)ModuleCatalog;
            catalog.AddModule(typeof(ModuleToolbox.ModuleToolbox));
            catalog.AddModule(typeof(ModuleWorkSpace.ModuleWorkSpace));
            catalog.AddModule(typeof(ModuleSolutionExplorer.ModuleSolutionExplorer));
            catalog.AddModule(typeof(ModuleProperty.ModuleProperty));
        }

        protected override void ConfigureContainer()
        {
            base.ConfigureContainer();

            RegisterTypeIfMissing(typeof(ITCService), typeof(TCService), true);
            RegisterTypeIfMissing(typeof(ISqliteService), typeof(SqliteService), true);

            Container.RegisterType<IApplicationCommands, ApplicationCommands>(new ContainerControlledLifetimeManager());
        }
    }
}

```

**Obr. 21** Vytvořený Bootstrap pro vývojové prostředí

Hlavní projekt využívá jako všechny projekty návrhový vzor MVVM. Proto jsem zde vytvořil ještě ViewModel, který předává aplikaci všechny příkazy, které jsou sdílené mezi všemy moduly.

Dále bylo zapotřebí vytvořit nové projekty, které budou fungovat jako moduly. Všechny tyto aplikace jsem vytvořil jako knihovnu tříd. Do všech této projektů byla opět přidána knihovna Prism.

Aby Prism dokázal daný modul identifikovat a přidat do kontejneru, bylo potřeba v každém projektu vytvořit třídu, která dědí vlastnosti ze třídy IModule knihovny Prism. V každém modulu je poté definováno View, které reprezentuje daný modul. Vytvořený design a funkce popsaných modulů bude popsána v následující kapitole.

```

namespace ModuleProperty
{
    public class ModuleProperty : IModule
    {
        IRegionManager regionManager;
        IUnityContainer container;

        public ModuleProperty(IRegionManager regionManager, IUnityContainer container)
        {
            this.regionManager = regionManager;
            this.container = container;
        }

        public void Initialize()
        {
            regionManager.RegisterViewWithRegion("ViewProperty", typeof(ViewProperty));
        }
    }
}

```

**Obr. 22** Ukázka tvoby jednoho z Modulů

Další projekt, který jsem vytvořil byla knihovna tříd, která obsahuje všechny objekty, modely, obrázky, modely agregátoru událostí, okna a příkazy aplikace. Tedy

všechny potřebné soubory nezávislé na modulu jsou umístěny zde, aby s nimi v případě potřeby mohl pracovat libovolný modul.

Předposlední dva projekty jsou také knihovny tříd. Tyto projekty mají za cíl zajišťovat data pro HMI. Dále budou popisovány jako sdílená služba ADS a sdílená služba SQLite.

Jak už název napovídá první z nich zajišťuje komunikaci dat z ADS zařízení a HMI komunikuje s touto službou pomocí vytvořeného rozhraní, které nabízí možnost přidání a odebrání ADS zařízení, zapsání do konkrétního ADS zařízení a zachytávání událostí o stavu ADS zařízení a změně jeho dat. Způsob připojení a vznik událostí je detailněji popsán v následující kapitole.

```
namespace TwincatService
{
    public interface ITCService
    {
        event EventHandler<DataChangeEvent> DataChanged;
        event EventHandler<DeviceChangeEvent> DeviceChanged;

        void Add(Device device);
        void Remove(Device device);

        void WriteData(Device device, string VariableName, object Value);
    }
}
```

**Obr. 23 Rozhraní sdílené služby ADS**

Druhá sdílená služba zajišťuje komunikaci mezi HMI a databázovým souborem vytvořeným pomocí SQLite. Sdílená služba nabízí opět rozhraní, které má přesně definované funkce, které může HMI využívat. Jedná se tedy o vytvoření databázového souboru při vytváření projektu, jeho načtení při otevření projektu. Dalšími funkcemi je přihlašování, přidávání záznamů do uživatelského a procesního žurnálu. Změna uživatele je indikována událostí „UserChanged“. Všechny funkce a vlastnosti jsou podrobněji popsány v následující kapitole.

```
namespace DbService
{
    public interface ISqliteService
    {
        event EventHandler<UserChangeEvent> UserChanged;

        void CreateProject(string ProjectPath);
        void OpenProject(string ProjectPath);
        bool Login(string UserName, string Password);

        ObservableCollection<string> GetAlarmTable(string ProjectPath);

        void AddUserJournal(string UserName, string Description, string New);
        void AddProcessJournal(string UserName, string Description);
    }
}
```

**Obr. 24 Rozhraní sdílené služby SQLite**

Předposledním vytvořeným projektem je runtime, který je vytvořený, stejně jako hlavní projekt, pomocí grafického subsystému WPF. Na rozdíl od hlavního projektu však neobsahuje žádné moduly a jeho třída bootstrap je mnohem jednodušší. Cílem toho projektu bude spouštět vytvořenou HMI ve vývojovém prostředí. Díky této vlastnosti není potřeba použití modulů, které slouží převážně k editaci projektu HMI a budou blíže

popsány v následující kapitole. Oba projekty však používají obě sdílené služby a proto je možné je nalézt v obou zdrojových kódech této struktury.

```
namespace DP_BeckhoffTwincat3_Runner
{
    public class Bootstrapper : UnityBootstrapper
    {
        protected override DependencyObject CreateShell()
        {
            return Container.Resolve<MainWindow>();
        }

        protected override void InitializeShell()
        {
            Application.Current.MainWindow.Show();
        }

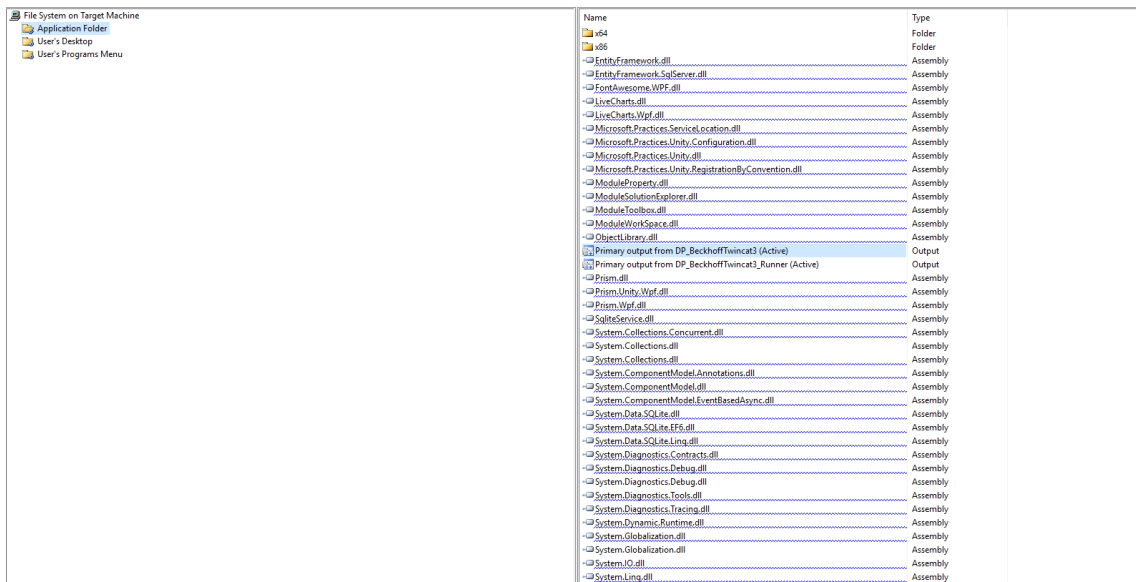
        protected override void ConfigureModuleCatalog()
        {
        }

        protected override void ConfigureContainer()
        {
            base.ConfigureContainer();

            RegisterTypeIfMissing(typeof(ITCService), typeof(TCService), true);
            RegisterTypeIfMissing(typeof(ISqliteService), typeof(SqliteService), true);
        }
    }
}
```

Obr. 25 Vytvořený bootstrap pro runtime

Posledním vytvořeným projektem je instalační projekt obou aplikací. Tento projekt se stará o sestavení instalačního souboru vývojového prostředí a runtime HMI. Při jeho tvorbě bylo zapotřebí přidat výstupy obou projektů do instalační části a poté nastavit vytvoření zástupců v start menu a na ploše.



Obr. 26 Ukázka vytváření instalačního souboru

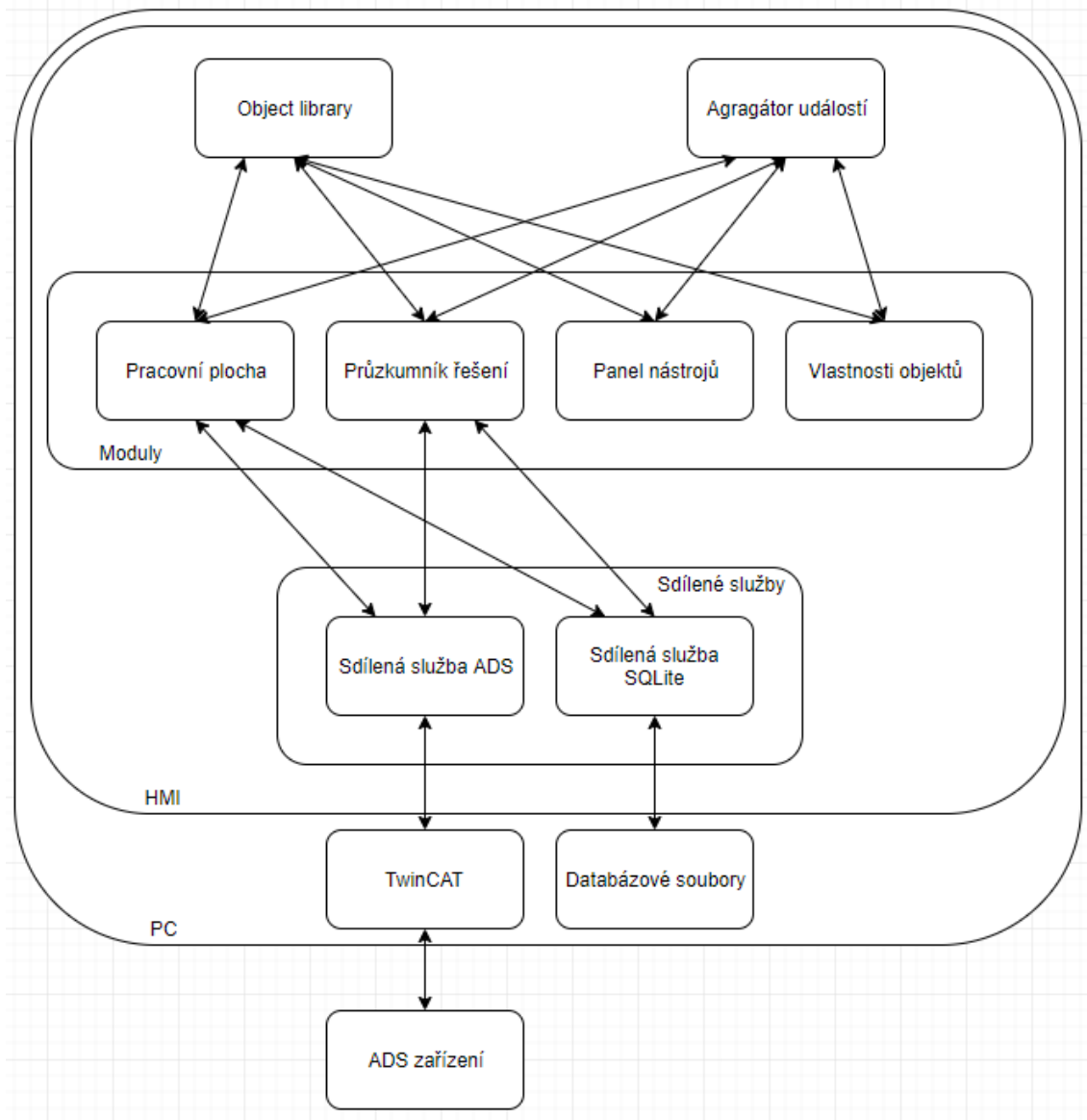
## **5. FUNKCE VÝVOJOVÉHO PROSTŘEDÍ A RUNTIME**

V této kapitole jsou popsány všechny vytvořené funkce a možnosti vývojového prostředí a runtime. Kapitola popisuje architekturu vývojového prostředí, funkci jednotlivých modulů a jejich vlastnosti. Jsou zde popsány všechny objekty, které jsem pro HMI vytvořil. Čtenář by měl z této kapitoly dostat celistvý přehled o tom, jak aplikace funguje.

### **5.1 Struktura vývojového prostředí**

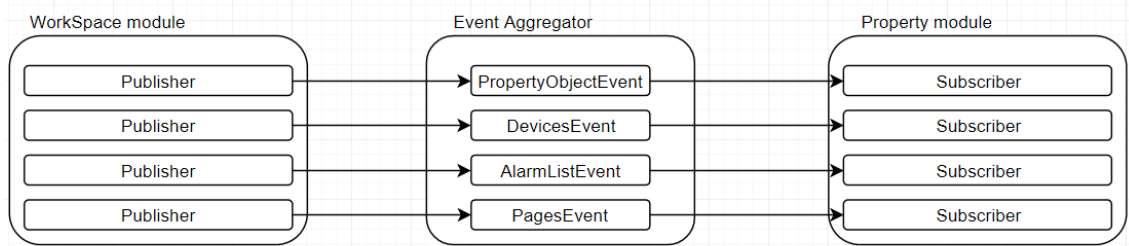
Vývojové prostředí jsem díky frameworku Prism rozdělil na moduly, které fungují zcela nezávisle, umožňují jednoduché testování a změna se dotkne vždy pouze konkrétního modulu, kde se změna provede. Moduly dle potřeby využívají sdílenou službu pro komunikaci se SQLite a ADS zařízením.

Vývojové prostředí jsem rozdělil do těchto modulů: vlastností, pracovní plochy, panele nástrojů a průzkumníku řešení. Jednotlivé moduly spolu komunikují pomocí agregátoru událostí (například při výběru objektu na pracovní ploše se do modulu Vlastností odešle reference daného objektu, aby zobrazil jeho konkrétní vlastnosti).



**Obr. 27** Struktura vývojového prostředí

Mezi moduly se přenáší data pomocí agregátoru událostí, což je primárně kontejner pro události, který umožňuje oddělení vydavatelů a odběratelů, aby se mohli samostatně vyvíjet. Agregátor událostí umožňuje vydavatelům a odběratelům vyhledat konkrétní událost. Je možné také přidat více vydavatelů a odběratelů. Moduly si tak díky těmto událostem mohou předávat struktury dat v reakci na nějakou událost v daném modulu. Modul nebo moduly na druhé straně tuto událost zachytí a zpracují dle svých potřeb. Každý z odběratelů tedy může zacházet s přijatými daty dle vlastního kódu.



**Obr. 28 Ukázka agregátoru událostí pro WorkSpace a Property moduly**

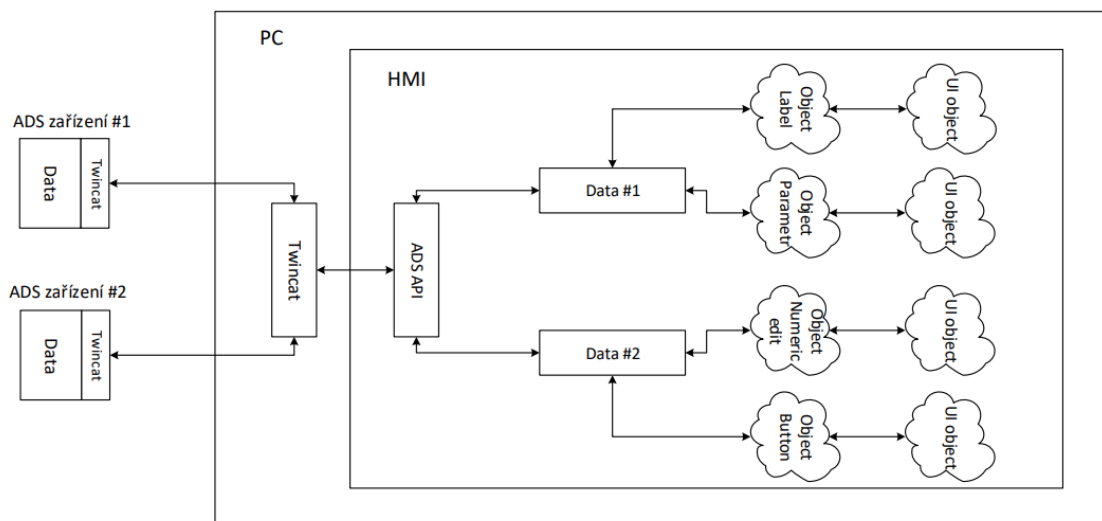
Všechny objekty a použité modely jsou uloženy v oddělené knihovně, která je společná jak pro vývojové prostředí, tak pro runtime. Díky tomu není potřeba uchovávat objekty na dvou místech současně a jejich úprava je mnohem jednodušší.

### 5.1.1 Sdílená služba ADS

Komunikace mezi HMI a ADS zařízením probíhá pomocí ADS API rozhraní společnosti Beckhoff. Ke své komunikaci využívá HMI sdílenou službu. Tato služba má vytvořené funkce pro přidávání a odebrání zařízení. Pro přidání a odebrání zařízení se používá struktura Device specifikuje název zařízení a jeho AmsNetId adresu.

Zařízení se strukturou Device však nejsou okamžitě přidány do této sdílené služby. Do sdílené služby jsou přidány zařízení až po spuštění vývojového prostředí do módu RUN. V tu chvíli se vytvoří struktura TwincatDevice, která okamžitě začne navazovat spojení s ADS zařízením. Pokud není nastavena adresa AmsNetId nebo není ADS zařízení dostupné, tak probíhá opětovný pokus o připojení.

Pokud je spojení s ADS zařízením úspěšně navázáno, tak se ze zařízení od komunikují všechny jeho symboly spolu s jejich datovým typem, místem v paměti, velikostí a názvem. Všechny tyto hodnoty jsou přidány do ADS oznámení, které automaticky generuje události při změně jejich hodnoty. Tyto změny se generují každých 200 milisekund, což by měl být dostatečný čas pro ochranu proti přehlcení HMI a zároveň o nepřetížení ADS zařízení.

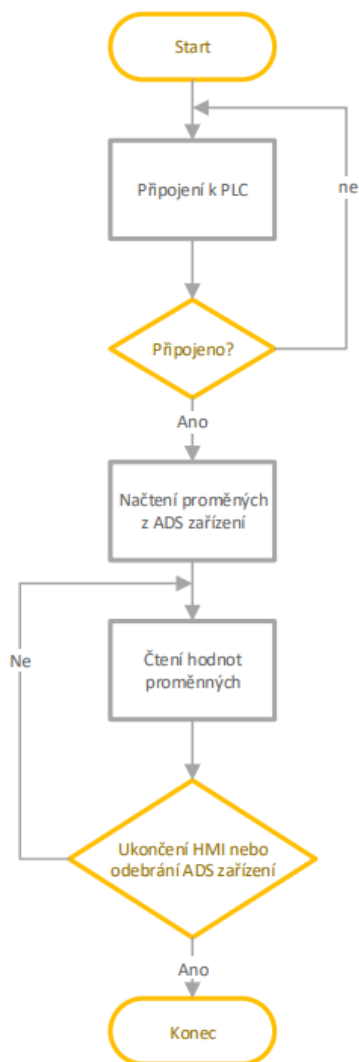


**Obr. 29 Datový tok mezi HMI a ADS zařízením**

Do této služby je možné přidat libovolný počet ADS zařízení se kterými vizualizace poté naváže spojení. Uživatel si poté u použitých objektů ve vizualizaci vybere dané ADS zařízení, se kterým chce daný objekt propojit. Data z ADS struktury poté pomocí eventu probublají až do daných zařízení, které obsahují dané zařízení a čtený tag, který způsobí vyvolání události.

Do ADS zařízení je také možné zapsat libovolný datový typ do libovolného tagu, který se v zařízení nachází. Například objekt tlačítko posílá do zařízení pouze 200 milisekundový puls v datovém typu Boolean. Parametr poté dokáže zapsat libovolnou hodnotu a libovolný datový typ do ADS zařízení. Díky tomu je možné nastavovat parametry libovolné datového typu a názvu.

Sdílená služba obsahuje tedy také dvě události. Jedna událost už byla výše zmíněna a slouží k zachytávání změn proměnných v ADS zařízení. Druhá se používá k diagnostice stavu zařízení. Pokud dojde ke ztrátě spojení nebo změně stavu PLC, tak kód danou změnu zachytí a vyšle událost k vyšší vrstvě, která danou informaci přidá indikujícímu objektu v HMI, pokud je použitý.



Obr. 30 Navázání komunikace s ADS zařízením

## 5.1.2 Sdílená služba SQLite

HMI je propojeno s databázovým souborem SQLite pomocí sdílené služby SQLite. Tato služba využívá pro tuto komunikaci knihovnu napsanou v C#. Moduly a runtime se k této sdílené službě připojují v závislosti potřeby archivace dat pro žurnál, případně pro přihlášení uživatelů. Při vytvoření projektu ve vývojovém prostředí HMI dojde k vytvoření dvou databázových souborů.

První soubor se jmenuje Alarm.sqlite a obsahuje tabulky se všemi listy alarmů, které jsou ve vývojovém prostředí vytvořeny. Název tabulky je vytvořen podle názvu seznamu alarmů. Tabulka obsahuje vždy následující sloupečky: id, popis, typ poruchy a IO. Id poruchy slouží jako jednoznačný identifikátor mezi polem hodnot v PLC a HMI. Popis specifikuje, o jakou poruchu se jedná. Typ poruchy rozeznává následující typy: neznámá, informační, varující a poruchová. IO slouží k definici vstupů a výstupu PLC, aby bylo možné rychleji reagovat na případnou poruchu či havárii.

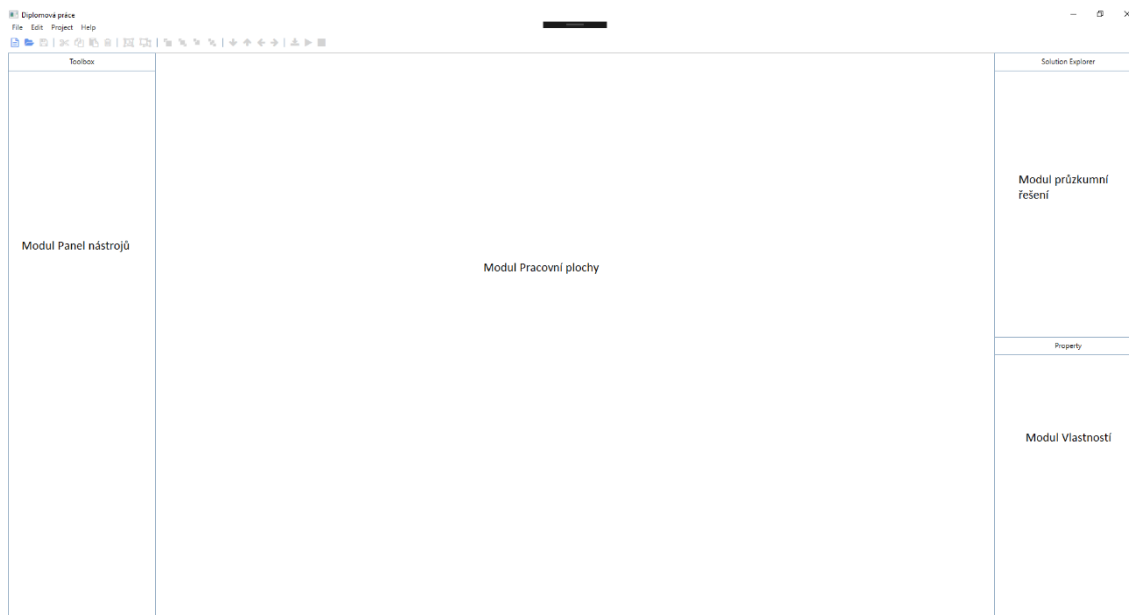
Druhý soubor jsem pojmenoval data.sqlite a obsahuje tři tabulky. První tabulka slouží k uložení uživatelů, kteří mají do HMI přístup. Obsahuje tedy přihlašovací jméno uživatele, jeho úroveň a heslo uživatele. Druhá tabulka ukládá uživatelský žurnál, kde je archivován každý zásah uživatele do chodu technologie. V poslední tabulce dochází k archivaci procesního žurnálu.

Při otevření projektu dochází k automatickému nastavení cesty na tyto soubory, se kterými se dá následovně pracovat stejně jako při jejich vytvoření. Všechny zápisy i čtení probíhá asynchronně a tím nezablokuje vlákno na dobu, než je signál vykonán.

## 5.2 Design vývojového prostředí

Jedná se o kontejner, do kterého jsou postupně načteny všechny moduly dle designu navrženém v XAML hlavního okna. Jeho hlavním úkolem je tedy spojit všechny moduly do výsledné aplikace, která vypadá poté jako jeden celek. Z pohledu uživatele tedy nelze poznat rozdíl, jestli je aplikace napsaná pomocí modulů a frameworku Prism však z vývojářského pohledu přináší obrovské výhody.

Pomocí hlavního okna je možné vyvolat celou řadu příkazů, které se vykonávají v příslušném module. Jednotlivé příkazy jsou umístěny do horního rozbalovacího menu a také jako ikony pod menu, aby bylo možné příkazy rychle použít.



**Obr. 31 Vzhled vývojového prostředí**

Každý z příkazů má svou vlastní ikonku, která slouží k jednoznačné identifikaci daného příkazu. Některé příkazy je možné používat pomocí standardizovaných zkratk. Ikony použité pro příkazy byly použité z volně dostupné knihovny fontawesome.wpf.

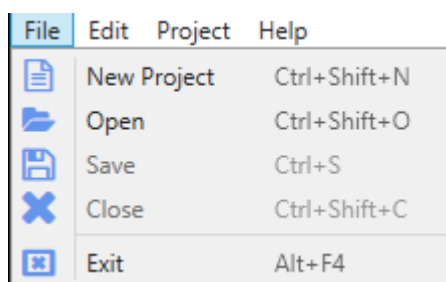
Jedním ze základních příkazů je otevření projektu, který má ikonku dokumentu. Při jeho zmáčknutí je uživateli spuštěn průvodce vytváření projektu, kde uživatel musí vyplnit název projektu, rozlišení projektu a umístění, kde se má projekt uložit. Po dokončení průvodce je vytvořena složka projektu na zadaném umístění v průvodci. Tato složka automaticky vytvoří podsložky pro ukládání databázových souborů, obrazovek a zařízení. Databázové soubory jsou jako jediné automaticky vygenerovány už při vytvoření projektu. Zkratka toho příkazu je Ctrl+Shift+N.

Každý vytvořený projekt je potřeba uložit. Proto vývojové prostředí nabízí příkaz k uložení daného projektu do dané cesty specifikované při tvorbě projektu. Při ukládání projektu se nejdříve vytvoří ve složce projektu soubor s názvem projektu a příponou .hmi. Tento soubor sebou nese informaci o jednotlivých obrazovkách a nastavení rozlišení vizualizace. Dále jsou do této cesty projektu uloženy soubory s designem a menu projektu. Dále jsou do složky obrazovky uloženy jednotlivé obrazovky vizualizace, které vždy mají svůj vlastní soubor pojmenovaný po názvu obrazovky. Do složky zařízení je uložený XML soubor se všemi ADS zařízeními v projektu. Zkratka toho příkazu je Ctrl+S.

Dalším nemálo důležitým příkazem je otvírání projektu, kterému jsem přidělil ikonu složky. Při jejím kliknutí je automaticky otevřený průzkumník souborů, který slouží k vybrání projektu. V tomto průzkumníku souborů je možné otevřít jen soubory s příponou \*.hmi. Díky tomu je zajištěno, že nebude otevřen žádný nevhodný soubor. Zkratka toho příkazu je Ctrl+Shift+O.

Každý otevřený projekt je možné zavřít pomocí příkazu, který má ikonu X. Při zavírání projektu dojde k vyvolání dialogového okna, které nabízí uživateli možnost daný projekt uložit. Zkratka toho příkazu je Ctrl+Shift+C.

Posledním příkazem této kategorie je známý příkaz se zkratkou ALT+F4, který ukončí aplikaci bez jakéhokoliv otálení.



Obr. 32 File menu

Další sada příkazů slouží k úpravě objektů na pracovní ploše. Příkazy je možné použít v přesně definovaných situacích, kdy dává použití daného příkazu smysl. U všech však musí být otevřený projekt a obrazovka vizualizace, což dává svým způsobem smysl, když se jedná úpravu objektu na pracovní ploše.

Prvním velice známým příkazem je vyjmout. Tento příkaz provede deserializaci daného objektu a uloží jej v podobě XAML do schránky. Tento objekt je poté z pracovní plochy odstraněn, aby bylo dosaženo požadovaného efektu vyjmutí. Příkaz jen symbolizován standardní ikonou otevřených nůžek. Pro použití tohoto příkazu musí být označen alespoň jeden soubor, avšak maximální počet není stanoven. Příkaz využívá standardizovanou zkratku Ctrl+X.

Dalším příkazem navazujícím je kopírování, které má ikonu dvou dokumentu přes sebe. Tento příkaz taktéž využívá svou standardizovanou zkratku, která je Ctrl+C. Zde dochází ke stejné akci jako při vyjmutí, avšak objekt není z pracovní plochy odstraněn.

Pro smysl těchto dvou příkazů je potřeba ještě vzpomenout příkaz vložení. Ten je možné použít pouze pokud schránka obsahuje data ve formě XAML a objekt je známý. Díky tomu je možné kopírovat pouze známé objekty. Zkratka příkazu je Ctrl+V.

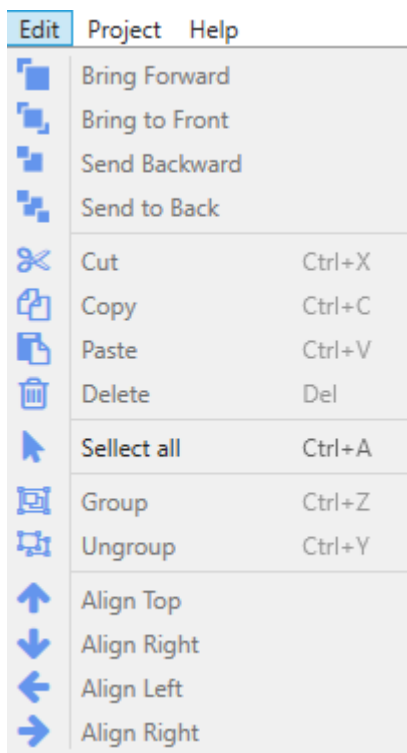
Pokud je objekty možné vyjmout, kopírovat, tak by bylo vhodné je umět i mazat. Proto byl napsán mazání, který nevratně smaže objekt z pracovní plochy. Tento objekt má ikonku popelnice a je možné jen použít klávesovou zkratku Del.

Pro výběr všech objektů na pracovní ploše byl vytvořen příkaz vyber vše. Tento objekt je symbolizován myší. Byl zvolena standardizovaná zkratka na Alt+A.

Dalšími příkazy jsou příkazy pro posouvání objektů mezi vrstvami. Tím je možné docílit požadovaného překrytí objektů dle potřeby dané vizualizace. Pro tuto vlastnost existují 4 typy toho příkazu. První z nich pošle objekt pouze o jednu vrstvu dozadu. Druhý pošle objekt úplně na nejspodnější vrstvu dané obrazovky. Stejným způsobem pracují i zbylé dva příkazy jen posílají objekty do popředí.

Ve vývojovém prostředí je také možné objekty spojovat do libovolných skupin a poté s nimi pracovat společně. Proto vznikl příkazy pro spojení a rozpojení skupiny. Tyto příkazy mají zkratku Ctrl+Z a Ctrl+Y.

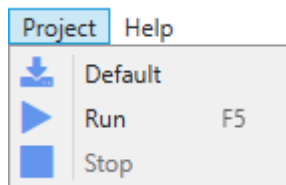
Posledními příkazy z této skupiny jsou příkazy pro zarovnání objektů na pracovní ploše. Tyto příkazy zarovnávají vždy minimálně dva objekty k sobě. Je možné tedy zarovnat objekty doleva, doprava, nahoru a dolů. Objekty se poté zarovnají co nejvíce na stranu k hraně objektu, který je co nejvíce pozičně na vybrané straně.



**Obr. 33 Editační menu**

Poslední sérií příkazů jsou příkazy pro samotnou interakci s ADS zařízením a nastavením defaultního projektu. Jak už napovídá, tak příkaz nastavení defaultního projektu nastaví projekt, který je poté spouštěný ve runtime. Konkrétně jde o uložení cesty projektu do registrů systému Windows, odkud si je poté runtime zase může vyčítat.

Příkazy pro interakci s ADS zařízením jsou Run a Stop. Příkaz Run přidá zařízení do sdílené služby ADS, která se poté snaží navázat spojení se zařízením. Pokud je navázání úspěšně provedeno, tak se začnou zobrazovat přímo data v objektech a uživatel, tak nemusí spouštět runtime, aby viděl, jestli má všechno správně a jak to bude vypadat. Příkaz Stop pak slouží k odpojení sdílené služby ADS od zařízení a následné odebrání těchto zařízení se sdílené služby.



**Obr. 34 Ukázka projektového menu**

## 5.2.1 Panel nástrojů

Tento modul slouží k zobrazení všech předdefinovaných objektů a jejich následnému použití. Vytvořené objekty přidané do panelu nástrojů je možné pomocí metody Drag&Drop libovolně přetahovat na pracovní plochu. Uživatel má tedy možnost si je vkládat na potřebná místa dle svého uvážení a potřeby.

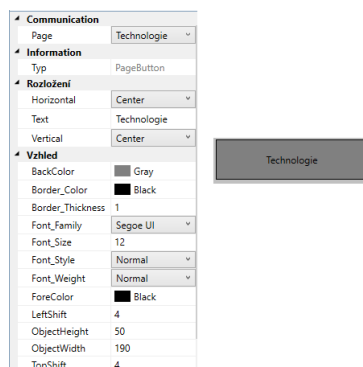
Každý z těchto objektů v panelu nástrojů je graficky zobrazený spolu se jménem objektu. Po najetí na objekt se kolem něj zobrazí čárkovaný obdélník, aby bylo vědět, který objekt se chystáme přesunout. Panel nástrojů nabízí následující objekty: PageButton, Zařízení, Alarmy, Žurnál, Label, Button, Ledka, Parametr, Trend, Čerpadlo, Ventil, Míchadlo, Tank, Sensor, PipeI, PipeL a PipeT.



Obr. 35 Ukázka objektů v panelu nástrojů

### 5.2.1.1 PageButton

Tento objekt funguje jako tlačítko pro přepínání mezi jednotlivými obrazovkami vizualizace. Tento objekt si uživatel může pomocí jeho vlastností nadefinovat podle jeho představ. Uživatel má celou řadu možností od typu a barvy písma před barvu pozadí, zarovnání textu po nastavení barvy a tloušťky ohraničení. Uživatel samozřejmě může nastavit text zobrazovaný na tlačítku, aby uživatel věděl, která stránka vizualizace se mu otevře po stisknutí tlačítka. Pro správnou funkci uživatel také musí nastavit obrazovku na kterou má tlačítko přepnout obsah stránky. PageButton je možné použít jak v menu, kde se jeho použití očekává, tak přímo na obrazovce pro rychlejší nebo logičtější přechod mezi stránkami.

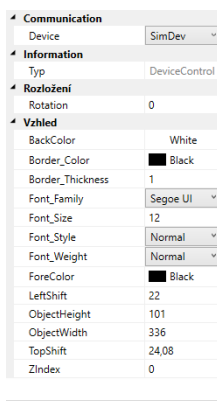


**Obr. 36 PageButton**

### 5.2.1.2 Zařízení

Zařízení je diagnostický objekt, který slouží k informaci o stavu ADS zařízení. Uživatel může umístit na jakoukoliv obrazovku umístit libovolný počet těchto objektů. Pokud dojde k výpadku ADS zařízení, tak v objektu dojde k okamžité změně stavu z Run na některý jiný. Objekt kromě informací o stavu ADS zařízení zobrazuje i jeho AmsNetId adresu a název. Pokud je tedy ve vizualizaci použito větší počet zařízení, tak je velice jednoduché dané zařízení od sebe rozlišit. Při tvorbě vizualizace je tedy dobré pojmenovávat jednotlivá ADS zařízení jednoznačně, nejlépe pomocí částí technologií, stroje, pod který spadají.

Na následujícím obrázku jsou vidět vlastnosti tohoto objektu. Jak je vidět, tak uživatel má zase možnost nastavit celou řadu vlastností. Díky tomu je možné docílit libovolného designu, a přitom dodržení šablonovitého použití i u dalších prvků.



**Obr. 37 Zařízení**

### 5.2.1.3 Alarmy

Objekt slouží jako diagnostický nástroj pro obsluhu dané vizualizace, kde může sledovat jednotlivé varování a poruchy na zařízení. Uživatel má možnost použít libovolný počet těchto zařízení a nemusejí být vždy použity pouze pro alarmy. Objektem je možné zobrazovat informaci o všech najetých trasách v technologii atd.

Teoreticky jej lze tedy využít na zobrazení jakéhokoliv obsahu informací, které lze v PLC vyjádřit jako pole hodnot datového typu Boolean.

Objekt zobrazuje pole hodnot pomocí tabulky, která má jasně definovanou strukturu. Při změně alespoň jedné hodnoty v poli dojde k okamžitému obnovené zobrazovaných informací. Uživatel má opět možnost nastavit celou řadu vlastností od barvy pozadí až po velikost písma.

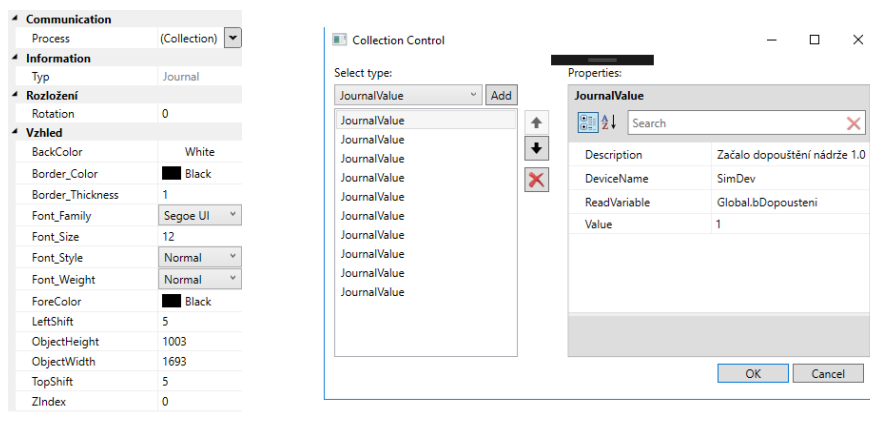
Id	Description	ID	AlarmLevel
1	Porucha čerpadla P-01A.	DD1	Error
2	Porucha čerpadla P-01B.	DD2	Error
3	Porucha čerpadla P-02A.	DD3	Error
4	Porucha čerpadla P-02B.	DD4	Error

Obr. 38 Alarmy

### 5.2.1.4 Žurnál

Objekt žurnál slouží k zobrazení archivovaných události, které zapříčinil uživatel svým zásahem a změnou stavu procesů dané technologie. Žurnál je rozdělen tedy do dvou různých typů: Uživatelský a procesní.

Uživatelský žurnál zobrazuje úkony, které byly provedeny ve vizualizace. Jsou zde zachyceny všechny přihlášení a odhlášení uživatelů. Všechny změny, které operátor z dané vizualizace provedl.



Obr. 39 Vlastnosti žurnálu

Uživatelský žurnál se generuje i bez předchozího nastavení a použití objektu. Funguje tedy automaticky v každé vizualizaci vytvořené tímto systémem a tento objekt slouží pouze jako jeho zobrazení. Technolog nebo technik poté mohou zpětně vyhodnocovat reakci operátoru na daný problém a tím zlepšení školení obsluhy.

Id	DateTime	Username	Description	New Value
111	6/5/2018 12:13:05 PM	Admin	Write to device SimDev variable Global.fSetPoint201A	40
110	6/5/2018 12:13:01 PM	Admin	Write to device SimDev variable Global.fSetPointDop	50
109	6/5/2018 12:12:58 PM	Admin	User logged.	Admin
108	6/5/2018 11:42:31 AM	Admin	Write to device SimDev variable Global.bAutomatStart	False
107	6/5/2018 11:42:30 AM	Admin	Write to device SimDev variable Global.bAutomatStart	True
106	6/5/2018 11:42:29 AM	Admin	User logged.	Admin
105	6/5/2018 9:48:09 AM	Admin	Write to device SimDev variable Global.AITTOAutomat	False
104	6/5/2018 9:48:09 AM	Admin	Write to device SimDev variable Global.AITTOAutomat	True

Obr. 40 Uživatelský žurnál

Procesní žurnál slouží k zobrazení procesních událostí, které jsou pro daný typ výroby či stroj charakteristické. Pro tuto akci už musí být objekt použit a nastaven na proměnné ADS zařízení, které indikují požadované stavy. Použití tohoto objektu může být zvláště vhodné pro zpětné vyhodnocování a sledování, jak rychle celý proces probíhá a kde jsou úzká místa výrobního celku.

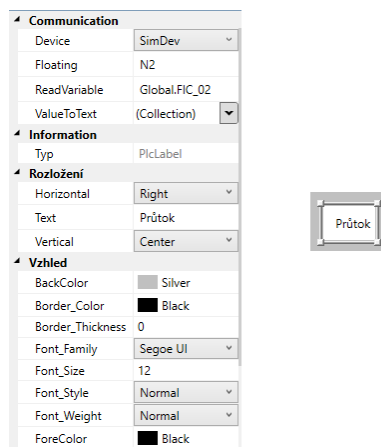
Id	DateTime	UserName	Description
253	6/5/2018 2:25:22 PM	Admin	Začalo přetečení z nádrže 1.01A do nádrže 2.01A.
251	6/5/2018 2:25:22 PM	Admin	Dokončeno dopouštění nádrže 1.01A.
250	6/5/2018 2:01:35 PM	Admin	Začalo dopouštění nádrže 1.01A.
249	6/5/2018 2:01:35 PM	Admin	Dokončeno vypouštění nádrže 2.01A.
248	6/5/2018 1:31:46 PM	Admin	Začalo vypouštění nádrže 2.01A.
247	6/5/2018 1:31:46 PM	Admin	Dokončeno míchání nádrže 2.01A.
246	6/5/2018 1:29:46 PM	Admin	Začalo míchání nádrže 2.01A.
245	6/5/2018 1:29:45 PM	Admin	Dokončeno filtrování v nádrži 2.01A.
244	6/5/2018 1:28:45 PM	Admin	Začalo filtrování v nádrži 2.01A.
243	6/5/2018 1:28:45 PM	Admin	Dokončeno přetečení z nádrže 1.01A do nádrže 2.01A.
242	6/5/2018 12:41:03 PM	Admin	Začalo přetečení z nádrže 1.01A do nádrže 2.01A.
241	6/5/2018 12:41:03 PM	Admin	Dokončeno dopouštění nádrže 1.01A.
240	6/5/2018 12:11:51 PM	Admin	Začalo dopouštění nádrže 1.01A.

Obr. 41 Procesní žurnál

### 5.2.1.5 Label

Jedná se o nejjednodušší a zároveň nejpoužívanější objekt vizualizace, který dokáže zobrazit hodnotu libovolné proměnné libovolného datového typu z ADS zařízení. Hlavní funkcí tohoto objektu je tedy zobrazení aktuální hodnoty měření, aktuální stav. Je s ním však možné zobrazit opravdu libovolnou hodnotu z ADS zařízení, kterou je potřeba předat operátorovi. Předpokládá se také, že tento objekt bude také použit k různým popiskům a označením zařízení na vizualizaci.

Jednou z jeho zajímavých vlastností je zobrazení stavu v podobě barvy a textu v závislosti na čtené hodnotě z ADS zařízení. Díky tomu je možné vytvořit stavový indikátor například pro mód zařízení. Kromě limitujícího omezení datového typu int, zde není žádné omezení v množství použitých stavů. Je tedy možné zobrazit pomocí labelu různé přechody, případně celé sekvence tak, aby uživatel přesně věděl, co se s technologií děje.

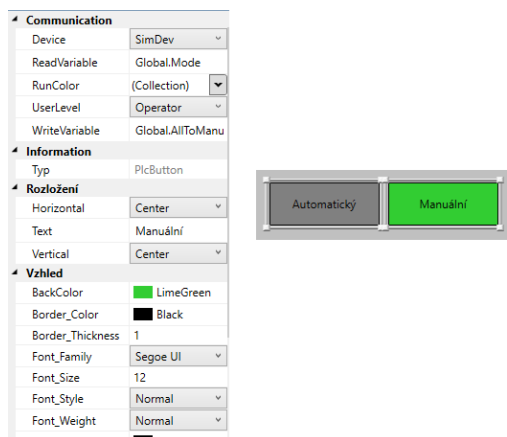


Obr. 42 Label

### 5.2.1.6 Button

Objekt tlačítka slouží převážně k poslání pulzu o šířce 200ms do zařízení ADS. Tento pulz je možné chytat například pomocí funkčních bloků SR nebo RS a poté dále vyhodnocovat. Button umí zobrazit stejně jako label stav v podobě barvy pozadí. Je taktéž možná nastavit libovolný počet stavů, které dokáže tlačítko zobrazit.

Tlačítko nabízí možnost libovolného nastavení vzhledu, který se dá poté šablonovitě použít. Každé tlačítko nabízí možnost nastavení pozadí, šířky ohraničení, barvu ohraničení, styl, typ a velikost písma a jeho barvu.



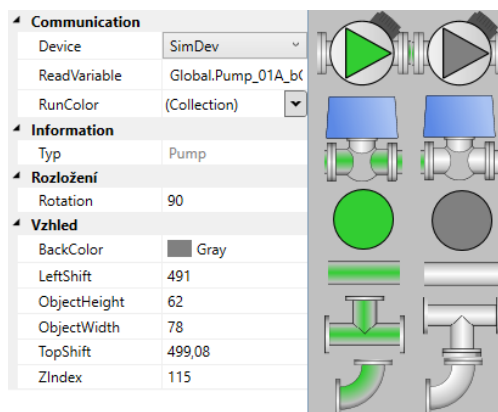
Obr. 43 Button

### 5.2.1.7 Ledka, Čerpadlo, Ventil, PipeI, PipeL, PipeT a Míchadlo

Následující vytvořené objekty slouží k indikaci stavu daného zařízení. Ve své podstatě všechny tyto zařízení fungují úplně stejně, avšak liší se konkrétním vzhledem. Všechny objekty je možné libovolně otáčet a tím je možné dosáhnout potřebné polohy.

Každý z vytvořených objektů má specifický design v závislosti na tom, jaké technologické zařízení představuje. Díky těmto objektům je možné vytvořit určitou část technologie. Díky potrubí je pak možné signalizovat, jestli daným potrubí něco protéká. Pokud je například potrubí sdílené pro více látek, tak je možné pomocí stavů indikovat i jaká látka se právě v potrubí nachází.

Všechny výše uvedené poruchy umí zobrazit poruchu daného zařízení. Tato porucha je indikována pomocí blikání daného zařízení červenou barvou v 500 milisekundových intervalech. Červenou barvu poruchy nelze změnit, aby bylo dodrženo standardní řešení. Je možné tedy předat informaci uživateli o poruše nejen v podobě alarmu, ale i pomocí zařízení, kterého se to týká.

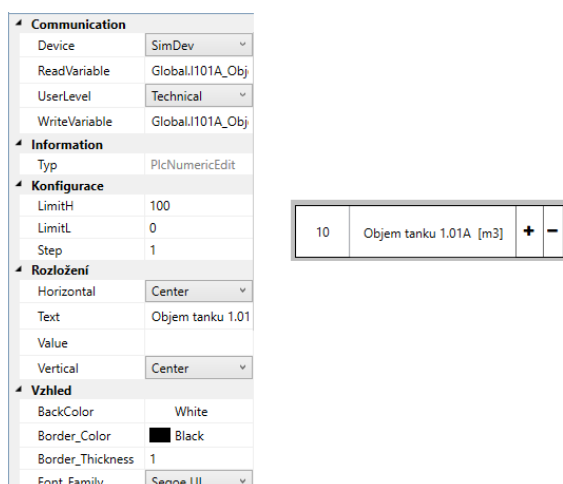


Obr. 44 Ledka, Čerpadlo, Ventil, PipeI, PipeL a PipeT

### 5.2.1.8 Parametr

Hlavním úkolem tohoto objektu je umožnit oprávněným osobám měnit parametry technologického procesu v PLC. Tento objekt nabízí možnost nastavování parametrů libovolného datového typu libovolné proměnné v ADS zařízení. Je možné konkrétní parametry omezit horní a spodní limitou, aby se hodnoty mohly pohybovat pouze v přesně definovaném interval. Je také možné nastavit hodnotu kroku o který se navýší nebo sníží hodnota po zmáčknutí příslušného tlačítka. Uživatel má taktéž možnost zadat hodnotu přímo do pole hodnoty pomocí klávesnice. Tato hodnota je zapsána do ADS zařízení až po stisknutí klávesy enter.

Parametru je možné nastavit různé vlastnosti. Důležitým parametrem kromě čtené a zapisované proměnné je text, který slouží k popsání daného parametru, aby bylo zřejmé, jaký parametr je právě upravován. Pozadí a text parametru je možné upravit stejně jako u předchozích objektů. Je možné také nastavit úroveň oprávnění, které uživatel musí mít při úpravě parametru.



Obr. 45 Parametr

### 5.2.1.9 Trend

Trend slouží k vynesení průběhů dané veličiny v závislosti na čase. Uživatel má možnost nastavit si periodu s kterou jsou vzorky pořizovány. Například u teploty, která se mění velice pomalu je zaznamenávání co 500 milisekund velice přehnané, avšak u rychlého polohování to může být žádoucí. Dále je potřeba nastavit počet vzorků, které mají být v daném grafu vyneseny. Z této hodnoty je poté dopočítána časový interval, který trend pokryje. Každý vzorek přesahující žádaný počet vzorků je postupně mazán, aby nedocházelo k přehlcování paměti.

Do trendu je možné v parametrech umístit libovolný počet zobrazovaných veličin. Tyto veličiny jsou poté vzorkovány současně a postupně vynášeny do trendu. Každé veličině je možné přiřadit libovolnou barvu, aby bylo možné danou sérii dat odlišit od ostatních. Je vhodné také každou sérii označit titulkem, který je poté zobrazený v legendě pod trendem



Obr. 46 Trend

### 5.2.2 Pracovní plocha

Pracovní plocha slouží k vytváření jednotlivých obrazovek vizualizace, tvorbě designu vizualizace a tvorbě menu. Nově otevřená obrazovka, design nebo menu je vložena do vlastní záložky, takže je možné jich mít současně otevřeno více. Záložky jsou opatřeny vždy názvem otevřeného objektu z průzkumníku řešení. Pokud se uživatel snaží otevřít už otevřený dokument, tak dojde pouze k přepnutí na konkrétní záložku.

Na pracovní plochu jsou umísťovány objekty z panele nástrojů. Při každém přidání objektu je vytvořen nový objekt daného typu. Všechny objekty vychází ze stejné třídy a díky tomu dědí ID a parentID.

ID slouží k jednoznačnému určení objektu na pracovní ploše. Každý objekt na pracovní ploše jej obsahuje a nikdy se dané ID neopakuje. ParentID poté slouží jako identifikátor všech objektů spojených do jedné skupiny. Všechny objekty stejné skupiny budou mít tedy společné ParentID.

Při přetažení objektu z panele nástrojů na pracovní plochu dojde k jeho vložení na místo, kde byl položen. Objekt se vytvoří s přesně danými rozměry a základním designem.

Každý objekt je možné kliknutím označit. Při označení se kolem objektu objeví rámeček, díky kterému je možné daný objekt libovolně zvětšovat a zmenšovat. Po najetí na objekt na pracovní ploše dojde ke změně kurzoru a objekt je možné posouvat na

libovolné místo na pracovní ploše. Díky tomu lze libovolně velké objekty libovolně rozmístit na pracovní ploše.



**Obr. 47 Ukázka objektu na pracovní ploše**

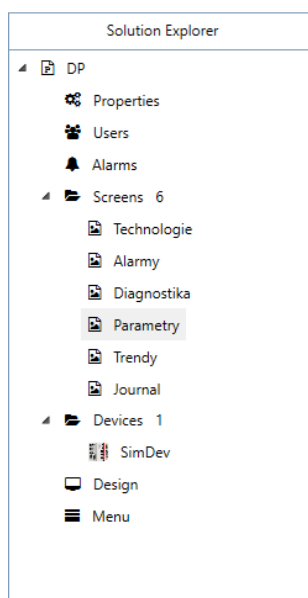
Objekty na pracovní ploše je možné spojovat do skupin, které je pak možné posouvat a zvětšovat vždy společně. Objekty jsou v různých vrstvách a je možné je přesouvat do pozadí i popředí. Objekty je možné taktéž vyjmout, kopírovat, vložit a smazat. Pokud jsou vybrány alespoň dva objekty, tak je možné je zarovnávat mezi sebou.

### 5.2.3 Průzkumník řešení

Jedná se o nejdůležitější modul celého vývojového prostředí, protože zajišťuje jeho základní funkce. V průzkumníku řešení jsou zobrazeny pomocí stromové struktury všechny vlastnosti projektu vizualizace. Uživatel může tyto vlastnosti libovolně editovat, přidávat a odebírat.

Každý projekt obsahuje následující vlastnosti: vlastnosti, uživatele, alarmy, obrazovky, zařízení, design a menu. Pro přidání další obrazovky a zařízení stačí kliknout pravým tlačítkem na nadpis obrazovek nebo zařízení a přidat jej. Stejným způsobem je možné danou obrazovku, případně zařízení smazat.

Každá část stromové struktury má svou jedinečnou a charakteristickou ikonu, kterou by měla uživateli pomoci se lépe v něm orientovat. V případě dalšího rozšiřování projektu je možné průzkumník řešení snadno doplňovat o další vlastnosti.



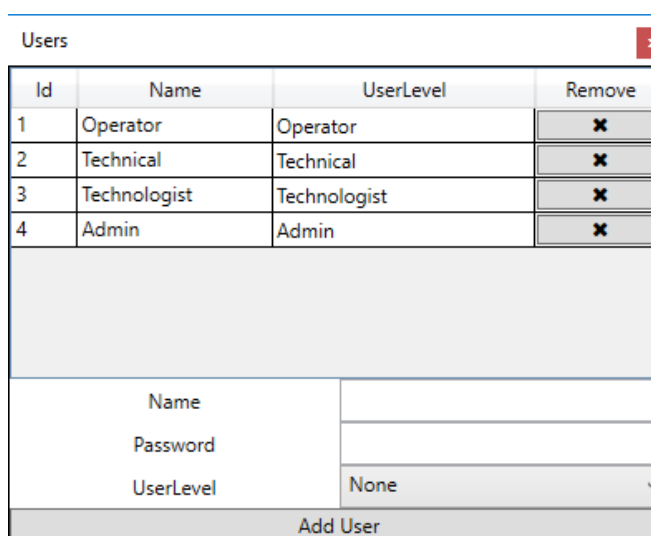
**Obr. 48 Průzkumník řešení**

### 5.2.3.1 Uživatelé

Při vytvoření projektu jsou základní uživatelé vytvořeni automaticky. Jedná se o uživatele „Operator“ s heslem „111“ a uživatelskou úroveň Operátor, „Technician“ s heslem „222“ a uživatelskou úroveň technik, „technologist“ s heslem „333“ a uživatelskou úroveň technolog a „Admin“ s heslem „999“ a uživatelskou úroveň administrátor.

Pomocí dvojitého kliku na uživatele dojde k otevření okna, ve kterém je možné editovat uživatele vizualizace. Je možné přidat libovolné množství uživatelů, kterým je potřeba specifikovat jméno, heslo a uživatelskou úroveň. Každý uživatel může být taktéž odebrán z vizualizace.

Vizualizace nabízí 5 různých uživatelských úrovní: none, operátor, technik, technolog a admin. Uživatel s vyšší oprávněním má vždy práva své úrovně a všech nižších úrovní.



Id	Name	UserLevel	Remove
1	Operator	Operator	✘
2	Technical	Technical	✘
3	Technologist	Technologist	✘
4	Admin	Admin	✘

Name	<input type="text"/>
Password	<input type="password"/>
UserLevel	None ▾

Add User

Obr. 49 Uživatelé

### 5.2.3.2 Alarmy

V alarmech se vytváří tabulky alarmů, které jsou poté propojeny s polem hodnot v ADS zařízení, podle nichž jsou zobrazovány. Je možné přidat libovolné množství listů alarmů, které jsou poté ukládány do SQLite. Každý list může mít libovolné množství prvků s následujícími sloupečky: ID, popis, IO a úroveň alarmu.

ID představuje jednoznačný identifikátor poruchy, přes který jsou data propojována s polem hodnot v ADS zařízení. Popis slouží k popsání alarmu, aby obsluha/uživatel byli schopni rozpoznat o jaký alarm se jedná. IO definuje o jaký vstup nebo výstup ADS zařízení se jedná, aby oprava dané poruchy byla rychlá. Úrovně alarmu mohou být následující: neznámá, informační, varující a poruchová.

Zde je také možné přidat libovolné množství seznamů poruch, které se poté propojí s objektem alarmů a ten daný list zobrazí. Může být velice vhodné při použití listů nejen na poruchy, ale i na interpretaci jiných dat uložených v poli hodnot v ADS zařízení.

Alarms			
Alarmy			Add AlarmList
Id	Description	IO	AlarmLevel
1	Porucha čerpadla P	DO1	Error
2	Porucha čerpadla P	DO2	Error
3	Porucha čerpadla P	DO3	Error
4	Porucha čerpadla P	DO4	Error
5	Ucpaný filtr.	DI1	Error
Add Row			

Obr. 50 Alarmy

### 5.2.3.3 Obrazovky

V této části jsou uchovávány všechny obrazovky projektu vizualizace. Každému projektu může být přidán libovolné množství obrazovek. Těmto obrazovkám je možné nastavovat barvu pozadí a jejich jméno. Toto jméno je pak použito při ukládání jako název souboru. Jméno je také zobrazováno při otevření stránky hlavičce záložky na pracovní ploše. Každá obrazovka může být odstraněna. Po odstranění je odstraněn i její XML soubor, takže neexistuje žádná možnost návratu.

Tyto objekty obrazovek jsou pro vizualizaci velice důležité, protože na pozadí v datové struktuře sebou nesou i všechny objekty dané obrazovky, které jsou při otevření projektu načteny z XML.

```
<?xml version="1.0" encoding="utf-8"?>
<Root>
  <Workspace>
    <Type>ObjectLibrary.WorkSpace, ObjectLibrary, Version=1.0.0.0, Culture=neutral, PublicKeyToken=null</Type>
    <ScreenName>Technologie</ScreenName>
    <Width>1705</Width>
    <Height>1015</Height>
    <BackColor>#FFC0C0</BackColor>
    <Border_Color>#FF0000</Border_Color>
  </Workspace>
  <Objects>
    <Object>
      <Type>ObjectLibrary.Tank, ObjectLibrary, Version=1.0.0.0, Culture=neutral, PublicKeyToken=null</Type>
      <ID>77665090-968a-4664-9d93-944979bdd82</ID>
      <IsGroup>false</IsGroup>
      <ParentID>00000000-0000-0000-0000-000000000000</ParentID>
      <Properties>
        <Width>172</Width>
        <Height>346.9999999999998</Height>
        <x>191</x>
        <y>173.1599999999998</y>
        <z>24</z>
        <Rotation>0</Rotation>
        <BackColor>#FFFFFF</BackColor>
        <Device />
        <ReadVariable></ReadVariable>
        <RunColors />
      </Properties>
    </Object>
    <Object>
      <Type>ObjectLibrary.Tank, ObjectLibrary, Version=1.0.0.0, Culture=neutral, PublicKeyToken=null</Type>
      <ID>192c459b-ffbd-4336-851d-7d1aafd6a6ac</ID>
      <IsGroup>false</IsGroup>
      <ParentID>00000000-0000-0000-0000-000000000000</ParentID>
      <Properties>
        <Width>172</Width>
        <Height>347</Height>
        <x>1220</x>
        <y>109.1599999999998</y>
        <z>4</z>
        <Rotation>0</Rotation>
        <BackColor>#FFFFFF</BackColor>
        <Device />
        <ReadVariable></ReadVariable>
        <RunColors />
      </Properties>
    </Object>
  </Objects>
</Root>
```

Obr. 51 Ukázka XML uložené obrazovky

### 5.2.3.4 Zařízení

Zde je potřeba přidat všechny ADS zařízení se kterými bude vizualizace komunikovat. Jejich množství není nijak omezena. Každému zařízení je potřeba specifikovat AmsNetId. Pomocí něj se poté k ADS zařízení HMI připojuje. Zařízení má také své jméno, pro jednodušší použití a orientaci při větším počtu zařízení.

```
<?xml version="1.0" encoding="utf-8"?>
<Root>
  <Devices>
    <Device>
      <Name>SimDev</Name>
      <AmsNetId>192.168.254.226.1.1</AmsNetId>
    </Device>
  </Devices>
</Root>
```

Obr. 52 Ukázka ukládání zařízení

### 5.2.3.5 Design

Design slouží k rozvržení vzhledu vizualizace do volné plochy. Každý projekt obsahuje v designu tyto čtyři objekty: lišty alarmů, přihlašování, obsah a menu. Je možné libovolně měnit jejich velikost. V závislosti na změně velikosti se mění velikosti podružných oken. Tedy změnou velikosti obsahu dojde i ke změně velikosti všech obrazovek, které jsou do daného místa vkládány. Lišta alarmů slouží k zobrazení poslední poruchy, která nastala.



Obr. 53 Ukázka lišty alarmů

Přihlašování slouží k přihlášení uživatelů vytvořených v uživateli. Každý uživatel musí vždy při přihlášení vyplnit své přihlašovací údaje, které jsou porovnány s údaji vytvořenými při vytváření uživatelů. Pokud je přihlášení úspěšné, tak dojde k zobrazení přihlášeného uživatele. Tento uživatel se může kdykoliv odhlásit pomocí tlačítka. Všechny přihlašovací akce jsou zároveň zaznamenávány do uživatelského žurnálu.



**Obr. 54 Design**

Design může být tvůrcem libovolně přeskládán do požadované podoby. Díky tomu je dosaženo skoro univerzálního designu. Je možné tedy menu libovolně přesouvat zprava do leva. Žádný z objektů tedy nemá absolutní pozici a velikost, ale je možné je libovolně měnit dle potřeby. Je možné také nastavit libovolnou barvu pozadí.

#### **5.2.3.6 Menu**

Slouží k vytvoření menu pro přepínání obrazovek v runtime. Každý projekt automaticky vygeneruje prázdné menu. Do menu ovšem může být vložen libovolný prvek z panelu nástrojů. Jedná se tedy o část designu, která je vždy načtena na pozici menu a je vždy viditelná. Uživatel si může rozvrhnout menu dle své potřeby a designu.

### **5.2.4 Vlastnosti objektu**

Při vybrání objektu na pracovní ploše dojde k odeslání reference objektu pomocí události do modulu vlastností. Je možné vybrat i zařízení z průzkumníku řešení. Poté jsou v tabulce vlastností zobrazeny jednotlivé vlastnosti objektu, které je možné nastavovat.

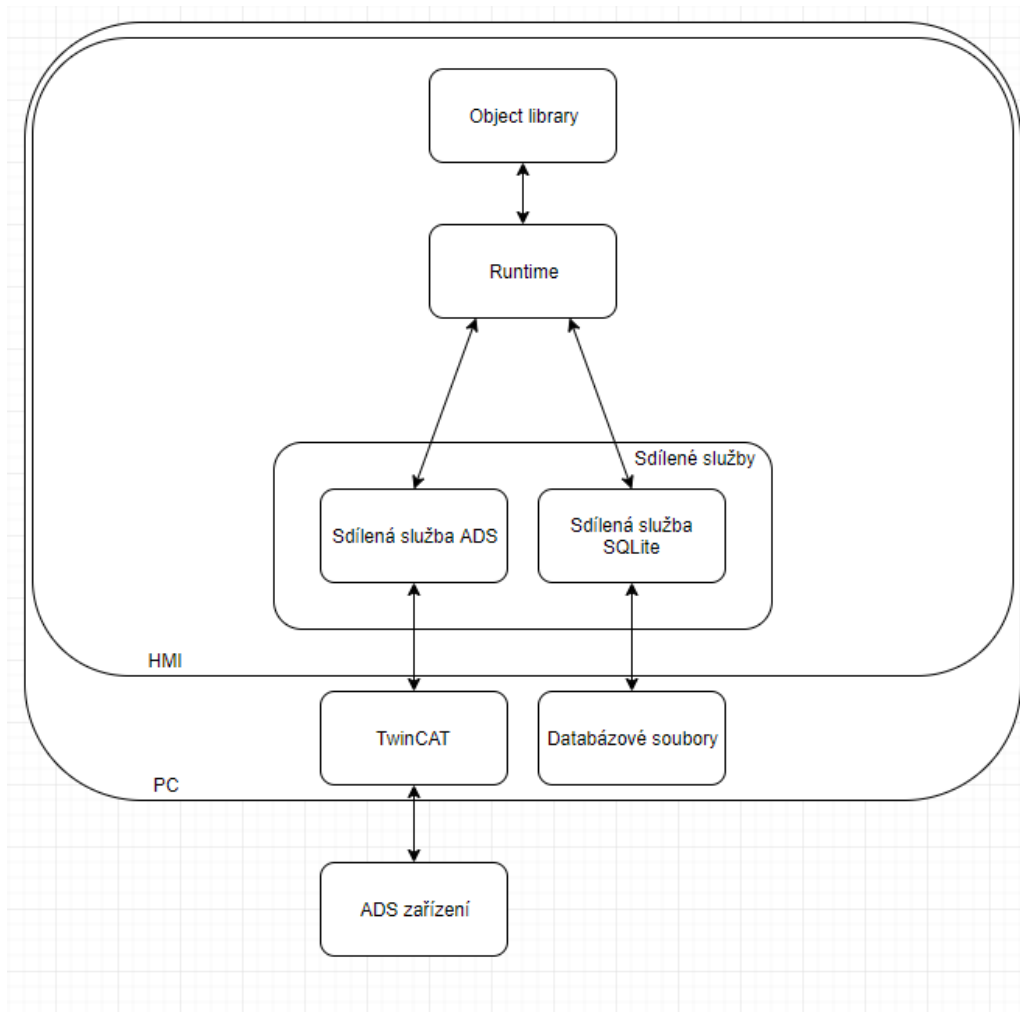
Objekt tabulka vlastností je použita z open source knihovny extended toolit, který se automaticky stará o zobrazení vlastností vybraného objektu na pracovní ploše.

## **5.3 Struktura Runtime**

Runtime je tvořen mnohem jednodušší strukturou než vývojové prostředí, protože zde nejsou potřeba jednotlivé moduly. Nejdříve se z registrů systému Windows zjistí, odkud

se má vizualizace začít načítat. Hlavní aplikace poté nejdříve načte design vytvořený ve vývojovém prostředí HMI. Poté postupně načte jednotlivé soubory jako jsou menu a zařízení. Nakonec se načtou všechny obrazovky a zadá se povel k připojení ADS zařízení. Poté je vizualizace plně načtena a je možné ji používat.

Všechny události a oznámení jsou zachytávány pomocí runtime, který následně jednotlivé prvky uživatelského rozhraní aktualizuje dle změny. Čtení hodnot tedy probíhá automaticky při jakékoliv změně ať už v ADS zařízení nebo v databázových souborech. Zápis poté probíhá v reakci na nějakou událost, které nastala.



**Obr. 55** Struktura Runtime

## 5.4 Design Runtime HMI

Design runtime aplikace vychází přesně z vytvořeného vzhladu ve vývojovém prostředí. Runtime si při spuštění načte design projektu. Poté postupně načte všechny obrazovky a menu. Po jejich načtení je vloží do designu na jejich pozici. Po tomto kompletním načtení vzhladu projektu se načtou ADS zařízení, které jsou hned automaticky připojeny a okamžitě začnou přenášet data. Na konci těchto kroku je runtime vizualizace plně načten a běží pro použití uživatele.



**Obr. 56 Ukázka načteného projektu v RUNTIME**

## **5.5 Instalační balíček HMI**

Posledním projektem vizualizace je instalační balíček s vývojovým prostředím a runtime HMI, který se nainstaluje na uživatelem zvolené místo na disku. Díky tomu je možné aplikaci volně šířit a jednoduše nainstalovat na konkrétní stroj, zařízení nebo PC.

Instalační balíček obsahuje všechny potřebné knihovny, aby bylo možné vše používat bez jakýchkoliv problémů. Vývojové prostředí je pojmenováno DP\_BeckhoffTwincat3 a runtime poté DP\_BeckhoffTwincat3\_Runner. Obě aplikace jsou plně funkční jako 32 bitová aplikace.

## 6. TVORBA UKÁZKOVÉHO ŘEŠENÍ

Nejprve jsem v mnou vytvořeném vývojovém prostředí vytvořil nový projekt s názvem DP. Po přemýšlení jsem se rozhodl vytvořit ukázkové řešení na dvojici tanků, kde do prvního tanku je kapalina napouštěna z vlaku. Poté pomocí čerpadla se záskokem přečerpána přes ventil do druhého tanku. V tomto tanku je pomocí čerpadla honěna kapalina dokola přes filtr, následně míchána a vypuštěna pryč.

### 6.1 Tvorba programu PLC

Nejprve bylo potřeba založit projekt ve Visual Studio pro TwinCAT XAE. Vytvořený projekt obsahuje předpřipravené nastavení PLC v podobě délky cyklu zařízení, počet úloh, typ systému a další.

K tomuto vytvořenému projektu je potřeba do kolonky PLC vytvořit podprojekt, který bude fungovat jako PLC program. Je možné zde využít i dalších vlastností, které TwinCAT 3 nabízí. Jedná se převážně o tvorbu bezpečnostních programů, projektů v c++ a nastavení I/O. Jelikož mnou vytvořený projekt je simulován na počítači v testovací verzi, tak kromě PLC projektu nebudeme potřebovat jiné vlastnosti.

Každý projekt obsahuje globální datové typy, reference na použité knihovny a složky pro tvorbu samotného programu. Jak už bylo zmíněno v 2. kapitole, tak ADS zařízení je možné programovat podle standardu IEC 61131-3. Pro mě nejbližší jsou jazyky strukturový text a funkční bloky.

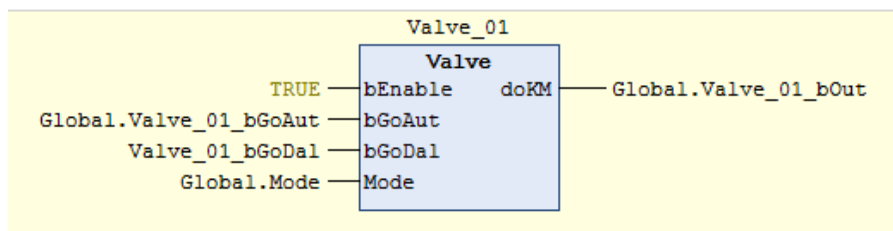
Nejprve bylo potřeba vytvořit funkční bloky pro jednotlivé zařízení. Proto jsem vytvořil postupně funkční bloky pro ventil, čerpadlo, tank a motor. Díky tomu přidání jednoho z těchto zařízení znamená použití jednoho konkrétního funkčního bloku. Kdokoliv, kdo bude upravovat zdrojový kód PLC pak musí znát jen funkci těchto bloků a nezáleží, jestli upravuje funkci zařízení číslo 10 nebo 100.

Poloautomatický a automatický režim poté využívá tyto funkční bloky a s jejich pomocí ovládá jednotlivé zařízení. Poloautomatický režim kromě ovládání zařízení přidává do řízení i různé procesní závislosti, které je potřeba hlídat při chodu technologie. Automatický režim poté už jen spouští poloautomatické část v přesně daných sekvencích, aby docílil požadovaného výsledku.

#### 6.1.1 Ventil

Pro ventily jsem vytvořil jako jednoduchý blok, který jej otvírá a zavírá dle povelu z aplikace nebo automatického režimu. Pokud je ventil přepnut do automatického režimu, tak jej není možné přepnout ručně z vizualizace, a naopak pokud je ventil v ručním režimu, tak není možné využívat automatický a poloautomatický režim.

Pro reálný systém by bylo potřeba doplnit ještě snímače koncové polohy, které se standardně používají u ventilů. Tam by samozřejmě bylo potřeba ošetření mezi polohového stavu doby přejetí, protože by mohla dojít k zaseknutí/zacpání ventilu.

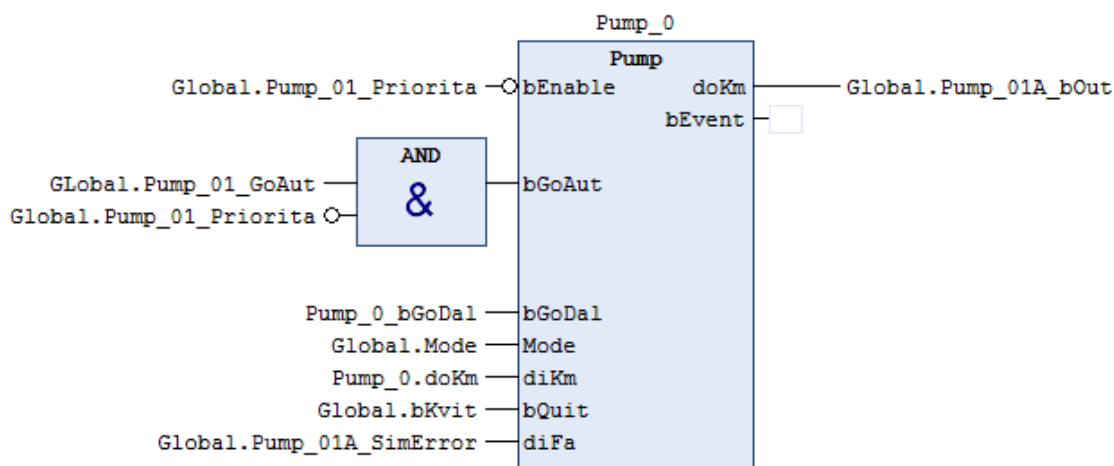


Obr. 57 Funkční blok ventilu

## 6.1.2 Čerpadlo

Čerpadlo má trochu složitější blok, protože kromě zapínacího a vypínacího povelu a změny módu, tak hlídá zpětně chod čerpadla. V praxi se vstup detekující chod používá k ověření, že čerpadlo jede. Většinou se tento vstup bere ze stykače před čerpadlem, pokud se nejedná o čerpadlo s frekvenčním měničem, který dokáže detekovat následující informaci například podle proudu. Chytřejší frekvenční měniče dokážou potom detekovat mnohem více poruch se kterými se dá poté pracovat. Každá porucha musí být poté potvrzena operátorem z vizualizace.

Jelikož čerpadla fungují vždy ve dvojici jako záskoková, tak je potřeba rozlišovat prioritu jednotlivých čerpadel. Obsluha pak může specifikovat, které zařízení si přeje prioritizovat. V automatickém režimu při poruše jednoho čerpadla poté najíždí automaticky druhé a údržba má potom čas dané čerpadlo opravit.



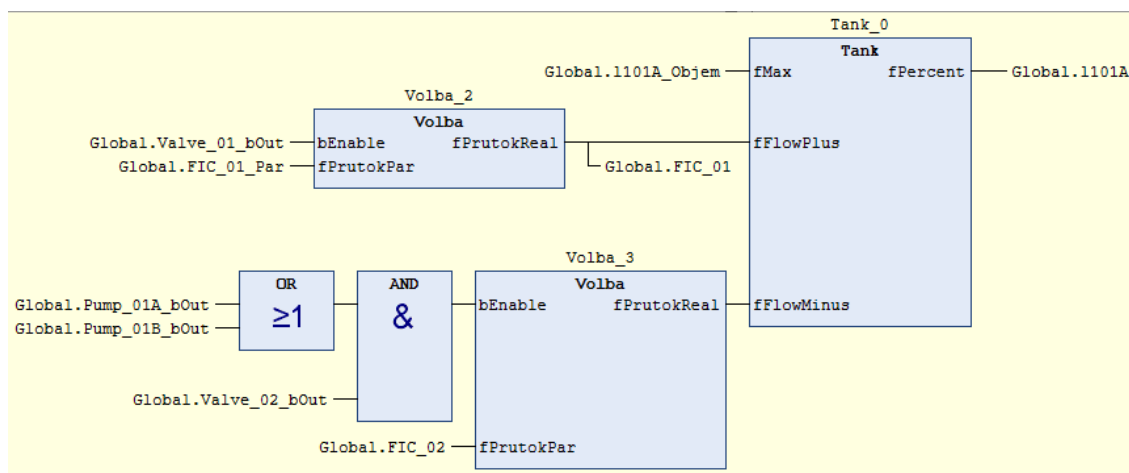
Obr. 58 Funkční blok čerpadla

## 6.1.3 Tank

Funkční blok tank slouží k vyhodnocování hladiny v tanku. Jelikož se jedná o simulaci, tak je potřeba simulovat měření hladiny. Uživatel v parametrech HMI definuje objem tanku a průtoky pro jednotlivé průtokoměry a z těchto hodnot je potom počítáno procentuální naplnění tanku. Tanky samozřejmě nejsou plněny a vypouštěny pořád, ale pouze v závislosti na zařízeních před a za tankem.

V praxi se hladina měří pomocí různých metod měření. Jednou z hodně používaných v současnosti je měření hladiny pomocí radaru. Jsou však místa, kde není

měření tak kritické, tak se hladina měří pomocí plováku. Existují tedy různé metody měření hladiny v tanku a jsou vybírány dle konkrétní aplikace a požadované přesnosti.



Obr. 59 Funkční blok Tanku

## 6.1.4 Poloautomatický režim

Poloautomatický režim lze rozdělit do pěti různých procesů, které popisují určitou část technologie. Pro námi vytvořenou technologii byli vymyšleny následující režimy: dopouštění, přečerpávání, filtrování, míchání a vypouštění.

Dopouštění je závislé na třech volitelných parametrech. Na průtoku průtokoměrem FIC-01, požadované hladině v tanku 1.01A a objemu tanku 1.01A. Tyto parametry je možné nastavit na obrazovce parametry. Po spuštění dopouštění je nejprve zkontrolováno, jestli požadovaná hladina je větší než aktuální. Pokud ano, tak se otevře ventil a začne se dopoušět na požadovanou hladinu. Po dopouštění se ventil opět uzavře a dopouštění je ukončeno. Pokud je aktuální hladina větší než požadovaná nebo nejsou zařízení přepnuta do automatu, tak napouštění nezačne.

Přečerpávání je závislé na čtyřech volitelných parametrech. Jsou to průtok FIC-02, požadovaná hladina přečerpání, objem tanku 2.01 a minimální hladiny tanku 1.01A. Před zapnutím čerpadla a přenastavení klapky na požadovanou trasu jsou zkontrolovány podmínky, jestli je požadovaná hodnota nižší než aktuální. Pokud ano, tak je zahájeno přečerpávání. Přečerpávání končí v tu dobu, když v tanku 1.01A není dostatek kapaliny nebo je dosaženo požadované hladiny v tanku 2.01A.

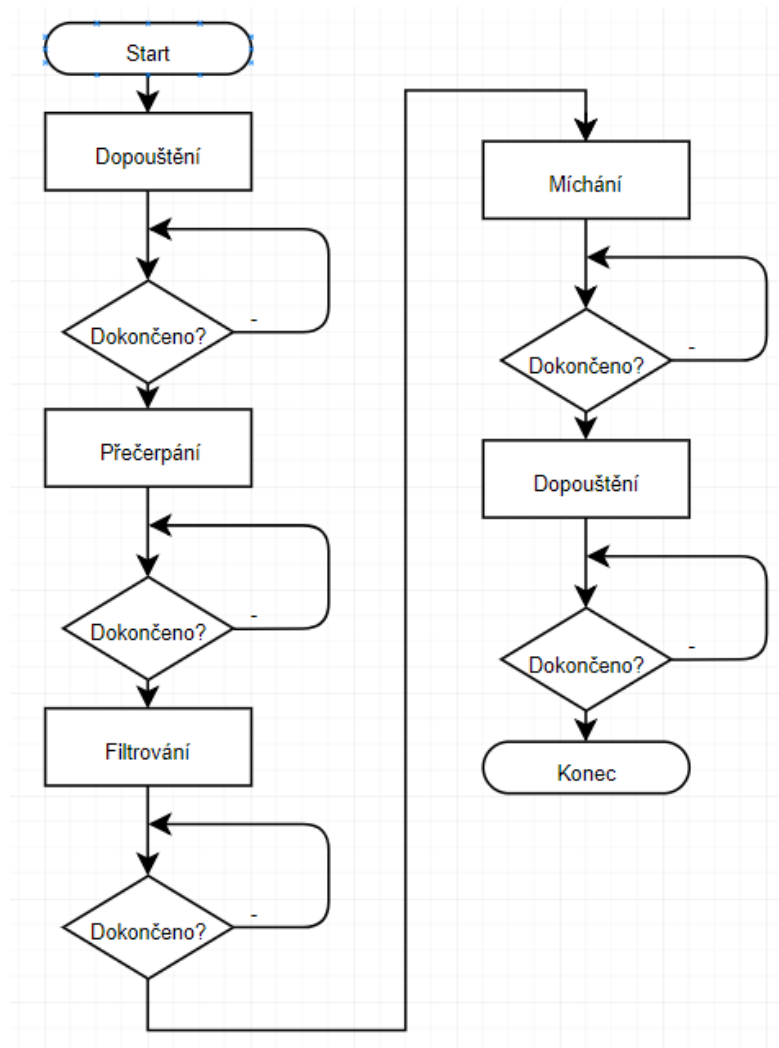
Filtrování se odvíjí pouze od požadované doby, kterou chceme danou kapalinu filtrovat. Kapalina je filtrována pomocí filtru F-01, přes který je hnána pomocí čerpadel P-02. V tomto stavu je potřeba, aby klapka LCV-03 byla otevřená a klapky LCV-02 a LCV-04 byly zavřené. Rychlost filtrování se samozřejmě odvíjí od výkonu čerpadel a je v parametrech simulována jako průtok průtokoměrem FIC-03. Na filtru jsou zároveň měřeny vstupní a výstupní tlak, aby bylo možné detekovat zanesení filtru. Toto zanesení je poté vyneseno do Alarmů, aby operátor dostal okamžitou informaci o problému.

Míchání probíhá v tanku 1.01A a odvíjí se pouze od času, jaký je potřeba danou kapalinu míchat. Míchadlo samozřejmě kontrolují i minimální hladinu, aby se netočilo naprázdno a tím nedošlo k jeho porušení.

Vypouštění tanku 2.01A se odvíjí od požadované hladiny, na kterou se má vypustit. Tento stav vyžaduje přepnutí ventilu LCV-03 do zavřeného stavu a otevření ventilu LCV-04. Kapalina je vypouštěna z tanku 2.01A pomocí čerpadel P-02, kde jejich výkon se dá simulovat pomocí průtoku průtokoměrem FIC-03 v parametrech.

### 6.1.5 Automatický režim

Automatický režim využívá pěti vytvořených poloautomatických režimů, které postupně v řadě za sebou spouští a čeká na jejich dokončení. Při zapnutí automatického režimu tedy začne nejprve dopouštět hladinu tanku 1.01A do požadované výšky. Poté přečerpá kapalinu z tanku 1.01A do tanku 2.01A v závislosti na minimální výšce v tanku 1.01A a požadované hladině v 2.01A (dosáhnutí jedné z nich danou část dokončí). Poté dojde k přepnutí trasy potrubí na filtrování a začne se filtrovat na požadovanou dobu. Po dokončení filtrování následuje míchání. Celý proces končí vypuštěním na požadovanou hladinu. Následně začíná celý proces znovu. Při vypnutí automatického režimu dojde k dojetí sekvence až do úplného konce, tedy po vypuštění tanku 2.01, a vypnutí celého procesu.



Obr. 60 Vývojový diagram automatického režimu

## 6.2 Tvorba HMI

Nejprve jsem vytvořil nový projekt ve vývojovém prostředí s názvem DP a rozlišením 1920x1080. Pote jsem se rozhodl, že vytvořím design následovně. Nahoře na bude lišta s objektem posledního alarmu. Vedle něj bude v pravém horním rohu nad menu umístěné přihlašování. Pod lištou alarmů budou tedy zobrazovány jednotlivé obrazovky a vedle nich bude umístěno menu. Výsledné rozložení je možné vidět na následujícím obrázku.

Díky tomuto designu bude neustále možné sledovat, jaký uživatel je přihlášený a dle potřeby jej změnit. Uživatel také okamžitě uvidí poslední vzniklý alarm, aniž by musel přepnout obrazovku vizualizace pryč z technologie. Díky tomuto nastavení získá obsluha větší přehled o dané situaci, aniž by musela vykonávat nějakou akci ve vizualizaci.



Obr. 61 Design HMI

Pro design bylo také potřeba menu, které se poté zobrazí v jeho sekci. Toto menu jsem vytvořil pomocí objektů PageButton. Každému z těchto tlačítek jsem přidělil název, podle kterého je možné danou stránku přesně identifikovat, takže uživatel ví, co zobrazuje. V menu byly tedy vytvořeny následující tlačítka: Technologie, Parametry, Trendy, Žurnál, Alamy a diagnostika.



**Obr. 62 Menu**

Čerpadla, ventily a míchadlo fungují ve dvou režimech: manuálním a automatickém. V manuálním režimu může obsluha daná zařízení přímo ovládat z HMI. V automatickém režimu jsou zařízení ovládána pomocí jednotlivých technologických celků, které lze pomocí poloautomatického chodu zapínat nebo spustit plně automatickou sekvencí, která automaticky spouští jednotlivé části dle parametrů.

Jednotlivé stavy zařízení jsou signalizovány uvnitř zařízení pomocí barev. Zelená barva u zařízení znamená zapnutý stav a šedá znamená vypnutý stav. Blikající červená barva potom znamená poruchu daného zařízení. Při výpadku komunikace jsou zařízení a stavy obarveny do šeda.

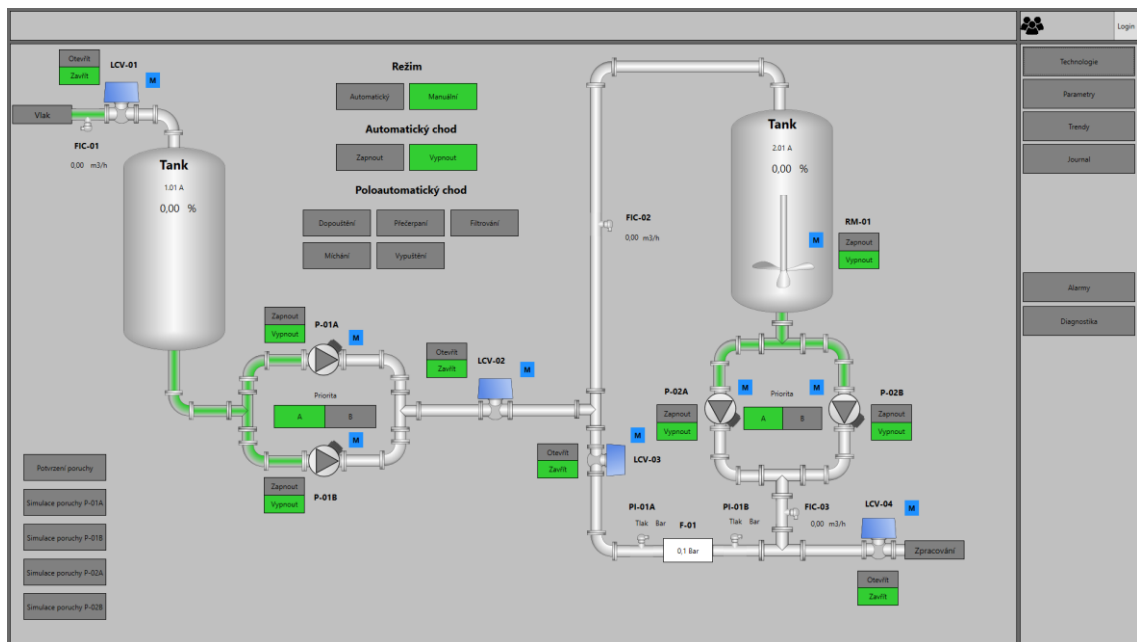
Poruchu zařízení je možné nasimulovat pomocí změny vstupního a výstupního tlaku v parametrech. Tato porucha se objeví pouze v liště alarmů jako poslední alarm a na obrazovce alarmů. Porucha na zařízení se dá nasimulovat pouze na čerpadlech, které fungují v záskokovém režimu. Pokud budeme tedy zkusit vyvolat poruchu čerpadla v automatické chodu, tak dojde k okamžitému najetí záskokového čerpadla. V případě poruchy obou zařízení dojde k pozastavení daného procesu. Poruchu je možné potvrdit pomocí potvrzujícího tlačítka nad tlačítky simulace poruch.

Alarms			
Alarmy			Add AlarmList
Id	Description	IO	AlarmLevel
1	Porucha čerpadla P-01A.	DO1	Error
2	Porucha čerpadla P-01B.	DO2	Error
3	Porucha čerpadla P-02A.	DO3	Error
4	Porucha čerpadla P-02B.	DO4	Error
5	Ucpaný filtr.	DI1	Error

**Obr. 63 Vytvořené alarmy**

Hodnoty jsou zobrazovány pomocí objektu label, který zobrazuje aktuální průtoky, hladiny, stavy a tlaky v technologii. Automatický a manuální režim u každého zařízení je zobrazen právě pomocí labelu ve kterém jsou nadefinovány potřebné texty a barvy. U hladin a průtoků jsou potom zobrazeny měnící se hodnoty, které mají v parametrech nastavenou přesnost na 2 desetinná místa.

Ovládací prvky, které jsou použity také jako signalizující, slouží k manuálnímu ovládání daných zařízení a k ovládání jednotlivých režimů. Pro všechny ovládací prvky je potřeba minimální úroveň operátor pro přihlášeného uživatele. Neautorizovaná osoba tedy nemá právo, jakkoliv zasahovat do technologie, avšak může si prohlížet její stavy a obsah.



**Obr. 64 Technologie ukázkového řešení**

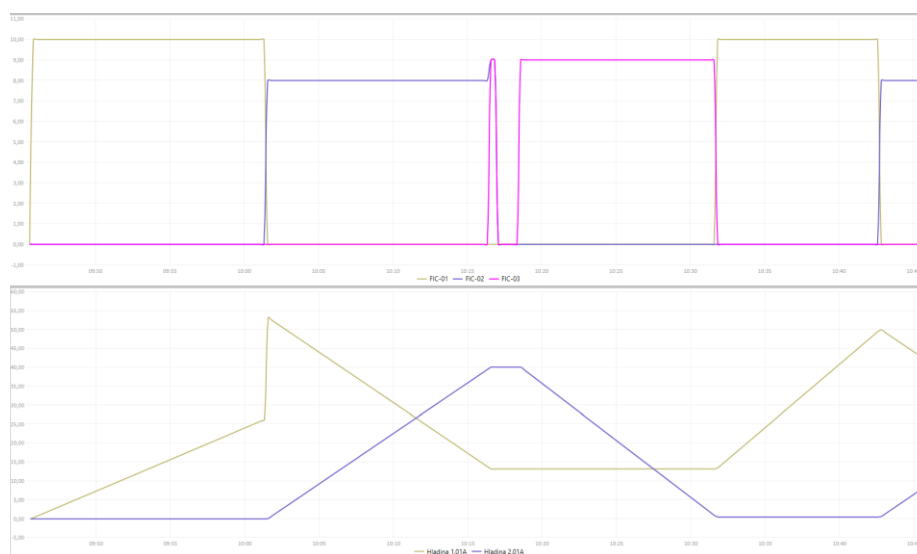
Jak už bylo zmíněno v předchozí kapitole, tak simulace je založená na celé řadě parametrů, bez kterých by automatický režim nemohl fungovat. Proto byla vytvořena obrazovka pro správu těchto parametrů. Minimální úroveň přihlášeného uživatele, který může nastavovat parametry, byla zvolena technik. Ostatní uživatelé s nižší úrovní mohou parametry pouze prohlížet.

V parametrech je možné nastavit objem tanků v  $m^3$ . Dále jsou zde průtoky jednotlivých průtokoměrů, kterým se nastavuje objemový průtok v  $m^3$  za hodinu. Nechybí ani minimální a požadované hladiny tanků při jednotlivých procesech v procentech. Je možné zde nastavit i tlaky před a za filtrem, které pak generují poruchu.

Parametry			
50	Požadovaná hladina doproubění [‰]	+	-
40	Požadovaná hladina přeféčerpání [‰]	+	-
0.5	Minimální hladina 1.01 při přeféčerpání [‰]	+	-
0.3	Minimální hladina 2.01 pro čerpání [‰]	+	-
10	Průtok FIC-01 [m <sup>3</sup> /h]	+	-
5	Průtok FIC-02 [m <sup>3</sup> /h]	+	-
8	Průtok FIC-03 [m <sup>3</sup> /h]	+	-
60	Délka filrování [s]	+	-
120	Délka míchání [s]	+	-
10	Objem tanku 1.01A [m <sup>3</sup> ]	+	-
10	Objem tanku 2.01A [m <sup>3</sup> ]	+	-

**Obr. 65 Parametry ukázkového řešení**

Na obrazovce trendů jsem vynesl průběhy průtoků FIC-01, FIC-02 a FIC-03. Periodu zobrazování dat jsem nastavil na 15 sekund a počet vzorku vyneseny v grafu na 240. Jedná se tedy o hodinový graf s 240 hodnotami. Krok mřížky jsem nastavil na 5 minut. Stejným způsobem jsou do grafu pod průtoky vyneseny hladiny tanků 1.01A a 2.01A. Na grafech jsou krásně vidět závislosti mezi průtoky a hladinami. Trendy jsou neustále aktualizovány novými hodnotami a díky tomu má operátor neustálý přehled o technologii.



**Obr. 66 Obrazovka trendů**

Další obrazovka je věnována žurnálu, kde se zaznamenávají všechny uživatelské a procesní změny. Jsou zde zaznamenány všechny interakce uživatele s vizualizací. Díky tomu je možné sledovat, zkoumat a vyhodnocovat, jak daný operátor postupoval v konkrétní situaci a jestli nezapříčinil daný problém.

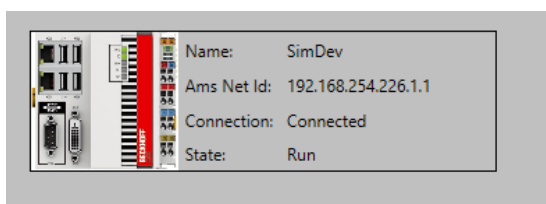
Procesní žurnál je poté takovým úložištěm, kdy nastal začal a skončil daný proces. Tyto informace mohou být vhodné při vyhodnocování, jestli vše funguje, jak má. Jestli se plní jednotlivé kroky včas, případně je možné pomocí něj hledat slabá místa technologie.

299	6/5/2018 10:43:08 PM	Admin	Dokončeno vypouštění nádrže 2.01A.
298	6/5/2018 10:13:18 PM	Admin	Začalo vypouštění nádrže 2.01A.
297	6/5/2018 10:13:18 PM	Admin	Dokončeno míchání nádrže 2.01A.
296	6/5/2018 10:11:18 PM	Admin	Začalo míchání nádrže 2.01A.
295	6/5/2018 10:11:18 PM	Admin	Dokončeno filtrování v nádrži 2.01A.
294	6/5/2018 10:10:18 PM	Admin	Začalo filtrování v nádrži 2.01A.
293	6/5/2018 10:10:18 PM	Admin	Dokončeno předčerpání z nádrže 1.01A do nádrže 2.01A.
292	6/5/2018 9:22:36 PM	Admin	Začalo předčerpání z nádrže 1.01A do nádrže 2.01A.
291	6/5/2018 9:22:35 PM	Admin	Dokončeno dopouštění nádrže 1.01A.
290	6/5/2018 8:58:49 PM	Admin	Začalo dopouštění nádrže 1.01A.
289	6/5/2018 8:58:49 PM	Admin	Dokončeno vypouštění nádrže 2.01A.
288	6/5/2018 8:29:00 PM	Admin	Začalo vypouštění nádrže 2.01A.
287	6/5/2018 8:29:00 PM	Admin	Dokončeno míchání nádrže 2.01A.
286	6/5/2018 8:27:00 PM	Admin	Začalo míchání nádrže 2.01A.
285	6/5/2018 8:26:59 PM	Admin	Dokončeno filtrování v nádrži 2.01A.
284	6/5/2018 8:25:59 PM	Admin	Začalo filtrování v nádrži 2.01A.
283	6/5/2018 8:25:59 PM	Admin	Dokončeno předčerpání z nádrže 1.01A do nádrže 2.01A.
282	6/5/2018 7:38:17 PM	Admin	Začalo předčerpání z nádrže 1.01A do nádrže 2.01A.
281	6/5/2018 7:38:17 PM	Admin	Dokončeno dopouštění nádrže 1.01A.
280	6/5/2018 7:14:31 PM	Admin	Začalo dopouštění nádrže 1.01A.
279	6/5/2018 7:14:31 PM	Admin	Dokončeno vypouštění nádrže 2.01A.
278	6/5/2018 6:44:41 PM	Admin	Začalo vypouštění nádrže 2.01A.
277	6/5/2018 6:44:41 PM	Admin	Dokončeno míchání nádrže 2.01A.
276	6/5/2018 6:42:41 PM	Admin	Začalo míchání nádrže 2.01A.
275	6/5/2018 6:42:41 PM	Admin	Dokončeno filtrování v nádrži 2.01A.
274	6/5/2018 6:41:41 PM	Admin	Začalo filtrování v nádrži 2.01A.
273	6/5/2018 6:41:40 PM	Admin	Dokončeno předčerpání z nádrže 1.01A do nádrže 2.01A.
272	6/5/2018 5:53:59 PM	Admin	Začalo předčerpání z nádrže 1.01A do nádrže 2.01A.
271	6/5/2018 5:53:58 PM	Admin	Dokončeno dopouštění nádrže 1.01A.
270	6/5/2018 5:30:12 PM	Admin	Dokončeno vypouštění nádrže 2.01A.
269	6/5/2018 5:30:12 PM	Admin	Začalo dopouštění nádrže 1.01A.
268	6/5/2018 5:00:23 PM	Admin	Začalo vypouštění nádrže 2.01A.
267	6/5/2018 5:00:23 PM	Admin	Dokončeno míchání nádrže 2.01A.
266	6/5/2018 4:58:23 PM	Admin	Začalo míchání nádrže 2.01A.
265	6/5/2018 4:58:22 PM	Admin	Dokončeno filtrování v nádrži 2.01A.
264	6/5/2018 4:57:23 PM	Admin	Začalo filtrování v nádrži 2.01A.
263	6/5/2018 4:57:22 PM	Admin	Dokončeno předčerpání z nádrže 1.01A do nádrže 2.01A.
262	6/5/2018 4:09:40 PM	Admin	Začalo předčerpání z nádrže 1.01A do nádrže 2.01A.

**Obr. 67 Ukázka procesního žurnálu automatického režimu**

V obrazovce alarmů jsou zobrazeny všechny aktuálně trvající alarmy, takže obsluha má úplný přehled o situaci a stavu technologie. Alarmy z listu zmizí po jejich potvrzení obsluhou, pokud už dále netrvají. Pokud alarm dále pokračuje, tak potvrzení obsluhy nemá žádný vliv a alarm je zobrazen dále. Operátor díky jejich rozlišení na informační, varovný a poruchový dokáže správně a okamžitě reagovat na danou situaci.

Obrazovka diagnostiky uchovává pouze objekt o stavu připojeného zařízení. Zobrazuje jeho název, adresu a stav. Pokud dojde ke ztrátě spojení se zařízením nebo změně jeho stavu (přepnutí zařízení z módu RUN do STOP), tak se okamžitě projeví na tomto objektu. V reálném projektu by mohlo být zajímavé umístit zde například všechny vstupy a výstupy ADS zařízení. Poté v případě nestandardního chování, případně poruše by bylo jednodušší detekovat a odstranit problém, protože na vizualizaci by byli přítomny všechny potřebné informace i pro oddělení MaR.



**Obr. 68 Diagnostika**

## 7. ZÁVĚR

V diplomové práci jsem se zabýval tvorbou HMI pro BECKHOFF TWINCAT 3 v programovacím jazyce C# pomocí grafického subsystému WPF. Vytvořil jsem vývojové prostředí pro tvorbu HMI pomocí předdefinovaných objektů, které lze snadno parametrizovat. Dále jsme vytvořil runtime, který spouští vytvořené HMI ve vývojovém prostředí. Použití, mnou vytvořených aplikací, bylo předvedeno na ukázkovém řešení.

Druhá kapitola práce se zabývá .NET frameworkem a jeho možnostmi. Jsou zde popsány všechny teoretické informace potřebné k tvorbě HMI. Jsou zde popsány také všechny jazyky, které byli při programování HMI použity. Popsán je zde i návrhový vzor, který je standardně používaný pro WPF aplikace.

Třetí kapitola je věnována softwarovým prostředkům, které byli použity při tvorbě HMI. Je zde popsáno vývojové prostředí Visual Studio od firmy Microsoft, ve kterém byla celá HMI vytvořena. Dále je zde popsána společnost Beckhoff s jejich nástrojem TwinCAT 3, který byl hlavním předmětem této práce.

Ve čtvrté kapitole se věnuji samotnému HMI. Je zde detailně popsána struktura a funkce vývojového prostředí. Jsou zde vysvětleny funkce a možnosti jednotlivých modulů vývojového prostředí spolu se sdílenými službami, které se starají o komunikaci dat. Dále následuje popsání struktury a funkce runtime HMI.

V páté kapitole je poté vytvořeno ukázkové řešení technologie. Je zde popsán postup, jakým byla ukázková technologie vytvořena. První část této kapitoly se věnuje vysvětlením, jak funguje vytvořený program v simulovaném PLC v software TwinCAT 3. Poté je zde popsán postup tvorby HMI ve vývojovém prostředí, které bylo vytvořeno.

Výsledkem této práce je tedy plně funkční vývojové prostředí a runtime pro HMI, který obsahuje všechny objekty specifikované v zadání práce. Aplikace je odzkoušena ve dvoudenním provozu, kde nevykazovala žádné známky nestability.

Aplikaci je možné libovolně rozšiřovat o další objekty a tím rozšířit možnosti použití na další odvětví. V případě použití při vývoji HMI na konkrétní stroj je možné do vizualizace doplnit libovolný 3D objekt daného stroje, případně robota a vizualizovat přímo jeho pohyby.

Diplomová práce byla zaměřena pouze na zařízení Beckhoff obsahující TwinCAT 3. Je však možné aplikaci jednoduše doplnit o API na nějaký jiný systém, případně na některý z používaných OPC serverů a tím bude HMI možné použít s libovolným zařízením.

V současné době aplikace využívá SQLite pro řešení projektů bez serverového řešení SQL. Zde je možné aplikaci také dále rozšiřovat a nabídnout poté zákazníkovi například napojení na jeho MSSQL/Oracle případně jiný databázový systém.

V mnoha firmách v dnešní době začíná být také populární se všude přihlašovat pomocí čtečky karet. Tuto vlastnost lze také velice jednoduše doplnit při správném zvolení čteček karet. Stačí jen spárovat kód karty s uživatelem a při přiložení karty ke čtečce karet jen porovnat kód karty s uloženými kódy karet u uživatelů.

Otázkou však zůstává, jakým směrem se budou vizualizace vyvíjet v následujících letech. Industrie 4, o kterém se v dnešní době mluví, nám přináší různé změny v propojení a komunikacích mezi zařízeními. Existuje velký nápor Cloudu a IOT. Firmy jako SAP a Microsoft se začínají svými řešeními ERP systému tlačit dolů v pyramidě. Je tedy otázka, jestli nakonec nepohltí vše a nevzniknou velké systémy, které budou umět vše od ERP po HMI.

# Literatura

- [1] The .NET Framework stack [online]. CC BY-SA 3.0, 2018 [cit. 2018-05-10].  
Dostupné z:  
[https://en.wikipedia.org/wiki/.NET\\_Framework#/media/File:DotNet.svg](https://en.wikipedia.org/wiki/.NET_Framework#/media/File:DotNet.svg)
- [2] Scriptol [online]. [cit. 2018-05-10]. Dostupné z:  
<https://www.scriptol.com/programming/cil.php>
- [3] Social Club [online]. 2014 [cit. 2018-05-10]. Dostupné z:  
<https://www.codeproject.com/Articles/607868/Social-Club-Sample-application-using-WinForms-Csha>
- [4] Telerik UI [online]. web [cit. 2018-05-10]. Dostupné z:  
<https://www.telerik.com/wpf/sample-applications>
- [5] Model-view-viewmodel [online]. 2009 [cit. 2018-05-10]. Dostupné z:  
<https://www.dotnetportal.cz/clanek/4994/MVVM-Model-View-ViewModel>
- [6] The ViewModel Pattern [online]. Microsoft, 2008 [cit. 2018-05-10]. Dostupné z:  
<https://blogs.msdn.microsoft.com/dphill/2009/01/31/the-viewmodel-pattern/>
- [7] Introduction to the Prism Library 5.0 for WPF [online]. Microsoft [cit. 2018-05-10]. Dostupné z:  
[https://msdn.microsoft.com/enus/library/ff921153\(v=pandp.40\).aspx](https://msdn.microsoft.com/enus/library/ff921153(v=pandp.40).aspx)
- [8] Beckhoff. In: Beckhoff Polska [online]. [cit. 2018-01-02]. Dostupné z:  
<https://cptrade.pl/wp-content/uploads/2016/06/Beckhoff-1-412x300.jpg>
- [9] BECKHOFF. Beckhoff [online]. [cit. 2.1.2018]. Dostupný na WWW:  
<https://www.beckhoff.com/images/twincat/popup/twincat-3-extendedautomation-engineering.png>
- [10] TwinCAT 3 – eXtended Automation Technology [online]. 2015 [cit. 2018-01-02].  
Dostupné z: [http://automa.cz/Aton/FileRepository/pdf\\_articles/53801.pdf](http://automa.cz/Aton/FileRepository/pdf_articles/53801.pdf)
- [11] BECKHOFF. Beckhoff [online]. [cit. 2.1.2018]. Dostupný na WWW:  
<https://www.beckhoff.com/images/twincat/popup/twincat-3-extendedautomation-performance.png>
- [12] BECKHOFF. Beckhoff [online]. [cit. 2.1.2018]. Dostupný na WWW:  
<https://infosys.beckhoff.com/content/1033/tcadscommon/images/TcAdsDevConcept.jpg>

## Seznam příloh

Příloha 1 - Zdrojový kód vývojového prostředí a runtime HMI je uložen na příloženém DVD.....	67
Příloha 2 - Zdrojový kód PLC programu je uložen na příloženém DVD.....	67
Příloha 3 - Instalační balíček HMI je uložen na příloženém DVD.....	67
Příloha 4 - Dokumentace zdrojového kódu vývojového prostředí a runtime je uložena na příloženém DVD.....	67

**Příloha 1 - Zdrojový kód vývojového prostředí a runtime HMI je uložen na přiloženém DVD**

**Příloha 2 - Zdrojový kód PLC programu je uložen na přiloženém DVD**

**Příloha 3 - Instalační balíček HMI je uložen na přiloženém DVD**

**Příloha 4 - Dokumentace zdrojového kódu vývojového prostředí a runtime je uložena na přiloženém DVD**

**Příloha 5 - Ukázkové řešení**