Univerzita Hradec Králové

Fakulta informatiky a managementu

Katedra informačních technologií

C# Aplikace pro konfiguraci bezpečnostního systému

Bakalářská práce

Autor: Jakub Šmída
Studijní obor: AI3-p

Vedoucí práce: Ing. Jaroslav Langer

Odborný konzultant: Bc. Jiří Šrámek

STARMON s.r.o.

Hradec Králové Únor 2022

Prohlášení:

Prohlašuji, že jsem bakalářskou práci zpracoval samostatně a s použitím uvedené literatury.

vlastnoruční podpis

V Hradci Králové dne 5.4.2022 Jakub Šmída

Poděkování:

Děkuji vedoucímu bakalářské práce Ing. Jaroslavu Langerovi za metodické vedení práce, odborné rady, věcné připomínky, vstřícný přístup a vše, co mi pomohlo při zpracování bakalářské práce.

Anotace

Název: C# Aplikace pro konfiguraci bezpečnostního systému

Práce se zabývá představením technologií elektronických zabezpečovacích zařízení, která je určena k zabezpečení objektů a staveb. Práce popisuje různé možnosti a způsoby konfigurace a zaměřuje se na tvorbu desktopové aplikace pro konfiguraci zabezpečovacího systému StarAlarm.

Obsahem práce je popis vývoje desktopové aplikace a k tomu použitých technologií. Aplikace je vyvíjena na platformě .NET Framework, za použití knihoven WinForms. Další použitou technologií je například protokol UPD, který je použit ke komunikaci mezi aplikací a ústřednou EZS.

V práci je také popsána aplikace a návod na její použití v rámci tvorby konfigurace a její odeslání do ústředny.

Annotation

Title: C# Application for configuration of security system

The work deals with the introduction of technologies of electronic security devices, which are used to secure buildings and structures. The thesis describes various options and methods of configuration and is part of the creation of a desktop application for the configuration of the StarAlarm security system.

The content of the work is a description of the development of a desktop application and the technologies used. The application is developed on the .NET Framework platform, using the WinForms libraries. Another technology used is, for example, the UPD protocol, which is used for communication between applications and the security alarm processor.

The work also describes the application and instructions for its use in the configuration and its sending to the processor.

Obsah

Seznam obrázků

Seznam ukázek konfigurace

Seznam ukázek kódu

Seznam zkratek

EZS ..... Elektronický zabezpečovací signalizace

GSM ..... Groupe Spécial Mobile

PCO ..... Pult centralizované ochrany

GC ..... Garbage collector

EZS ..... Elektronický zabezpečovací signalizace

IoT ..... Internet of things

PIR ..... Passive infrared sensor

API ..... Application Programming Interface

WPF ..... Windows Presentation Foundation

ISO ..... International Organization for Standardization

LAN ..... Local Area Network

IP ..... Internet Protocol

UDP ..... User Datagram Protocol

DNS ..... Domain Name System

1 Úvod

Konfigurace alarmového systému je nutnou částí jeho instalace a správy. Některé systémy používají ke konfiguraci webová rozhraní, klávesnice, nebo desktopové aplikace, které komunikují s ústřednou pomocí ethernetu. Velmi záleží na účelu bezpečnostního zařízení. Existují spousty systémů určených pro běžné uživatele, kteří si například chtějí zabezpečit rodinný dům. V takovém případě je důležité, aby byla konfigurace co nejjednodušší a intuitivní. Ale v případě složitějších systémů určených pro průmyslové použití je vyžadována odborná instalace a není možné zaručit jednoduchost. Systém StarAlarm 2.0 je určen pro použití v průmyslu a jeho konfigurace je prováděna zaměstnanci, kteří musí umět systém správně nakonfigurovat. Pro konfiguraci je používána konfigurace, která je následně přeložena externí aplikací do binárního kódu a pomocí dalšího softwaru nahrána do ústředny alarmového systému. Tento proces je zdlouhavý a vyžaduje, aby zaměstnanec přesně znal konstrukci textové konfigurace a všechny postupy zápisu. Z toho důvodu byl vznesen požadavek na vytvoření konfiguračního software, který tento proces usnadní a sjednotí.

Tato práce popisuje aplikaci, která spojuje všechny používané aplikace a rozšiřuje je o funkcionality umožňující jednoduchou práci se systémem StarAlarm. Pro lepší přehled v konfiguraci bude zobrazována, kromě textové podoby, také graficky, tak aby byla pro uživatele přehledná a umožňovala mu snadno provádět změny jednotlivých částí, bez nutnosti otevřít textový editor. Kromě toho umožní zaměstnanci přímo v aplikaci konfiguraci přeložit a odeslat do ústředny pomocí zabudovaného komunikačního rozhraní. Jako další vlastnost aplikace je možnost stažení konfigurace z paměti ústředny pomocí UDP protokolu, aby nebylo nutné si konfiguraci ukládat lokálně. Pro obsažení všech funkcí pro správu, je také implementována možnost vyčtení, úpravy a nahrání systému přístupových karet, které slouží k ovládání systému.

2 Cíl práce

Na začátku této práce autor popíše technologie využívané v praktické části. Cílem práce je popsat technologie jazyka C# a bezpečnostního systému, pro který je určena výsledná aplikace. Aplikace musí načíst a uložit textovou konfiguraci, zobrazit grafický návrh konfigurace a přeložit konfiguraci do binárního kódu. Dále musí být schopna odeslat binární konfiguraci do alarmového systému pomocí UDP protokolu.

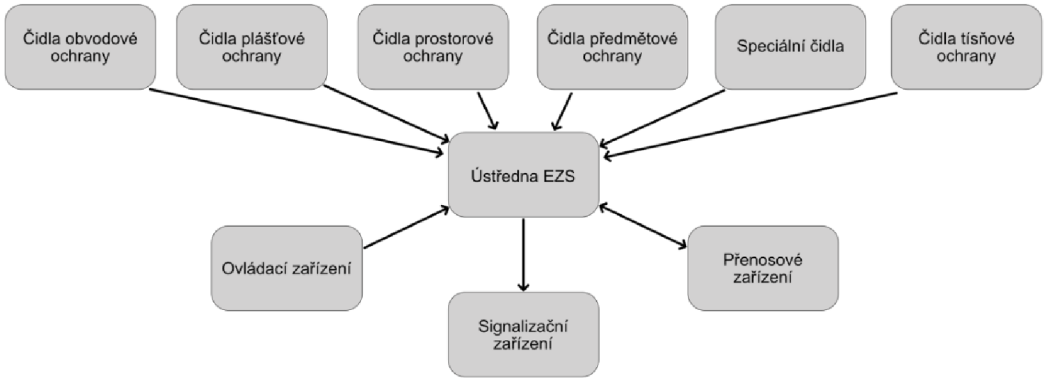
Cílem této bakalářské práce je vytvořit funkční aplikaci pro využití v praxi.

3 Teoretické zpracování

V této části bakalářské práce se autor zabývá teoretickou částí zvoleného tématu. Dojde zde k vysvětlení pojmů a popisu technologií pojící se s tématem.

3.1 Elektronická zabezpečovací signalizace

Definice prvků EZS je dána evropskou normou EN ČSN 50 131-1. Centrem bezpečnostního systému je poplachová ústředna. Jejím úkolem je vyhodnocovat stavy připojených zařízení a v případě poplachu zareagovat definovaným způsobem. Nejčastěji spuštěním optické a zvukové signalizace, ale ústředna může být vybavena GSM modulem, připojením k ethernetu nebo pultem centralizované ochrany neboli PCO. Což je zpoplatněná služba, obsahující i odborný zásah a kontrolu objektu. Možností zapojení systémů EZS je několik. Následující obrázek zobrazuje přímé propojení všech čidel s ústřednou. (1)



Obrázek 1 - Obecné blokové schéma EZS, převzato (1) a upraveno

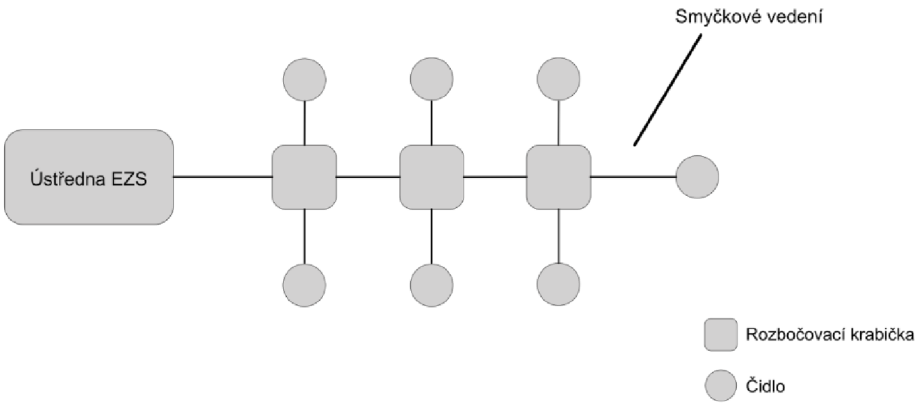
3.1.1 Poplachová ústředna

Poplachová ústředna je základní částí bezpečnostního systému, která musí být konfigurovatelná, aby se mohla použít v různých prostředích a objektech. Způsobů konfigurace je několik. Jedním ze způsobů je konfigurace pomocí ovládací klávesnice. Tento přístup ke konfiguraci je poměrně komplikovaný a zdlouhavý. Mnohem lepší je konfigurace pomocí speciálního softwaru, nebo webového rozhraní. (1)

3.1.1.1 Smyčková ústředna

Smyčková ústředna má většinou vstupy, které jsou pro více zabezpečovacích smyček. U tohoto typu ústředny je vstupní vyhodnocovací obvod obvykle řešen jako dělič napětí s komparátorem nebo také jako vyvážený měřící můstek. Většinou je možné zapojit hodně zabezpečovacích smyček, pro které je vyhodnocovací obvod vždy samostatný. Jde o proudovou smyčku, kde je obvykle zátěž v podobě zatěžovacího odporu. A nejjednodušeji jsou do této smyčky v sérii zapojeny sabotážní i poplachové kontakty každého čidla, které je do smyčky připojené. Odpor smyčky vyhodnocuje ústředna.

Jak z výše uvedeného vypovídá, čidla, která jsou určena pro tyto smyčkové ústředny, musí mít rozpínací kontakt na výstupu. Součástí tohoto kontaktu je obvykle elektromagnetické relé. V klidovém stavu je relé sepnuté, po aktivaci čidla dojde k rozepnutí. Tím se také rozpojí proudová smyčka. Velkou nevýhodou této ústředny je velké množství kabelů, kvůli zapojení každého čidla. Každé čidlo totiž musí být samostatně připojeno přes rozbočovací krabičku k vyhodnocovacímu obvodu dané smyčky. (2)



Obrázek 2 - Blokové schéma zapojení smyčkové ústředny, převzato (2) a upraveno

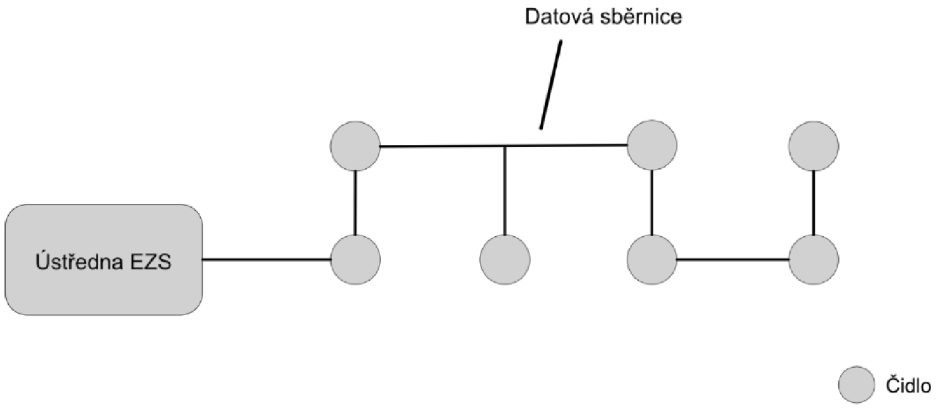
3.1.1.2 Ústředna s přímou adresací čidel

V případě ústředny s přímou adresací čidel komunikaci mezi poplachovou ústřednou zprostředkovává datová sběrnice. Ta se skládá ze čtyř vodičů, dva z nich slouží k napájení jednotlivých čidel a zbylé vodiče jsou součástí datové sběrnice.

Komunikace polo duplexní a master-slave orientovaná. Ústřednu tvoří zařízení master a čidla jsou zařízení slave. Komunikace probíhá ve směru ústředna-čidla. Každé čidlo je připojeno na sběrnici a má svou specifickou adresu. Komunikace se odehrává tak, že poplachová ústředna vytváří adresy dílčích čidel, které jsou připojené na sběrnici a diagnostikuje jejich stavy.

Největší předností tohoto systému je snadná rozpoznatelnost místa čidla, které se aktivovalo. A to díky tomu, že u čidla, které se aktivovalo, je vždy známá adresa čidla a okolnost, která vyvolala změnu.

Tento druh ústředny je tedy nejvhodnější pro rozsáhlé objekty, díky okamžitému a přesnému určení místa aktivního čidla. Právě toto není možné u smyčkových ústředen. Při aktivaci čidla bychom znali pouze smyčku, ve které se čidlo nachází. Jenže na takové smyčce může být čidel více. Ústředny s přímou adresací však mají na sběrnici omezený počet čidel. Je to kvůli úbytku napětí na napájecích vodičích. Obvykle na jednu sběrnici umístit až desítky čidel při celkové délce vedení v řádu až stovek metrů. (3)



Obrázek 3 - Blokové schéma zapojení ústředny s přímou adresací čidel, převzato (3) a upraveno

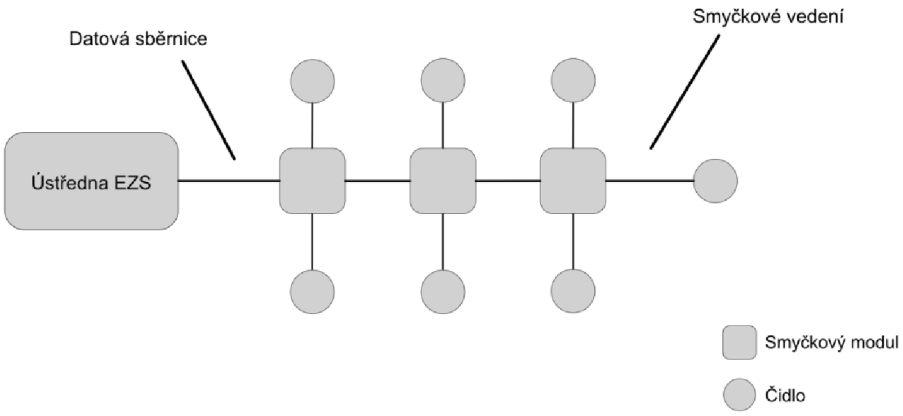
3.1.1.3 Ústředna smíšeného typu

Tento typ ústředny je velmi podobný předchozímu typu. Rozdíl je ve větších možnostech délky kabelového vedení, které nemá tak velké omezení.

Komunikace u tohoto typu ústředny se odehrává po analogové nebo digitální sběrnici. Odehrává se mezi ústřednou a tzv. smyčkovým modulem (koncentrátorem). Na tento modul jsou pak pomocí proudových smyček připojena dílčí čidla. Vyhodnocení signálu z dílčích čidel se provádí dvěma způsoby.

Prvním ze způsobů je koncentrátor a analogovým multiplexem. Multiplex jednotlivě přepíná smyčky, které jsou připojeny na koncentrátor. Odpor smyček je pomocí analogové sběrnice vyhodnocen v poplachové ústředně.

Druhým způsobem je vyhodnocení odporu smyčky v koncentrátoru. V druhé možnosti vyhodnocení signálu probíhá komunikace s ústřednou v datové podobě. Celková délka datového vedení může být až 1 km. Tuto délku je možné markantně zvětšit tím, že použijeme opakovač mezi ústřednu a koncentrátor. Velká výhoda je použití totožných druhů čidel s reléovým výstupem. Tato čidla jsou významně levnější než čidla bezdrátová případně čidla s přímou adresací. (4)



Obrázek 4 - Blokové schéma zapojení ústředny smíšeného typu, převzato (3) a upraveno

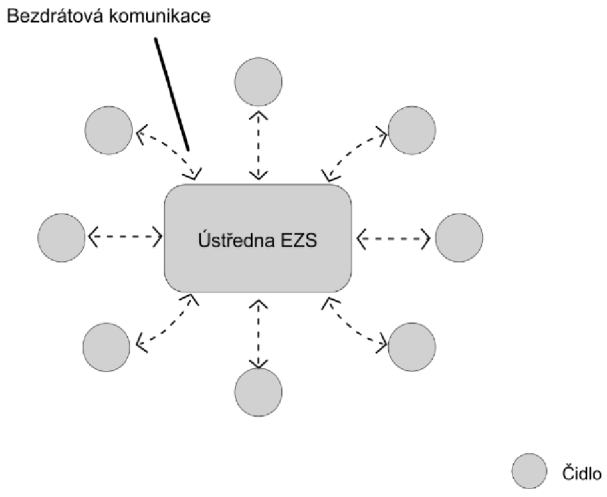
3.1.1.4 Ústředna s bezdrátovým přenosem

Tato ústředna, ke komunikaci s dalšími prvky nepotřebuje žádné kabelové vedení mimo napájení. Komunikace této ústředny se obvykle odehrává v kmitočtovém pásmu 433 MHz nebo 868 MHz. Komunikace může být duplexní nebo polo duplexní. Při polo duplexní komunikaci smí komunikovat jen čidlo případně jiná součást systému s poplachovou ústřednou. Nevýhodou této možnosti je neschopnost ústředny kontrolovat, zda nedošlo k závadě nebo záměrnému poškození některého

dalšího člena systému. Z tohoto hlediska je značně bezpečnější duplexní přenos, u kterého je komunikace obousměrná. Ústředna tedy může sledovat chod všech členů pomocí kontrolních zpráv, které jsou generovány buď periodicky, anebo náhodně. Na kontrolní zprávy musí jednotlivý členové odpovídat, tím je možné zajistit správnou funkci každého zařízení či odhalení eventuální poruchy nebo úmyslného poškození.

Protože je tento systém bezdrátový, každé zařízení musí mít bateriové napájení. Stav baterií tedy je nutno měřit samotným zařízením. Detekci nutnosti náhrady baterie lze provádět buď komunikací s poplachovou ústřednou, anebo místní akustickou signalizací. Jde o zabezpečovací systém, je tedy nutnost rádiovou komunikaci šifrovat a zabezpečit proti rušení. Pokud je současný kanál rušen, novodobé ústředny mají funkci automatického přeladění rádiové komunikace na odlišný kanál, tím zajišťují bezpečnou komunikaci a funkční činnost celého systému.

Tyto poplachové ústředny mají velkou výhodu. Lze je totiž dodatečně namontovat do zrekonstruovaných budov. Při takovéto dodatečné montáži jsou instalační zásahy minimální. Pokud se ale jedná o novostavbu, je výhodnější se zabezpečovací technikou již dopředu počítat. Použitím např. smyčkových ústředen se navíc minimalizují finanční náklady, bezdrátová zařízení jsou totiž mnohem nákladnější než zařízení drátová. (4)



Obrázek 5 - Blokové schéma komunikace ústředny s bezdrátovým přenosem, převzato (3) a upraveno

3.1.2 Prvky zabezpečovacího zařízení

Prvky zabezpečovacího zařízení rozdělujeme podle jejich funkce a umístění.

3.1.2.1 Prvky plášťové ochrany

Prvky plášťové ochrany jsou částí bezpečnostního systému, která přichází jako první do styku s narušitelem objektu nebo prostoru, který je chráněn. Do určité míry mohou mít i preventivní efekt na útočníka, ale také jsou nejčastěji obětmi pokusů o sabotáž. Je efektivní kombinovat je s jiným typem ochrany. (3)

3.1.2.1.1 Magnetické kontakty

Tento snímací prvek je vždy tvořen dvěma díly. Jedním je permanentní magnet, který je tvořen, zmagnetizovaným válečkem z feritu a je vždy připevněn k pohyblivé části. Druhým dílem je jazýčkový kontakt tvořený skleněnou trubičkou s dvěma feromagnetickými kontakty. V klidovém stavu je magnet blízko jazýčkového kontaktu a ten je v sepnutém stavu. Při oddálení magnetu se kontakt rozepne. (5)

3.1.2.2 Prvky prostorové ochrany

Jsou nejčastěji používaný na místech s častým pohybem, například na chodbách a schodištích. Rozdělují se na aktivní a pasivní. Aktivní samy skenují svoje okolí a vytváří prostředí v němž detekují změny, oproti tomu pasivní prvky pouze registrují změny ve fyzikálním prostředí. (6)

3.1.2.2.1 Pasivní infračervené čidlo (PIR)

Tyto čidla pracují na principu zachycení změny vlnové dílky v infračerveném spektru. Čidlo rozdělí oblast, kterou hlídá do zón a v každé zóně měří hodnotu infračerveného záření, která je v klidovém stavu konstantní. Ve chvílí, kdy do sledovaného prostoru vstoupí osoba nebo jiný živočich vyzařující teplo, změní se hodnota infračerveného záření mezi zónami a čidlo vyhlásí poplach. Nevýhodou tohoto čidla je citlivost na zdroje tepla a světelné rušení. (7)

3.1.2.2.2 Mikrovlnné čidlo

Čidlo na mikrovlnném principu používá ke své činnosti dopplerovský radar k detekci pohybujících se objektů. Princip fungování čidla je na základě odražených mikrovln. V případě, že je prostředí konstantní, nic se v něm nepohybuje, interference vyslaných a odražených vln je také konstantní. Pokud se v prostředí snímaném mikrovlnným čidlem vyskytne pohybující se předmět, změní se interferenční frekvence a čidlo vyhlásí poplach. (8)

Výhodou tohoto čidla oproti infračervenému je schopnost detekovat neživé předměty. Také je nezávislé na vnějších vlivech jako jsou, teplota, hluk nebo světelné záření. Záření vyzařované čidlem není pro člověka škodlivé, ale může být nebezpečné pro menší živočichy.

3.1.2.3 Ovládací a indikační prvky

Umožňují ovládání stavu bezpečnostního zařízení a předání informace o jeho stavu. K tomuto účelu může sloužit například jednoduchá klávesnice, čtečka čipů nebo přístupový terminál. Některé jednodušší systémy mohou být ovládány jednoduchým radiofrekvenčním ovládáním. Nejdůležitější možnost ovládání je stav systému. Tedy takzvané jeho odjištění.

Jako indikační zařízení mohou sloužit například diody, které vyjadřují stav, různé druhy narušení a poplachu. Komplexnějším způsobem zobrazení mohou být různé typy displejů. (5)

Obrázek 6 - Kódová klávesnice, (27)

3.2 Programovací jazyk C#

Programovací jazyk C# je vysokoúrovňový a objektově orientovaný jazyk. Vyvinula jej firma Microsoft na základě jazyků C++ a Java. Spolu s tímto jazykem přišla i platforma .NET Framework[1]. Jazyk byl schválen standardizačními komisemi ECMA a ISO, konkrétně to jsou normy ECMA-334 a ISO/IEC 23270. C# má velice podobnou syntaxi s jazyky Java a C++, proto je velice jednoduché přejít z těchto jazyků na C#. Hlavní využití najde při programování desktopových, mobilních nebo webových aplikací s napojením na databázové systémy. (9)

```
using System;
namespace HelloPeople
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Ahoj lidi.");
            Console.WriteLine("Dnes je krásný den.");
        }
    }
}
```

Ukázka kódu 1 - Ukázka jazyka C#, vlastní implementace

3.2.1 Porovnání C# a C++

C# je v porovnání s C++ mnohem jednodušší vzhledem k zápisu a syntaxi, ale také mnohem robustnější. Navíc obsahuje takzvaný garbage collector, který automaticky vyhledává a uvolňuje již nepotřebné části paměti, také zabraňuje některým běhovým chybám, například úniku paměti (v angličtině memory leak). Oproti tomu C++ má složitější zápis, syntaxi a ani neobsahuje GC. To znamená, že o správu paměti se musí starat programátor. Oba jazyky jsou objektové, umožňují použití

[1] Softwarová struktura, sloužící jako podpora programování, vývoje a organizace jiných softwarových projektů. (25)

objektových vlastností jako je dědičnost, polymorfismus, interface a další. C# umožňuje vícenásobnou dědičnost pouze u rozhraní. V C# nelze použít proměnnou před její inicializací. Také není možné použít ukazatele mimo „nebezpečný kód“. (10)

3.2.2 .NET

Vývojová platforma .NET je Open Source prostředí pro tvorbu různých aplikací, například (11):

- Webové aplikace
- Cloudové aplikace
- Mobilní aplikace
- Desktopové aplikace
- Hry
- IoT (Internet of Things)
- Strojové učení
- Konzolové aplikace
- a další…

V .NET lze vyvíjet pro různé platformy od Windows až po iOS a také různé typy procesorů (x64, x86, ARM32, ARM64). U této technologie je možné používat funkce, které jsou specifické pro danou platformu, například API operačního systému. Skvělým příkladem jsou WinForms a WPF u operačního systému Windows.

Pro platformu .NET lze vyvíjet programy ve třech jazycích. Konkrétně C#, F# a VisualBasic. Všechny tyto jazyky jsou integrovány v sadě Visual Studio. (11)

3.2.3 Windows Forms

Framework Windows Forms (zkráceně WinForms) je framework uživatelského rozhraní, který umožňuje tvorbu desktopových aplikací pro operační systémy Windows. Jeho hlavní výhodou je jednoduchost v tvorbě formulářů pomocí drag and drop designeru. Součástí frameworku jsou komponenty pro tvorbu formulářů, jako

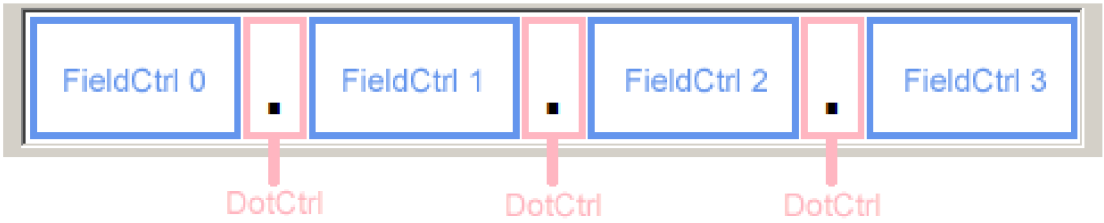
jsou textová pole, popisky, tlačítka, panely pro uspořádání komponent nebo komponenty pro zobrazení dat z databáze. (12)

3.2.3.1 Komponenty

Ovládací prvky Windows Forms jsou opakovaně použitelné součásti, které zapouzdřují funkce uživatelského rozhraní a používají se v aplikacích založených na systému Windows na straně klienta. Windows Forms poskytuje nejen mnoho ovládacích prvků připravených k použití, ale poskytuje také infrastrukturu pro vývoj vlastních ovládacích prvků. Stávající ovládací prvky je možné kombinovat, rozšiřovat nebo vytvářet vlastní ovládací prvky. (13)

3.2.3.1.1 IpAddressControl

V sadě nástrojů WinForms chybí jakákoli možnost pro zápis IP adresy. Je sice možné použít pro zápis IP adresy textové pole, ale je nutné ošetřit chybný vstup. Tato komponenta rozšiřuje obyčejné textové pole pro pohodlné zapsání IP adresy a neumožňuje zapsat jiné znaky. (14)



Obrázek 7 - Ukázka IpAddressControl, převzato (14)

3.2.3.1.2 ComboCheckBox

Komponenta, která spojuje dvě základní komponenty WinForms - ComboBox a CheckBox. Umožňuje tedy úsporu místa při tvorbě formuláře. (15)

3.3 Komunikační protokoly

Komunikační protokol je systém pravidel, který umožnuje dvou a více entitám komunikačního systému přenést informace pomocí různých variací fyzikálních veličin. Protokol definuje, jak má vypadat syntaxe, sémantika, a jaká pravidla platí

pro synchronizaci, komunikaci a možné opravy chyb. Implementace protokolů může být hardwarová, softwarová, nebo kombinace těchto dvou možností. (16)

3.3.1 Protokol UDP

Protokol UDP (anglicky User Datagram Protocol) je často označován jako nespolehlivý, jednoduchý a bez záruky. Protokol totiž neumožňuje ověřit, zda datagram došel k plánovanému příjemci. Data se mohou lehce ztratit. Též protokol nezachovává pořadí při odeslání. Nelze tedy předpovědět, zda adresátovi dorazí ve stejném pořadí.

Tento protokol je vhodný pro servery, které zaopatřují mnoho uživatelů. Také je užitečný pro aplikace, které fungují systémem otázka-odpověď. Příklad takového systému je DNS případně i sdílení souborů v síti LAN. Nebo také tam, kde se předpokládají ztráty datagramů.

Hlavička UDP protokolu obsahuje čtyři políčka. Jsou to políčka cílového a zdrojového portu, políčko obsahující délku UDP paketu a políčko kontrolního součtu. Tyto informace umožňují ochranu před špatně orientovanými datagramy.

Políčko zdrojového portu je volitelné, odesílatel datagramu totiž nemusí požadovat odpověď. Pokud se políčko nepoužívá, automaticky je vložena hodnota nula. Toto políčko i políčko cílového portu je šestnáctibitové a identifikuje přijímající a odesílající proces.

Políčko délky UDP je v bytech, jeho minimální hodnota je tedy osm. Toto políčko je povinné, nelze ho vynechat.

Kontrolní součet je šestnáctibitová doplňková informace, která je určena jako kontrola, zda při přenosu dat nedošlo k vadě. Adresát má možnost si součet kdykoliv ověřit a zjistit, zda součet souhlasí. Pokud však nesouhlasí, došlo k poškození a ztrátě dat. (17)

3.3.2 Protokol TCP

Protokol TCP (anglicky Transmission Control Protokol) je druhým z dvojice protokolů transportní vrstvy OSI modelu síťové komunikace, nebo do třetí vrstvy modelu TCP/IP. Tento protokol umožnuje aplikaci vytvořit spojeni pro spojitý a seřazený přenos dat s kontrolou doručení. Navíc pomocí portů umožňuje rozlišení různých aplikací spuštěných na jedno zařízení. Protokol používá pro doručení paketu nespolehlivý protokol IP, kde spolehlivost zaručuje odesláním potvrzení o doručení dat. Před samotn7m odesláním dat je nutné navázat spojeni. Tento proces navázání spojení má tři fáze (anglicky three-way handsake). V průběhu navázání se obě strany dohodnou na číslu potvrzení a číslu sekvence. Ukončení spojení probíhá podobným způsobem jako navázání pouze s jiným typem příznaku paketu.(18)

3.3.3 Protokol 104

Protokol IEC 60870-5-104 (též známý jako IEC 104) je složkou zařízení a systémů IEC Telecontrol. Tento protokol je rozšířením protokolu IEC 101, změny jsou jen v přenosové, síťové a fyzické vrstvě. Standard IEC 60870-5 zajišťuje komunikační profil pro odesílání a přijímání základních ovládacích a stavových zpráv mezi dvěma systémy.

IEC 104 umožňuje síťový přístup k IEC 60870-5-101 (někdy též jako IEC 101) pomocí standardního přenosového profilu. Protokol 104 také umožňuje komunikaci mezi řídicí stanicí a rozvodnou přes standartní TCP/IP. Komunikace je založena na modelu klient-server. (19)

4 Návrh a implementace programu

V této části autor popíše požadavky, které má výsledná aplikace splňovat. Jak postupoval při jejich analýze, návrhu a implementaci výsledného programu.

4.1 Funkční požadavky

Aplikace bude splňovat tyto požadavky:

1. Uložení a načtení textového souboru s konfigurací
2. Grafické zobrazení konfigurace
3. CRUD operace s moduly konfigurace
4. Překlad pomocí externího programu do binárního kódu
5. Odeslání konfigurace do ústředny EZS
6. Zpětné načtení konfigurace z ústředny EZS
7. Správa přístupových karet

4.2 Non funkční požadavky

Aplikace je vyvíjena v prostředí Visual Studio 2019 Professional na počítači s operačním systémem Windows 10, na platformě .NET Framework 4.8 s použitím WinForms.

Pro ladění komunikační části aplikace je použita ústředna Star Alarmu s firmwarem jádra verze 5_022. Další možností je použití simulátoru ústředny, který je spuštěn lokálně na zařízení. Pro odchycení UDP paketů byl použit software WireShark.

4.3 Analýza požadavků

Pro urychlení a usnadnění procesu konfigurace EZS je nutné, aby byli všechny požadavky zapsané výše zapsané splněny.

4.3.1 Uložení a načtení konfigurace

Aplikace musí obsahovat textový parser, který musí být schopný správně načíst textový zápis konfigurace. Nejdůležitější je flexibilita parseru, protože většina

konfiguraci byla dříve psána člověkem, je tedy velká pravděpodobnost chyb, překlepů nebo prohození parametrů.

4.3.2 Grafické zobrazení

Grafické zobrazení v hlavním okně musí zobrazit všechny moduly, se základními informacemi o jejich nastavení, například jejich název, adresu nebo zónu, do které patří. V horní části hlavního okna je prvek Ústředna, který zobrazuje základní parametry a nastavení ústředny.

4.3.3 CRUD operace s moduly

Hlavní okno aplikace bude umožňovat smazání, přidání a editaci modulů a jejich vstupů a výstupů. Každý prvek modulu obsahuje tlačítko, které otevře nastavovací formulář modulu a tlačítko pro smazání modulu.

4.3.4 Překlad konfigurace

O překlad konfigurace se stará externí program, který je psán v jazyce C++ a je distribuován jako exe [2]soubor. Aplikace tento soubor při spuštění ukládá do složky AppData. Před každým překladem je uložena textová konfigurace. Při přechodu do komunikačního okna aplikace proběhne překlad a binární soubor pro odeslání je uložen do složky AppData. Aplikace umožňuje překlad a uložení binárního souboru do adresáře zvoleného uživatelem.

4.3.5 Odeslání konfigurace

Odeslání přeložené konfigurace, bude možné pouze po úspěšném překladu. V případě chyby dojde k přerušení překladu a zobrazení dialogového okna.

---

[2] Soubor spustitelný v operačním systému Windows

4.3.6 Načtení konfigurace

Zpětné načteni konfigurace je také důležitou části pro pohodlnou práci se systémem. Tato možnost je dostupná jen pro otevření nové konfigurace, a nachází se v komunikačním formuláři.

4.3.7 Správa přístupových karet

Bezpečnostní systém obsahuje seznam přístupových karet pro přístup a ovládání systému. Aplikace umožní načíst a uložit seznam karet, a to z textového souboru nebo z ústředny pomocí UDP protokolu.

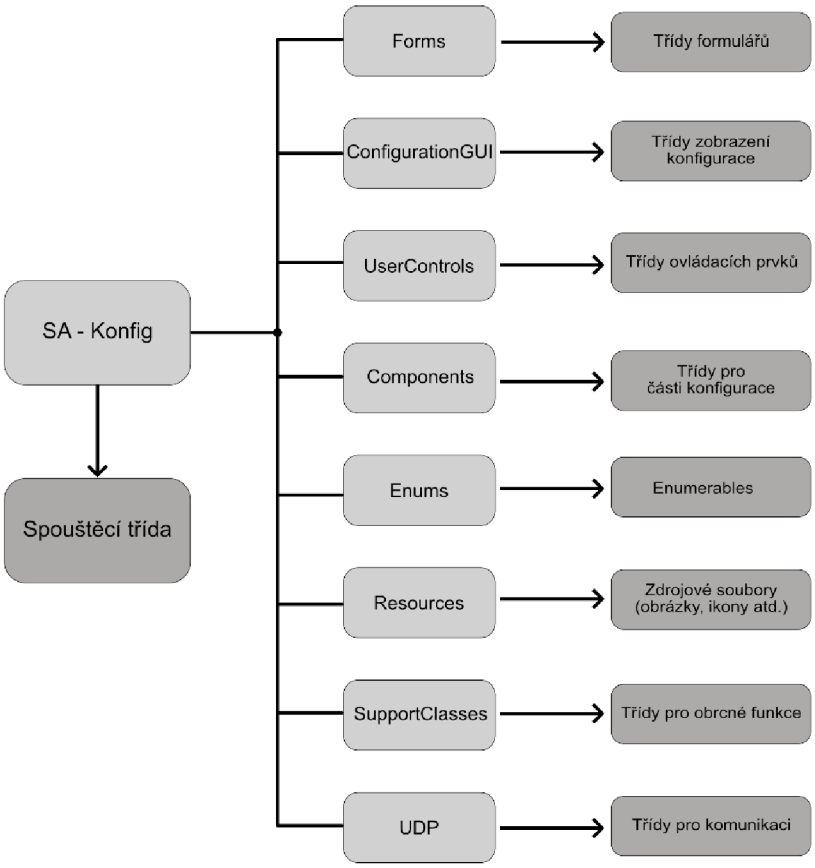
4.4 Návrh aplikace

Třídy aplikace SA-Konfig jsou rozděleny do jmenných prostorů[3], tak jak je zvykem u aplikací na platformě .NET. Toto rozdělení pak odpovídá struktuře adresářů v tomto řešení. Pro tuto aplikaci byl navržen následující způsob dělení:

- SAKonfig: Obsahuje pouze spouštěcí třídu.
- SAKonfig.Forms: V tomto namespace se nachází všechny okenní formuláře aplikace.
- SAKonfig.ConfigurationGUI: Tento jmenný prostor bude obsahovat prvky důležité pro vykreslení grafické konfigurace.
- SAKonfig.UserControls: Namespace určený pro ovládací prvky vytvořené autorem.
- SAKonfig.Components: Zde budou umístěny všechny komponenty konfigurace.
- SAKonfig.Enums: V tomto namespace budou umístěny výčtové typy Enumerables

[3] Anglicky „namespaces“

- SAKonfig.Resources: V této složce budou veškeré potřebné zdroje pro aplikaci. Jako například ikony, exe a dll soubory potřebné pro chod programu.
- SAKonfig.SupportClasses: Namespace pro třídy, které obstarávají například načítání a ukládání souborů.
- SAKonfig.UDP: Zde se nachází všechny třídy potřebné pro UDP komunikaci.



Obrázek 8 – Schéma rozdělení aplikace, vlastní zpracování

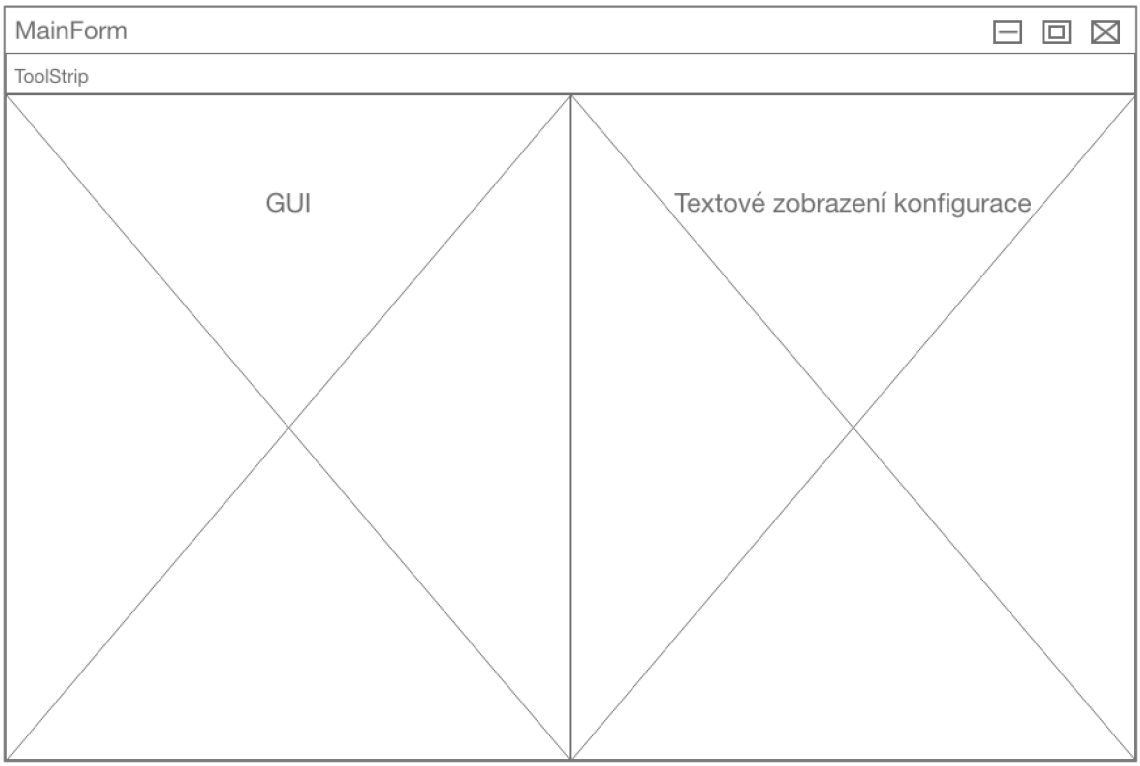
4.4.1 Návrh GUI

Zde autor popíše, jak je aplikace navrhnuta a krátce popíše jednotlivé důležitější formuláře, které aplikace obsahuje. Aplikace je navrhnuta tak, aby umožňovala jednoduchou tvorbu a editaci konfigurace pro systém StarAlarm. Je určena pro

zkušeného uživatele tohoto systému i pro člověka, který nezná textovou konfiguraci tohoto systému.

4.4.1.1 Hlavní okno aplikace

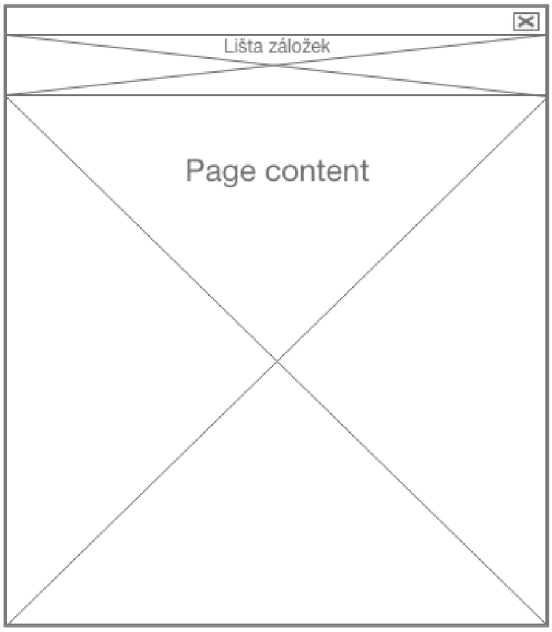
Hlavní okno aplikace je navrhnuto tak, aby bylo možné mít zobrazeno grafickou konfiguraci a zároveň náhled textové formy konfigurace. Zobrazení textu je možné vypnout (například v případě, že je nutné graficky zobrazit větší konfiguraci). Základní návrh hlavního formuláře je rozdělen na polovinu. Na první polovině je zobrazen grafický návrh a na druhé polovině je textový zápis konfigurace. Šířku je možné přizpůsobit pomocí myši a panel textového zobrazení je možné skrýt pomocí tlačítka.



Obrázek 9 – Drátěný model hlavního okna aplikace, vlastní zpracování

4.4.1.2 Okno nastavení ústředny

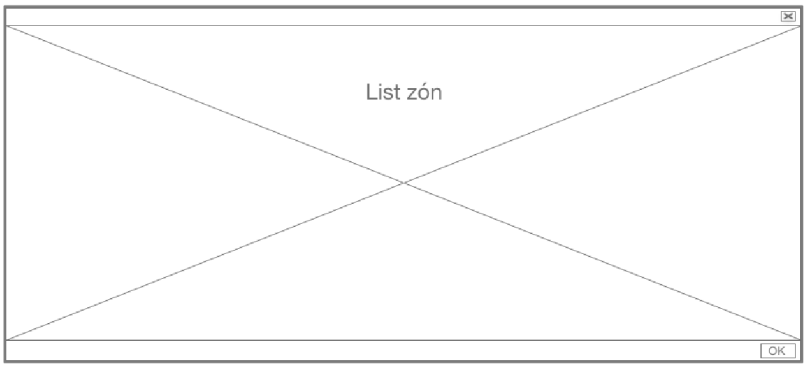
Okno pro nastavení ústředny se skládá z několika záložek, aby bylo možné jednotlivé části nastavení přehledně oddělit.



Obrázek 10 - Drátěný model formuláře pro nastavení ústředny, vlastní zpracování

4.4.1.3 Okno nastavení zón

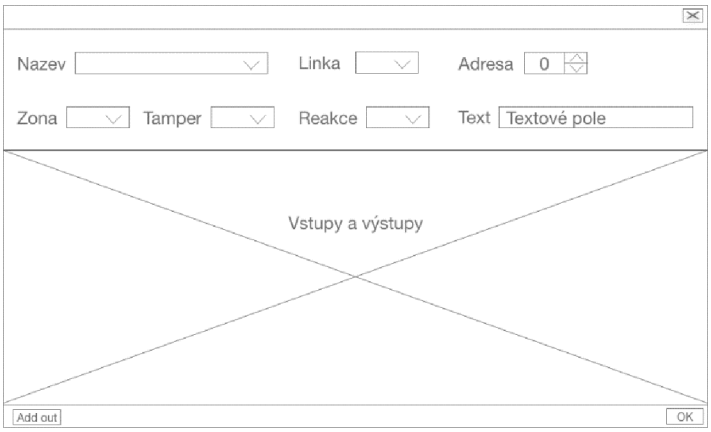
Pro nastavení zón je navrženo okno, kde je zobrazen seznam všech zón a jejich nastavení. Tedy jeden panel pro každou zónu.



Obrázek 11 - Drátěný model okna pro nastavení zón, vlastní zpracování

4.4.1.4 Okno úpravy modulů

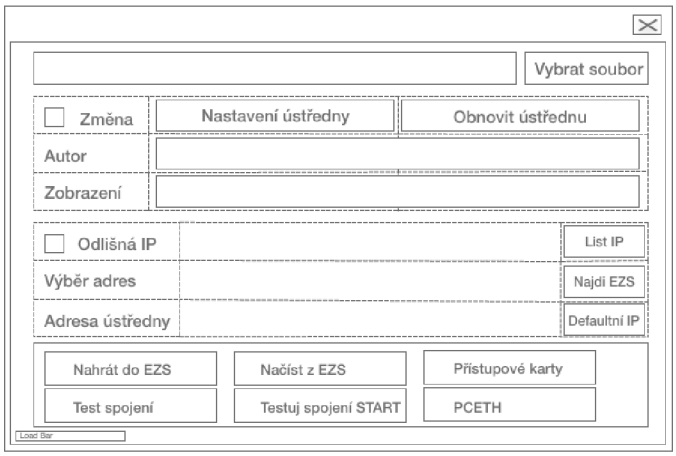
Další důležitou částí aplikace je okno pro nastavení modulu a jejich vstupů i výstupů. Toto okno má dvě možnosti otevření. A to otevření prázdného okna při přidání nového modulu a úprava již exitujícího modulu.



Obrázek 12 - Drátěný model formuláře pro editaci a přidání modulu, vlastní zpracování

4.4.1.5 Okno komunikace s ústřednou

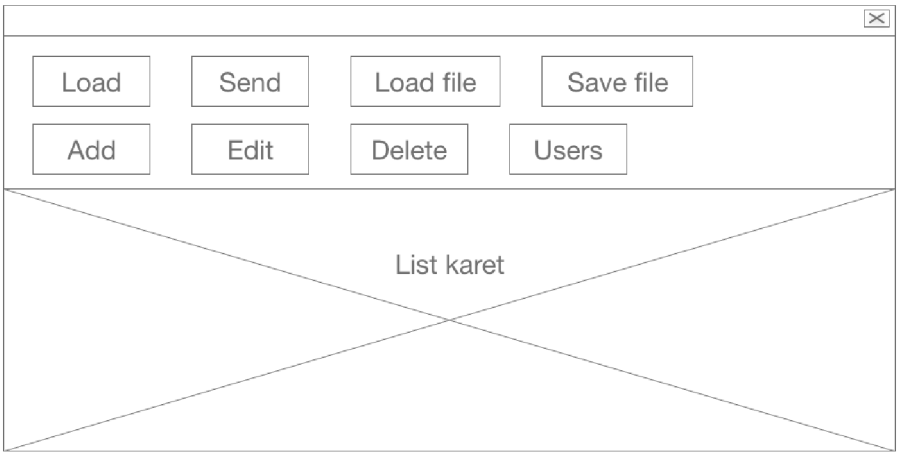
Jednou z posledních důležitějších součástí je komunikační okno, kde probíhá spojení s ústřednou. Toto okno obstarává všechny možnosti komunikace pomocí UDP. Z tohoto okna je možné také otevřít externí soubor pro monitorování systému StarAlarm.



Obrázek 13 - Drátěný model komunikačního okna, vlastní zpracování

4.4.1.6 Okno správy karet

Toto okno obsahuje možnost načíst a odeslat seznam přístupových karet do ústředny EZS. V horní části okna se nachází ovládací tlačítka a ve spodní části je tabulka se seznamem karet.



Obrázek 14 - Drátěný model formuláře pro správu přístupových karet, vlastní zpracování

4.4.1.7 Dialogová okna

Aplikace obsahuje ještě několik dialogových oken, které slouží k některým akcím, pro které není možné použít dialogových oken integrovaných v prostředí .NET, například MessageBox,[4].

4.5 Implementace do kódu

V této části autor popíše implementaci některých částí aplikace.

4.5.1 Popis tříd konfigurace

Pro snadnější práci s konfigurací je pro každou specifickou část konfigurace vytvořena vhodná třída. Pro uložení konfigurace ústředny je implementována třída

[4] Dialogové okno, které zobrazuje zprávu pro uživatele a blokuje všechny ostatní akce. (24)

Ustredna, která obsahuje instance podřízených částí jako je *Note*, *Ethernet*, *Time*, *Stanice* a *Termostat*. Kromě těchto instancí je zde definováno i několik kolekcí, například pro linky RS485 nebo reakce.

Uložení modulu je implementováno v třídě *Modul*, kdy tato třída obsahuje dvě kolekce třídy *InOut* pro uložení vstupů a výstupů. Další kolekce je zde pro *Temperature* a reakce. Modul je ve výchozím stavu vytvořen jako neaktivní, není zobrazen v grafice. Všechny hodnoty jsou implementovány jako vlastnosti a mají nastaveny potřebné přístupnosti. Pro některé hodnoty jsou definovány výčtové typy.

```
public Modul()
        {
            Aktiv = false;
            IsComented = false;
            IsIO = false;
            Adresa = 0;
            Name = ModulName.Default;
            LinkaRS485 = 0;
            Tamper = Tamper.NE;
            Parametr = false;
            Zona = StorageClass.DefaultZona;
            Text = "";
            Coment = "";
            Temperatures = new List<Temperature>();
            Temperatures.Add(new Temperature(1));
            Temperatures.Add(new Temperature(2));
            InputList = new List<InOut>();
            OutputList = new List<InOut>();
            Ant = new InOut();
            Prot104 = new Prot104Items<ModulTag>();
            Reaction = new List<string>();
        }
```

Ukázka kódu 2 – Konstruktor nového modulu, vlastní implementace

4.5.2 Načtení a uložení konfigurace

Pro načítání a ukládání souborů je vytvořena třída *FileManager*, která obstarává rutiny okolo načítání a ukládání souborů. Obsahuje metody pro načtení a uložení binárních a textových souborů.

4.5.3 Parsování textové konfigurace

Po načtení je nutné konfiguraci rozparsovat a uložit do tříd, tak aby s konfigurací mohl program pracovat. Parser se nachází ve třídě *ControlMain*. Tato třída, obsahuje parsery pro textovou i binární konfiguraci, také se do ní ukládají všechny součásti

konfigurace. Mimo zón, které jsou uloženy v statické třídě *StorageClass*, kvůli nutnosti přístupu z více míst v aplikaci.

Pro parsování textové konfigurace slouží funkce *StringParser*. Tato funkce, dostane jako vstupní parametr pole stringů[5], což je načtený soubor rozdělený na řádky. Po inicializaci proměnných následuje *for* cyklus, který prochází jednotlivé řádky a v těle tohoto cyklu se s nimi dále pracuje. Dále je v těle cyklu takzvaný try/catch blok, který ošetřuje případné chyby na jednotlivých řádcích konfigurace. Před průchodem řádku, se vymažou proměnné a jako první se zjistí, zda řádek není zakomentovaný.

```
if (isComented = line.Contains(COMENTS))
    {
        if (line.TrimStart().StartsWith(COMENTS))
        {
            line = line.TrimStart('/');
        }
        if (line.Contains(COMENTS))
        {
           comentFind = line.Substring(line.IndexOf(COMENTS));
           line = line.Replace(comentFind, "");
           comentFind = comentFind.Replace(COMENTS, "");
           comentFind = comentFind.TrimStart();
       }
    }
```

Ukázka kódu 3 – Blok pro detekci komentářů, vlastní implementace

Po odstranění komentářů následuje metoda *GetTag(string line)*, která na základě řádku, který dostane jako parametr zjišťuje, jaký tag řádek popisuje. Tato metoda je řešena pomocí podmínek if a metody *String.Contains*. (20) Při splnění podmínky dojde k dosazení příslušného ukazatele z *Enum.Pointer,* aby bylo později možné rozdělení pomocí přepínače *switch/case.*

Ještě před blokem přepínače se nachází část kódu, která kontroluje adresaci protokolu 104. Tedy uložení adresy do proměnné a kontrola duplicity.

Poslední potřebná věc je zakódování čárek v textu. Pro tento účel je implementována metoda *CodeTextInLine(string line)*, která opět přijímá jako parametr řádek a vrací

[5] Pole znaků, které slouží k reprezentaci textu. (28)

jej v zakódované podobě. Tedy pouze odstraní čárky z textu mezi uvozovkami, kvůli možnosti následného rozdělení.

```
private string CodeTextInLine(string line)
        {
            if (line.Contains("\""))
            {
                string s = StorageClass.BetweenStrings(line, "\"", "\"");
                string ns = TextCode(s);
                if (!s.Equals(ns))
                        line = line.Replace(s, ns);
            }
            line = line.Replace("\"", "");
            return line;
        }
```

Ukázka kódu 4 – Metoda zakódování textu, vlastní zpracování

Následuje blok *switch/case*, kde jsou jednotlivé prvky uloženy do proměnných, aby s nimi bylo možné později snadno pracovat.

4.5.3.1 Parsování Note

V této části probíhá načtení části konfigurace obsahující poznámky a informace o autorovi. Nejdříve je rozdělen první řádek, na kterém je informace o autorovi. Každá informace v konfiguraci je oddělena pomocí čárky. Rozdělení textu umožňuje metoda *String.Split*, která přijímá jako parametr charakter pro rozdělení a je zavolána na instanci třídy *String*. Tato metoda vrátí pole stringů.

Dále je v podmínce kód pro zjištění přítomnosti klíčového slova, za kterým se nachází hodnota. V tomto případě je to název autora.

Následně je pomocí *for* cyklu načten text poznámky, cyklus začíná na dalším řádku a postupuje, dokud nenarazí na prvek *USTREDNA*. Poté je od hlavního počítadla řádků odečten počet řádků, který prošel cyklus pro načtení poznámky.

```
case Pointer.note:
     splitLine = new List<string>(line.Split(commaSpliter));
     if (splitLine.Exists(x => x.Contains(_Autor)))
     {
         _temp = splitLine.Find(x => x.Contains(_Autor));
         Ustredna.Note.Autor = TextDecode(_temp.Substring(_temp.IndexOf(EQ) + 1).Trim());
         _temp = string.Empty;
     }
     for (int j = i + 1; j < loadedText.Length; j++)
     {
         line = loadedText[j];
         line = line.Replace("\"", "");
         if (loadedText[j].Contains(USTREDNA))
     {
         i = j - 1;
         break;
     }
     if (line.TrimEnd().TrimStart() != "")
     {
        Ustredna.Note.AddLine(line.TrimEnd().TrimStart());
     }
}
break;
```

Ukázka kódu 5 - Parsování prvku NOTE

4.5.3.2 Parsování modulů

Při detekování řádku popisující modul konfigurace jsou vymazány kolekce obsahující vstupy a výstupy. Dalším krokem je vytvoření instance konkrétního modulu pomocí zkráceného konstruktoru. Po přidání tagů protokolu 104 následuje metoda *ParseModulLine(modul, line)*, která rozděluje hodnoty na řádku modulu. Tato metoda má návratovou hodnotu instanci třídy *Modul*, která je následně přiřazena do proměnné *actualObject* kvůli následnému přidání vstupů a výstupů.

Metoda *ParseModulLine* je implementována stejně jako všechny ostatní části, které slouží k dělení konfigurace. Tedy rozdělí řádek na texty obsahující klíčové slovo a hodnotu. Následně pomocí vyhledávacích metod z jmenného prostoru *System.Collections.Generic* je zjištěna existence klíčového slova a vyhledána jeho hodnota.

Na konci metody je detekováno, zda je modul částí ústředny. V případě, že tomu tak není, je modul přidán do seznamu všech modulů.

```
case Pointer.kon4:
      InputList = new List<InOut>();
      OutputList = new List<InOut>();
      modul = new Modul(true, isComented, ModulName.KON4, comentFind, InputList, OutputList);
      modul.Prot104 = Get104Items<ModulTag>(prot104items);
      actualObject = ParseModulLine(modul, line);
      break;
```

Ukázka kódu 6 - Case parsování modulu, vlastní implementace

```
private Modul ParseModulLine(Modul modul, string line)
{
   string _temp;
   List<string> splitLine;
   splitLine = new List<string>(line.Split(commaSpliter));
   if (splitLine.Exists(x => x.Contains(_Linka))){
       _temp = splitLine.Find(x => x.Contains(_Linka));
       modul.LinkaRS485 = Convert.ToInt32(_temp.Substring(_temp.IndexOf(EQ) + 1).Trim());
       _temp = string.Empty;
    }
    if (splitLine.Exists(x => x.Contains(_Adresa))){
    _temp = splitLine.Find(x => x.Contains(_Adresa));
    modul.Adresa = Convert.ToInt32(_temp.Substring(_temp.IndexOf(EQ) + 1).Trim());
        _temp = string.Empty;
    }
    if (splitLine.Exists(x => x.Contains(_Zona))){
        _temp = splitLine.Find(x => x.Contains(_Zona));
        modul.Zona=StorageClass.Zony[Convert.ToInt32(
                                _temp.Substring(_temp.IndexOf(EQ)+1).Trim())];
    _temp = string.Empty;
    }
    if (splitLine.Exists(x => x.Contains(_Text))){
        _temp = splitLine.Find(x => x.Contains(_Text));
        modul.Text = TextDecode(_temp.Substring(_temp.IndexOf(EQ) + 1).Trim());
        _temp = string.Empty;
    }
    if (!modul.IsIO){
        Moduls.Add(modul);
    }
    return modul;
}
```

Ukázka kódu 7 - Metoda ParseModulLine, vlastní implementce

4.5.4 Překlad konfigurace

Pro překlad konfigurace je implementována metoda *Translate*. Tato metoda dostane jako parametr adresu pro uložení výsledného souboru s binární konfigurací. Metoda vrací proměnnou typu string, ve které je uložen výstup externí aplikace.

Pro spuštění aplikace je použita metoda *Process.Start(startInfo)*, která přijme jako parametr třídu *ProcesStartInfo*. Tato třída obsahuje informace potřebné pro spuštění exe souboru. Například, zda má být aplikace otevřena v okně, nebo pouze na pozadí. Dalšími informacemi je adresa vedoucí k spustitelnému souboru nebo parametry, které mají být použity při spuštění. V tomto případě parametry

umožňují určení souboru, který má být přeložen a kam má být uložen výsledný binární soubor.

```
public string Translate(string file)
        {
            string s = string.Empty;
FileManager.SaveStringFile(StorageClass.TempTextFileToTranslate, OutTextToTranslate());
            ProcessStartInfo startInfo = new ProcessStartInfo();
            startInfo.CreateNoWindow = false;
            startInfo.UseShellExecute = false;
            startInfo.RedirectStandardError = true;
            startInfo.RedirectStandardOutput = true;
            startInfo.FileName = StorageClass.TranslatePath;
            startInfo.WindowStyle = ProcessWindowStyle.Normal;
            startInfo.Arguments = "\"" + StorageClass.TempTextFileToTranslate
                                      + "\" " + "\"" + file + "\"";
            try
            {
                using (Process exeProcess = Process.Start(startInfo))
                {
                    s = exeProcess.StandardOutput.ReadToEnd();
                    s += exeProcess.StandardError.ReadToEnd();
                    if (exeProcess.StandardError.ReadToEnd() != "")
                    {
                    MessageBox.Show(exeProcess.StandardError.ReadToEnd(), "Chyba překladu!");
                    }
                    exeProcess.WaitForExit();
                }
            }
            catch (Exception e)
            {
                Console.WriteLine(e.Message);
            }
            return s;
        }
```

Ukázka kódu 8 - Metoda pro překlad konfigurace, převzato (21) a upraveno

4.5.5 Kontrola délky textu

Pro usnadnění tvorby konfigurace aplikace kontroluje uživatelem zadané hodnoty, aby nedošlo k neočekávané chybě. V případě textu, je nutné dodržet maximální délku 25 znaků. Problémem jsou znaky obsahující diakritiku. Jeden tento znak je v binární podobě značen třemi znaky z ASCII tabulky. Z toho důvodu bylo nutné vytvořit vlastní kontrolu textu napsaného do textového pole. O kontrolu zapsaného textu, se stará metoda *StringCutter(TextBox t*), která jako parametr převezme *TextBox,* s jehož textem je třeba pracovat. Tato metoda je volána vždy při změně textu. Metoda zkontroluje zapsaný text, pokud je delší než 24 znaků, je zkrácen.

```
public static void StringCutter(TextBox t)
        {
            if (GetTextLenght(t.Text) > 24)
            {
                int select = t.Text.Length - 1;
                t.Text = t.Text.Remove(t.TextLength - 1, 1);
                SystemSounds.Hand.Play();
                t.Select(select, 0);
            }
        }
```

Ukázka kódu 9 - Metoda pro zkrácení textu v TextBoxu, vlastní implementace

Další metodou pro ošetření délky textu je metoda CutString(string temp), která zkrátí text, obsažený v přijímaných parametrech a vrátí výsledek jako návratovou hodnotu. Je využita při tvorbě textové konfigurace pro překlad.

```
private string CutString(string temp)
        {
            while (StorageClass.GetTextLenght(temp) > textLenght)
            {
                temp = temp.Substring(0, temp.Length - 1);
            }
            return temp;
        }
```

Ukázka kódu 10 - Metoda pro zkrácení textu, vlastní implemetace

4.5.6 UDP komunikace

O komunikaci pomocí UDP obsluhuje třída *UDPComunication*. Tato třída obsahuje metody pro komunikaci s EZS. Konkrétně jsou to metody FindEzsIPAddress a SendAndRecive. Obě metody slouží pouze pro vyřízení komunikace. Jako parametr dostávají pole byte, které obsahuje hotový packet pro odeslání.

Při inicializaci třídy je v parametru konstruktoru předána IP adresa ústředny, případně broadcast[6] adresa. V konstruktoru jsou nastaveny porty pro odeslání a příjímání paketů a end point pro odesílaní.

[6] Zpráva pro všechna připojená síťová rozhraní. (29)

```
public UDPComunication(IPAddress serverIP)
        {
            this.receivePort = 10700;
            this.sendPort = 10710;
            this.sendEndPoint = new IPEndPoint(serverIP, sendPort);
        }
```

Ukázka kódu 11 - Konstruktor třídy UDPComunication, vlastní implementace

4.5.6.1 Metoda FindEzsIPAddress

Metoda slouží k vyhledávání ústředen na síti. Pro komunikaci je využívána třída *UdpClient*. Po inicializaci proměnné udpClient je nutné použít metodu udpClient.Client.Bind, která umožňuje svázat socket k použití klienta. Dalším krokem je nastavení ReciveTimeout. Pro uložení nalezených adres je definován list IP adres, který metoda po přijmutí odpovědí vrátí.

4.5.6.2 Metoda SendAndRecive

Metoda pro účel komunikace s ústřednou. Funguje na principu okamžité odpovědi. Ústředna odpoví na každý rozpoznaný dotaz. Na začátku je definována instance třídy *Socket*, která nám umožňuje komunikaci pomocí UDP protokolu. Třida *Socket* obsahuje metodu *Bind*, která slouží k definování portu, na kterém budou přijímána data. Následuje inicializace pole pro přijatá data, nastavení příchozího a odchozího časového limitu. Dále v bloku Try/Catch již probíhá komunikace. Pomocí metody *Connect* je nastaven end point, na který komunikace směřuje, tedy adresa ústředny.

```
public byte[] SendRecive(byte[] data, int timeout = 3000)
        {
Socket server = new Socket(AddressFamily.InterNetwork,SocketType.Dgram, ProtocolType.Udp);
            server.Bind(new IPEndPoint(IPAddress.Any, 0));
            byte[] reciveData = new byte[1051];
            server.SendTimeout = timeout;
            server.ReceiveTimeout = timeout;
            try
            {
                server.Connect(sendEndPoint);
                server.Send(data);
                Remote = new IPEndPoint(IPAddress.Any, receivePort);
                server.ReceiveFrom(reciveData, SocketFlags.None, ref Remote);
            }
            finally
            {
                server.Close();
            }
            return reciveData;
        }
```

Ukázka kódu 12 - Metoda SendAndRecive, vlastní implementace

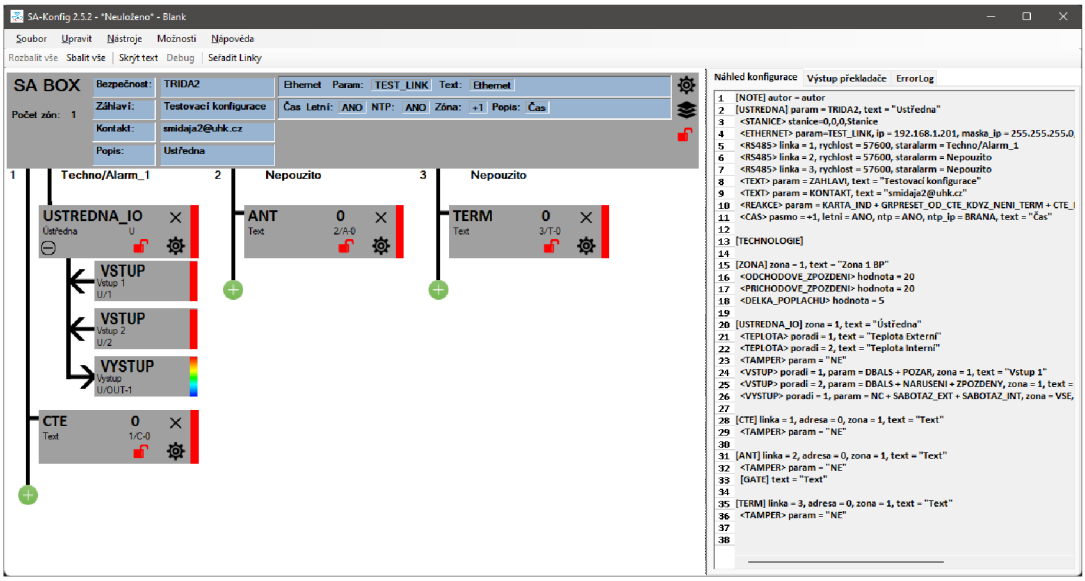
4.5.7 Formuláře

Aplikace obsahuje několik formulářových oken. Hlavním oknem aplikace je již výše zmíněný FormMain. Dalšími formuláři aplikace jsou:

- FormAdd – umožňuje přidávání a úpravu modulů a jejich vstupů.
- FormUstredna – nastavení ústředny, a jejích součástí
- FormZony – obsahuje nastavení zón včetně jejich parametrů
- FormEZSComunication – okno pro komunikaci s ústřednou, například nahrání nebo vyčtení konfigurace
- FormEZSCards – správa přístupových karet
- FormSettings – uživatelské nastavení aplikace

4.5.7.1 FormMain

Hlavní částí okna je panel pro grafické zobrazení konfigurace, k tomuto účelu jsou vytvořeny prvky pro jednoduší grafické zobrazení. Těmito prvky jsou UstrednaGUI, ModulGUI a IoGUI. Tyto prvky grafického zobrazení jsou hlavními částmi pro zobrazení konfigurace, implementují události pro přenesení kliknutí na tlačítka, která zobrazují, ale jejich metody jsou vykonávány v hlavním formuláři.



Obrázek 15 – Ukázka FormMain, vlastní zpracování

4.5.7.2 FormEZSComunication

Formulář určený pro komunikaci s ústřednou. Implementuje metody pro nahrávání a vyčítání konfigurace. Tyto metody využívají komponentu *BackgroundWorker*, aby nedocházelo k zamrznutí programu při komunikaci s ústřednou. *BackgroundWorker* umožňuje v průběhu své činnosti předávat informace do hlavního vlákna, například měnit stav v stavovém řádku a aktualizovat *ProgressBar*.

Formulář má dvě možnosti otevření, tedy dva konstruktory. Výchozí obsahuje vstupní parametr třídy *Ustredna*, kde je předána informace o IP adrese, na kterou bude směřována komunikace, ale v případě, že je to konfigurace nové ústředny je možné komunikační adresu v okně změnit. V konstruktoru je nastaveno zapnutí nebo vypnutí některých tlačítek dle způsobu spuštění. Také je nastaveno pole pro uložení binární konfigurace, která je předána v parametru konstruktoru. Na konci konstruktoru je metoda *LoadIpAddreses*, která slouží k načtení seznamu IP adres známých ústředen pro snadnější správu.

```
private void BwSendToEZS_DoWork(object sender, System.ComponentModel.DoWorkEventArgs e)
{
    try
    {
        BackgroundWorker worker =(BackgroundWorker) sender;
        worker.ReportProgress(0);
        if (GetEZSVersion()){
            throw new Exception("Připojujete se k ustředně v1");
        }
        worker.ReportProgress(20);
        tRX_Paket_StarAlarm_DataCommit tRX_Paket_StarAlarm_DataCommit =
        byteArrayToStructure<tRX_Paket_StarAlarm_DataCommit>(b.EraseFlash());
        worker.ReportProgress(30);
        if (DataComitTest(tRX_Paket_StarAlarm_DataCommit.Potvrzeni_dat)){
            worker.ReportProgress(40);
            b.CopyAllToFlash(DISTA, structRS485, configFile);
            worker.ReportProgress(60);
        }
    e.Result = ByteArrayToStructure<tRX_Paket_StarAlarm_DataCommit>(b.UpdateConfiguration());
    worker.ReportProgress(90);
    }
    catch (Exception ex)
    {
         throw ex;
    }
}
```

Ukázka kódu 13 - Metoda BackgroundWorkeru pro odeslání konfigurace, vlastní implementace

```
public FormEZSComunication(Ustredna ustredna)
        {
            InitializeComponent();
            this.ustredna = new Ustredna(ustredna);
            ustrednaBackUp = ustredna;
            configFile = new byte[0];
            btnLoadFromEzs.Enabled = true;
            ipConnect.IPAddress = ustredna.Ethernet.IP;
            txtAutor.Text = ustredna.Note.Autor;
            cmbDisplay.SelectedIndex = 1;
            stations = StorageClass.LoadStations();
            b = new BinaryStructures(ipConnect.IPAddress);
            LoadIpAddreses();
        }
        public FormEZSComunication(Ustredna ustredna, byte[] configFile):this(ustredna)
        {
            this.configFile = configFile;
            selectedBinaFileName.Text = "AKTUALNÍ KONFIGURACE";
            btnSelectBina.Enabled = false;
            btnLoadFromEzs.Enabled = false;
        }
```

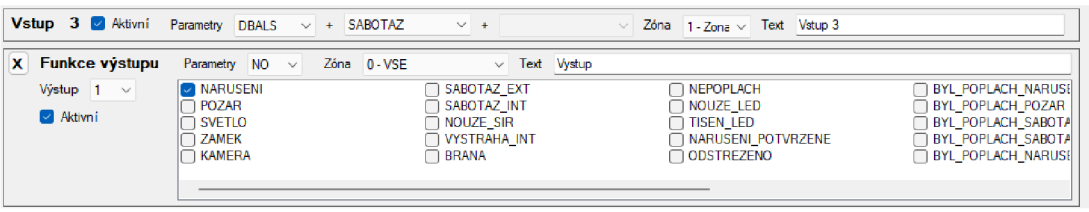
Ukázka kódu 14 - Konstruktory FormEzsComunication, vlastní implementace

4.5.8 Vlastní ovládací prvky

Pro zjednodušení tvorby formulářů obsahující opakující se prvky jsou vytvořeny vlastní ovládací prvky. Využití této funkcionality frameworku velice usnadňuje vykreslení některých částí aplikace, kde se vyskytuje mnoho opakujících se prvků. Například při přidávání modulů, kde je třeba vypsat všechny vstupy a výstupy, nebo pro nastavení zón, kde je 20 stejných prvků nastavení.

4.5.8.1 PanelVstup a PanelVystup

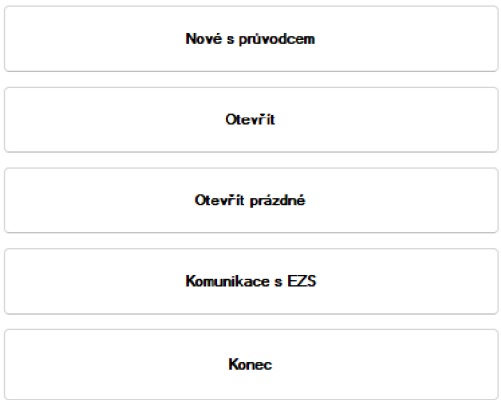
Tyto panely jsou používány k zobrazení vstupů a výstupů ve formuláři pro úpravu a přidání modulů. Slouží pouze k zobrazení a kontrole některých podmínek. Obsahují metody pro získání třídy *InOut*, která obsahuje všechny informace o vstupu, nebo výstupu.



Obrázek 16 – Ukázka zobrazení vstupu a výstupu, vlastní zpracování

4.5.8.2 StartupDialog

Tento prvek je vyvolán při spuštění aplikace, slouží jako rozcestník pro uživatele. Po kliknutí na tlačítko je vyvolána událost *ClickResult*, která v proměnné *sender* předá hodnotu, která umožní v následujícím *switch/case* provést zvolenou akci.



Obrázek 17 – Ukázka StartupDialogu, vlastní zpracování

4.6 Struktura textového výstupu aplikace

V této části bakalářské práce autor popíše, jak vypadá textový výstup aplikace, jaké jdou použity tagy a jak se zapisují. Tagy mají různé parametry a vlastnosti. U téměř každého tagu se nachází parametr text, který slouží k pojmenování nebo nastavení určité textové hodnoty.

4.6.1 Prvek NOTE

Tento prvek slouží jako úvodní poznámka. Parametr *autor* je povinným parametrem, který by měl obsahovat určení autora dokumentu. Pod tímto prvkem je možné popsat konfiguraci. Text mezi uvozovkami je povolen až do začátku prvku *USTREDNA*.

```
[NOTE] autor = "Jakub Šmída"
"Testovací konfigurace pro účely BP"
```

Ukázka konfigurace 1 – Ukázka prvku NOTE, vlastní zpracování

4.6.2 Prvek USTREDNA

Prvek *USTREDNA* provádí nastavení ústředny EZS. V celém systému může být pouze jedna ústředna. Ústředna má dva parametry – *param* a *text*. Pod tento prvek spadají tagy *ETHERNET*, *CAS*, *PROT104*, *TEXT* a *REAKCE*.

Parametr *param* má dvě vlastnosti. První z nich je vlastnost *TRIDA3*, která nastavuje chování ústředny podle III. třídy zabezpečení. Jeho druhou vlastností je *TRIDA2*. Ta ovlivní chování ústředny podle II. třídy zabezpečení. Pokud vlastnost nevyplníme, automaticky se doplní *TRIDA2*.

Parametr *text* se nachází na téměř každém řádku konfigurace. Slouží k pojmenování prvku. V tomto případě slouží k pojmenování ústředny nebo definování umístění.

```
[USTREDNA] param = TRIDA3, text = "Ústředna Bakalářská Práce"
```

Ukázka konfigurace 2 – Prvek USTREDNA, vlastní zpracování

4.6.2.1 Tag ETHERNET

Tento tag aktivuje ethernetové spojení. Pokud se tag v konfiguraci nenachází, není možné, aby spojení probíhalo. Stejně jako u prvku *USTREDNA* má tento tag dva parametry – parametr *param* a parametr *text*, který pojmenovává spojení.

První parametr *param* má čtyři vlastnosti, kterými jsou *TEST_NE*, *TEST_LINK*, *TEST_UDP* a *TEST_104*. Pokud není u tohoto parametru vlastnost zadaná, výchozí nastavení je *TEST_LINK*. Vlastnost *TEST_NE* znamená, že se připojení neověřuje a netvoří žádnou překážku v zastřežení[7]. Druhá vlastnost *TEST_LINK* kontroluje, zda je ethernetový port zapojen do zásuvky. Vlastnost *TEST_UDP* sleduje, zda přichází pravidelné UDP pakety na poplachovou ústřednu. Poslední vlastnost *TEST_104* sleduje funkčnost spojení pomocí protokolu IEC 104, který je popsán výše.

[7] Překážkou v zastřežení je stav, který brání v uvedení systému do střežícího stavu.

```
<ETHERNET> param = TEST_LINK, text = "Připojení na síť"
```

Ukázka konfigurace 3 – Tag ETHERNET, vlastní zpracování

4.6.2.2 Tag CAS

Čas v ústředně je nastaven vždy podle výchozího nastavení. Tento tag pouze mění parametry onoho výchozího nastavení. Tag má čtyři vlastnosti a jeden parametr, který pojmenovává čas.

Vlastnost *pasmo* určuje, v jakém časovém pásmu má čas běžet. Udává se v hodnotách od -12 do +12. Zda se mění letní a zimní čas provádí vlastnost *letni*, může nabývat pouze hodnot ano, nebo ne. Vlastnost *ntp* synchronizuje čas z NTP[8] serverů, může mít pouze hodnoty ano, nebo ne. IP adresu, ze kterého se čas synchronizuje udává vlastnost *ntp_ip*. Tato vlastnost je povinná a její výchozí hodnota je IP adresa brány (anglicky gateway[9]).

```
<CAS> pasmo = +1, letni = ANO, ntp = ANO, ntp_ip = BRANA, text = "Čas"
```

Ukázka konfigurace 4 – Tag CAS, vlastní zpracování

4.6.2.3 Tag PROT104

Tag PROT104 slouží k nastavení protokolu IEC 104, který se používá ke komunikaci na drážní síti. V případě že je zapsán v konfiguraci, je třeba ještě ke všem modulům, vstupům, zónám, ethernetu a ústředně nutno doplnit adresy pro jejich rozlišení pro tento protokol. Tag má čtyři parametry – *trat*, *stanice*, *param* a *text*. Parametr *text* opět pojmenovává spojení.

---

[8] NTP (anglicky The Network Time Protocol) je protokol pro synchronizaci vnitřních hodin počítačů a síťových zařízení, převzato (23)

[9] Gateway převádí komunikaci mezi lokální sítí a internetem. (26)

Parametr *trat* udává číslo trati, nabývá tedy hodnot od 0 do 255. Parametr *stanice* udává číslo stanice, nabývá hodnot od 0 do 255. Parametr *param* má čtyři vlastnosti. První vlastností je *CAS_SYNC*. Tato vlastnost povoluje synchronizaci času po protokolu 104. Neustálé hlášení čidel v od střeženém stavu provádí vlastnost *STALA_INDIKACE*. Vlastnost *GRPRESET_1* je pomocná vlastnost určená pouze pro instalace, kde je třeba, aby byla hodnota 1 akceptována pro restart zón[10]. Poslední vlastností parametru *param* je *NULUJ_PRIHLASENIM,* která způsobí vynulování všech aktivací čidel po přihlášení.

```
<PROT104> trat = 101, stanice = 20,
param = CAS_SYNC + STALA_INDIKACE + NULUJ_PRIHLASENIM, text = "Kom po 104"
```

Ukázka konfigurace 5 – Tag PROT104, vlastní zpracování

4.6.2.4 Tag TEXT

V tagu TEXT mohou být dva parametry, parametr *param* a parametr *text*. Parametr *param* má dvě vlastnosti, které určují, v jaké části obrazovky se zobrazí text, který je uvedený v parametru *text*. Vlastnost *ZAHLAVI* způsobuje to, že se text zobrazí v horní části obrazovky. A druhá vlastnost *KONTAKT* zapříčiňuje to, že se text zobrazí v dolní části obrazovky terminálu.

```
<TEXT> param = ZAHLAVI, text = "Testovací konfigurace"
<TEXT> param = KONTAKT, text = "Kontakt pro testování"
```

Ukázka konfigurace 6 – Tagy TEXT, vlastní zpracování

4.6.2.5 Tag REAKCE

Tento tag ovlivňuje chování ústředny. Tag má jeden parametr *param*, který má několik vlastností, které lze mezi sebou kombinovat. Níže každou vlastnost autor krátce popíše.

[10] Ve většině případů je hodnota pro restart zón defaultně hodnota 2.

Vlastnost *AKTIVACE_NEOMEZENY* ruší výchozí nastavení, které odpojí čtecí zařízení (TERM, CTE) po pěti neúspěšných aktivacích. Druhá vlastnost *OSTROVNI* zruší závislost zastřežení na aktivním ethernetovém připojení. Tedy neaktivní připojení nebrání v zastřežení. Další vlastností parametru param je *KARTA_IND*, která způsobuje, že zařízení schopná číst přístupové karty, budou zvukovou indikací indikovat kartu v blízkosti. Vlastnost *TAMPER_VSE* znamená, že bude nastaven tamper[11] u všech modulů. *BLOKOVANI_DO_RESETU* je vlastnost, která obnoví blokované prvky po restartu ústředny. *PORUCHA_SIR_SABOTAZ* se týká signálu *PORUCHA_SIR* a znamená, že tento signál bude zpracován jako sabotáž. V případě že není tato vlastnost zvolena, je signál *PORUCHA_SIR* zpracován jako porucha výstražného zařízení. Restart zón se uskuteční při načtení známé karty na *CTE* nebo *TERM* a má tři možné vlastnosti. V jednu chvíli může být aktivní pouze jedna z následujících vlastností:

- *GRPRESET_OD_CTE_VZDY* – vyvolá restart zón vždy, když CTE přečte známou kartu
- *GRPRESET_OD_CTE_NIKDY* – CTE nikdy nevyvolá restart zón
- *GRPRESET_OD_CTE_KDYZ_NENI_TERM* – výchozí nastavení, ústředna se tak chová, v případě že žádná z možností GRPRESET není nastavena v konfiguraci

```
      <REAKCE> param = AKTIVACE_NEOMEZENY + TAMPER_VSE + KARTA_IND + OSTROVNI
+ GRPRESET_OD_CTE_VZDY
```

Ukázka konfigurace 7 – Tag REAKCE pro ústřednu, vlastní zpracování

4.6.3 Prvek ZONA

Zóny slouží k rozdělení modulů do skupin, může jich být maximálně dvacet. Každá zóna má vlastnost *zona*, kde se nastaví pořadí zóny, tedy 1 až 20. Další vlastností je *text,* který slouží k popisu zóny.

[11] Bezpečnostní kontakt, který chrání před sabotáží krabičky.

Každá zóna může obsahovat několik tagů:

- *PRICHODOVE_ZPOZDENI, ODCHODOVE_ZPOZDENI* – nastavení příchodového a odchodového zpoždění alarmu v dané zóně, může obsahovat hodnotu 0 až 255 sekund. Pokud není v konfiguraci nastaven, je použito výchozí nastavení 20 sekund.
- *DELKA_POPLACHU* – umožňuje nastavit dobu po jakou bude v zóně hlášen zvukový poplach. Výchozí hodnota je 5 sekund a rozsah hodnot je 0–255 sekund.
- *AUTOMATICKE_ZASTREZENI* – tento tag nastavuje za jakou dobu v rozpětí 0 až 255 minut se zóna automaticky zastřeží, výchozí nastavení pro tento tag je 0, to znamená, že se nepoužívá automatické zastřežení.
- *UPRESNENI* – slouží k určení vlastností zóny, může mít následující hodnoty:
  - *NEAKTIVNI* – deaktivace zóny
  - *KRATKA_SABOTAZ* – sabotážní poplach trvá pouze 5 sekund
- *SPOLECNA_ZONA* – lze nastavit závislost na jiné zóně, tedy pokud zóna nastavená v tomto tagu nebude zastřežena, nebude možné zastřežit ani tuto zónu.

```
[ZONA] zona = 1, text = "Test Zona 1"
 <PRICHODOVE_ZPOZDENI> hodnota = 25
 <ODCHODOVE_ZPOZDENI> hodnota = 25
 <DELKA_POPLACHU> hodnota = 25
 <AUTOMATICKE_ZASTREZENI> hodnota = 25
 <UPRESNENI> param = 25
```

Ukázka konfigurace 8 – Nastavení zóny v konfiguraci, vlastní zpracování

4.6.4 Prvek TECHNOLOGIE

Neobsahuje žádné parametry, slouží pouze k oddělení části konfigurace, která je určena pro konfigurátor a není překládána do binární podoby externím programem.

4.6.4.1 Tag TERMOSTAT

```
[TERMOSTAT] adresa = 92, HM = 1, DM = 1, HPM = 1, DPM = 1, text = "Termostat"
```

Ukázka konfigurace 9 – Nastavení termostatu, vlastní zpracování

V tomto tagu jsou nastaveny meze termostatu, který je obsažen na desce ústředny. Povinné parametry jsou:

- *adresa* – adresa termostatu v případě EZS je vždy hodnota 92
- *HM* – horní mez
- *DM* – dolní mez
- *HPM* – horní mez poruchová
- *DPM* – dolní mez poruchová
- *text* – popis termostatu

4.6.5 Prvky modulů

```
[KON8] linka = 1, adresa = 0, zona = 1, text = "SA-KON8"
  <TAMPER> param = ANO
  <VSTUP> poradi = 1, param = NC + NARUSENI + ZPOZDENY, zona = 1,
       text = "Narušení zpožděˇ."
  <VSTUP> poradi = 2, param = DBALS + NARUSENI + OKAMZITY, zona = 1,
       text = "Narušení okamž."
  <VSTUP> poradi = 3, param = NC + POZAR, zona = 1, text = "Požár"
  <VSTUP> poradi = 4, param = NC + MASKOVANI, zona = 1, text = "Maskování"
  <VSTUP> poradi = 5, param = NC + PORUCHA_SIR, zona = 1,
       text = "Porucha sirén"
  <VYSTUP> poradi = 1, param = NO + SVETLO, zona = 1, text = "Světlo"
  <VYSTUP> poradi = 2, param = NO + KAMERA + SABOTAZ_INT + VYSTRAHA_INT,
       zona = 1, text = "Kamera"
  <VYSTUP> poradi = 3, param = NO + ZAMEK, zona = 1, text = "Zámek"
  <VYSTUP> poradi = 4, param = NO + BRANA, zona = 1, text = "Vrata"
```

Ukázka konfigurace 10 – Konfigurace modulu se vstupy a výstupy, vlastní zpracování

K nastavení modulů nacházejících se v konfiguraci je používán prvek obsahující zkratku modulu, která je v hranatých závorkách.

Možné typy modulů jsou:

- ANT – Přístupová anténa je určena například k otevření vstupních dveří do budovy. K tomuto modulu je nutné přidat modul GATE, který slouží jako bezpečnostní most do systému.
- CTE – Čtecí zařízení na přístupové karty. Hlavním účelem je možnost zastřežit a odstřežit zónu, ke které je definována. Další možností použití je jako přístupový bod k otevření dveří. Modul obsahuje dva vstupy a jeden výstup.
- KON4 – Koncentrační modul obsahující čtyři vstupy a dva výstupy.
- KON8 – Koncentrační modul čítající osm vstupů a čtyři výstupy.
- LOCK – Modul určený k otevírání dveří. Obsahuje jeden výstup, určený k připojení elektronického dveřního zámku. Také obsahuje čtyři vstupy.
- TERM – Modul terminálu s dotykovým displejem, který slouží ke komplexnímu ovládání zabezpečovacího zařízení. Modul také obsahuje čtečku přístupových karet. Po přihlášení pomocí platné karty lze ovládat zóny a procházet poplachy. Modul dále obsahuje dva vstupy a jeden výstup.

Každý prvek musí obsahovat následující parametry:

- *adresa* – pořadí modulu v konfiguraci, musí se shodovat s hardwarovým nastavením na desce modulu
- *linka* – slouží k nastavení linky, ke které je modul připojen
- zona – určení, do které zóny modul náleží
- *text* – slouží k popsání modulu

Možný je výskyt parametru *param* s hodnotou KON_V_EZS, který byl používán k nastavení interních vstupů a výstupů na desce ústředny. Nyní je k tomuto účelu vytvořen prvek *USTREDNA_IO*, který na rozdíl od ostatních modulů nemá definované parametry *linka* a *adresa*. Tento prvek má tři vstupy, dva výstupy a jeden reléový výstup, který je určen k spínání výkonových zařízení. Může například spínat klimatizaci, která může být řízena zabudovaným termostatem.

4.6.5.1 Tag TAMPER

Tento tag slouží k nastavení tamperu modulu. Obsahuje jediný parametr *param*, který může nabývat třech hodnot. Těmi jsou hodnoty *ANO*, *NE* a *GLOBAL*. Hodnoty *ANO* a *NE* jsou nadřazeny globálnímu nastavení v ústředně.

4.6.5.2 Tag TEPLOTA

V případě modulu *USTREDNA_IO* se tímto tagem nastavuje použití teplotního čidla na modulu nebo EZS. Parametry pro tento tag jsou *poradi* a *text*. Parametr *poradi* určuje, které čidlo je použito dle hodnoty - 1 pro externí čidlo vyvedené z EZS a 2 pro čidlo na základové desce EZS. V případě modulu jde pouze o číslování.

4.6.5.3 Tagy VSTUP a VÝSTUP

Slouží k určení jednotlivých vstupů a výstupů modulu. Jejich počet je různý dle modulů. Musí obsahovat parametry *zona*, *poradi*, *param* a *text*.

- *zona* – Určuje, do jaké zóny EZS je vstup zařazen. Možné hodnoty tohoto parametru jsou 1 až 20. V případě výstupu je možné použít hodnotu *VSE*, která určuje, že výstup patří do všech zón.
- *poradi* – Pořadí vstupu číslováno od 1. Jejich maximální počet je určen dle typu modulu.
- *param* – Určuje vlastnosti a chování vstupu nebo výstupu.
- *text* – Tento parametr slouží k popsání prvku.

4.6.5.3.1 Parametry vstupu

V případě vstupu jsou parametry rozděleny do tří skupin. Z každé skupiny je vybrán jeden a jsou spojeny pomocí „+“.

V první skupině jsou parametry zapojení vstupu:

- NC
- NO
- DBALS

Druhá skupina obsahuje typové značení:

- NEPOUZIT
- NARUSENI
- POZAR
- SABOTAZ
- TISEN
- NOUZE
- MASKOVANI
- PORUCHA
- NEPOPLACH
- PORUCHA_230V
- PORUCHA_DC
- BATERIE_VYBITA
- PORUCHA_SIR
- ZRUSENI_NOUZE

4.6.5.3.2 Parametry výstupu

U výstupu jsou pouze dvě skupiny parametrů. Je vždy vybrán pouze jeden parametr z první skupiny a z druhé mohou být označeny parametry všechny. Hodnoty se také spojují pomocí znaménka „+“.

První skupina NC a NO

Druhá skupina:

- NEPOUZIT
- NARUSENI
- POZAR
- SVETLO
- ZAMEK
- KAMERA
- SABOTAZ_EXT
- SABOTAZ_INT
- NOUZE_SIR
- VYSTRAHA_INT
- BRANA
- NEPOPLACH
- NOUZE_LED
- TISEN_LED
- NARUSENI_POTVRZENE
- ODSTREZENO
- BYL_POPLACH NARUSENI
- BYL_POPLACH_POZAR
- BYL_POPLACH_SABOTAZ_EXT
- BYL_POPLACH_SABOTAZ_INT
- BYL_POPLACH_NARUSENI_POTVRZENE

5 Shrnutí výsledků

5.1 Zhodnocení splnění požadavků z analýzy

Vytvořená aplikace umožňuje načtení, zobrazení i překlad všech doposud používaných konfiguračních souborů. Všechny požadavky, které byly zadány analýzou byly splněny. Jedním z hlavních bodů analýzy byla možnost odeslání konfigurace do ústředny, aby nebylo nutné používat jiný software. Toho bylo dosaženo pomocí integrace externího překladače, který umožní překlad do binární podoby. Uživatel nemusí složitě ukládat svoji práci a otevírat jiný software. Vše je vykonáno automaticky při zvolení požadavku na odeslání konfigurace.

Uživatel je schopen pracovat s konfigurací, přidávat moduly, měnit jejich vlastnosti a také je mazat. Stejně tak je schopen upravit jakoukoli jinou potřebnou část konfigurace. Výhodou je, že všechny úpravy jsou kontrolovány tak, aby nedošlo k chybě. V případě, že uživatel kdekoli zadá špatnou hodnotu, je zobrazeno varování a musí být provedena oprava.

Další bod analýzy byla správa uživatelských přístupových karet. I tato část byla úspěšně splněna. Uživatel je schopen vyčíst karty z ústředny a také nemá problém je zpět nahrát. Kromě toho je možné je uložit i načíst ze souboru.

5.2 Ukázka uživatelského použití aplikace

Aplikace je distribuována jako samostatný exe soubor, takže není třeba žádná instalace. Stačí uložit soubor aplikace do samostatného adresáře. Aplikace si při prvním otevření extrahuje několik souborů důležitých pro chod aplikace. Většina je uložena ve složce *Temp* v adresáři uživatele a některé se extrahují do adresáře, kde je uložen exe soubor. Po spuštění se zobrazí obrazovka s úvodním menu.

Zde si můžeme zvolit:

- Nové s průvodcem – tvorba nové konfigurace
- Otevřít – načtení již vytvořené textové konfigurace
- Otevřít prázdné – otevře prázdnou konfiguraci s výchozím nastavením

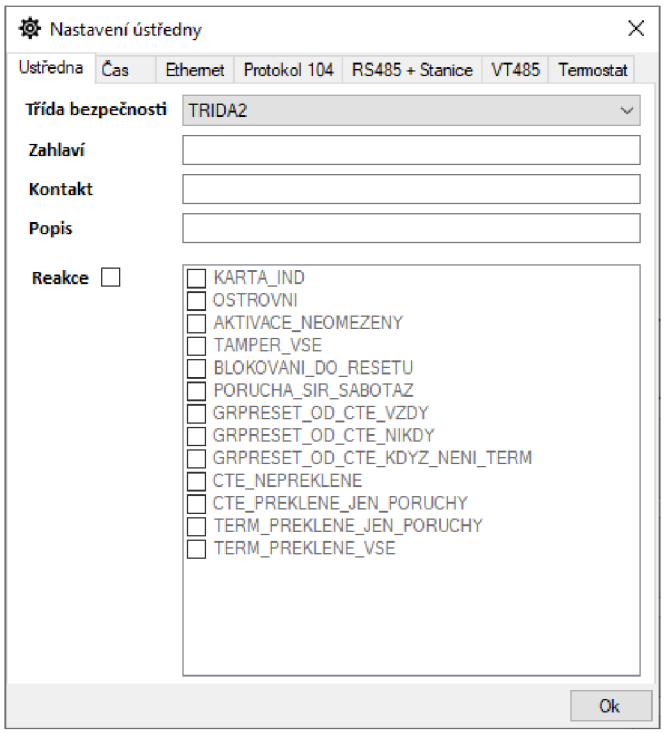
- Komunikace s EZS – otevře okno Komunikace s EZS, slouží k načtení konfigurace
- Konec – zavře aplikaci

5.2.1 Vytvoření nové konfigurace

Pokud uživatel zvolí volbu „Nové s průvodcem“, otevře se postupně formulář pro nastavení ústředny a nastavení zón. Následně je zobrazeno hlavní okno aplikace.

5.2.1.1 Nastavení ústředny

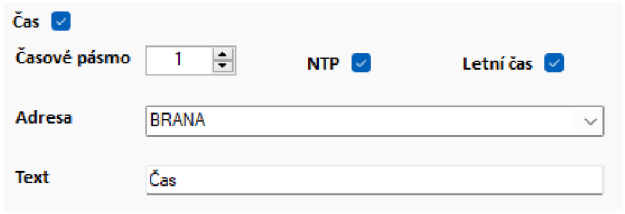
Tento formulář obsahuje několik záložek, do kterých je nastavení kvůli přehlednosti rozděleno. Na záložce „Ústředna“ je možné nastavit úroveň zabezpečení ústředny, texty „Záhlaví“ a „Kontakt“ umožňují nastavit texty, které se zobrazují na displeji terminálu. Pole „Popis“ umožňuje nastavení textu ústředny. Zaškrtávací políčko „Reakce“ nastaví, zda bude v konfiguraci zobrazen tag *REAKCE*. Po jeho zaškrtnutí je možné zvolit, jaké reakce mají být použity.



Obrázek 18 – Okno nastavení ústředny, vlastní zpracování

5.2.1.1.1 Nastavení času

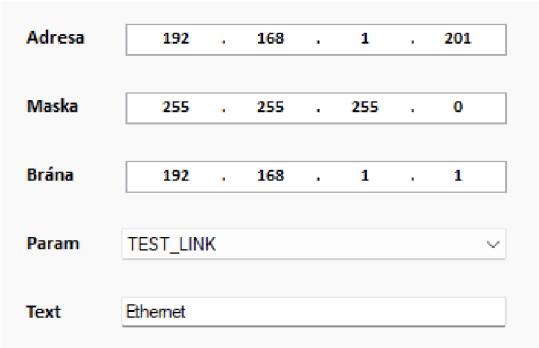
Na záložce „Čas“, najdeme nastavení času ústředny. Na této záložce je možné nastavit časové pásmo, letní čas nebo možnost získávat čas pomocí NTP a jeho adresu, která je ve výchozím stavu nastavena na síťovou bránu. Jako u všech prvků je možné i zde nastavit popis v poli Text.



Obrázek 19 – Nastavení času, vlastní zpracování

5.2.1.1.2 Nastavení ethernetu

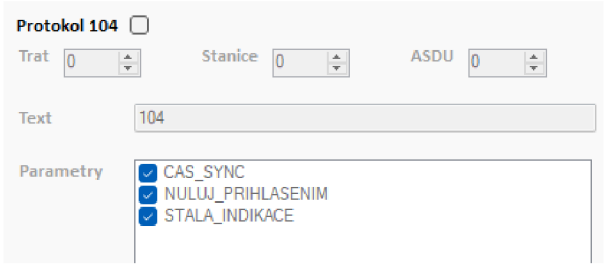
Nastavení ethernetu je nutné pro možnost vzdálené správy. Z bezpečnostních důvodů ústředna neumožňuje získání adresy pomocí DHCP a musí být vždy manuálně nastavena. Dále je možné nastavit parametr pro testování ethernetového spojení, například je možné vyvolat poplach v případě odpojení od ethernetu.



Obrázek 20 – Nastavení IP adresy ústředny, vlastní zpracování

5.2.1.1.3 Nastavení protokolu 104

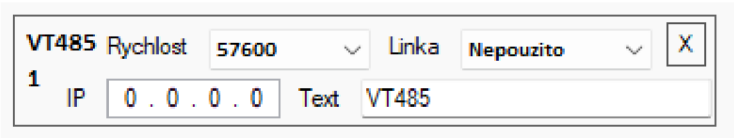
Tento komunikační protokol je používám v případě, že je systém nasazen v drážním prostředí. kde je tento komunikační protokol často využíván.



Obrázek 21 – Nastavení protokolu 104, vlastní zpracování

5.2.1.1.4 Nastavení RS485 a VT485

Pod těmito položkami je nastavení komunikačních linek alarmového systému. Lze nastavit tři hardwarové linky a až dvanáct virtuálních linek, které jsou následně připojeny pomocí ethernetu. Vzhledem k častému použití v prostorech staničních budov je možné nastavit stanici. U linek je možné nastavit jejich rychlost a určení, kterou linku zastupují. V případě nastavení virtuální linky je nutné ještě nastavit její IP adresu a popis pro přehlednost.



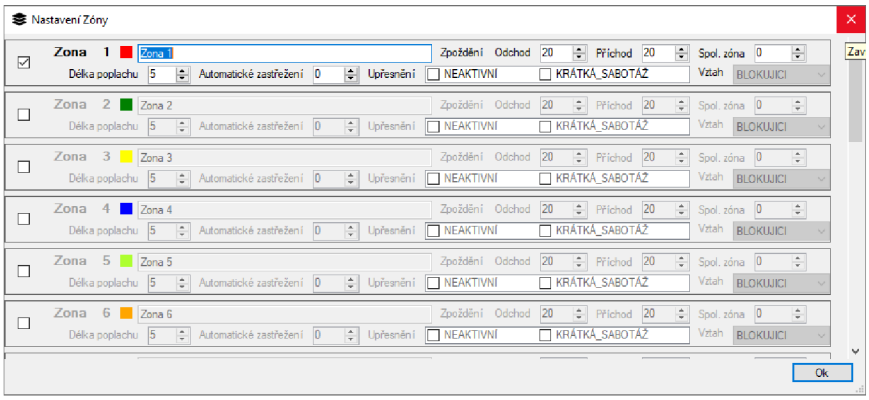
Obrázek 22 – Nastavení virtuální linky 485, vlastní zpracování

Obrázek 23 – Nastavení hardwarových linek 485 a stanice, vlastní zpracování

5.2.1.2 Nastavení zón

Po zavření následuje formulář pro nastavení zón. V tomto formuláři jsou zobrazeny všechny zóny a jejich možnosti. Ke každé zóně je možné nastavit příchozí a odchozí zpoždění, společnou zónu, délku poplachu, automatické zastřežení a také je možné

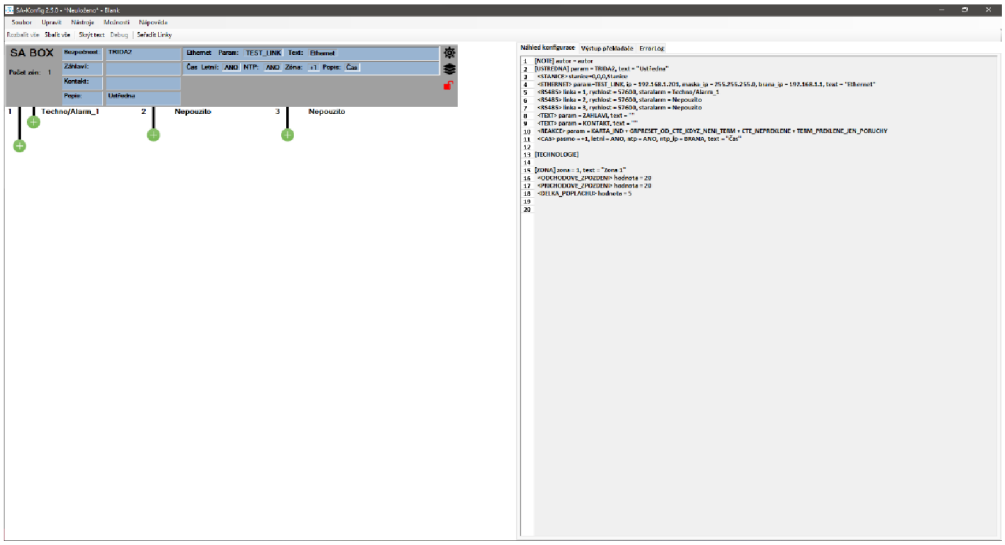
nastavit upřesnění zóny, tedy je možné jí deaktivovat nebo nastavit možnost krátká sabotáž. Pro pozdější přehlednost je vhodné si zónu pojmenovat. V případě nastavení společné zóny je možné ještě nastavit vztah.



Obrázek 24 – Nastavení zón, vlastní zpracování

5.2.1.3 Hlavní okno aplikace

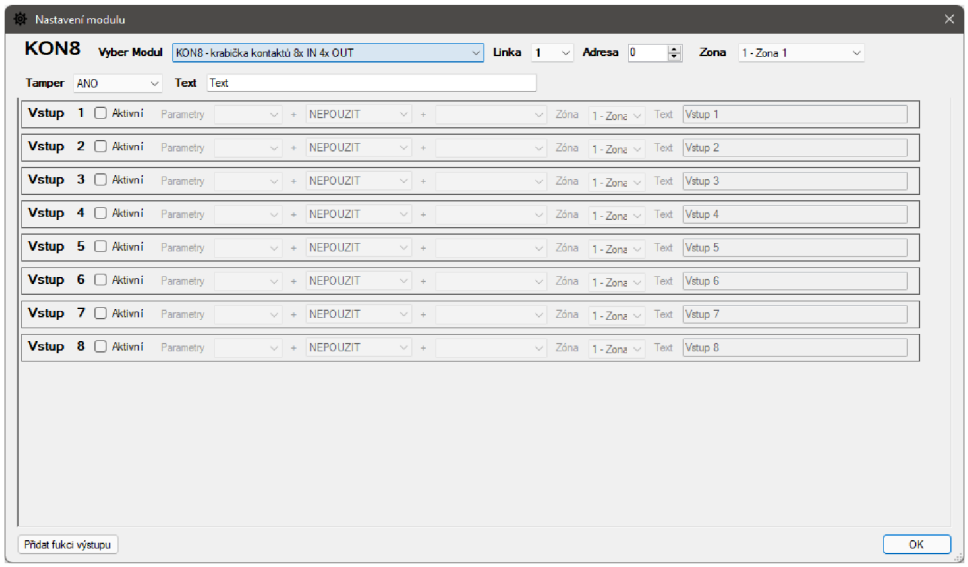
Po základních nastaveních se uživatel dostaneme do hlavního formuláře aplikace, kde vidíme ústřednu a všechny její linky. Ke každé lince můžeme pomocí tlačítka „+“ přidat jednotlivé moduly. U první linky je navíc ještě jedno tlačítko „+“, které slouží k přidání modulu *USTREDNA_IO*.



Obrázek 25 – Hlavní okno aplikace s prázdnou konfigurací modulů, vlastní zpracování

5.2.1.4 Vytvoření modulu

Při vytvoření modulu se uživateli otevře okno, kde je možné zvolit, jaký modul chce uživatel přidat. Dle typu modulu se zobrazí příslušný počet vstupů, které je možné jednotlivě nastavit dle potřeby. Pro přidání výstupů se v dolní části okna nachází tlačítko „Přidat funkci výstupu“. V tomto okně je možné také zvolit linku, na které se modul nachází a jeho adresu, která je předvyplněna softwarem na základě počtu již přidaných modulů. Pro všechny vstupy a výstupy je možné nastavit zónu, stejně tak i pro samotný modul. U některých modulů je možné zvolit jejich reakci.



Obrázek 26 – Formulář pro nastavení a přidání modulu, vlastní zpracování

6 Závěry doporučení

Aplikace značně ulehčila správu systému StarAlarm. Nyní je možné pohodlně používat jediný software, který dokáže zastoupit všechny potřebné náležitosti pro běžného uživatele tohoto systému.

Vzhledem k dnešním trendům jsem měl k vývoji aplikace zvolit spíše novější framework, například .NET Core. Tento framework by totiž umožnoval snadnější implementaci na jiné platformy, i když to nikdy u této aplikace nebylo nutné, protože se aplikace používá výhradně na systému Windows.

Také jsem udělal chybu ve výběru WinForms. Pro jakékoli menší a jednoduché aplikace je to velmi efektivní nástroj s užitečným designerem, ale pro účely této práce je to nedostatečné. V případě, že je načtena velká konfigurace, ve které se nachází větší počet modulů, aplikace nestíhá vykreslovat pohyb s konfigurací. Vzhledem k tomu, že WinForms nepodporují hardwarovou akceleraci grafického zobrazení, není možné urychlit proces vykreslení.

Také bych rád v budoucnu upravil některé formuláře, například formulář pro nastavení zón. Je velmi nepřehledný a také by mohl být v lepší podobě, například jako řádkový seznam. Ten by umožňoval přechod do nastavení zóny. Bylo by vhodné přidat možnost organizovat pořadí jednotlivých zón.

V budoucnu bych rád přidal možnost pro kontrolu konfigurace a její ladění. Vzhledem k velkému počtu starých konfigurací, které jsou z doby před používáním konfiguračního softwaru je to důležité pro jejich správné načtení a použití.

Práce mi umožnila zabývat se zajímavou problematikou usnadnění procesu konfigurace a správy bezpečnostního zařízení. Vnesla mi vhled do programování síťové komunikace a vytvoření rozsáhlé aplikace s mnoha funkcemi. Jsem rád, že se mohu podílet na vývoji takto užitečné aplikace.

7 Seznam použité literatury

1. **Černohlávek, Petr.** Elektronická zabezpečovací a monitorovací jednotka. *Univerzita Pardubice.* [Online] 10. květen 2013. [Citace: 28. červen 2021.] https://dk.upce.cz/bitstream/10195/52414/2/CernohlavekP_ZabezpecovaciJednotka_LH_2013.pdf.

2. **Dokulil, Zdeněk.** Modul elektrické zabezpečovací ústředny. *Vysoké učení technické v Brně.* [Online] 2010. [Citace: 30. Ledene 2022.] https://www.vut.cz/www_base/zav_prace_soubor_verejne.php?file_id=28004.

3. **Hladík, Drahoslav.** Střední odborné učiliště elektrotechnické, Plzeň . *Elektronické zabezpečovací systémy a elektronická požární signalizace.* [Online] 2010. [Citace: 6. Leden 2022.] https://www.souepl.cz/wp-content/ucitele/hladik/opvk2009/Ukazka-skripta/Skripta_ukazka.pdf.

4. **Martínek, Marek.** DOMOVNÍ ZABEZPEČOVACÍ SYSTÉM. *VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ.* [Online] 2018. [Citace: 30. Leden 2022.] https://www.vut.cz/www_base/zav_prace_soubor_verejne.php?file_id=174400.

5. **Křeček, Stanislav.** *Příručka zabezpečovací techniky.* místo neznámé : Cricetus, 2002. 8090293824.

6. **Autor, Neznámý.** *Slezská univerzita.* [Online] [Citace: 6. Leden 2022.] https://www.slu.cz/math/cz/knihovna/ucebni-texty/Ochrana-osob-a-majetku/Bezpecnostni-system-na-ochranu-majetku.pdf/.

7. **ada, lady.** PIR Motion Sensor. *Adafruit.* [Online] 15. Listopad 2021. [Citace: 8. Leden 2022.] https://cdn-learn.adafruit.com/downloads/pdf/pir-passive-infrared-proximity-motion-sensor.pdf.

8. **Heide, Patric.** Comercial microwave sensor technology: an emerging business. *Microwave Journal.* [Online] 1. Květen 1999. [Citace: 13. Leden 2022.] https://www.microwavejournal.com/articles/2641-commercial-microwave-sensor-technology-an-emerging-business.

9. **Contributors, Microsoft.** Introduction to C#. *Microsoft.* [Online] Microsoft, 7. Leden 2017. [Citace: 9. Srpen 2021.] https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/language-specification/introduction.

10. **Fawcett, Jim.** *Comparison of C++ and C#.* [Prezentace] Syracuse : Syracuse University, 2005.

11. **Microsoft.** Průvodce platformou .NET. *Microsoft Docs.* [Online] Microsoft. [Citace: 11. Srpen 2021.] https://docs.microsoft.com/cs-cz/dotnet/core/.

12. **Gorge, Andy De.** Desktop Guide (Windows Forms .NET). *Microsoft Docs.* [Online] Microsoft, 16. Listopad 2021. [Citace: 3. Leden 2022.] https://docs.microsoft.com/en-us/dotnet/desktop/winforms/overview/?view=netdesktop-6.0.

13. —. Overview of using controls (Windows Forms .NET). *Microsoft Docs.* [Online] Microsoft, 30. Červenec 2021. [Citace: 13. Leden 2022.] https://docs.microsoft.com/cs-cz/dotnet/desktop/winforms/controls/overview?view=netdesktop-6.0.

14. **Chapman, Michael.** ipaddresscontrollib. *Github.* [Online] 22. Srpen 2020. [Citace: 23. Červen 2019.] https://github.com/m66n/ipaddresscontrollib.

15. **Lottering, Martin.** CheckBox ComboBox Extending the ComboBox Class and Its Items. *CodeProject.* [Online] 29. Duben 2008. [Citace: 2020. Listopadu 16.] https://www.codeproject.com/Articles/21085/CheckBox-ComboBox-Extending-the-ComboBox-Class-and.

16. **Neznámý, Autor.** Wikipedia. *Protokol.* [Online] [Citace: 13. Březen 2022.] https://cs.wikipedia.org/wiki/Protokol_(informatika).

17. **Postel, Jon.** User Datagram Protocol. *HJP.* [Online] srpen 1980. [Citace: 10. srpen 2021.] https://www.hjp.at/doc/rfc/rfc768.html.

18. **Tirinda, Viktor.** vut.cz. *VYUŽITÍ PROTOKOLU TCP V SIMULAČNÍM.* [Online] 28. Květen 2008. [Citace: 28. Březen 2022.]

https://dspace.vutbr.cz/bitstream/handle/11012/18905/final-thesis.pdf?sequence=10.

19. **Matoušek, Petr.** Description and analysis of IEC 104 Protocol. *Fakulta informačních technologií VUT v Brně.* [Online] prosinec 2017. [Citace: 11. srpen 2021.] https://www.fit.vut.cz/research/publication-file/11570/TR-IEC104.pdf.

20. **Microsoft.** String.Contains Metoda. *Microsoft Docs.* [Online] Microsoft. [Citace: 26. Srpen 2021.] https://docs.microsoft.com/cs-cz/dotnet/api/system.string.contains?view=net-5.0.

21. **Lehman, Logan.** How can I run an EXE file from my C# code? *Stacoverflow.* [Online] STACK OVERFLOW, 13. Březen 2012. [Citace: 19. Leden 2020.] https://stackoverflow.com/questions/9679375/how-can-i-run-an-exe-file-from-my-c-sharp-code.

22. **Stroustrup, Bjarne.** *The C++ programming language .* Ann Arbor : Addison–Wesley, 2013. 978-0-321-56384-2.

23. **PCMag.** Definotion of NTP. *PCMag.* [Online] [Citace: 16. srpen 2021.] https://www.pcmag.com/encyclopedia/term/ntp#:~:text=A-,N,connected%20wall%20and%20desk%20clocks..

24. **Microsoft.** MessageBox Třída. *Microsoft Docs.* [Online] Microsoft. [Citace: 26. Srpen 2021.] https://docs.microsoft.com/cs-cz/dotnet/api/system.windows.forms.messagebox?view=net-5.0.

25. **Autor, Neznámý.** Wikipedia. *Framework.* [Online] 15. Prosinec 2020. [Citace: 4. Leden 2022.] https://cs.wikipedia.org/wiki/Framework.

26. **Gartner.** Gateway. *Gartner Glossary.* [Online] [Citace: 13. Listopad 2021.] https://www.gartner.com/en/information-technology/glossary/gateway.

27. **Jablotron.** Klávesnice a přístupové m. *Jablotron.* [Online] Jablotron. [Citace: 8. Leden 2022.] https://www.jablotron.com/cz/katalog-produktu/alarmy/jablotron-100/ovladaci-prvky/klavesnice-a-pristupove-m/.

28. **Microsoft.** String Třída. *Microsoft Docs.* [Online] Microsoft. [Citace: 19. Leden 2022.] https://docs.microsoft.com/cs-cz/dotnet/api/system.string?view=net-6.0.

29. **Microsoft Academic.** Broadcast. *Wikipedia.* [Online] Wikipedia. [Citace: 19. Leden 2022.] https://cs.wikipedia.org/wiki/Broadcast.

Oskenované zadání práce

UNIVERZITA HRADEC KRÁLOVÉ
Fakulta informatiky a managementu
Akademický rok: 2020/2021

Studijní program: Aplikovaná informatika
Forma studia: Prezenční
Obor/kombinace: Aplikovaná informatika (ai3-p)

Podklad pro zadání BAKALÁŘSKÉ práce studenta

Jméno a příjmení:	**Jakub Šmída**
Osobní číslo:	**I1900258**
Adresa:	Olešnice 24, Olešnice, 51736 Olešnice u Rychnova nad Kněžnou, Česká republika
Téma práce:	C# Aplikace pro konfiguraci bezpečnostního systému
Téma práce anglicky:	C# Application for configuration of security system
Vedoucí práce:	Ing. Jaroslav Langer Katedra informatiky a kvantitativních metod

Zásady pro vypracování:

Cílem práce je popsat technologie jazyka C# a bezpečnostního systému, pro který je určena výsledná aplikace. Aplikace musí načíst a uložit textovou konfiguraci, zobrazit grafický návrh konfigurace a přeložit konfiguraci do binárního kódu. Dále musí být schopna odeslat binární konfiguraci do alarmového systému pomocí UDP protokolu.
Cílem této bakalářské práce je vytvořit funkční aplikaci pro využití v praxi.
Osnova

- Úvod
- Cíl práce
- Teoretiká východiska
- Praktická část
- Shrnutí výsledků
- Závěr
- Použitá literatura

Seznam doporučené literatury:

Černohlávek, Petr. Elektronická zabezpečovací a monitorovací jednotka. *Univerzita Pardubice*. [Online] 10. květen 2013. [Citace: 28. červen 2021.] https://dk.upce.cz/bitstream/10195/52414/2/CernohlavekP_ZabezpecovaciJednotka_LH_2013.pdf.
Contributors, Microsoft. Introduction to C#. *Microsoft*. [Online] Microsoft, 7. Leden 2017. [Citace: 9. Srpen 2021.] https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/language-specification/introduction.
Microsoft. Průvodce platformou .NET. *Microsoft Docs*. [Online] Microsoft. [Citace: 11. Srpen 2021.] https://docs.microsoft.com/cs-cz/dotnet/core/.

Podpis studenta: Datum:

Podpis vedoucího práce: Datum: