TECHNICKÁ UNIVERZITA V LIBERCI

Fakulta mechatroniky a mezioborových inženýrských studií

Katedra Aplikované Informatiky Akademický rok: 2003/2004

ZADÁNÍ DIPLOMOVÉ PRÁCE

pro: **Tomáše Knajfla**

studijní program: M 2612 - Elektrotechnika a informatika

obor: 3902T005 - Automatické řízení a inženýrská informatika

Vedoucí katedry Vám ve smyslu zákona o vysokých školách č.111/1998 Sb. určuje uto diplomovou práci:

Název tématu: **Grafická nadstavba s možností ovládání požárního systému.**

Zásady pro vypracování:

1. Seznámení se elektronickou požární signalizací a ústřednou essertronic 8007.
2. Tvorba SW pro grafickou nadstavbu.
3. Možnost zobrazovat stav systému.
4. Možnost zápisu a kontroly historie událostí.
5. Aplikace klient/server (server je třeba naprogramovat na konkrétní typ zařízení).
6. Zabezpečený přenos dat klient/server.
7. Ovládání jednoduché s identifikací uživatele.

Rozsah grafických prací: dle potřeby dokumentace

Rozsah průvodní zprávy: cca 40 až 50 stran

Seznam odborné literatury:

[1] Libor Dostálek a kolektiv.: Velký průvodce protokoly TCP/IP: Bezpečnost 2.aktualizované vydání ,Computer Press Praha 2003.

[2] Josef Pirkl.: Komponenty v Delphi,Computer Press 2002.

[3] Orfali, R. et al: The Essential Client/Server Survival Guide, John Wiley & Sons 1994.

[4] Dudáček, A.: Požárně bezpečnostní zařízení (EPS). Ostrava, VŠB-TUO 1996.

Vedoucí diplomové práce: RNDr. Pavel Satrapa

Konzultant: Ing. Michal Suchánek (Vegacom)

Zadání diplomové práce: **23. 10. 2003**

Termín odevzdání diplomové práce: **21. 5. 2004**

L.S.

Vedoucí katedry

Děkan

V Liberci dne: 15.10. 2003

Studijní program: M 2612 – Elektrotechnika a informatika

Obor: 3902T005 – Automatické řízení a inženýrská informatika

Grafická nadstavba s možností ovládání požárního systému

(Grafic user interface with possibility controlling of fire system)

Tomáš Knajfl

Vedoucí práce: RNDr. Pavel Satrapa, Ph.D.

Technická univerzita v Liberci

Rozsah práce a příloh:

Počet stran textu: 54

Počet obrázků: 20

Počet tabulek: 2

Počet příloh: CD-ROM

Datum: 14.5.2004

Anotace

Cílem diplomové práce je realizace grafické nadstavby pro elektronickou požární signalizaci (EPS), jenž vizualizuje a ovládá požární hlásiče připojené k ústředně Essertronic 8007.

Realizace se skládá ze dvou aplikací.

1. Klientská - provádějící vlastní vizualizaci s ovládáním požárních hlásičů, připojená k serverové aplikaci přes protokol TCP/IP.
2. Serverová - dekódující data z EPS, jenž komunikuje s ústřednou přes sériový port.

Aplikace je základem pro komerční využití.

Summary

The target of my diploma work is a realization of grafic superstructure for an electronic fire site signalling (EPS), which visualitezes and control of fire detectors connected to fire alarm computer Essertronic 8007.

The realization consists of two application.

1. Client application - make self visualizing with controlling of fire detectors, which is connected to server application through protocol TCP/IP.
2. Server application – decoded data from EPS, which communicate with fire alarm computer through serial port.

Applocation is basis for commercial occupancy.

Prohlášení

„Místopřísežně prohlašuji, že jsem diplomovou práci vypracoval samostatně s použitím uvedené literatury."

Byl jsem seznámen s tím, že na mou diplomovou práci se plně vztahuje zákon č. 121/2000 o právu autorském, zejména § 60 (školní dílo) a § 35 (o nevýdělečném užití díla k vnitřní potřebě školy).

Beru na vědomí, že TUL má právo na uzavření licenční smlouvy o užití mé práce a prohlašuji, že **souhlasím** s případným užitím mé práce (prodej, zapůjčení apod.).

Jsem si vědom toho, že užít své diplomové práce či poskytnout licenci k jejímu využití mohu jen se souhlasem TUL, která má právo od mne požadovat přiměřený příspěvek na úhradu nákladů, vynaložených univerzitou na vytvoření díla (až do jejich skutečné výše).

V Liberci, dne 14.5.2004

Podpis: Tomáš Kugl

Poděkování

Rád bych touto cestou poděkoval vedoucímu diplomové práce RNDr. Pavlovi Strapovi, Ph.D. za podnětné rady a zodpovědné vedení. Dále bych chtěl poděkovat firmě VEGACOM a to především ing. Michalu Suchánkovi za ochotu a přístup při její realizaci.

V neposlední řadě bych chtěl poděkovat celé své rodině, přítelkyni a příbuzným za podporu při studiu i při tvorbě diplomové práce.

Obsah

Seznam obrázků

Seznam tabulek

Seznam příloh

Seznam použitých značek a zkratek:

EPS	Elektronická požární signalizace
OP	Osobní počítač
GN	Grafická nadstavba
TCP/IP	Přenosový a komunikační protokol
RS	Sériové rozhraní
LED	Světlo vyzařující dioda
IP	Internet address
DB	Databáze
UPC	Nepřerušitelný zdroj elektrické energie
API	Aplikační programové rozhraní
HT	Hvězdičkový tvar
VCL	Knihovna vizuálních komponent
RSA	Šifrovací algoritmus
COM	Sériové komunikační rozhraní

1 Úvod

Již před mnoha lety, kdy na planetě žili prehistoričtí živočichové, byl největším nepřítelem pro všechen život oheň. Tehdy se primitivním formám života nepodařilo tento živel podmanit, nebo ho alespoň zlikvidovat, jedinou možnou záchranou byl útěk. A proto v dnešní pokrokové civilizaci se výskytu požáru maximálně předchází, aby nedocházelo ke škodám na majetku a bezhlavému chaotickému útěku osob.

K tomuto účelu existují elektronické požární signalizace, které detekují výskyt požáru již v jeho začátcích. Na tyto systémy se vytvářejí grafické nadstavby, umožňující z jakéhokoli místa sledovat stavy prvků a ovládat daný systém (aktivace, deaktivace hlásičů).

Úkolem, jenž byl zadán firmou VEGACOM s.r.o., je vytvoření grafické grafické nadstavby pro elektronickou požární signalizaci, skládající se ze dvou aplikací, klientské a serverové, komunikující mezi sebou pomocí protokolu TCP/IP. Serverová komunikuje s ústřednou Essertronic7008, ke které je připojen přes sériový port RS 232, získaná data z ní dekóduje a posílá je klientské aplikaci, jenž realizuje vlastní zobrazení prvků se zápisem dat do souboru pro zpětnou kontrolu. Systém dovoluje aktivovat nebo deaktivovat jednotlivé prvky z grafické nadstavby.

2 Seznámení s EPS a ústřednou Essertronic © 8007

2.1 Význam včasné detekce vzniku požáru

Nezbytným předpokladem pro úspěšnou evakuaci osob, zvířat a materiálu při požáru a pro účinný protipožární zásah je včasné zjištění vznikajícího požáru [1].

Je proto výhodné neponechávat zjištění vzniku požáru a případně i provedení některých dalších operací v podstatě na náhodě, ale použít vhodného technického zařízení, které vznikající požár zjistí, vyhlásí požární poplach a případně provede i další potřebná opatření. K tomuto účelu slouží zařízení Elektrické požární signalizace.

2.2 Elektronická požární signalizace

Elektrická požární signalizace [2] slouží k včasné detekci a signalizaci vznikajícího požáru v objektu, kde je to vyžadováno dle požárního posouzení nebo na základě vlastního přání zákazníka. Samočinně nebo prostřednictvím lidského činitele urychluje předání informace o detekci osobám určeným k zásahu, případně uvádí do činnosti zařízení, která buď zabrání rozšíření požáru, nebo provádějí přímý protipožární zásah.

Detekce vzniká na základě signálů od čidel (nebo-li detektorů či hlásičů), které jsou rozmístěny tak, aby sledovaly celý střežený prostor. Tyto signály jsou dále vyhodnoceny přímo v detektoru, nebo předány dále k ústředně, která je vyhodnotí. Má-li ústředna na svém vstupu od detektorů signály odpovídající poplachovému stavu (tedy pravděpodobnost vzniku požáru), vyhlásí poplach a vykoná předem naprogramované funkce (ovládá ostatní protipožární zařízení, dálkově přenáší signál

poplach na určené stanoviště atd.). A tak předejde k ohrožení života či zdraví osob a velkým škodám na majetku.

2.2.1 Ústředny EPS

Základním a také nejdůležitějším prvkem je bezpochyby ústředna. Na světovém trhu je několik firem, zabývající výrobou ústředen EPS, my používáme ústřednu Essertronic 8007 viz 2.2.3.

Ústředny [2] komunikují s hlásiči tak, že jsou schopny v každém okamžiku znát aktuální hodnotu stavu hlásiče. Ústředna zajišťuje nepřetržité napájení požárních hlásičů nebo prvků EPS. Sdružuje a vyhodnocuje všechny informace, které nám posílají tyto prvky ze kterých, pak může generovat poplachový signál. Při poplachovém signálu lze ovládat připojené zařízení prostřednictví výstupních prvků (odblokování únikových cest, řízení vzduchotechniky, spouštění stabilního hasícího zařízení,..) a použít požární poplachové zařízení, které se používá při vyhlášení požáru, jedná se o zdroj zvuku či optickou signalizaci.

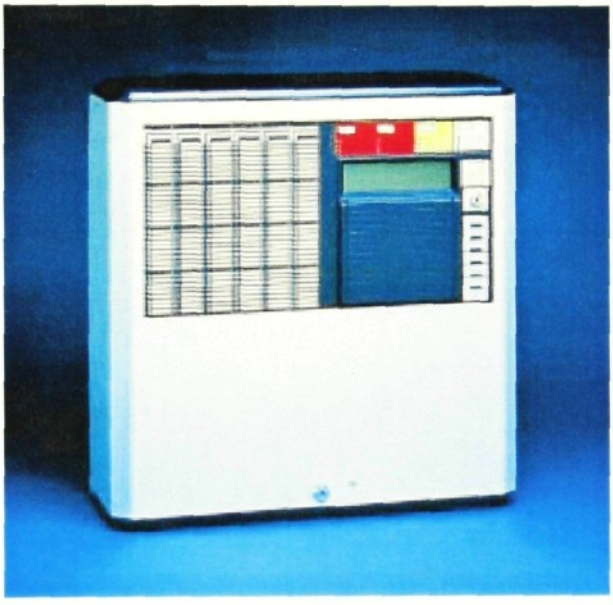
Ústředna zobrazuje data o aktuálních stavech, všech připojených hlásičů. Může komunikovat i nadstavbovými systémy (obsluhy systému z jakéhokoli osobního počítače) či možnost připojení systému EPS s pultem centralizované ochrany.

2.3 EPS – Essertronic® 8007

Ústředny EPS Essertronic 8007 [3], obr.1 jejímž výrobcem je rakouská firma ESSER reprezentují nejmodernější trend požární signalizace. Pomocí modulární koncepce výstavby je možné splnit téměř libovolné požadavky kapacity systému. Při

použití inteligentních hlásičů (multisenzory) je zaručená včasná detekce požáru při maximálním potlačení falešných poplachů.

Na kruhové vedení esserbus viz 2.3.1 je možné připojit 127 účastníků při délce vedení až 2 km.. Účastníkem vedení mohou být jak hlásiče požáru, tak vstupně-výstupní prvky, pro připojení speciálních hlásičů (např. lineární, nasávací atd.), nebo pro řízení a ovládání jiných technologii.

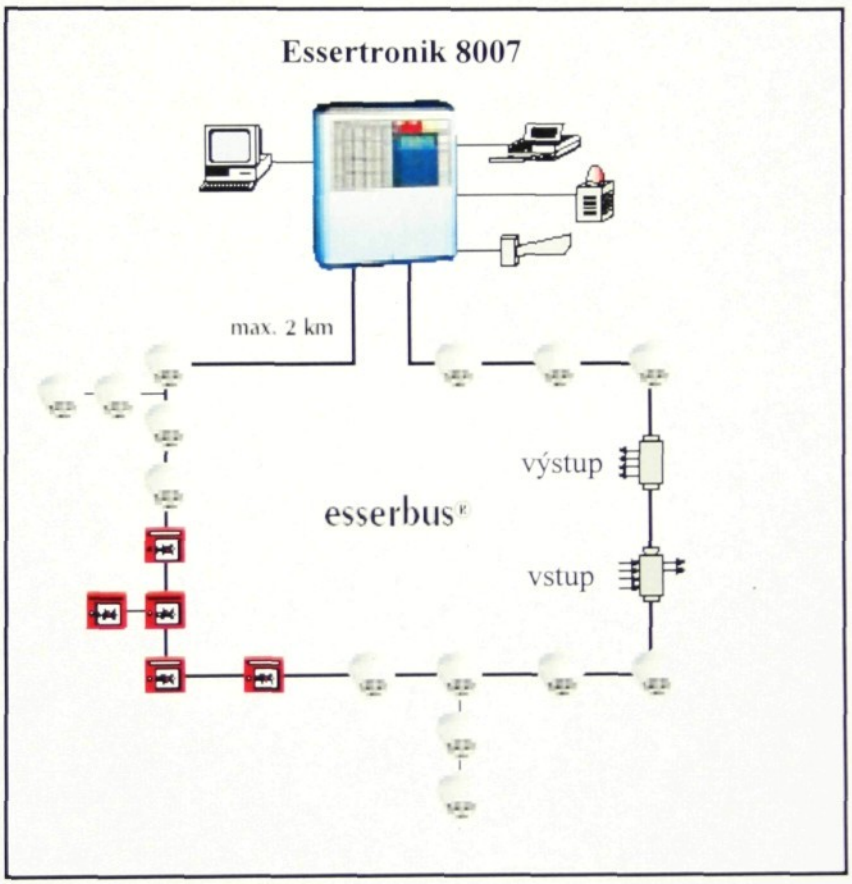


Obr.1: Ústředna EPS essertonic© 8007 [5]

Sběrnicová technika je využitá i při vzájemném propojení až 31 ústředen, PC stanic, ukazovacích a ovládacích tabel aj. Hlásiče připojené přímo na esserbus mají svoji vlastní inteligenci (mikropočítač) a vyhodnocují situaci přímo na místě. Z toho pak vyplývají další přednosti, jako je autodiagnostika celého systému, softwarová adresace, dálkový servis atd..

2.3.1 Esserbus®

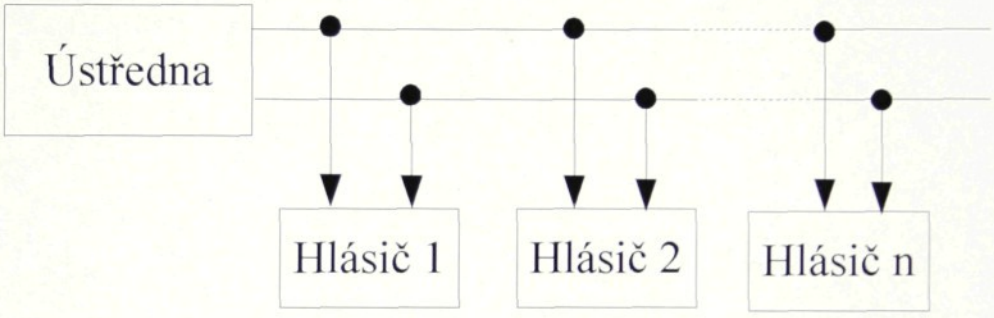
Micromodul Esserbus [4] je datové, z obou stran napájené a kontrolované 2 žílové vedení (obr.2) s kruhovou topologií pro připojení 127 vstupních a výstupních prvků tedy hlásičů a esserbus Kopplerů na ústřednu je omezen normou VdS popř. návrhem normy EN-54. Jestliže dojde k přerušení obvodu, tak všechny detektory v kruhové smyčce budou aktivní a funkční, nedojde k jejich odpojení.



Obr.2: Esserbus [5]

Topografie vedení esserbus umožňuje vytvořit na libovolném místě kruhu odbočku s tím, že prvky na kruhu a na odbočce jsou si funkčně rovnocenné.

Speciální busový protokol nepřetržitě monitoruje všechny účastníky kruhu, stav vedení a zabezpečuje přenos informací mezi účastníky vedení opatřené vlastní inteligencí a ústřednou. Hlásiče umístěné přímo na vedení esserbus, jsou zásadně vybaveny vlastní inteligencí a vyhodnocují situaci přímo na místě. Proto se zde často nasazují multisenzorové hlásiče 3D a 4D, kde je vlastní inteligence pro vyhodnocení velkého množství informací. Z toho vyplývají další přednosti, jako je softwarová adresace, dálkový servis hlásičů atd.



Obr. 3: Připojení hlásičů

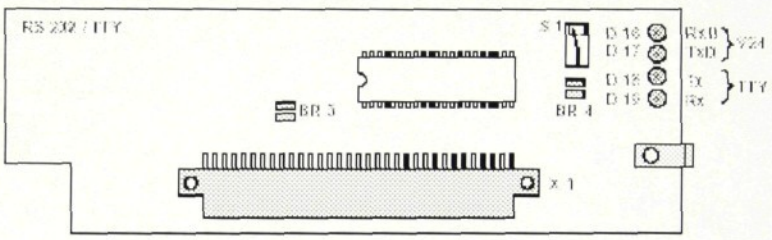
Ústředna komunikuje se senzorem pomocí paralelní adresace obr.3, tím že vyšle jeho adresu s dalšími povely (např. pro ovládání signalizace v senzoru apod.). Všechny senzory a adresovatelná zařízení na hlásicí lince dekódují adresy vysílané ústřednou. Senzor (hlásič), který dekóduje svoji adresu, přijme povely vysílané ústřednou a potom odpoví svým stavovým údajem, který obsahuje všechny potřebné informace pro ústřednu (naměřená hodnota sledované veličiny, druh senzoru, povely z ústředny uložené v paměti senzoru, adresa senzoru apod.).

2.3.2 Micro moduly

Micro moduly [6] jsou přídavné karty, které se vkládají do slotů ústředny. Moduly jsou použity jako vstupně/výstupní prvky (essernet, esserbus, RS232/TTY, atd.) pro komunikaci s okolím.

2.3.2.1 RS 232/TTY modul

Tento modul [6] poskytuje buď sériovou linku RS 232 nebo TTY rozhraní (20mA) pro výměnu dat s externím připojeným zařízením jenž obvykle bývá osobní počítač. V našem případě budeme používat pouze sériové rozhraní RS 232 ke komunikaci se servrem.



Obr.4: Modul RS232/TTY [6]

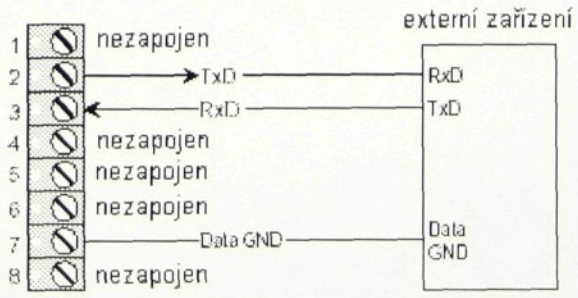
Podrobnější vysvětlení modulu RS 232/ TTY (obr.4):

X1 - 64 pinový konektor sloužící k připojení modulu do ústředny essertronic ®8007

BR 4 - konektor (jumper) který slouží k aktivaci svítivých diod (LED) D16 až D19.

D16,D17 - LED diody zobrazující přenos dat po RS 232

D18,D19 - LED diody zobrazující přenos dat po TTY



Obr.5: Detailní připojení externího zařízení

Detailní připojení externího zařízení k modulu přes sériové rozhraní obr.5. Maximální délka kabeláže pro RS 232 je 15 metrů, jenž je úměrná rychlosti přenosu, která může být max. 19200 bit/s.

2.3.4 Parametry ústředny Essertronic® 8007

Ústředna Essertronic® 8007 se skládá z několika modulů, jejímž hlavním modulem je základní (basic) deska na kterou jsou připojovány různé zařízení nebo mikro moduly. Tato ústředna je napájena ze sítě, která dobijí i vlastní dvě akumulátorové baterie.

Technické parametry [6]**:**

Princip napájení	: primární spínací zdroj
napájecí napětí	: 230/50Hz
Spotřeba	: 150VA
Nabíjecí napětí baterii	: 13.8 V
Kapacita baterii	: max. 80 Ah (2 x 40 Ah)
Provozní napětí	: 12 V DC
Klimatická odolnost	: R14 DIN 50019
Rozměry (Š. x V x H)	: 486 x 643 x 293 mm

2.4 Požární hlásiče obecně

Požární hlásiče [1] se používají pro sledování, měření a případně i vyhodnocování fyzikálních parametrů a jejich změn provázejících vznik požáru se používá hlásičů požáru.

Požární hlásič snímá jednu nebo více těchto fyzikálních veličin:

- Nárůst teploty.
- Přítomnost viditelných či neviditelných zplodin kouře.
- Infračervené či ultrafialové spektrální složky světla při hoření plamenem.

Výběr fyzikální veličiny (či jiného průvodního jevu požáru), kterou budeme snímat pro detekci požáru určíme na základě provozních podmínek, kam hlásič umístíme. Při špatné volbě se může stát, že hlásič detekuje tzv. "falešné poplachy", tj. signalizuje požár na základě běžných provozních podmínek střeženého prostoru. Hlásiče požáru (senzory) můžeme rozdělovat podle celé řady různých kritérií viz následující kapitoly. Základní rozdělení je na hlásiče tlačítkové a hlásiče samočinné.

2.4.1 Základní rozdělení hlásičů

Tlačítkové hlásiče [1]

Aktivují se prostřednictvím lidského činitele, který musí tuto změnu vyhodnotit a potom stiskem tlačítkového hlásiče předat údaj o požáru do ústředny EPS. Z pohledu planých poplachů je signalizace z tlačítkových hlásičů považována za velmi spolehlivou. Tlačítka v hlásičích bývají často vybavena buď mechanickou nebo magnetickou aretací stisknuté (aktivní) polohy. U takových hlásičů musí obsluha po signalizaci "Požár" nejprve hlásič odaretovat a teprve potom je možné zrušit signalizaci požáru na ústředně.

Samočinné hlásiče

Reagují na výskyt nebo změnu fyzikálních parametrů požáru, bez nutnosti zásahu lidského činitele.

2.4.2 Rozdělení samočinných požárních hlásičů

Teplotní hlásiče

- Hlásič teploty statický
- Hlásič teplot diferenciální
- Lineární tepelný detektor - teplotní kabel

Kouřové hlásiče

- Hlásič kouře ionizační
- Hlásič kouře optický

Optické hlásiče

- Optický hlásič plamene

2.4.3 Hlásiče EPS série 9200

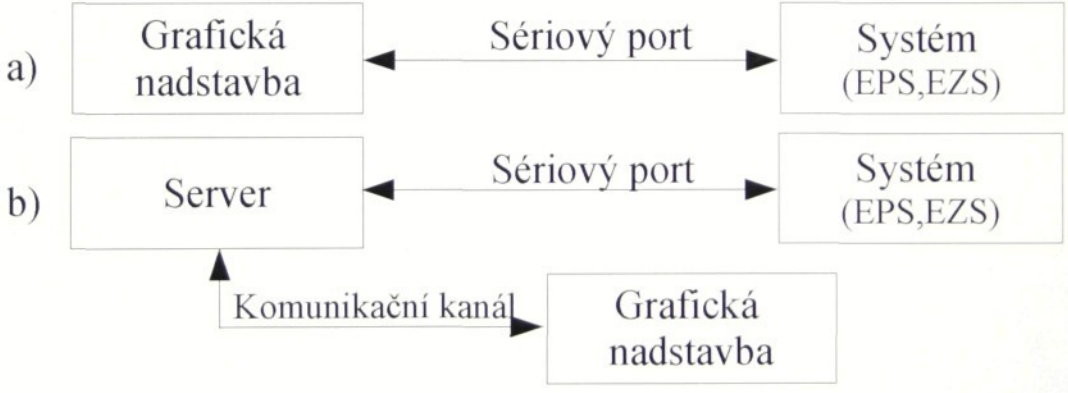
Hlásiče série 9200 [23] od firmy Esser jsou určeny výhradně pro ústředny EPS 8007. Obsahují vlastní inteligenci a veškeré zpracování dat provádí hlásič sám. Unikátní multisenzorová technika 3D a 4D umožňuje vícekriteriální rozhodování s konstantní citlivostí na různé druhy požárů. Při malé citlivosti na falešné a plané poplachy a vysokou citlivostí na pravé projevy hoření pomocí inteligentního algoritmu vyhodnocování časového průběhu jednotlivých signálových křivek.

2.5 Grafická nadstavba

Grafická programová nadstavba vytváří efektivní integrační prostředek pro vizualizaci a ovládání zařízení na zobrazovací jednotce, ve většině případu se jedná o monitor nebo displej. Nadstavba by měla být koncipována obecně tak, aby umožňovala integraci většiny zařízení s jednoduchou obsluhu.

Nadstavba komunikuje s ústřednou přes sériový port obr. 6 a) toto řešení se může používat pouze do 15 metrů a to je velmi malá vzdálenost, proto se v praxi nevyskytuje a používá se v kombinaci se serverem. Server přijímá data od ústředny a posílá je po komunikačním kanálu obr.6 b) do grafické nadstavby. Některé technologie komunikačních přenosů:

- LAN pomocí protokolu **TCP/IP,** tento přenos informací se používá nejčastěji pro jeho rychlost a vzdálenost.
- Sériový port RS 485, jehož délka vedení může být maximálně 1200 metrů.
- Pomocí telefonní linky přes internet TCP/IP. Drahý (placení telefonních poplatku).
- Jiné.



Obr. 6: Způsoby komunikace mezi EPS a GN

2.5.1 Grafická nadstavba pro EPS

Grafická nadstavba systému EPS je určena pro střední a velké objekty s velkým množstvím hlásičů a ovládacích prvků. Její používání lze realizovat i pro menší objekty s menší efektivitou (cena produktu).

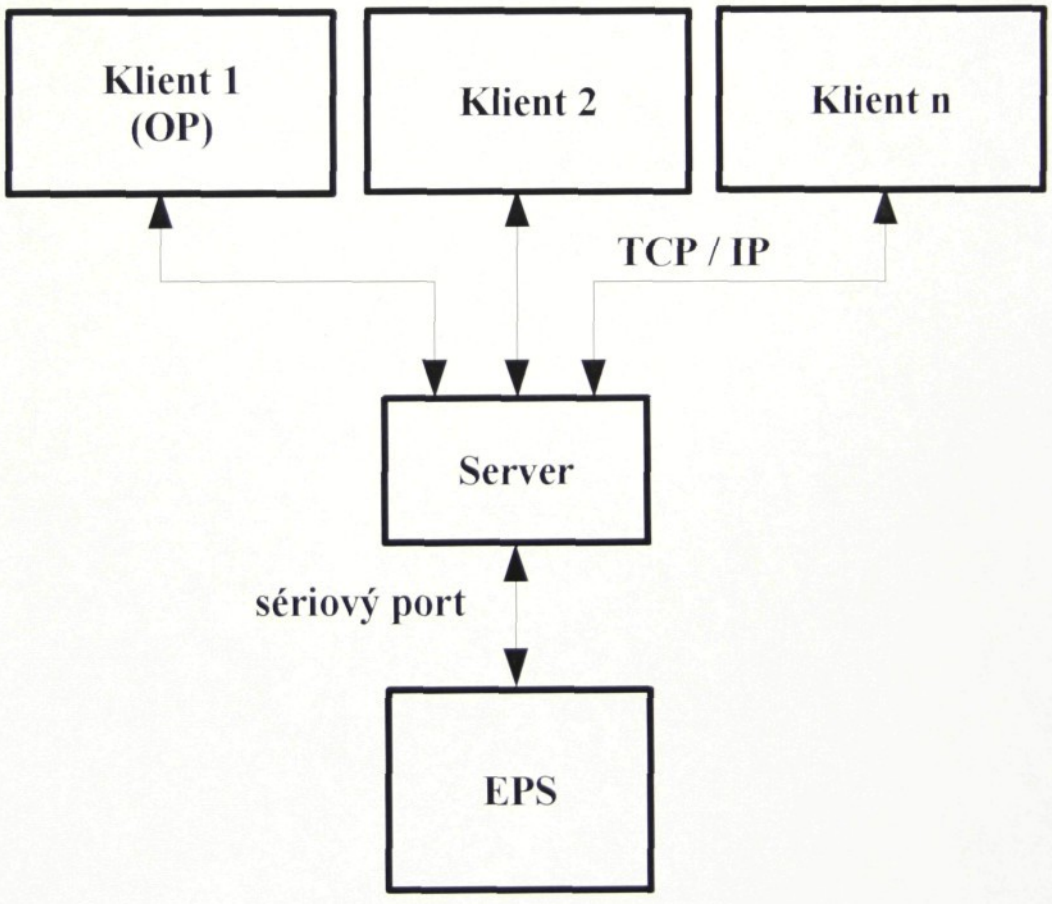
GN umožňuje:

- Dobrou orientaci obsluhy.
- Kvalitní a rychlou informaci o poplachu.
- Přehled aktuálních stavů V/V prvků.
- Možnost akustické signalizace.

3 Aplikační část

3.1 Popis aplikačního modelu

Hlavní blokové schéma grafické nadstavby na EPS se zobrazeno na obr.7. Jedná se o nadstavbu obr.6 b) se síťovou komunikaci TCP/IP mezi serverem a klientem na němž probíhá vlastní grafická vizualizace požární ochrany.



Obr. 7: Blokové schéma GN

Nadstavba se skládá ze dvou aplikací běžících na různých PC:

1. **Klientská** aplikace připojující se na server, jenž mu posílá aktuální data o hlásičích přes TCP/IP, které porovná se svoji databází a provádí zobrazení hlásičů na OP. Komunikace je obousměrná přičemž ve směru klient server probíhá pouze zřídka.

2. Aplikace **server** přijímající data o snímačích požáru, které posílá ústředna Essertronic© 8007 po sériové lince RS232, proto musí být server umístěn do 15 metrů od ústředny. OP s aplikací „server" by měl být napájen stálou elektrickou energií, čímž dosáhneme záložním zdrojem (UPS), aby nedocházelo k výpadku elektrické energie a tím k vypnutí aplikace. Která by přerušila komunikaci s klientem a došlo by k nefunkčnosti vizualizace v reálném čase.

3.2 Aplikace „Klient"

Jedná se o nejdůležitější univerzální a robustní aplikaci vytvořenou v programovacím jazyku Delphi 5 od firmy Borland©.

Program se skládá z hlavní formuláře (zobrazující prvky) a čtyř pomocných (přihlášení uživatele, vytvoření projektu, DB a legendy).

Základní činností „Klient" aplikace je zobrazování požárních hlásičů, které představují určité bitmapy, zobrazené na nadřazených bitmapách zobrazující aktuální patro objektu. Při změně stavu hlásiče se automaticky změní i bitmapa hlásiče a informace o stavu se zapíší do viditelného textové bloku a souboru (historie dat).

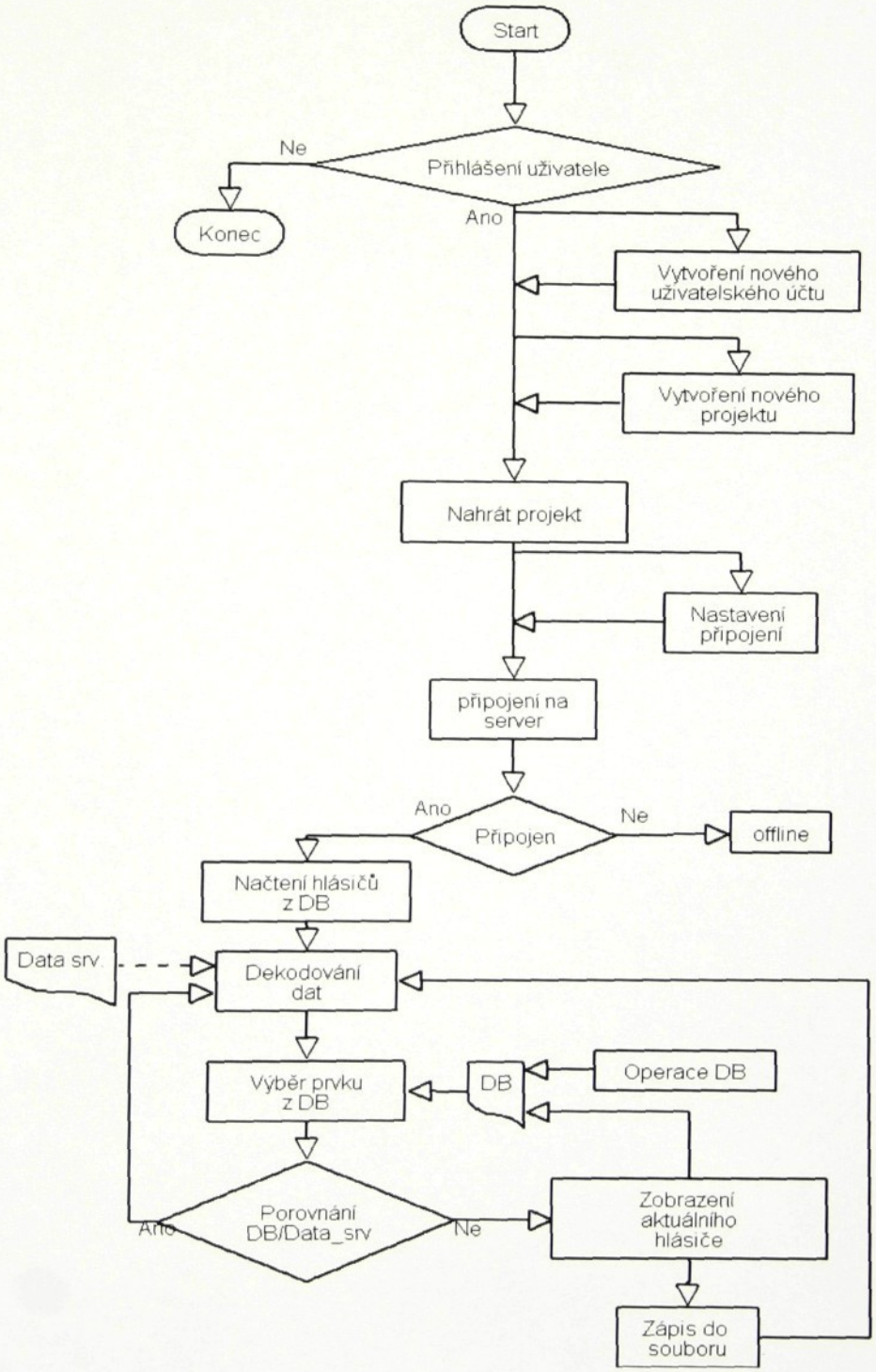
3.2.1 Popis algoritmu „Klient“

Algoritmus aplikace „Klient“ je zobrazen na obr.8. Jeho základem je spuštění konzolové aplikace. Abychom mohli používat aplikaci, musíme se nejdříve přihlásit jako právoplatný uživatel. Pro vytvoření nového uživatelského účtu, musíme mít určité uživatelské práva, které se nastavují při vytváření účtu.

Po přihlášení následuje nahraní projektu ze souboru je-li vytvořen. Projekt obsahuje název objektu (budovy, projektu, atd.), IP adresu k serveru a cesty k bitmapám (podlaží) zobrazené na monitoru ve formě záložek. Poslední akcí uživatele je připojení na server (aplikace „Server“). Navážel-li klient spojení se servrem dochází k automatickému načtení prvků z vlastní databáze hlásičů (DB) a následné zobrazení na jednotlivých patrech.

Při příchodu dat od serveru (Data_srv) se dekódují a převádějí na číslo hlásiče a jeho stav. Poté se porovnávají stavy db_stav (stav záznamu v DB) a stav_srv při stejném čísle hlásiče. Při jeho rozdílu nastává změna záznamu v DB a zobrazení nového stavu hlásiče se zápisem do textového souboru a lisboxu. A čekáme na nová data od serveru. Jsou-li data (stav) stejná s DB, stav nebyl změněn, tudíž nedochází ke změně bitmapy (hlásiče) a opět čekáme na nová data.

V algoritmu (obr.8) záměrně vynechávám některé bloky pro snadnější pochopení orientaci algoritmu. V dalších podkapitolách budou podrobně rozebírány jednotlivé bloky a soustavy bloků.



Obr. 8: Algoritmus aplikace „Klient“

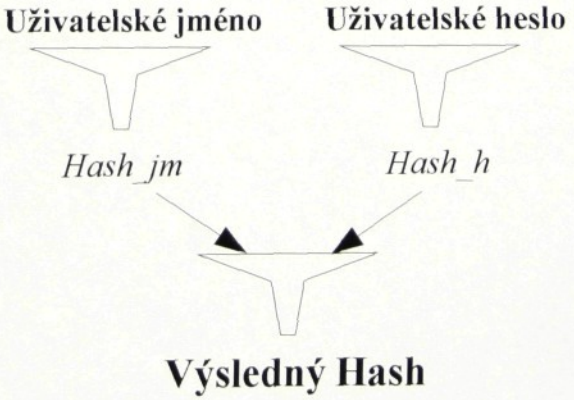
3.2.2 Ověření uživatele

Ověření uživatele (autentizace) je část procesu přihlašování, při kterém uživatel zadá svoje tajné heslo a tím potvrdí oprávnění na použití zadaného uživatelského jména.

Při ověřování uživatelského hesla se nejvíce používají HASH [10] kódy. Nejvýstižnějším názvem je však kryptografický kontrolní součet. Jedná se o jednosměrný algoritmus s jedinečnými vlastnostmi:

- Ze stejného vstupu vytváří vždy stejný výstup.
- Algoritmus by neměl být snadno odvoditelný či invertovatelný.
- Malá změna na vstupu má za následek velké změny ve výstupu.
- Ze vstupu proměnné délky vytváří malou hodnotu.

V mém případě používám ELFhash (**E**xecutable and **L**inking **F**ormat) pro hashování uživatelského hesla a uživatelského jména jeho použití je zobrazeno na obr.9.



Obr. 9: Algoritmus ukládání hashe

Z obrázku vidíme hashování jak uživatelského hesla (heslo), tak i jména (user) tyto hashe se násobí a z výsledku se provede poslední hash (hash) z obou hashů. Zde je ukázka algoritmu z obr.9:

```
user:=elfhash(a);                          //hesh jména; a=uziv_jm
heslo:=elfhash(b);                         //hesh hesla;b=uziv_heslo
hash:=elfhash(inttostr(user*heslo));       //hesh vysledný
```

Elfhash je funkce s návratovou hodnotou Integer → výsledkem je tedy kladné celé číslo, zdrojový kód hashovací funkce:

```
function TForm4.ElfHash(const Value: string): Integer;
var
  i, x: Integer;
begin
  i:=0;x:=0;
  Result := 0;
  for i := 1 to Length(Value) do
  begin
    Result := (Result shl 4) + Ord(Value[i]);
    x := Result and $F0000000;
    if (x <> 0) then
      Result := Result xor (x shr 24);
    Result := Result and (not x);
  end;
end;
```

Výsledky hashovací funkce ElfHash zobrazené v tab.1:

Vegacom	209549837
Tomáš Knajfl	45275660
45275660	140151520

Tab. 1: Výsledky hashovací fce.

Při získání výsledného hashe, porovná s hashem uloženým v inicializačním souboru, sloužícího k uchovávání uživatelských údajů, jestliže jsou stejné, dojde k akceptování uživatelského účtu.

Na obr. 10 je zobrazení přihlašovacího formuláře s možností vytvoření nového uživatelského učtu s určitými právy (vkládání se vygeneruje po zadání uživat. jména a hesla a kliknutím na tlačítko „Nový uživ."). Všechny informace o novém uživatelském účtu se ukládají do inicializačního souboru (Data.dat) ve formátu (hash výsledný = práva uživatele) :

[Users]
189252099=4
3437713=8

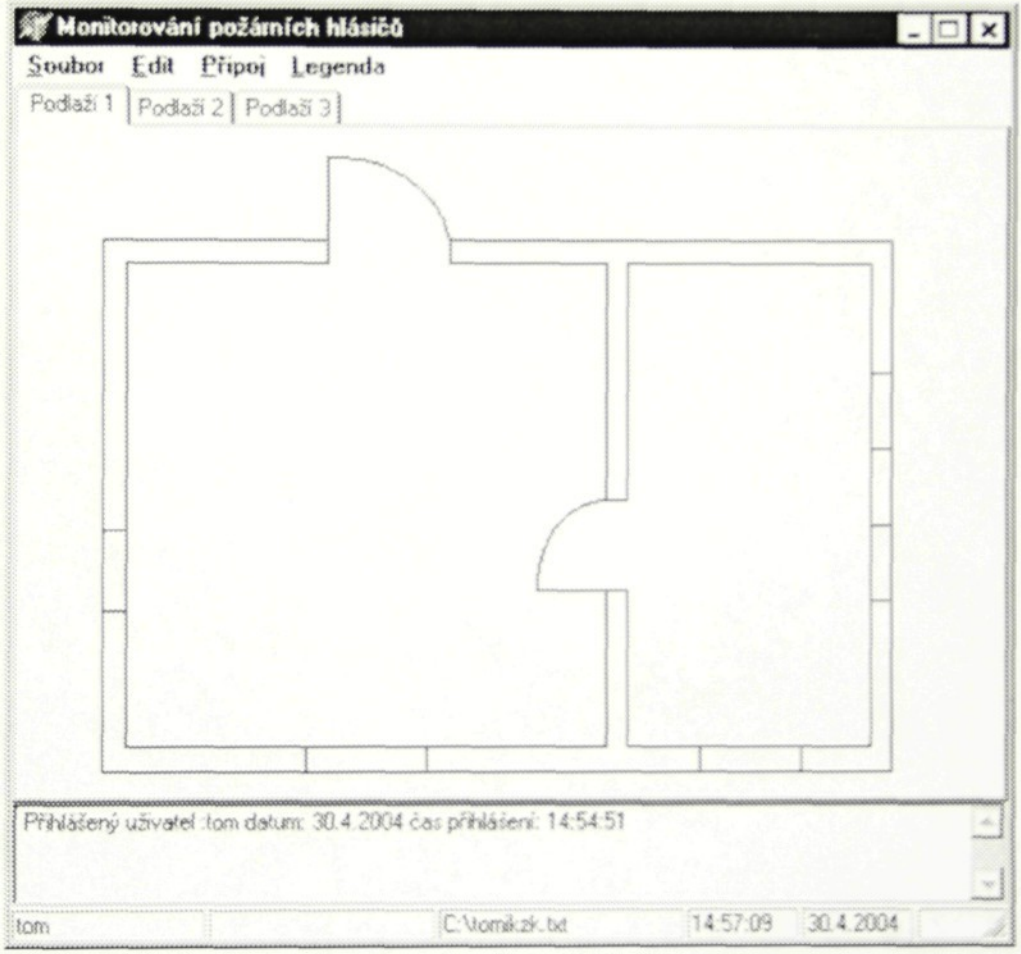
Obr. 10: Přihlašovací formulář

Program kontroluje zda již není uživatelské jméno obsazeno, tím ze hashuje nové uživatelské jméno a porovnává s uloženým hashem v ini souboru. Pro se ukládají i heshované uživatelské jména do skupiny [UsersName].

Pro zabezpečení se mohou používat i různé bezpečnostní prvky, jako např. čipová karta, biometrické čtečky pro kontrolu otisku prstů či identifikace podle oční rohovky, identifikace podle hlasu mluvčího nebo bezpečnostní klíče jako jsou např. USB token nebo Security box, atd.

3.2.3 Projekt

Jedná se o skupinu cest k bitmapám, představující jednotlivé půdorysy pater objektu, uložené v textovém souboru. Při nahráti projektu se automaticky vygenerují bitmapy, jejíž cesty jsou uložené v souboru. Jednotlivá patra zobrazujeme pomocí záložek a lze mezi nimi jednoduše přepínat. Z obrázku obr.11 máme aktivní třetí podlaží(bez hlásičů).



Obr.11: Zobrazení projektu (bez hlásičů)

Ukázka výpisu text. souboru (projektu) liberec.txt :

```
*VEGACOM*
3
```

```
127.0.0.1
C:\lib_1.JPG
C:\lib_2.JPG
C:\lib_3.JPG
Pospíšilova 15
```

1.ř. - Jedná se o kontrolní textový řetězec, kontrolovaný při načítaní projektového souboru.

2.ř. - Počet pater v projektu = P.

3ř. - IP adresa serveru ke kterému je připojena ústředna.

4.ř – (P+3).ř - Cesty bitmap.

(P+4) – N ř. - Popis modelu (podlaží), jediný je nepovinný. N, počet řádku souboru.

Obr.12: Vytvoření nového projektu

Neexistuje-li model projektu musí být vytvořen (v menu Edit → nový objekt). Po vygenerování nového formuláře (obr.12) sloužícího pro založení nového projektu musíme vyplnit následující povinné data, název souboru v našem případě „Liberec" , počet pater a IP adresu serveru. Poslední údaj je nepovinný a pouze informativní a slouží k popisu modelu či jiných dat. Po kliknutí na tlačítko „Ulož" se otevírají

dialogové okna a postupně vybíráme jednotlivé bitmapy (podlaží), které se ukládají do textového souboru „liberec.txt".

3.2.4 Připojení k serveru

Posledním uživatelským bodem je přihlášení na aplikaci „server". Síťová komunikace mezi serverem a klientem je realizována pomocí soketů, které dokáží navzájem komunikovat, a to typicky pomocí protokolu TCP/IP. Aplikace bude sloužit jako jednoduchý TCP/IP klient, takže bude umožňovat zadání IP adresy serveru a připojení k tomuto serveru. Další podporovanou operací bude zaslání dat serveru. V aplikaci využíváme komponentu `ClientSocket1`, jenž je součástí palety `internet`.

Základní vlastnosti komponenty clientsocker:

- Active - Aktivace klienta.
- Address - Nastavení IP adresy pro aktuální server.
- Port - Specifikace portu.

Události (events) komponenty clientsocker:

- OnConnect - Připojení na serveru.
- OnConnecting - Připojování.
- OnDisconnecting - Odpojování.
- OnRead - Čtení dat přicházející do serveru.
- On Write - Odeslání dat serveru.

Čtení dat se provádí pomocí funkce `ReceiveText: string` a na opak zápis dat na server funkcí `SendText(const S: string): Integer`, funkce vrací hodnotu nula jestliže transfer dat proběhl úspěšně v jiném případě (chyby) je hodnota nenulová. Obě funkce jsou součástí třídy `TcustomWinSocket`.

Nastavení IP adresy serveru ke kterému se chceme připojit, bylo provedeno již ve vytvoření nového projektu viz 3.2.3. Abychom mohli přistupovat k serveru musíme uvést také číslo portu, jenž je nastaveno globálně na hodnotu „2003".

Po vygenerování projektu, se pouze stačí připojit (v menu→ Připoj) v hlavním programu. Při úspěšném připojení se ve stavovém řádku hlavního programu zobrazí název PC, ke kterému jsme připojeni.

3.2.5 Databáze hlásičů

Nejzákladnějším prvkem algoritmu je databáze hlásičů, jenž sdružuje data potřebná k zobrazení hlásičů na jednotlivých patrech projektu. Popis DB tabulky zobrazené na obr.13:

`Cislo_prvku` – Klíčový údaj DB popisující číslo jednotlivého hlásiče.

`Stav_prvku` – Stav hlásiče (požár, zapnut, odpojen, ...).

`Cislo_patra` – Číslo patra na niž zobrazujeme prvek.

`Poznámka` – Nepovinný údaj, pouze informativní.

`X,Y` – Poloha daného hlásiče na aktuálním patře v pixelech .

`Vyska_H,Delka_H` – Rozměry zobrazeného hlásiče v pixelech.

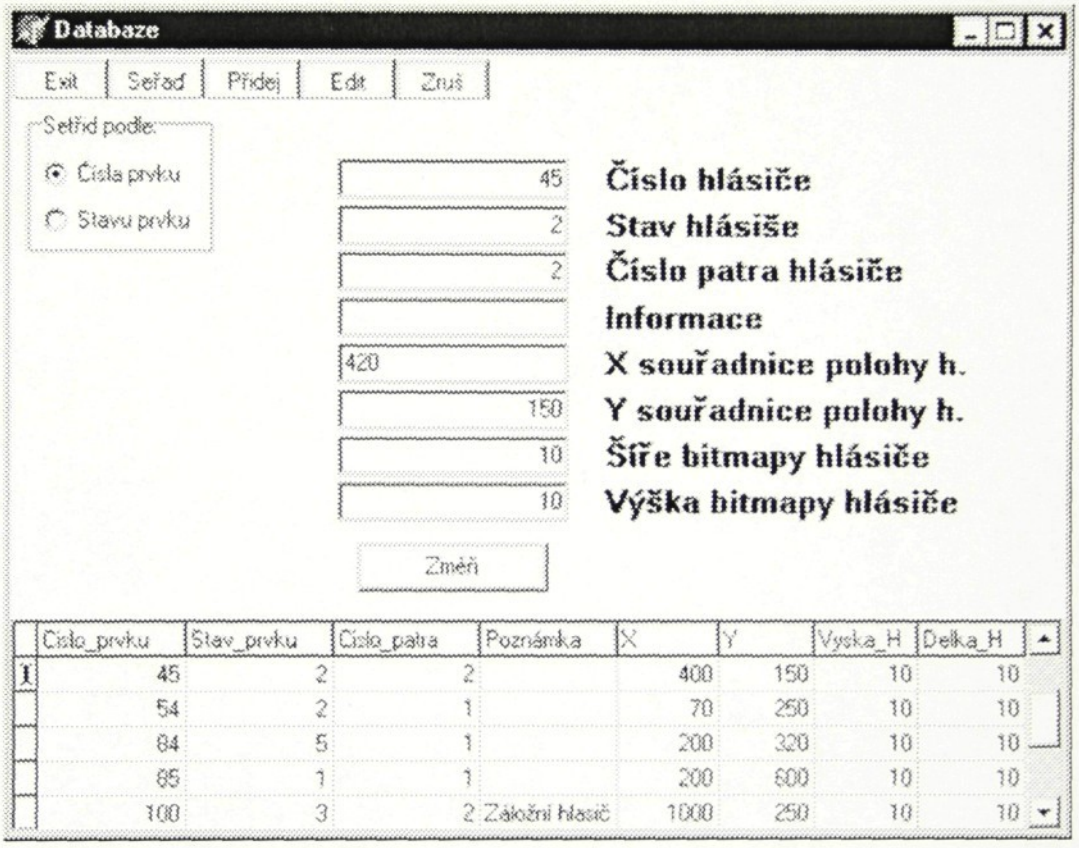
	Cislo_prvku	Stav_prvku	Cislo_patra	Poznámka	X	Y	Vyska_H	Delka_H
	1	4	1		200	400	10	10
	6	2	1		70	100	10	10
▶	21	2	1		290	100	10	10
	45	2	2		260	260	10	10
	54	2	1		70	250	10	10
	84	5	1		200	320	10	10
	85	1	1		200	600	10	10

Obr.13: Databáze hlásičů

První dvě informace, (Cislo_prvku, Stav_prvku) nám zasílá server a ostatní si

musí uživatel doplnit sám, podle výkresů ve kterých je zakreslena poloha prvku (hlásiče). Při nedoplnění informací, nemůže být prvek graficky zobrazen, informace o jeho stavu jsou pouze v textové formě, která se ukládá do souboru .

S databázovou tabulkou je možné provádět několik běžných operací zobrazených na obrázku obr.14:

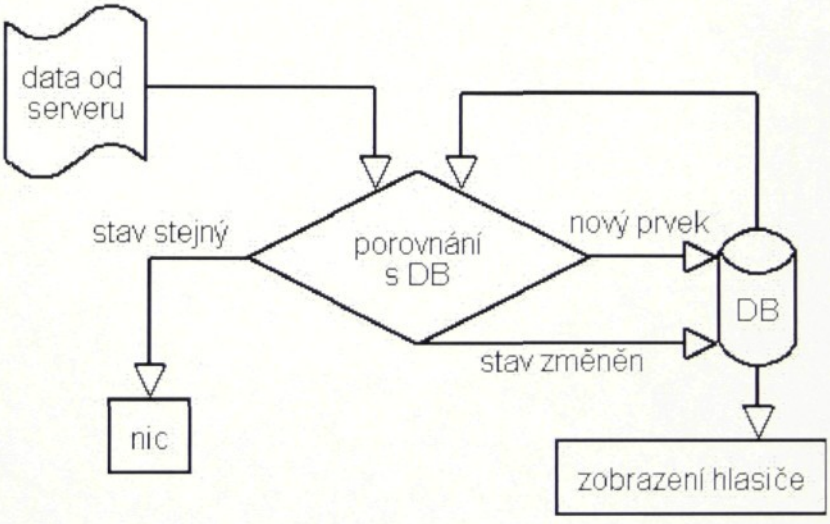


Obr.14: Operace s DB

- **Setřídění tabulky**(položka v menu „Seřad“) se provádí podle dvou kritérií: čísla prvků nebo stavu hlásičů.
- **Vložení nového záznamu** (položka v menu „Přidej“).
- **Editování** aktivního záznamu (položka v menu „Edit“), kromě položky „Cislo_prvku“, ostatní položky se mění v editačních polích zobrazený na obr.14.
- **Zrušení** vybraného záznamu (položka v menu „Zruš“).

3.2.6 Algoritmus zobrazování hlásičů

Jednotlivé dekódované prvky porovnáváme s databází hlásičů, při nenalezení čísla hlásiče (Cislo_prvku) v DB uživatel volí, zda-li se prvek do přidá DB či bude dokonce činnosti programu ignorován. Prvek přidaný do DB není zobrazen pro neúplnost informaci viz 3.2.5., proto při připojení se uživateli zobrazí informace o všech neúplných hlásičích. Je-li prvek v DB obsažen, pak rozhoduje údaj o stavu hlásiče (Stav_prvku). Při rozdílném stavu s DB ,se nový stav zapíše do DB, z obnovené DB se zobrazí prvek (bitmapa) s novým stavem. Stane-li se, že stavy jsou identické nedojde k žádné změně. Algoritmus je zobrazen na obr.15.



Obr.15: Algoritmus zobrazení hlásičů

Hvězdičkový tvar (přijímaná data)

Server nám posílá data ve "hvězdičkovém" tvaru **A*100+B*C***, přičemž **A** je číslo skupiny (1-9999) a **B** je číslo hlásiče (1-32), obě tyto hodnoty představují „Cislo_prvku" u vedené v DB. **C** představuje stav prvku „Stav_prvku" s rozmezí

hodnot (1-9), hvězdička ' * ' odděluje „Cislo_prvku" od „Stav_prvku" a ukončuje řetězec. Data převedeme na „Cislo_prvku" a „Stav_prvku", se kterými nadále pracujeme.

Př.: **A** (číslo skupiny) = 68, **B** (číslo hlásiče) = 12, **C** (stav prvku) = 3

výsledný HT 6812*3* , 6812 – číslo prvku uložené v DB .

Stav prvku

Každý hlásič má několik stavů, jenž charakterizují jeho činnost. Přehled stavů (obr.16) je graficky znázorněn i v hlavní aplikaci (v hl. menu→`Legenda`).

Popis jednotlivých stavů v aplikaci:

1 – Hlásič odpojen, deaktivace(data\odpojen.bmp).

2 – Zapnut, aktivace (data\zapnuto.bmp).

3 – Hlásí poplach (data\pozar.bmp).

4 – Technický alarm (data\tech_alarm.bmp).

5 – Porucha hlásiče (data\porucha.bmp).

Obr.16: Legenda hlásičů

V závorkách jsou uvedeny cesty k bitmapám, které si uživatel může libovolně upravovat či vytvářet nové v jakémkoli grafickém editoru.

3.2.7 Textové informace

Veškeré informace o hlásičích jsou zobrazovány, také jako textové zprávy ve spodní části hlavního formuláře (`form1.memo2`), uživatel tak přesně vidí jaký prvek právě změnil svůj stav. Všechny informace se zapisují do souboru historie dat (data\historie.txt), ve kterých lze zpětně dohledat jakékoli změny. Z ukázky lze zjistit kdy a jaký uživatel byl připojen či odpojen, tím získáme veškeré informace o prvkách, které mají jiný stav než-li aktivní (zapnut).

Ukázka souboru historie.txt:

```
Přihlášený uživatel : T_K datum: 29.4.2004 čas přihlášení: 22:45:09
 29.4.2004 22:45:13 Hlásič č.1 je v technickém alarmu
 29.4.2004 22:45:13 Hlásič č.84 je v poruše
 29.4.2004 22:45:13 Hlásič č.85 je odpojen
 29.4.2004 23:21:24 Hlásič č.1 je aktivní a hlásí nečinnost v okolí
Odhlášen uživatel: T_K datum: 29.4.2004 čas odhlášení: 23:45:32
```

3.2.8 Aktivace a deaktivace prvků

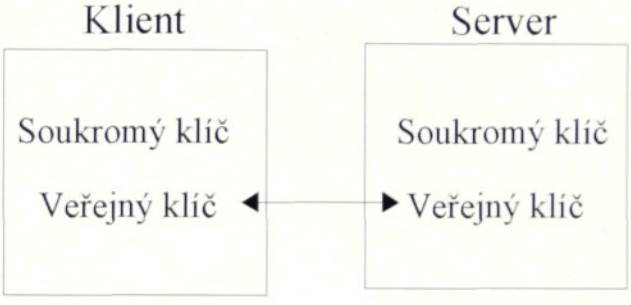
Aktivace či deaktivace prvků, se provádí kliknutím pravým tlačítkem myši nad hlásičem, klient odešle data serveru ve HT šifrované algoritmem RSA viz 3.2.9, který je ihned zpracuje dekóduje a posle ústředně, která nám pošle odpověď.

Pro tuto událost byla vytvořena procedura `akt_hla` reagující na stisk pravého tlačítka myši nad hlásičem. Procedura vyhodnotí je-li prvek ve stavu „odpojen“ (deaktivován) či v jiném (aktivován), a na základě výsledků nastaví do HT na pozici stavu v HT '1' (deaktivace) nebo '0' (aktivace) a odešle serveru.

3.2.9 Šifrování RSA

Algoritmus RSA je založený na asymetrickém kódování používající dva klíče: **soukromý** a **veřejný**. Cokoli je zašifrováno veřejným klíčem, lze dešifrovat pouze klíčem soukromým.

Dojde-li k navázání komunikace posílají se navzájem veřejné šifrovací klíče obr.17, se kterými se budu šifrovat data. Dešifrování se provádí soukromým klíčem. Přeruší-li se komunikace, veřejné klíče jsou nulovány v obou aplikacích. Šifrování a dešifrování se provádí v obou aplikacích.



Obr.17: Princip posílání klíčů

V aplikacích je požita freewarová komponenta LbRSA, jenž generuje oba klíče, každý klíč se skládá z mantisy a exponentu, délka šifrovacích klíčů je nastavena na 128bitů. Šifrování se provádí funkcemi `lbrsa1.EncryptString (data)` a naopak dešifrování `LbRSA1.DecryptString(š_data)`

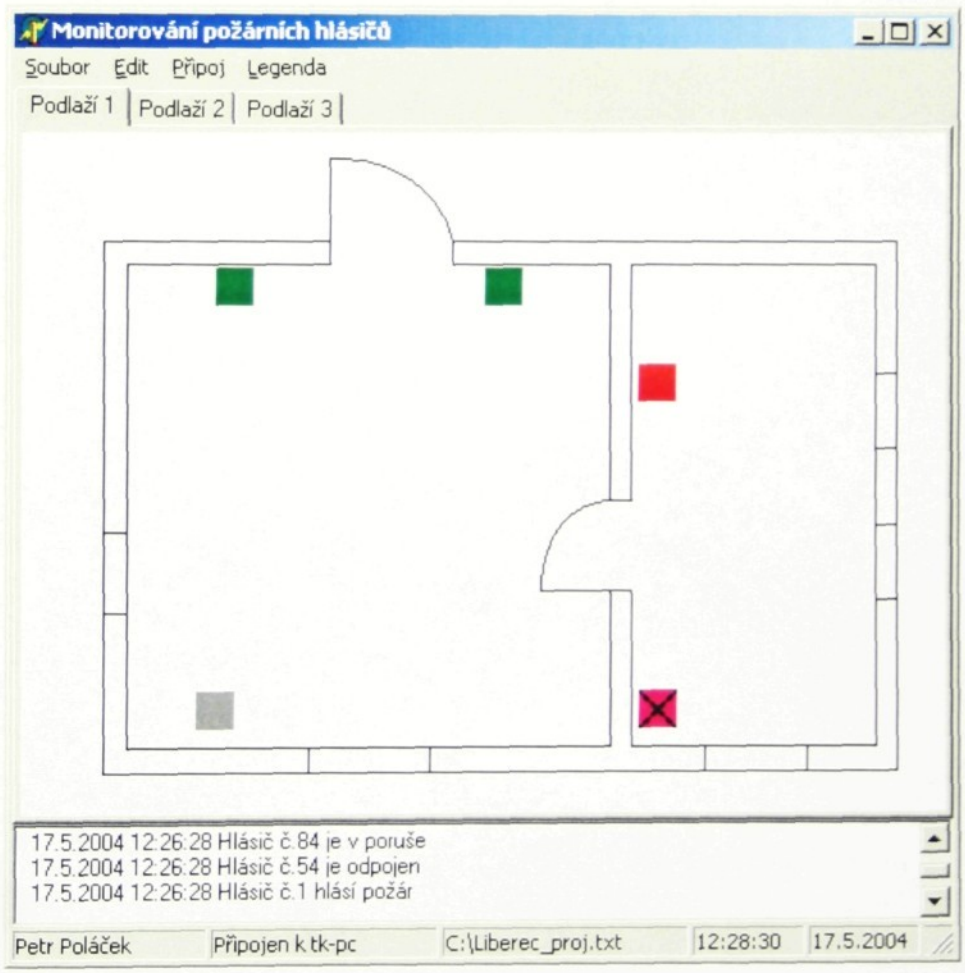
3.2.10 Uživatelské hlavní okno

Hlavní okno (`Monitorování požárních hlásičů`) zobrazené na obrázku (obr.19) je uživatelský přehledné, s textovými a grafickými informacemi o hlásičích. Důležité data o uživateli jsou uvedeny ve stavovém řádku hlavní aplikace, jenž se dělí na pět informativních panelů.

Informace ve stavovém řádku:

1) Jméno (Nick) přihlášeného uživatele.
2) Název serveru ke, kterému jsem připojeni.
3) Cesta k načtenému projektu.
4) Aktuální čas.
5) Datum.
6) Zobrazuje aktuální pozici kurzoru myši nad jednotlivými patry.

Z uživatelské nabídky (menu) ovládáme celou aplikaci prostřednictvím položek z nich některé jsou vnořené (Soubor, Edit).



Obr.18: Uživatelské hlavní okno „Klient"

Položky v menu:

- **Soubor** - Vnořená položka způsobující rozvinutí další úrovně menu.
  - **Nahrát projekt** – Načítá uložený projekt ze souboru.
  - **Offline** – Načtení hlásičů z databáze (nemusí být připojen na server)
  - **Přihlášení** – Změna přihlášeného uživatele.
  - **Konec** – Ukončení aplikace Klient.
- **Edit** – Vnořená položka.
  - **Databáze** – Spuštění DB formuláře (obr.13).
  - **Nový projekt** – Spuštění formuláře na vytvoření nového projektu (obr.12).
- **Připoj** – Připojí klientskou aplikaci k serverové.
- **Legenda** – Graficky zobrazí legendu o stavech hlásičů (obr.16).

3.2.11 Shrnutí aplikace „klient“

Vlastní uživatelský program reaguje, na změny stavů požárních hlásičů, které vizualizuje ve vytvořeném projektu a informace ukládá do textového souboru. Aplikace je přehledná, s možností uživatelské konfigurace a úpravou databáze hlásičů. Veškeré zdrojové kódy jsou obsaženy v příloze na CD-ROM.

3.3 Aplikace „Server“

Aplikace je vytvořena pro konkrétní ústřednu Essertronic 7008 se kterou komunikuje. Program je naprogramován také v softwarovém prostředí Borland Delphi 5, ale pouze jedním formulářem.

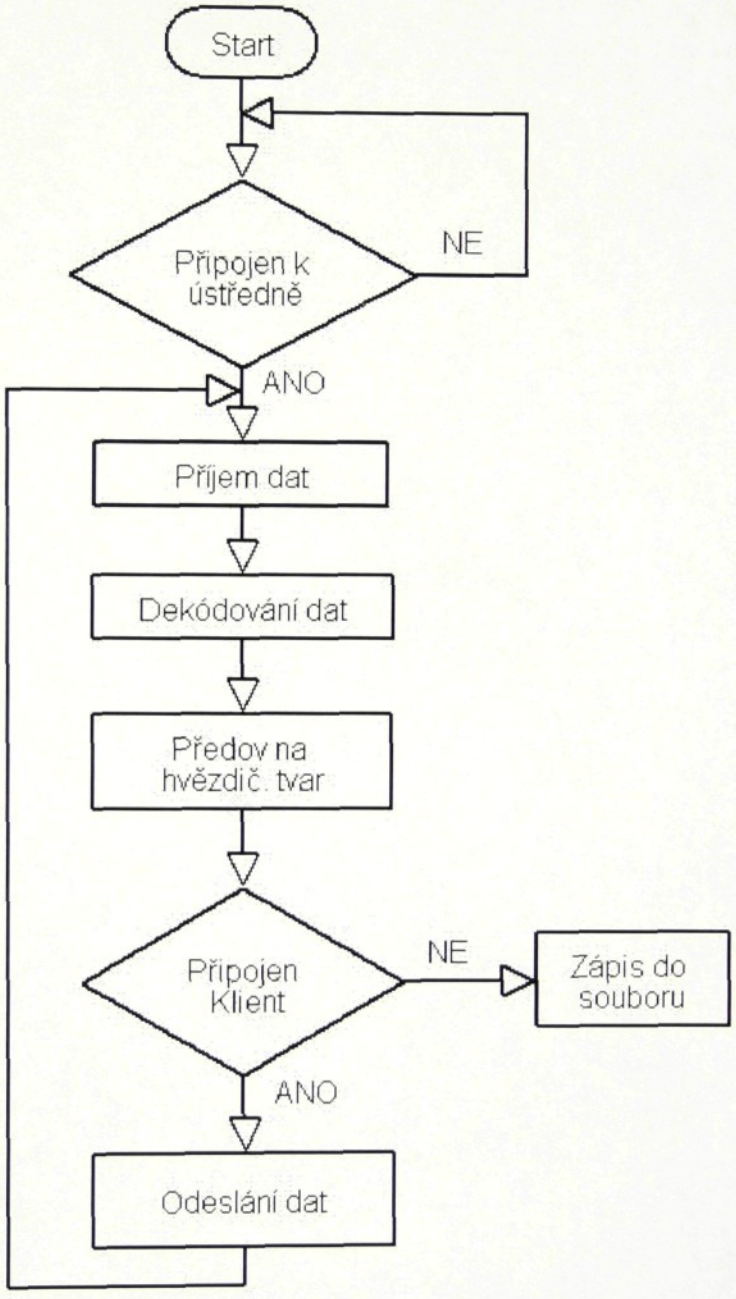
Základní činností „Server“ aplikace je dekódování protokolu BMZ8007/8008, jenž posílá ústředna, po převedení dat na hvězdičkový tvar se tyto data posílají do „klientské“ aplikace.

3.3.1 Popis algoritmu „Server“

Algoritmus aplikace „Server“ je zobrazen na obr.19. Jeho základem je spuštění konzolové aplikace, jenž čeká na připojení k ústředně Essertronic 8007. Při úspěšném připojení přijímáme data, která nám posílá ústředna. Data se dekódují a následně převedou na hvězdičkový tvar (HT) viz 3.3.6. Je-li k „serveru“ připojený klient posílají se mu data v HT, který je šifrovány algoritmem RSA, opačném případě (není-li připojen) se data ukládají do textového souboru. V dalších kapitolách budou jednotlivé bloky podrobně rozebrány.

3.3.2 Sériové připojení k ústředně

Připojení k ústředně Essertronic 7008 se provádí pomocí Win32 API funkcí. COM porty se chovají v systému téměř stejně jako soubory. Server se připojí sériovým kabelem k modulu RS232/TTY obr.5.



Obr.19: Algoritmus aplikace „Server“

API

Soubor programů, knihoven a rutin, které slouží k programování aplikací. API soubory obsahují definice obecně používaných, zejména ovládacích prvků (okénka, menu), způsob komunikace s uživatelem či periferiemi.

Použité Win32 API funkce v programu:

`Createfile` - Otevření sériového portu.

`Setupcomm` - Nastavení velikosti vyrovnávacích pamětí.

`Getcommstate` - Zjištění nastavení sériového portu.

`Setcommstate` - Nastavení sériového portu.

`WriteFile` - Zápis dat na sériový port.

`ReadFile` - Čtení dat ze sériového portu.

`CloseHandle` - Zavření sériového portu.

`ClearCommError` - Informace o chybách při komunikaci.

Zdrojový kód programu k připojení ústředny :

```
procedure TForm1.Button4Click(Sender: TObject);
var
  Pname: Pchar;
  dcb: Tdcb;
begin
  case radiogroup1.Itemindex of          // výběr portu se kterým
    1: pname:=pchar('com1');             // komunikujeme
    2: pname:=pchar('com2');
  end;
    hcom:= createfile(pname, generic_read or generic_write,
          0,nil,Open_existing,0,0); // otevře následující objekt a vrátí
handle
  if hcom <> invalid_handle_value then  //ověří je-li fce. úspěšná
  begin
    setupcomm(hcom,2048,2048);
    getcommstate(hcom, DCB);
    with dcb do
    begin
      baudrate:= 600;                          // komunikační rychlost
```

```
      bytesize:= 8;                                    // počet bitů
      stopbits:= onestopbit;                           // stop bity
      parity:= noparity;                               // parita
     end;
    fcom:=setcommstate( hcom,dcb); // vrací nenulové číslo je-li fce. v
                                                       // pořádku
      if fcom then label2.Caption:=('Připojen'); // je-li připojen na
                                                       // port
   end;
end;
```

Vlastnosti (komunikační rychlost, počet bitů, stop bity, parita) komunikačního sériového přenosu jsou nastaveny, aby odpovídali nastavení ústředny Essertronic 7008 (protokolu BMZ8007/8008) viz. 3.3.4

3.3.3 Příjem dat z ústředny

Přijímání dat z ústředny se rovněž provádí pomocí API příkazů viz 3.3.2.

Část kód programu pro příjem dat :

```
procedure TForm1.Timer1Timer(Sender: TObject);
var                              // deklarace proměných
  i: integer;
  err,r: dword;                  // double word (4 byty integer)
  cstat: tcomstat;               // struktura obsahující informace o
komunikačním                                         // zařízení
begin
  if fcom  then                  // připojen
  begin
    clearcommerror(hcom, err,@cstat);
    while cstat.cbInQue > 0 do // číslo bitu přijímajícího od zařízení
     begin
       readfile(hcom,b,1,r,nil); // čtení dat z portu a zápis proměnné b,
       clearcommerror(hcom, err,@cstat);//pokračování proc. v příloze na
                                                       CD
```

3.3.4 Dekódování dat z ústředny

Ústředna Esssertronic 8007 posílá po sériové lince protokol BMZ8007/8008 [13], který musíme rozluštit a převést na: stav (6 bitů), číslo skupiny (15 bitů) a číslo hlásiče (6 bitů). Protokol se skládá z 6 bytů, jejich rozpoznání zjistíme nahrubo z prvních dvou bitů.

Protokol BMZ8007/8008

1. Byte – Tento byte může mít pouze šest hodnot, jedná se tudíž o stav prvku. Jeho hodnoty s přiřazenými stavy a prioritami jsou uvedeny v následné tabulce (Tab.2).

Hodnota (Hex)	Stav prvku	Priorita
28H	požár	6
04H	porucha	5
38H	technický alarm	4
58H	odpojen	3
68H	zkušební stav	2
60H	zapnuto	1

Tab.2

2. Byte – Ve tvaru $\mathbf{o_5\ o_4\ o_3\ o_2\ o_1\ o_0\ 0\ 1}$, kde **o** jsou bity č. skupiny (low) první dva bity slouží k rozpoznání bytu.

3. Byte – $\mathbf{b_6\ a\ a\ a\ o_7\ o_6\ 1\ 0}$, zde **b** je nejvyšší připravenost bitu pro č. skupinu (high), bity **a** jsou pouze pomocné, jejich posloupnost je 0 0 0.

4. Byte – $\mathbf{b_5\ b_4\ b_3\ b_2\ b_1\ b_0\ 1\ 1}$, bity pro č. skupinu (high).

5. Byte – m_5 m_4 m_3 m_2 m_1 m_0 0 1, kde **m** jsou bity čísla prvku ve skupině.

6. Byte – 0 0 1 1 0 0 1 0, poslední a kontrolní bytu přenosu informací jednoho prvku (stav, č. skupina, číslo prvku).

Číslo prvku jednoduše převedeme z binárního 6 bitového čísla (5.byte) na decimální. U čísla skupiny je to již složitější, to se skládá ze 15 bitů: $b_6b_5b_4b_3b_2b_1b_0$ $o_7o_6o_5o_4o_3o_2o_1o_0$, jež také převedeme na decimální číslo. A získáme veškeré informace o stavu prvku.

Ukázka dekódování bytů 1 a 2:

Při dekódování prvního bytu používám příkaz **case**, na zbývající byty jsem vytvořil dvě funkce (**nastav, zjisti**).

```
if zjisti(0,pom) = 0 then          // testování 0 bitu
  if zjisti(1,pom) = 0 then        // testování 1 bitu
    begin
      ss_code:=true;               // je-li testování v pořádku
      c_hlasice:=0;                // nulování proměnných
      objekt:=0;
      bn:=0;
      bajt:=1;                     // nastavení bytu
        case pom of
          $58: stav:=1;            //odpojen
          $60: stav:=2;            //zapnuto
          $28: stav:=3;            //požár
          $38: stav:=4;            //technický alarm
          $04: stav:=5;            //porucha
        else
          begin
            beep;                  // zvuková signalizace při neznámé stavu
          end;
        end;
    end
  else
    begin
      if bajt = 1 then                    // testování prvního bytu
        if zjisti(0,pom) = 1 then         // testování 0 bitu
          begin
```

```
if zjisti(1,pom) = 0 then          // testování 1 bitu
  begin
    for i:=2 to 7 do
     if zjisti(i,pom) = 1  then
      objekt:=nastav(i-2,objekt);// nastavení 6 bitu
                                    skupiny
    bajt:=2;                      // nastavení druhého
    exit;                         // konec procedury
  end
    else
      ss_code:=false;
  end
    else
      ss_code:=false;
```

Použité funkce:

1. Nastavení bitu (co) na zvolenou pozici.

```
function Tform1.nastav (poz: integer; co: byte): byte;
var
c_bit: integer;
begin
  c_bit:=trunc(Power(2,poz));
  result:= co or c_bit;
end;
```

2. Vrátí hodnotu bitu při zadané pozici (poz).

```
function Tform1.zjisti (poz: integer; co: byte): byte;
 var
 c_bit: integer;
 begin
   c_bit:=trunc(Power(2,poz));
   if  (co and c_bit) <> 0  then result:= 1;
    else
      result:= 0;  end;
```

3.3.5 Odeslání dat ústředně

Po příjmu data z klientské aplikace ve HT viz 3.2.9, se musí nejprve převést data na číslo hlásiče (c_hlasice) a skupinu (skupina), kterou rozdělíme na dva byty (skup_b1, skup_b1) viz 3.3.4. Po zjištění, zdali de o aktivaci či deaktivaci viz 3.2.9 na plníme byty, podle protokol BMZ8007/8008 viz 3.3.4 s využitím funkcí

nastav, zjisti. Po naplnění, odešleme všechny byty pomocí funkce `writefile (hcom,s[1],length(s),w,nil)`, po sériovém portu do ústředny.

3.3.6 Převod na hvězdičkový tvar

Převádíme na HT pro snadnější získání hodnot na straně klienta viz 3.2.6. Pro převod dat byla vytvořena funkce P_TCP jenž vrací celý HT řetězec. Každý hlásič je zapsán do textového souboru (HistorieS.txt).

```
function   Tform1.P_TCP(stav_a,objekt_a   ,bn_a   ,c_hasice_a   :
integer):string;
var
  c_hla,skupina: integer;          // pomocné poroměné
  c_str,a1,vys: string;
begin
  a1:=inttohex(bn_a,2);            // převod b bitů na hex.(high)
  vys:='$'+vys+a1;
  a1:=inttohex(objekt_a,2);        // převod o bitů na hex.(low)
  vys:=vys+a1;                     // spojení obou řetězců
  skupina:=strtoint(vys);          // převod na číslo skupiny v hex. tavaru
  c_hla:=skupina*100+c_hasice_a;   // aktuální číslo prvku. v klientské
                                      aplikaci
  c_str:=inttostr(c_hla)+'*'+inttostr(stav_a)+'*'; // celý HT

  zapis_file(DateToStr(Date)+ ' ' +TimeToStr(Time)+ '  Hlásič
          č.'+inttostr(c_hla)+ ' ve stavu:'+inttostr (stav_a));
   //zapisuje hlásiče do textového souboru, pro zápis byla vytvořena procedura
          zapis_file
 result:=c_str;          // vrací celý HT
end;
```

3.3.7 Připojení klienta

Aplikace slouží jako TCP/IP server. Pro vytvoření serveru použijme komponentu `ServerSocket`, která je součástí VLC Delphi 5. Server naslouchá

na nastaveném portu (`ServerSocket1.Port:=2003`) klienty a čeká na jejich připojení.

Událost `ServerSocket1ClientConnect` zjišťuje, zda je připojen klient, jestliže ano pak logickou proměnou 'pripoj', dáme do TRUE v opačném případě FALSE s proměnou nadále využíváme. Veškeré informace o klientu (čas připojení a odpojení) se ukládají do textového souboru (HistorieS.txt). Není-li k serveru připojen klient, rovněž se ukládají informace o hlásičích textového souboru viz 3.3.6.

3.3.8 Odeslání a příjem dat klientské aplikaci

Odesílání a příjem dat je provedeno pomocí komponenty `ServerSocket` viz 3.3.7, komponenta má stejné funkce a události jako `ClientSocket` viz 3.2.4.

Odesílaní dat se byla vytvořena nová procedura `posli_client` , která má dva úkoly šifrování viz 3.2.9 dat (HT) a odesílání dat klientské aplikaci pomocí konponenty `ServerSocket`.

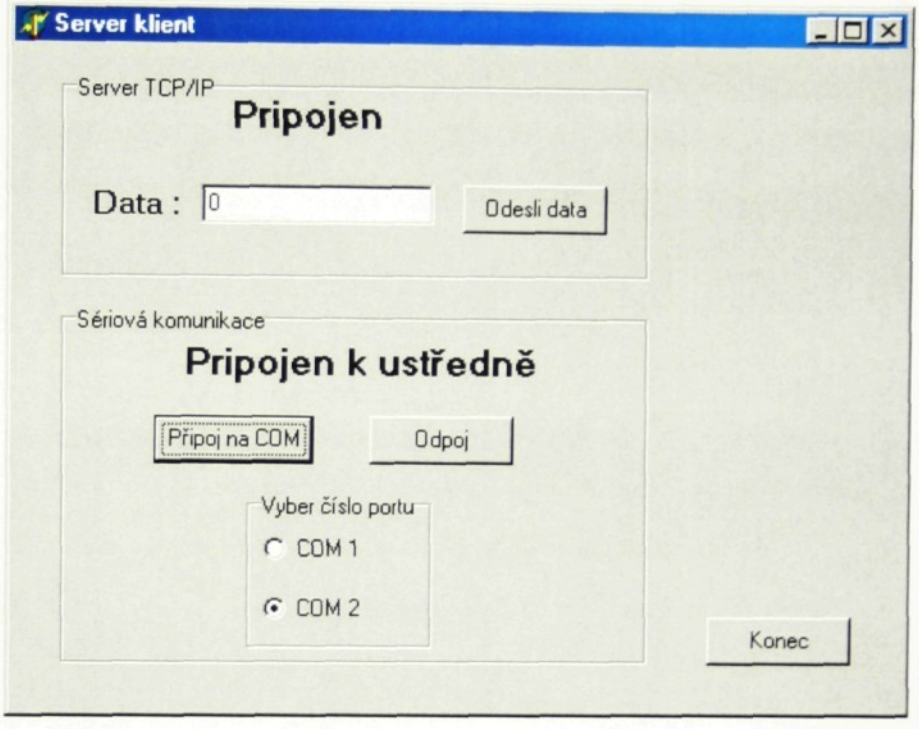
```
procedure Tform1.posli_client(client_str:string) ;
begin
      ServerSocket1.Socket.Connections[0].SendText
            (lbrsa1.EncryptString (client_str));
end;
```

3.3.9 Vizuální popis aplikace „Server“

Aplikace má jedno okno obr. 20 rozdělené na dvě skupiny:

1. Server TCP/IP – Zde mám možnost posílat (testovat) data klientské aplikaci ve tvaru HT a indikaci o stavu připojení.

2. Sériová komunikace – Uživatel zde má možnost výběru portu (COM1,2) a připojení (tlačítko „Připoj na COM") či odpojení („Odpoj") serveru k ústředně. Také je tu zobrazen stav připojení.



Obr.20: Uživatelské hlavní okno „Server"

3.3.10 Ověření funkčnosti

Nebyla zde možnost připojení aplikace „Server" k ústředně, ve firmě VEGACOM nebyla k dispozici ústředna Essertonic8007© se zapojenými hlásiči, proto funkčnost byla ověřena vytvořením nové testovací aplikace komunikující s COM2 a přes hardwarový přípravek (sériový komunikační kabel) spojený s COM1, se kterým komunikoval „Server". V testovací aplikaci jsme zadávali hodnoty jednotlivých bytů, tyto posloupnosti bytů „Server" dekódoval správně. Tím je zaručena komunikace s ústřednou.

3.3.11 Shrnutí aplikace „Server“

Aplikace je naprogramována na konkrétní zařízení Essertronic8007©, komunikace s klientem a ústřednou je obousměrná. Zdrojové kódy jsou obsaženy v příloze na CD-ROM.

4 Závěr

V rámci diplomové práce byla vytvořena grafická nadstavba, jejímž cílem je vizualizace požárních hlásičů, jenž jsou rozčleněny na jednotlivá patra. Detekuje-li prvek vznikající požár, aplikace reaguje na změnu zobrazením nového stavu, doplněného textovou informací po případě zvukovou signalizací. Aplikace umožňuje aktivaci či deaktivaci požárních hlásičů v systému. Komunikace mezi serverem a klientem, byla zabezpečena šifrovacím algoritmem RSA, tím dochází k utajení informací.

Funkčnost byla ověřena simulováním výstupních hodnot ústředny Essertronic7008©, neboť zde nebyla možnost vyzkoušet software na fungující ústředně s připojenými hlásiči. Jakmile bude ústředna dodána, celá aplikace bude v praxi vyzkoušena.

Do budoucna je počítáno se spoluprací s firmou VEGACOM na dalším vývoji této aplikace, kterým mohou být doplňkové nástroje jako 3D obraz celé budovy, vytváření bitmap zobrazujících jednotlivá patra a jiné. Dalšími možnostmi, které lze použít pro zabezpečení identifikace uživatele, jsou externí zařízení jako čipové karty nebo biometrické čtečky.

Aplikace byla vytvořena na základě všech požadavků firmy a bude nadále využívána jako doplňková služba při projektech elektronické požární signalizace.

5 Literatura

[1] ALEŠ DUDÁČEK.: Spojení a signalizace v PO a Požárně bezpečnostní zařízení (EPS), Ostrava 1994

[2] VAGACOM.: http://www.vegacom.cz/admin/files/eps.pdf

[3] ALPOS s.r.o. Ostrava.: http://www.alpos.cz/elektric.htm

[4] ALGIS s.r.o.: http://www.alsig.cz/old/pozar.htm

[5] ESSER.:8007_GB

[6] ESSER INSTALLATION INSTRUCTIONS ESSERTRONIC® 8007 FIRE ALARM CONTROL PANEL

[7] LOŠÁK,J.-BRADÁČOVÁ,I.-DUDÁČEK,A.-FILIPI,B.: Vybavování objektů požárně bezpečnostními zařízeními, (Závěrečná zpráva RU-99-093-87), Ostrava

[8] ČSN 342710, Předpisy pro zařízení elektrické požární signalizace

[9] ELEKTRIKA.CZ: http://www.elektrika.cz/win/paralela/advp1021119.html

[10] ROOT.CZ.: Šifrování - úvod do problematiky: http://www.root.cz/clanek/229

[11] ALENA KABELOVÁ, LIBOR DOSTÁLEK.: Velký průvodce protokoly TCP/IP a systémem DNS, 3. aktualizované a rozšířené vdání,Computer Press, a.s., s.592

[12] LIBOR DOSTÁLEK A KOLEKTIV.: Velký průvodce protokoly TCP/IP Bezpečnost, 2.aktualizované vydání, Computer Press 2003, s.571

[13] ESSER GmbH, Protokol BMZ8007/8008

[14] DUDÁČEK, A.: Požárně bezpečnostní zařízení (EPS). Ostrava, VŠB-TUO 1996

[15] BURKHARD KAINKA, HANS-JOACHIM BERNDT.: Využití rozhrání PC pod Windows, Hel 2000, s.151

[16] DALIBOR KAČMÁŘ.: Programujeme v COM a COM+,Computer Press, s.326

[17] FRANK ELLER.: Delphi 6 příručka programátora, Grada publishing, s.272

[18] VÁCLAV KADLEC.: Učíme se programovat v Delphi a jazyce Object Pascal, Computer Press 2001, s.288

[19] MARCO CANTÚ.: Myslíme v jazyku Delphi 6 – 1. díl, Grada publishing, Praha 2002, s.496

[20] ORFALI, R et al.: The Essential Client/Server Survival Guide, Jonh Wiley & Sons 1994

[21] RAY LISCHNER.: Delphi v kostce, Computer Press Praha 2000, s.550

[22] JOSEF PIRKL.: Komponenty v Delphi,Computer Press 2002

[23] ELMONT GROUP, a.s.: http://www.elmontgroup.cz/eser.html