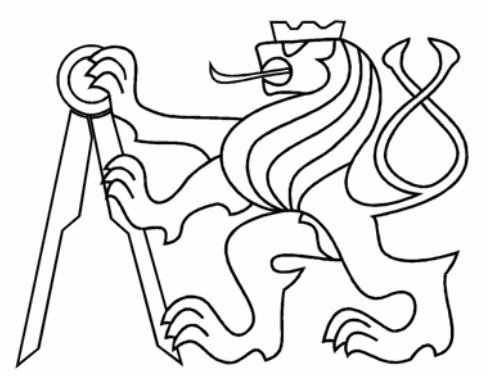
České vysoké učení technické v Praze

Fakulta elektrotechnická

Katedra řídicí techniky



Bakalářská práce

Rozšíření modelu rybí farmy

Praha 2007 Jan Baláš

Prohlášení

Prohlašuji, že jsem svou bakalářskou práci vypracoval samostatně a použil jsem pouze podklady (literaturu, projekty, SW atd.) uvedené v přiloženém seznamu.

V Praze, dne ……. …….

podpis

Poděkování

Na tomto místě bych chtěl poděkovat všem lidem, kteří mi při tvorbě této práce pomáhali. Velký dík patří hlavně ing. Janu Kelbelovi, vedoucímu práce, za ochotu, pomoc a vedení v průběhu tvorby této práce.

Dále bych chtěl poděkovat svým rodičům, kteří mě po celou dobu studia vytrvale a s nekončící trpělivostí podporovali.

Abstrakt

Cílem této práce je navrhnout a realizovat rozšíření existujícího modelu automatizované rybí farmy, které spočívá ve vytvoření hardwarově nenáročného uživatelského prostředí pro řídicí program. Aplikace funguje na operačním systému Linux a mohla by být doplněna o ovládání pomocí dotykového displeje. Pro realizaci bylo užito programovacího jazyka C a knihoven ncurses a ORTE.

Abstract

This thesis aims to design and implement the extension of an already existing automatic fish farm model. The extension subsists in creating a user friendly interface for the control program which is not hardware-demanding. The application runs under Linux operating system and could be subsequently expanded by touch screen control. C programming language and ncurses and ORTE libraries were used during implementation.

Obsah

ÚVOD **4**

1. RYBÍ FARMA **5**

1.1 Popis rybí farmy 5

1.2 Použitá komunikace 6

2. SÍTĚ **6**

2.1 ISO/OSI model 6

2.2 Protokoly 7

2.3 Model publish-subscribe 8

2.3.1 Publish-subscribe architektura 8

2.3.2 Publish-subscribe protokol a RTPS 8

2.4 RTPS protokol 10

2.5 Composite state transfer protokol 11

2.6 ORTE 12

3. POUŽITÝ SOFTWARE **13**

3.1 Operační systém 13

3.2 Programovací jazyk 13

3.3 Vývojové nástroje 14

4. TEXTOVÉ UŽIVATELSKÉ PROSTŘEDÍ **15**

4.1 Knihovna Ncurses 15

4.1.1 Windows 15

4.1.2 Knihovna panel 15

4.1.3 Knihovna menus 16

5. STRUKTURA APLIKACE **16**

5.1 Obecný popis programu 16
5.2 Grafická část 16
5.2.1 Ovládací menu 16
5.2.2 Spořič obrazovky 18
5.2.3 Zadávací kalkulačka 18
5.3 Datová část 19
5.3.1 Načítání nových dat 19
5.3.2 Zapisování dat 20

6. POPIS APLIKACE 20
6.1 Grafické rozhraní 20
6.1.1 Základní obrazovka 20
6.1.2 Přihlašování 21
6.1.3 Zadávání nových dat 21
6.1.4 Spořič obrazovky 22
6.2 Parametry programu 22
6.3 Jazykové mutace 23

7. ZÁVĚR 24

POUŽITÁ LITERATURA 25

PŘÍLOHA A - OBSAH PŘILOŽENÉHO CD 26

PŘÍLOHA B - POPIS KONFIGURAČNÍCH SOUBORŮ 27

Seznam zkratek

API *(Application Programming Interface)* Sbírka funkcí, procedur a tříd dané knihovny, které může programátor, využívající tuto knihovnu, použít.

CD *(Compact Disc)* Kompaktní disk – optické záznamové médium.

CDK *(Curses Developement Kit)* Widgets určené pro rychlejší a snazší tvorbu uživatelských rozhraní využívající knihovnu curses.

EPL *(Elipse Public License)* Licence používaná organizací Eclipse Foundation pro šíření jejího SW. Licence vhodná pro šíření svobodného SW v obchodním prostředí.

GUI *(Graphical User Interface)* Grafické uživatelské rozhraní skládající se z interaktivních grafických prvků.

GNU GPL *(GNU General Public License)* Licence pro svobodný SW. Zdrojové kódy šířené pod touto licencí mohou být upravovány a používány, šířeny však musí být opět pod licencí GNU GPL.

MIT license Licence pocházející z Massachusetts Institute of Technology umožňující užití kódů šířených pod touto licencí v proprietálním SW za podmínky, že tato licence bude šířena spolu s tímto SW.

ORTE *OCERA Real Time Ethernet*

OSI model *(Open Systems Interconnection Basic Reference Model)* Sedmivrstvý abstraktní popis pro komunikaci a design síťových protokolů.

pH *(Potential of Hydrogen)* Kyselost roztoku.

RS485 Vícesignálová sériová průmyslová sběrnice.

RT *Real Time*

RTAI *(Real-Time Application Interface)* Rozšíření pro jádro Linuxu, které umožňuje psaní aplikací s přesným časováním.

RTOS *(Real-time Operating System)* Multitaskový operační systém určený pro RT aplikace.

RTPS *(Real-Time Publish-Subscribe)* Protokol vyvinutý firmou Real-Time Innovations, Inc jako síťový protokol pro distribuované systémy.

SW Software

TCP *(Transmission Control Protocol)* Connection oriented reliable protokol 4. síťové vrstvy.

TUI *(Text User Interface)* Uživatelské tozhraní nezaložené na bitmapách, ale pouze na textových symbolech.

UDP *(User Datagram Protocol)* Connection oriented unreliable protokol 4. síťové vrstvy.

X11 *(X Windows system)* Zobrazovací protokol v unixových systémech poskytující GUI založené na konceptu bitmapového okna.

Úvod

Cílem této bakalářské práce bylo rozšířit existující model rybí farmy umístěný v laboratoři K09 o uživatelské rozhraní, přes které by bylo možné ovládat řídicí program. Jelikož práce, na které jsem navazoval, byly zaměřeny na nízkou cenu komponent i programového vybavení, i já jsem pro tvorbu využil nástroje, které jsou šířeny pod GNU GPL. Výhodou užití softwarového vybavení šířeného pod touto licencí jsou nízké pořizovací náklady pro vývoj a široká podpora komunity, která existuje okolo svobodného software.

Tento dokument v první kapitole seznamuje s existujícím modelem rybí farmy a s řídicím programem. Druhá kapitola se zabývá způsobem komunikace v síti a použitým komunikačním protokolem RTPS. Třetí kapitola se věnuje softwaru a systému použitému při tvorbě aplikace. Ve čtvrté kapitole jsou popsány prostředky využité při tvorbě uživatelského rozhraní. V páté kapitole je vysvětlena struktura aplikace a použitých funkcí. V šesté kapitole je popis samotné aplikace.

1. Rybí farma

Model rybí farmy vznikl jako součást projektu IFiBO. Model byl vytvořen v rámci diplomových prací Jana Kelbela [1], Ondřeje Špinky [2] a Tomáše Cíle podle pokusné automatické rybí farmy, která byla navržena ve Výzkumném ústavu rybářském a hydrobiologickém ve Vodňanech.

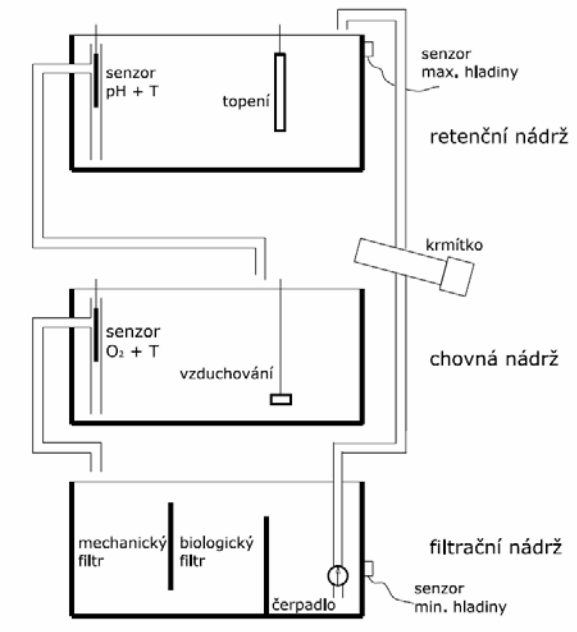
1.1 Popis rybí farmy

Model rybí farmy se skládá ze tří nad sebou postavených nádrží a automatického krmítka (viz obr. 1.1 – převzat z diplomové práce Jana Kelbela [1]).

První, nejvýše položená nádrž slouží k ohřevu vody. V jejím odtoku je umístěn senzor teploty, kyselosti (pH) a kapacitní senzor hladiny, který brání přetečení nádrže.

Druhá nádrž, sloužící k samotnému chovu ryb, je opatřena vzduchováním. V jejím odtoku jsou umístěny senzory teploty a kyslíku.

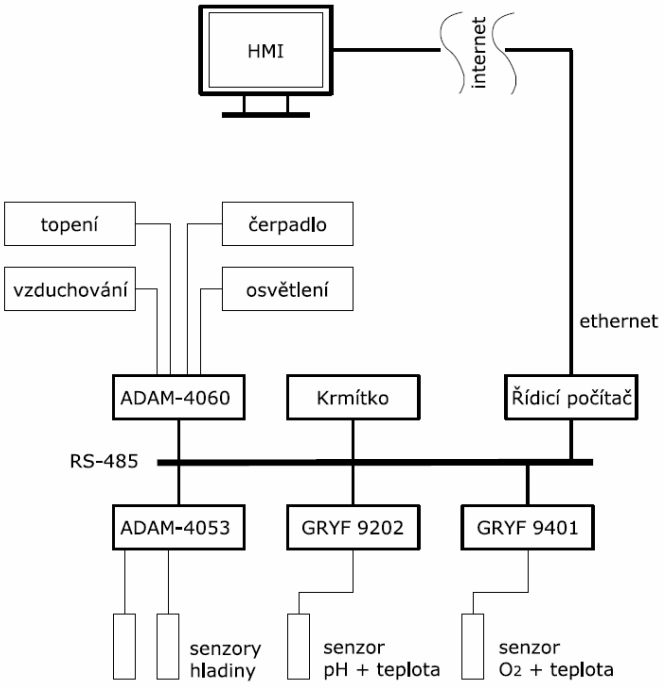
Poslední, třetí nádrž slouží k mechanické a biologické filtraci. Pročištěná voda se z ní vrací zpět do první nádrže. V této nádrži je umístěn kapacitní senzor, který brání chodu čerpadla na prázdno.



Obr. 1.1 Model rybí farmy

1.2 Použitá komunikace

Všechna zařízení rybí farmy komunikují s řídicím programem po sběrnici RS485. Řídicí program komunikuje s aplikací navrženou pro tuto práci, která mu po síti posílá nové parametry pro senzory a nastavení a také z něj přijímá aktuální informace o senzorech a nastaveních (viz obr. 1.2 – převzato z diplomové práce Jana Kelbela). S řídicím počítačem komunikuje aplikace pomocí protokolu ORTE (viz Sítě).



Obr. 1.2 – Blokové schéma systému

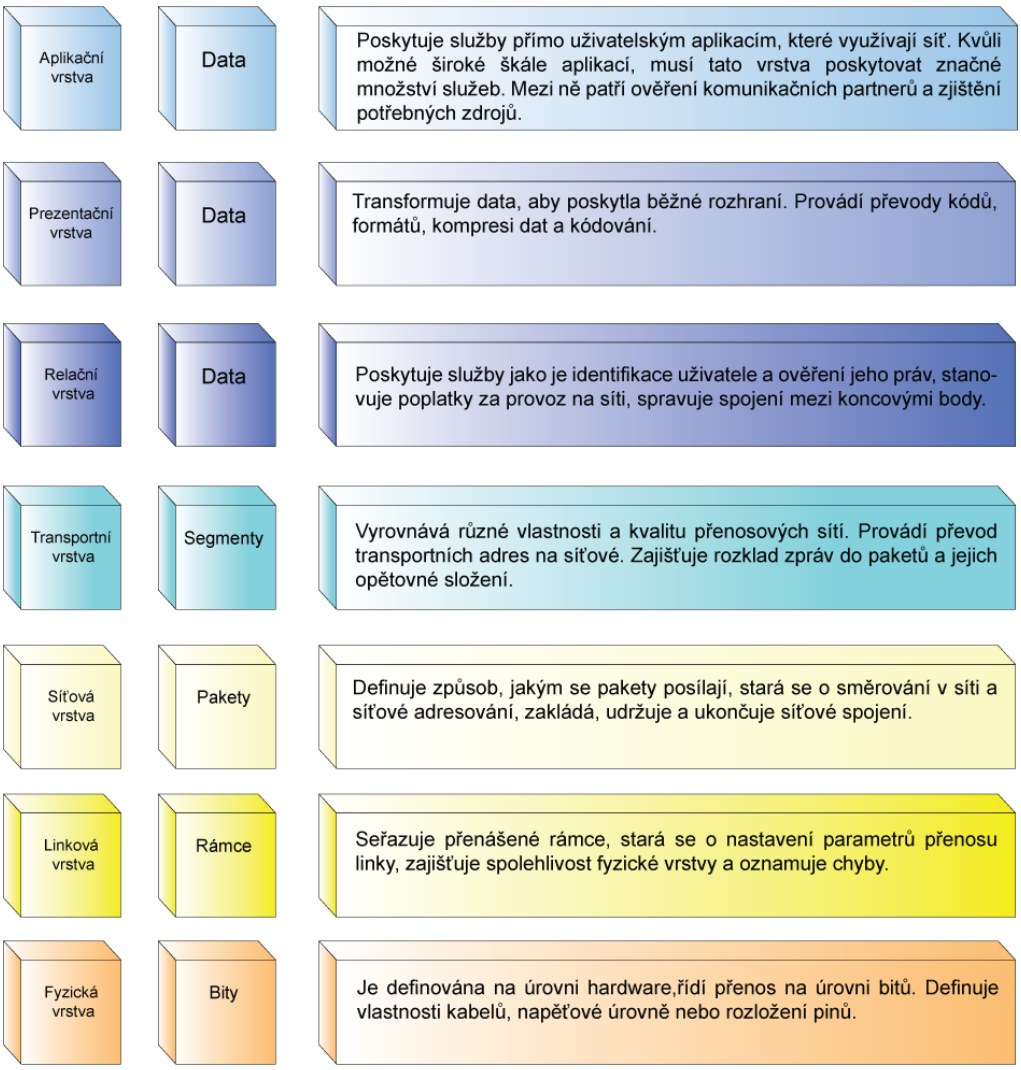
2. Sítě

2.1 ISO/OSI model

V rámci snahy o standardizaci počítačových sítí byl vytvořen a roku 1984 přijat jako mezinárodní norma ISO 7498 sedmivrstvý referenční model ISO/OSI. Tato norma nespecifikuje přesnou implementaci systémů, ale pouze uvádí všeobecné principy, které by měly být užity při propojování systémů.

Každá vrstva modelu zastává definovanou skupinu funkcí (viz obr 2.1), které jsou potřeba pro úspěšnou komunikaci mezi systémy. Každá vrstva může využívat služeb vrstvy o jednu nižší a stejně tak poskytuje své služby vyšší vrstvě. Komunikace mezi vrstvami

v jednom systému se řídí pravidly, která se obvykle nazývají *interface*. Komunikace mezi různými systémy na stejné vrstvě se řídí pomocí protokolů – viz 2.2.



Obr. 2.1 Struktura ISO/OSI modelu

2.2 Protokoly

Protokol je dorozumívací standard, který umožňuje spojení, komunikaci, a přenos dat mezi dvěma koncovými body. Protokoly mohou být definovány jako pravidla řídící syntax, sémantiku a synchronizaci komunikace.

Podle způsobu doručení dat můžeme protokoly rozdělit do dvou skupin

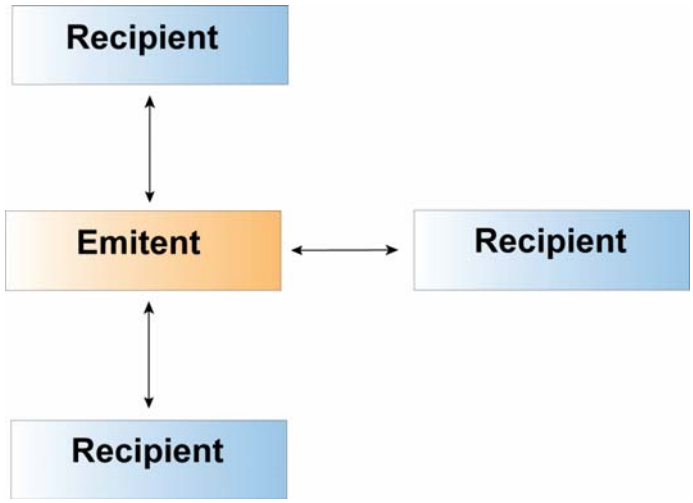
- **Spolehlivé** (*reliable*) – garantuje spolehlivé doručení a doručení ve správném pořadí. V případě nedoručení jsou data poslána znovu. Toto ověřování je na úkor rychlosti a vytíženosti linky. Příkladem takového protokolu může byt TCP.

- **Nespolehlivé** (*unreliable*) – protokol negarantuje doručení paketu, ani pořadí, ve kterém dojdou. Výhodou je vyšší rychlost a menší vytížení linky. Příkladem takového protokolu je UDP.

2.3 Model publish-subscribe

2.3.1 Publish-subscribe architektura

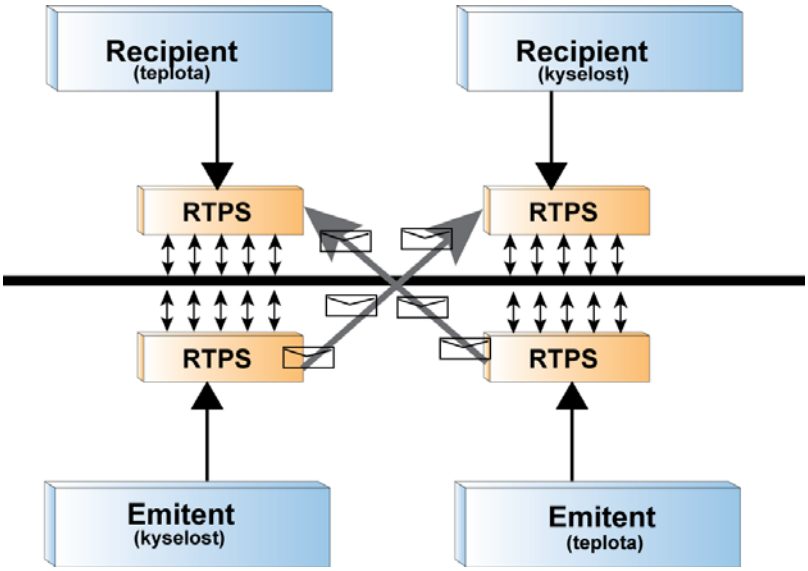
Tento model distribuce dat se vyznačuje tím, že jednotliví účastníci komunikace se dělí do dvou skupin. Na emitenty (*publishers*) a recipienty (*subscibers*). Emitent může dodávat data jednomu i více recipientům. Model komunikace mezi emitenty a recipienty je znázorněn na obr. 2.2.



Obr. 2.2 Model publish-subscribe architektury.

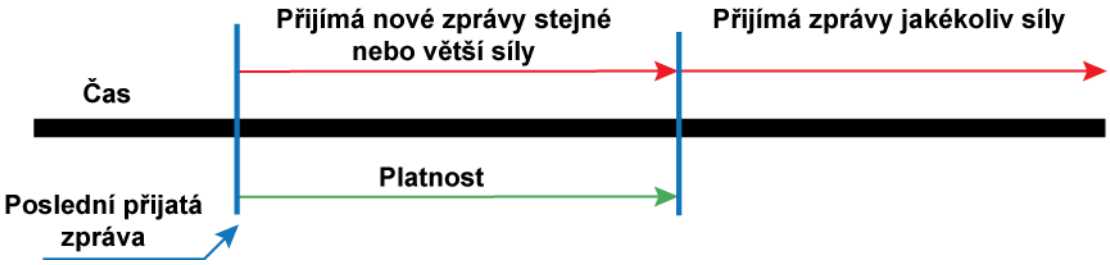
2.3.2 Publish-subscribe protokol a RTPS

Publish-subscribe protokol byl implementací do RTPS rozšířen o časové parametry a možnost nastavení kvality služeb. Komunikace zde probíhá vždy mezi emitenty a recipienty (viz obr 2.3).



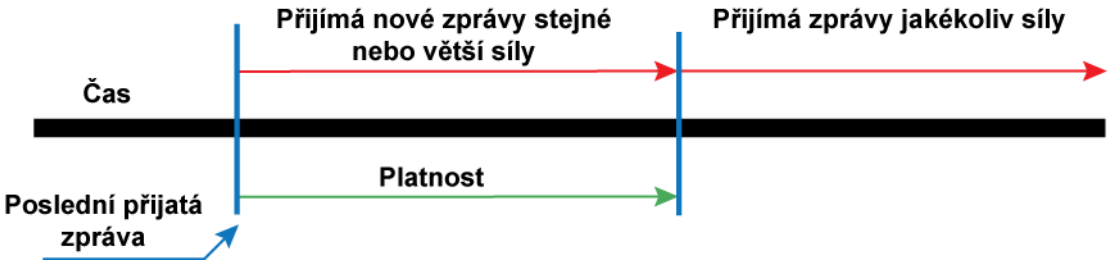
Obr. 2.3 Publish-subscribe protokol implementovaný do RTPS

- **Emitent** (*publisher*) je charakterizován čtyřmi parametry: předmětem (*topic*), typem (*type*), silou (*strenght*) a platností (*persistance*). Předmět a typ musí být vždy předem registrovány. Síla udává váhu emitenta vůči ostatním emitentů se stejným předmětem. Platnost udává dobu, po kterou jsou data od daného emitenta platná. Data jsou publikována tak rychle, jak je emitent obstarává. Je-li v síti víc emitentů se stejným předmětem, je publikován ten s největší silou. Dojde-li k vypršení platnosti nebo poruše emitenta s největší silou, jsou publikována data emitenta s druhou největší silou. Po obnovení činnosti nebo získání nových dat nejsilnějšího emitenta jsou publikována opět jeho data. Grafické znázornění chování emitenta je vidět na obr 2.4.



Obr. 2.4 Grafické znázornění výběru emitenta

- **Recipient** (*subscriber*) je charakterizován čtyřmi parametry: předmětem (*topic*), typem (*type*), minimálním odstupem (*minimum separation*) a nejzazší mezí (*deadline*). Předmět a typ musí korespondovat s předmětem a typem datového toku od emitenta. Minimální odstup definuje dobu, během které recipient nepřijímá žádná nově publikovaná data. Nejzazší mez značí dobu, po kterou je recipient ochoten čekat na nová data. Pokud do této doby nová data nedorazí, je oznámena chyba. Grafické znázornění chování recipienta je vidět na obr 2.5.



Obr. 2.5 Grafické znázornění chování recipienta

2.4 RTPS protokol

RTPS protokol byl vyvinut firmou Real Time Innovation, Inc. jako síťový protokol pro datově distribuovaný systém. RTPS je navržen tak, aby fungoval nad nespolehlivým protokolem, jako je UDP/IP. Hlavním cílem tohoto protokolu je:

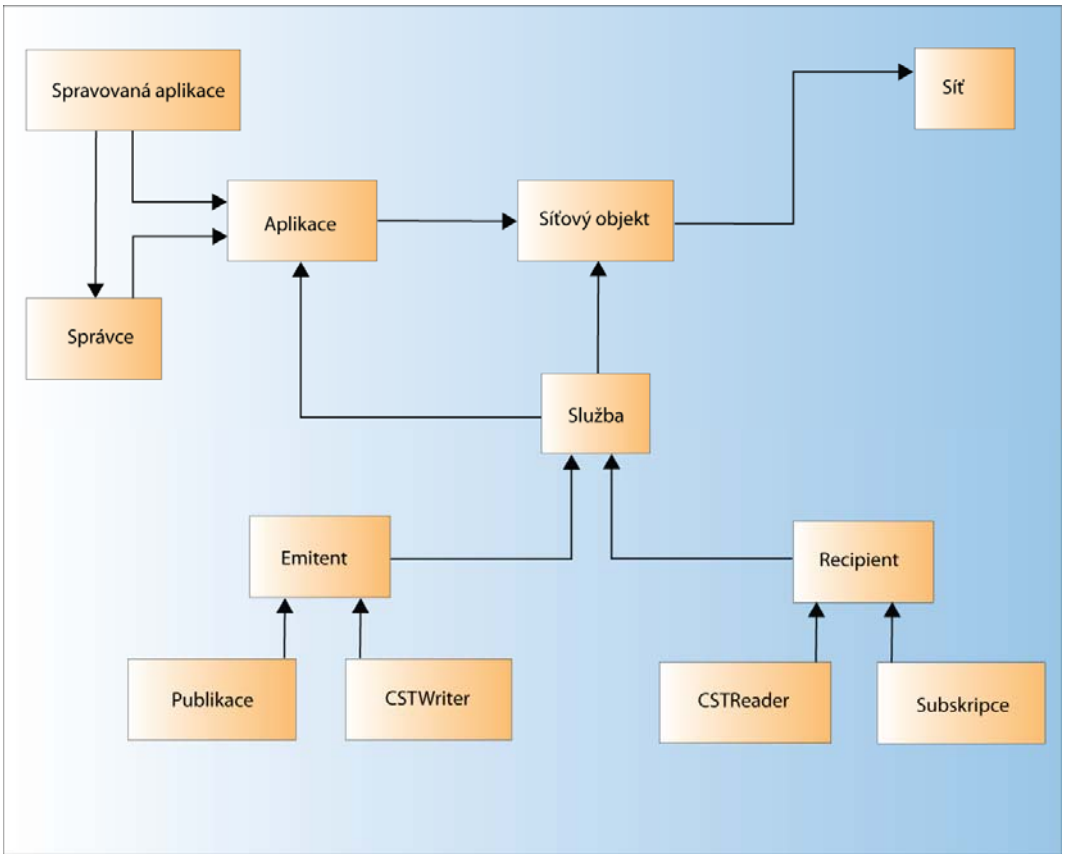
- Plug and play konektivita, aby nové aplikace a služby byly automaticky objeveny a mohly se kdykoliv připojit nebo odpojit bez potřeby rekonfigurace
- Možnost nastavení kvality služeb podle kritérií na maximální rychlost doručení (*best effort*) nebo maximální spolehlivost doručení (*reliable*).
- Rozšiřitelnost umožňující systémům růst do větších sítí.
- Možnost časování zpráv.

RTPS protokol poskytuje dvě základní funkce:

- **Distribuci dat** – RTPS specifikuje formát zpráv a komunikační protokoly, které podporují publish-subscribe protokol (viz 2.3.2).

- **Správu informací** – RTPS protokol využívá CST protokol (viz 2.5) k tomu, aby umožnil aplikacím získat informace o existenci a atributech ostatních aplikací a služeb v síti. Tyto pomocné informace pomohou každé aplikaci získat komplexní přehled o aplikacích, manažerech, emitentech a recipientech v síti a umožní tím poslat data na správné místo a správně interpretovat data přijatá.

Objektový model RTPS protokolu je vidět na obr.2.6.



Obr. 2.6 Objektový model RTPS protokolu

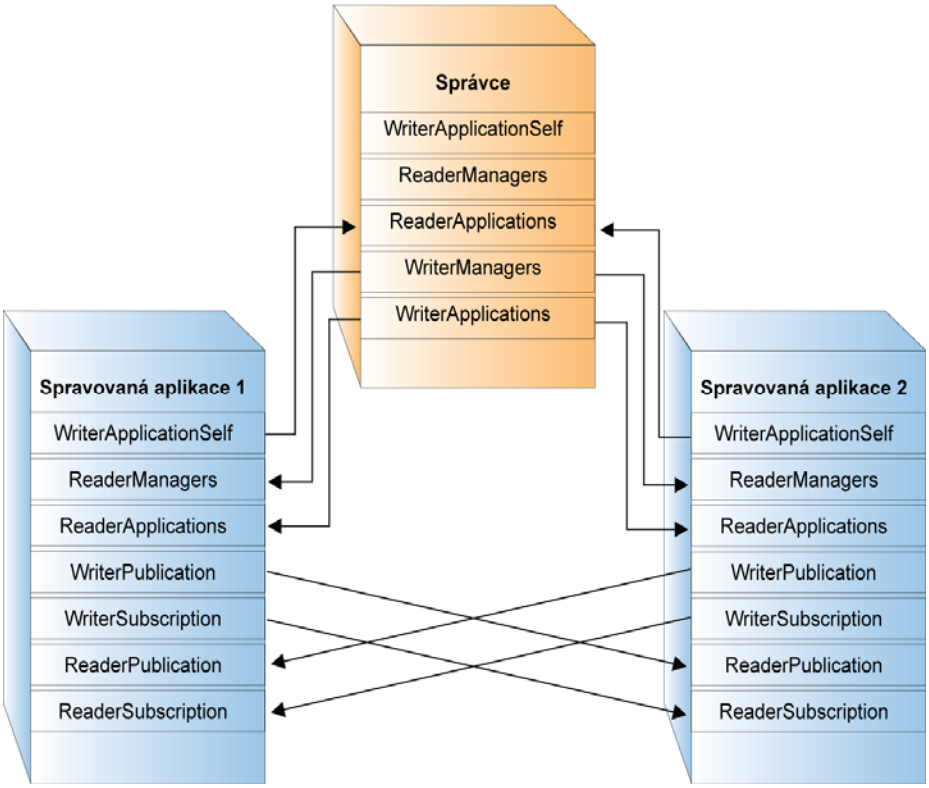
2.5 Composite state transfer protokol

Composite state transfer protokol je určen k výměně stavových informací (*composite state*) mezi účastníky komunikace. *Composite state* je složen z atributů síťového objektu. Komunikace vždy probíhá od CSTWriter k CSTReader.

RTPS využívá CST k automatickému objevení aplikací a služeb potřebných pro danou aplikaci. Pro tuto potřebu jsou definovány dva typy síťových aplikací: správce (*manager*) a spravovaná aplikace (*manager application*).

- **Správce** – sleduje své spravované aplikace a jejich atributy pomocí Registration protokolu. Jednotliví správci mezi sebou komunikují prostřednictvím Inter-Manager protokolu. Navzájem si vyměňují seznamy spravovaných aplikací.
- **Spravovaná aplikace –** může být spravována jedním i více správci. Správce v síti vyhledává prostřednictvím Manager-Discovery protokolu. Application-Discovery protokol umožňuje najít další spravované aplikace v síti. Služby lze najít pomocí Service-Discovery protokolu.

Grafické znázornění komunikace mezi správcem a spravovanými aplikacemi a komunikace mezi jednotlivými spravovanými aplikacemi je znázorněna na obr. 2.7.



Obr. 2.7 Model komunikace využívající síťových objektů správce a spravovaná aplikace

2.6 ORTE

ORTE je open-source implementace komunikačního protokolu RTPS. ORTE je podporováno v Linuxu, FreeBSD, Solaris, MacOS, PharLap, RTAI s Renet a dokonce i v OS Windows.

Základem každé aplikace využívající ORTE je tzv. doména (*domain*). Ta spravuje celou komunikaci včetně jejího vytvoření i ukončení. Jelikož ORTE používá strukturu

publish-subscribe, obsahuje funkce pro tvorbu emitentů (*ORTEPublicationCreate()*) i recipientů (*ORTESubscriptionCreate()*). Každý emitent si registruje datové typy, které bude publikovat, a každý recipient si registruje typy, které bude přijímat. Dále je nutné registrovat serializační a deserializační funkce. Ty slouží k převodu dat uložených v operační paměti do datového proudu a zpět. Pro tvorbu serializačních a deserializačních funkcí existuje IDL compiler, který je schopen automaticky generovat zdrojové kódy těchto funkcí v jazyce C na základě jmen proměnných a jejich datových typů.

ORTE umožňuje dva režimy komunikace – *best effort* a *strict reliable* (viz 2.4). ORTE poskytuje další množství užitečných funkcí, které usnadňují tvorbu aplikací. Například uspání aplikace, podrobné výpisy, okamžité odeslání zprávy nebo registrace funkcí, které jsou provedeny při odeslání nebo přijetí nové zprávy.

3. Použitý software

Jedním z podstatných záměrů projektů, na které tato práce navazuje, bylo vytvoření levného programového vybavení. Proto je veškerý používaný software šířen pod licencemi pro svobodný software jako GNU GPL, EPL nebo MIT.

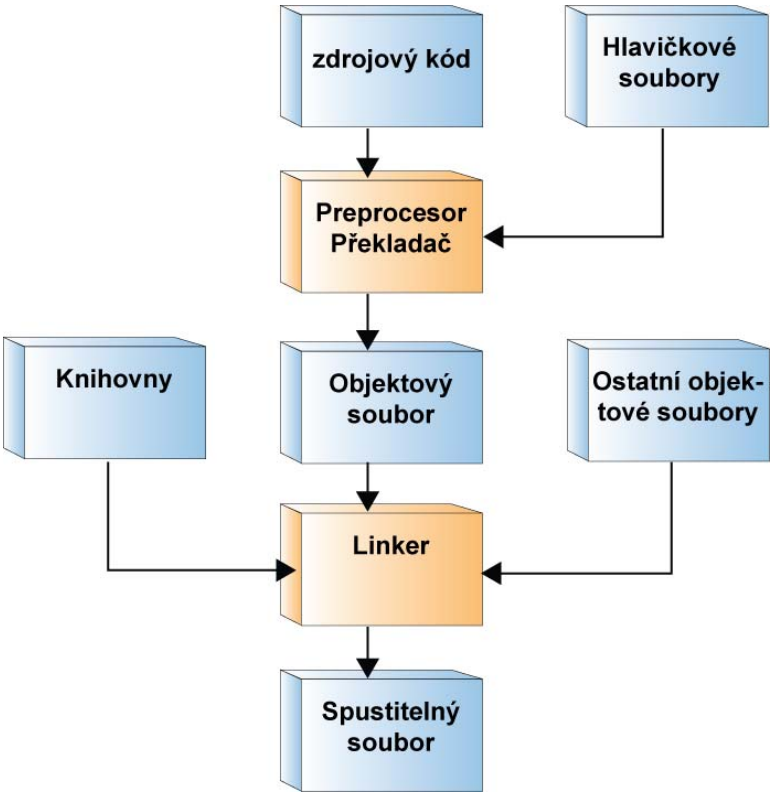
3.1 Operační systém

Pro vývoj aplikace je použit systém Linux – distribuce Fedora 6, pro testování je použita distribuce Fedora 7 a Debian, které jsou běžně používány pro klasické stolní počítače. Nízká paměťová i výpočetní náročnost předurčuje aplikaci k běhu na Linuxových systémech využívaných ve vestavných počítačích, jako je například MITELinux nebo BlueCat.

3.2 Programovací jazyk

Pro aplikaci byl vybrán jazyk C, jelikož je úzce spjat s vývojem unixových operačních systémů, a je proto mezi nimi snadno přenositelný. Jedná se o velmi zavedený jazyk, který je podporován výrobci hardwaru s existencí překladačů pro většinu typů mikroprocesorů. Mimo jiné existují i propracované nástroje pro optimalizaci a validaci kódu. Obecný způsob zpracování kódu je vidět na obr. 3.1. Další výhodou jazyka C je, že nepotřebuje náročnou runtime podporu jako např. jazyk Java.

Nástroje pro tvorbu GUI nejsou v C příliš propracované, proto je vytvořeno TUI, pro které již vcelku kvalitní nástroje existují. Ani zde však není možno – vzhledem k netypickému ovládání aplikace – při tvorbě použít existující ncurses widgets (CDK).



Obr. 3.1 Způsob zpracování kódu v C

3.3 Vývojové nástroje

V Linuxu existuje řada kvalitních textových editorů s funkcemi pro usnadnění psaní kódu. Mezi ně patří například velmi oblíbený Vim. Pro práci je využit textový editor Kate, který je přímo součástí základního balíčku KDE. Jeho hlavní výhodou je malá náročnost na systémové prostředky, zabudovaný příkazový řádek a zvýrazňování syntaxe. S rostoucí komplikovaností projektu však Kate začal být nedostačující, a proto jsem přešel k dokonalejšímu vývojovému prostředí – Eclipse s CDT pluginem, který umožňuje použít toto vývojové prostředí k tvorbě C/C++ kódu. Eclipse je v současné době zřejmě nejlepší grafické open-source vývojové prostředí. Nabízí kvalitní náhled do struktury programu, dobrý debugger i nápovědu k používaným funkcím. Velkou výhodou Eclipse je také jeho univerzálnost. Díky flexibilnímu návrhu této platformy se dá seznam podporovaných jazyků rozšířit nejen o C/C++, ale i Javu (pro tu byl Eclipse původně vyvinut), PHP, HTML či XML. Vývoj podpory jednotlivých pluginů je veden v samostatných projektech, takže se liší i jejich

kvalita. Podstatné však je, že všechny mohou těžit ze základního IDE prostředí, které je ve srovnání s jinými open-source grafickými vývojovými nástroji velmi rychlé, a to nejen na platformě Linux (existují balíčky pro KDE, GTK i Motif), ale i na Windows.

4. Textové uživatelské prostředí

TUI bylo vybráno pro tuto aplikaci ze dvou hlavních důvodů. Prvním z nich je velmi nízká hardwarová náročnost ve srovnání s GUI běžícím pod X11. Tato skutečnost je důležitá zejména kvůli nižším výkonům vestavných počítačů používaných běžně v průmyslu a samozřejmě také kvůli ceně. Druhým důvodem je jeho snadná použitelnost pomocí vzdálené příkazové řádky (*remote shell*).

4.1 Knihovna Ncurses

Tato knihovna vznikla z knihovny curses. Zkratka Ncurses znamená „*new curses*“. Poskytuje API pro tvorbu TUI. TUI vytvořená pomocí této jsou téměř nezávislá na terminálu, což činí aplikaci velmi portabilní. Problém vzniká pouze u některých komerčních unixových distribucí, které používají místo ncurses původní knihovnu curses.

4.1.1 Windows

Okna (windows) jsou nejdůležitějším konceptem curses. S jejich pomocí se dá manipulovat s částmi obrazovky samostatně a aktualizovat pouze potřebné části obrazovky, což umožní tvorbu obdoby klasického bitmapového GUI v textovém režimu. Nevýhodou oken je pouze to, že při tvorbě větších aplikací se manipulace s nimi stává značně nepřehlednou. Pak je řešením použití přídavné knihovny panel viz 4.1.2.

4.1.2 Knihovna panel

Při využívání velkého množství navzájem se překrývajících oken a podoken je značně obtížné dodržovat obnovování jednotlivých složek. Proto je použita tato knihovna, která umožňuje snadné přiřazení skupin oken k jednotlivým panelům, které se chovají jako vrstvy. S každým panelem se pak dá snadno pohybovat v pomyslné hromadě nahoru a dolů, čímž se nastavuje jeho viditelnost, a tím i viditelnost všech oken k němu přiřazených.

4.1.3 Knihovna menus

Knihovna *menus* je rozšířením klasických *curses*. Poskytuje sadu funkcí pro tvorbu klasických menu, která známe z GUI. Umožňuje také tvorbu menu s možností vybrat více položek. Pro tvorbu této aplikace jsou nejdůležitější funkce pro zpracování uživatelských vstupů (menu_driver()) a funkce pro tvorbu objektů menu (new_menu()).

5. Struktura aplikace

V této kapitole bude popsána struktura celé aplikace, včetně popisu funkcí užitých pro posílání a přijímání dat, tvorbu TUI a nastavení aplikace.

5.1 Obecný popis programu

Celá aplikace se skládá ze dvou samostatně běžících vláken. První vlákno má na starost načíst z řídicího počítače periodicky každých 5 sekund data, druhé vlákno se stará o grafické výstupy a ovládání programu. Je-li zadán požadavek na vložení nových dat do řídicího programu, je spuštěno třetí vlákno, které tento požadavek provede a poté se ukončí. Při ukončení aplikace druhé vlákno ukončí zbylá běžící vlákna, čímž se aplikace ukončí.

5.2 Grafická část

Tvorba grafického menu je rozdělena do tří základních částí. První část se zabývá samotným ovládacím menu, kde jsou zobrazeny všechny výpisy. Druhá část, nazvaná „spořič obrazovky“, je určena pro chod po většinu času se zobrazením výstupů ze senzorů. Poslední část „kalkulačka“ slouží k zadávání nových hodnot do řídicího programu a přihlašování do administrátorského režimu.

5.2.1 Ovládací menu

- **`init_set_menu()`**

  Funkce pro vytvoření objektů menu z knihovny menus a nastavení vlastností ncurses, jako je způsob čtení vstupů, barevné kombinace a nastavení kurzoru na požadované hodnoty pro menu.

- **`create_window_elements_menu()`**

  Funkce pro vytvoření objektů typu WINDOW, které jsou použity v menu. V průběhu práce byla do funkce přidána část, která použitá okna slučuje pod objekty typu PANEL

- **`exit_window(int password)`**

  Toto okno je použito pouze pro potvrzení ukončení aplikace.

  `password` – určuje, zda se aplikace nachází v uživatelském, nebo administrátorském módu. Tato kontrola je zde proto, aby nedocházelo k nechtěnému odhlašování. 1 znamená, že program je v administrátorském režimu, 0 v uživatelském režimu.

- **`refresh_menu_panel (int cislo_menu1, int cislo_menu2)`**

  Tato funkce slouží k vykreslení hodnot, vybraných v menu. Funkce byla navrhnuta tak, aby ji bylo možné využít pro ovládání pomocí dotykového panelu.

  `cislo_menu1` – označuje vybranou položku v horním menu.

  `cislo_menu2` – označuje vybranou položku v dolním menu.

- **`move_menu(int menu_number, int direction)`**

  Tato funkce slouží k pohybu v menu pomocí vstupu z klávesnice. Podle vybrané možnosti v tomto menu se provádí příslušné překreslení pomocí refresh_menu_panel()

  `menu_number` – určuje aktivní menu (horní nebo dolní).

  `direction` – určuje směr pohybu.

- **`menu_read_from_keyboard(int password)`**

  Tato funkce slouží pro načítání vstupu z klávesnice při zobrazeném menu.

  `password` – slouží pro určení, v jakém režimu se aplikace nachází. Je-li v administrátorském režimu, je uživateli zpřístupněno třetí menu, ve kterém lze zadávat nová data do řídicího programu.

- **`exit_menu_cleanup()`**

  Tato funkce slouží k uvolnění paměti, která byla ručně alokována pro menu (jazyk C to nedokáže udělat automaticky), a dále k ukončení módu ncurses a k obnově terminálu.

5.2.2 Spořič obrazovky

- **`init_set_screensaver()`**

  Tato funkce slouží k nastavení ncurses pro potřeby spořiče obrazovky obdobně jako init_set_menu a init_set_calc.

- **`create_window_elements_screensaver()`**

  Obdobná funkce jako create_window_elements_menu(), až na to, že vytváří objekty pro spořič obrazovky.

- **`draw_static_window_elements_screensaver()`**

  Vykreslí prvky spořiče, které nejsou závislé na získaných datech (popisky, barevné zvýraznění). Tyto prvky již nebudou překreslovány.

- **`draw_dynamic_window_elements()`**

  Tato funkce slouží k přepsání všech hodnot v okamžiku, kdy jsou načtena nová data.

- **`screensaver_read_from_keyboard(int password)`**

  Tato funkce čeká na vstup z klávesnice (případně může být změněno na vstup z dotykového panelu). Po zaznamenání aktivity pouští menu v režimu, který je dán proměnnou password.

  `password` - jako v ostatních funkcích slouží ke zjištění, zda je aplikace v administrátorském nebo uživatelském režimu.

5.2.3 Zadávací kalkulačka

- **`init_set_calc(int password)`**

  Tato funkce slouží k nastavení ncurses pro potřeby spořiče obrazovky obdobně jako init_set_menu a init_set_screensaver.

- **`create_window_elements_calc()`**

  Obdobná funkce jako create_window_elements_menu(), až na to, že vytváří objekty pro kalkulačku.

- **`draw_window_elements_calc(int password)`**

  Tato funce vykreslí všechny prvky kalkulačky.

`password` – určuje, zda je kalkulačka volaná pro zadání nových dat do řídicího programu, nebo pro zadání hesla. Podle toho se v popiscích kalkulačky nastaví příslušné texty.

- **`calc_flash_number (WINDOW *window,int time_us, char *symbol)`**

  Tato funkce slouží ke znázornění zmáčknutí klávesy v kalkulačce.

  `window` – určuje, jaké okno v kalkulačce má být zvýrazněno.

  `time_us` – určuje dobu, po kterou má být zmáčknuté tlačítko zvýrazněno.

  `symbol` – určuje, jaký symbol má být ve vybraném okně zobrazen při zvýraznění a po něm

- **`calc_read_from_keyboard(int password)`**

  Tato funkce slouží k zadávání hodnot pomocí kalkulačky.

  `password` – tato proměnná určuje, zda je kalkulačka volaná pro zadávání hesla, nebo nové hodnoty do řídicího programu. Je-li volána pro zadání hesla, jsou zadané číslovky zobrazované hvězdičkami. Po potvrzení je zadaná hodnota porovnána s hodnotou v konfiguračním souboru. Jsou-li hodnoty totožné, je aplkace přepnuta do administrátorského režimu. Je-li aplikace volána pro zadání nové hodnoty, je po potvrzení spuštěna funkce pro odeslání požadované hodnoty řídicímu programu send_new_value().

5.3 Datová část

Datová část je rozdělena do dvou podčástí. První má na starost periodické vyčítání hodnot ze senzorů a nastavení řídicího programu, druhá má na starost zapisování uživatelem zadaných hodnot.

5.3.1 Načítání nových dat

- **`data_z()`**

  Tato funkce je spuštěna při startu vlákna určeného pro načítání dat z řídicího programu. Má na starost iniciaci ORTE a vytvoření ORTE domény, která bude spravovat všechny příchozí zprávy.

- **`Basin1_Temp_Subscription()`**

  V této funkci dochází k registrování všech přijímaných datových typů pro použitou ORTE doménu. Dále se zde vytvoří recipienti pro všechna přijímaná data.

- **`recvCallBack_type()`**

  Tato skupina funkcí slouží k zaregistrování tzv. callbackových funkcí pro jednotlivé recipienty. Každý datový typ, který je přijímán, má registrovanou svou vlastní callbackovou funkci. V těchto funkcích dochází v okamžiku příchodu nových dat k jejich převedení z bufferu do globální proměnné, kterou dále používají další části programu.

5.3.2 Zapisování dat

- **`sendNewValue(int okno,int okno2, name,int row, flov value)`**

  Tato funkce je volána z kalkulačky při potvrzení zadání nové hodnoty. Obdobně jako data_z() má na starost vytvoření domény, která bude spravovat komunikaci pro zápis nových dat, registraci použitého typu proměnné a vytvoření samotného emitenta. Po zapsání dat pak tuto doménu zruší.

  `okno` – poloha v horním menu

  `okno2` – poloha v dolním menu

  `row` – konkrétní položka ve vybrané kombinaci horní a dolní menu

  value – zapisovaná hodnota. Tato hodnota je zadána z kalkukačky a poslána emitentem řídicímu programu.

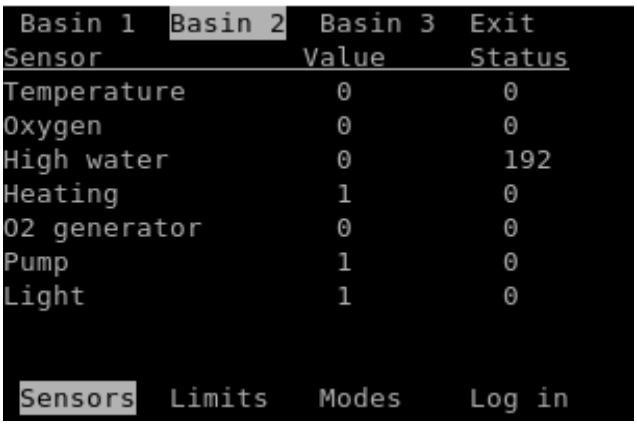
6. Popis aplikace

6.1 Grafické rozhraní

6.1.1 Základní obrazovka

Základní obrazovka (obr. 6.1) se objeví vždy při spuštění programu či při návratu ze spořiče obrazovky. Všechny senzory a nastavení řídicího programu zde jsou rozděleny podle nádrže a podle typu nastavení řídicího programu. Mezi horním a dolním menu se přechází

pomocí tlačítka tabulátor. Po úspěšném přihlášení přibude 3. menu, v němž lze zapisovat nové hodnoty, které budou poslány přímo řídicímu programu.



Obr. 6.1 Základní obrazovka

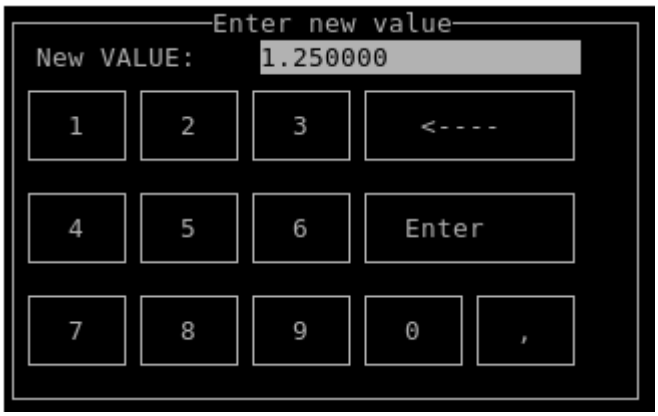
6.1.2 Přihlašování

Přihlašovací obrazovka (obr. 6.2) slouží k zadání hesla do administrátorského režimu. Není-li zadané správné heslo, nedostane uživatel přístup k menu, odkud je možné zasílat data řídicímu programu.

Obr. 6.2 Zadávání hesla

6.1.3 Zadávání nových dat

Po zadání hesla se uživatel dostane do administrátorského módu, kde může vybrat položku na základní obrazovce a změnit její hodnotu (nejde měnit status senzoru, jelikož to je jeho vlastnost). Pro zadání nové hodnoty je použito obdobné menu jako u zadávání hesla při přihlašování (obr. 6.3). Zadaná hodnota je po potvrzení poslána řídicímu programu.



Obr. 6.3 Zadávání hodnot

6.1.4 Spořič obrazovky

Spořič obrazovky (obr. 6.4) slouží k přehlednému náhledu na nejdůležitější hodnoty bez potřeby listováním složitějším menu. Spořič se automaticky zapne, pokud je doba nečinnosti delší, než je doba zadaná v konfiguračním souboru (viz 6.2). Další možnost, jak jej aktivovat, je klávesou F1. Při jakékoli aktivitě uživatele se spořič vypne a spustí se základní obrazovka v módu obyčejného uživatele (není možné měnit nastavení řídicího programu).

Sensor	Value	Status
Basin 1		
Temperature	0	0
pH	0	0
High water level	0	192
Basin 2		
Temperature	0	0
Oxygen	0	0
High water level	0	192
Basin 3		
Low water level	1	192

Obr. 6.4 Spořič obrazovky

6.2 Parametry programu

Základní vlastnosti programu se dají nastavit v souboru config. Tyto parametry se projeví v programu bez nutnosti nového překladu programu.

Možnosti nastavení jsou:

- **heslo** – nastavuje heslo pro přístup do editačního modu. Heslo může být jakékoliv kladné celé číslo do velikosti 16 bitů.

- **screensaver time** – nastavuje čas, po kterém se zapne spořič obrazovky (viz 6.1.4). Čas se nastavuje v desetinách sekundy a maximální hodnota je 255. Je-li jeho hodnota 0, dá se spořič obrazovky aktivovat pouze klávesou F1 (časový limit je nastaven na nekonečno).
- **jazyk** – nastavuje se zde jazyk aplikace (viz 6.3). Parametrem je vždy název souboru, ve kterém jsou uloženy překlady pro daný jazyk. V současné době to jsou možnosti „l-cestina“, „l-english“, „l-deutsch“

6.3 Jazykové mutace

Pro zvětšení uživatelského pohodlí se v konfiguračním souboru config (viz 6.2) dá nastavit jazyková mutace celé aplikace. Konkrétní názvy pro jednotlivé položky jsou uloženy v samostatném souboru a nastavují se podle přiložené tabulky (viz tabulka v příloze). Není-li nějaká položka aplikace pojmenována, je automaticky nahrazena prázdným znakem. Každý jazyk má svůj vlastní konfigurační soubor. Nyní jsou vytvořeny překlady pro češtinu (cestina), angličtinu (english) a němčinu (deutsch). Pokud obsahuje text pro menu jiné než ASCI znaky, je potřeba zajistit jejich podporu pro používanou příkazovou řádku. Pokud není tato podpora, dojde k jejich chybnému zobrazení (např. při použití české diakritiky).

7. Závěr

Výsledkem této práce je spustitelná aplikace, schopná komunikovat přes protokol RTPS s existujícím řídicím programem prostřednictvím emitentů a recipientů. Aplikace je ovládána klávesnicí a funguje korektně i při vzdáleném přístupu využívajícím ssh. Výstup z aplikace je TUI s rozměrem ¼ VGA, což je velikost dotykových displejů dodávaných například společností MITE k některým jejich vestavěným počítačům.

Aplikace rozlišuje dva typy uživatelů. Jeden má práva zapisovat do řídicího programu, druhý má pouze práva pozorovatele. Veškerá nastavení jsou uložena v konfiguračním souboru, pro jejich změnu tak stačí upravit tento soubor a není třeba znovu kompilovat celou aplikaci. Pro vyšší použitelnost je TUI vytvořeno ve třech jazykových mutacích. Názvy pro jednotlivé mutace jsou pro každý jazyk uloženy v samostatném konfiguračním souboru.

Vylepšením současného stavu by mohlo být vytvoření ovládání přes dotykový displej. Dále by bylo možné vytvořit programové rozšíření, které by v pravidelných intervalech zapisovalo údaje ze senzorů a nastavení do databáze, odkud by tyto údaje mohly být dále použity při pozdějším zpracování.

Použitá literatura

[1] Kelbel, J.: *Řídicí systém chovu ryb*. Diplomová práce
Praha, ČVUT, Elektrotechnická fakulta, Katedra řídicí techniky, 2004

[2] Špinka, O.: *Demonstrační prostředí pro vestavěné systémy*. Diplomová práce
Praha, ČVUT, Elektrotechnická fakulta, Katedra řídicí techniky, 2004

[3] Pokorný, L.: *Podpůrné nástroje pro RTPS komunikaci*. Diplomová práce.
Praha, ČVUT, Elektrotechnická fakulta, Katedra řídicí techniky, 2005

[4] RTI: *Real-Time Publish-Subscribe (RTPS) Wire protocol specification*, 2002 verze 1.7 [PDF]

[5] Smolík, P.:*ORTE – OCERA Real-Time Ethernet*, 2004 [PDF]

[6] Padala, P.: *NCURSES Programming HOWTO* verze 1.9, 2005 [PDF]

[7] *Cisco Network Academy* [online], 2006
http://netacad.siliconhill.cz/

[8] Herout, P.: *Učebnice jazyka C,* 4. přepracované vydání
Kopp nakladatelství, České Budějovice, 2005

[9] Kocourek, P.: *Přenos informace*, 2. vydání
Vydavatelství ČVUT, 1999

[10] *The C++ Resources Network* [online]
http://www.cplusplus.com/

[11] *Tutoriál k programu make*
http://www.eng.hawaii.edu/Tutor/Make/

[12] *Encyklopedie*
http://www.wikipedia.org/

Příloha A - Obsah přiloženého CD

docs – obsahuje elektronickou podobu tohoto textu

sources – obsahuje zdrojové kódy aplikace

executable – spustitelný soubor (spust)

controler – obsahuje program (controler), který běží na řídicím počítači, kte kterému se aplikace připojuje

libraries – obsahuje zdrojové kódy použitých knihoven a použitého programu make

make – použitý make, verze 3.81 beta1

orte – knihovna ORTE, verze 0.3.1

ncurses – knihovna NCURSES, verze 5.6

Příloha B - Popis konfiguračních souborů

- **config**

heslo	heslo pro vstup do administrátorského menu
screensaver time	čas v desetinách sekundy
jazyk	soubor s názvy položek menu v daném jazyce

- **cestina**

označení		*popisek*	*označení*		*popisek*
l_tl1	–	limita pro min. teplotu – nádrž 1	s_t2	–	senzor teploty v nádrži 2
l_th1	–	limita pro max. teplotu – nádrž 1	s_ox	–	senzor kyslíku
l_tl2	–	limita pro min. teplotu – nádrž 2	s_ph	–	senzor kyselosti
l_th2	–	limita pro max. teplotu – nádrž 2	s_hw	–	senzor max. hladiny vody
l_ol	–	limita pro min. kyslíku	s_lw	–	senzor min. hladiny vody
l_oh	–	limita pro max. kyslíku	s_p	–	stav pumpy
l_hw	–	limita pro max. hladinu vody	s_l	–	stav světla
l_lw	–	limita pro min. hladinu vody	s_o2	–	stav vzduchování
l_phl	–	limita pro min. kyselost	s_ht	–	stav topení
l_phh	–	limita pro max. kyselost	tm_1	–	horní menu – 1. položka
m_t1	–	mód teploty v nádrži 1	tm_2	–	horní menu – 2. položka
m_t2	–	mód teploty v nádrži 2	tm_3	–	horní menu – 3. položka
m_o	–	mód kyslíku	tm_4	–	horní menu – 4. položka
m_ph	–	mód kyselosti	bm_1	–	dolní menu – 1. položka
m_l	–	mód světla	bm_2	–	dolní menu – 2. položka
m_p	–	mód pumpy	bm_3	–	dolní menu – 3. položka
m_og	–	mód vzduchování	bm_4	–	dolní menu – přihlášeno
s_t1	–	senzor teploty v nádrži 1	bm_5	–	dolní menu – odhlášeno