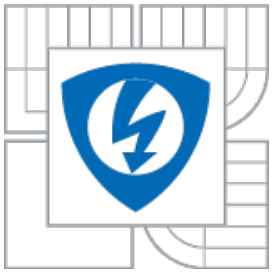
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV BIOMEDICÍNSKÉHO INŽENÝRSTVÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF BIOMEDICAL ENGINEERING

SOFTWARE PRO ZPRACOVÁNÍ DAT ZE SNÍMAČE POLOHY HLAVY

SOFTWARE FOR PROCESSING OF HEAD POSITION SENSOR DATA

DIPLOMOVÁ PRÁCE
MASTER´S THESIS

AUTOR PRÁCE — Bc. Vladimír Slávik
AUTHOR

VEDOUCÍ PRÁCE — Ing. Martin ČÍŽEK
SUPERVISOR

BRNO 2010

Anotace

Práce stručně uvádí do problematiky odchlípení sítnice, vitrektomie a pooperační péče. Dále nastiňuje způsob zvýšení pacientského komfortu použitím systému pro průběžnou signalizaci a záznam poloh hlavy. Softwarová část tohoto systému je realizována za použití programového prostředí C++ Builder a pomocných knihoven výrobců hardware využitého v systému. Tato softwarová aplikace umožňuje analýzu dat získaných systémem a jeho správu. Je popsáno podrobně její grafické uživatelské rozhraní, použití uživatelem a zejména vnitřní uspořádání a rozdělení funkcí do podsystémů.

Abstract

The paper introduces briefly topics of retinal detachment, vitrectomy and applicable postoperative care. A system for increasing patient comfort is described, which continuously records and signalizes head tilt. The software part of aforementioned system is implemented, using C++ Builder programming environment and helper libraries supplied by system's hardware component vendors. This software application allows user to analyze recorded data and manage the system. Application's user interface and usage is described in detail, as well as implementation details concerning internal functions and subsystem organization.

Klíčová slova

náklon hlavy, pozice hlavy, odchlípení sítnice, vitrektomie, software, záznam, vizualizace, bezdrátový přenos, komunikační protokol, RS-232, USB, C++, C++ Builder

Keywords

head tilt, head posture, retinal detachment, vitrectomy, software, recording, visualization, wireless transmission, communication protocol, RS-232, USB, C++, C++ Builder

Citace

SLÁVIK, V. *Software pro zpracování dat ze snímače polohy hlavy.* Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2010. 56 s. Vedoucí diplomové práce Ing. Martin Čížek.

Prohlášení

Prohlašuji, že svou diplomovou práci na téma „Software pro zpracování dat ze snímače polohy hlavy“ jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.

V Brně dne 20. května 2010 .....
podpis autora

Poděkování

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

V Brně dne 20. května 2010 .....
podpis autora

Obsah

Seznam ilustrací

1. Úvod

Odchlípení sítnice je stav, při němž se sítnice oddělí od její nosné tkáně. Může nastat samostatně či v důsledku pooperačních komplikací. Léčba zahrnuje tři možné různé operační procedury, z nichž při dvou (preferovaných) je po samotné operaci pro přidržení sítnice při hojení použita plynová či olejová tamponáda. Ta je náchylná ke změně pozice, a proto je nutno po několik dní až týdnů udržovat hlavu v předepsané pozici, v níž je bublina na svém místě.

Pro zhodnocení výsledků je potřeba vědět, zda k případným pooperačním komplikacím došlo následkem nedodržení polohy hlavy či samovolně. V současnosti jsou využívána spíše preventivní opatření, kdy je hlava pacienta fixována. Vyvíjen je systém, jenž je ve formě drobného zařízení připevněn k pacientovi a informuje jej při nedodržení předepsané pozice. Kromě toho provádí nepřetržitě záznam a umožňuje lékaři pozdější vyhodnocení chování pacienta [16][17].

1.1 Cíl práce

Cílem diplomové práce je vyvinout softwarovou část systému umožňujícího vyhodnotit průběh náklonu hlavy v čase. Systém se v plném rozsahu skládá ze zařízení pro sledování a záznam náklonu pacientovy hlavy (snímač), komunikačního rozhraní připojovaného k USB portu počítače a konečně software pro komunikaci se snímačem a prohlížení záznamů. Vyvíjený software tedy musí být schopen jednak ovládat celý komunikační řetězec, druhak umožnit snadnou a vizuálně orientovanou práci s daty.

1.2 Definice

Pro všechen následující text práce je termín „formulář“ ekvivalentní pojmu „okno“. Stejně tak jsou ekvivalentní termíny „aplikace“ a „program“. Také lze v jednoznačném kontextu považovat termíny „zařízení“, „senzor“ či „snímač“ za totožné se „zařízení pro sledování a záznam náklonu“.

2. Motivace – pozadí problému

Při odchlípení sítnice je nutno sítnici operativně navrátit zpět. Vzhledem k fyziologii oka, kde sítnice směřuje „dovnitř“ koule, je dále zapotřebí sítnici přidržet *zevnitř*. Na pacienta jsou přitom kladeny velké nároky ohledně pozice hlavy.

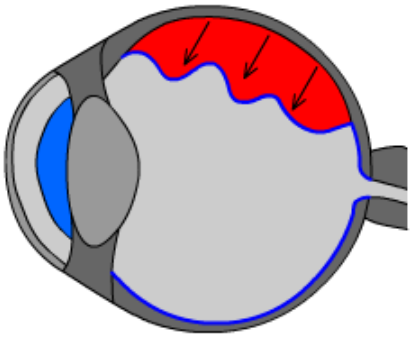
2.1 Relevantní část fyziologie oka

Sítnice je tenká (méně než půl milimetru) vrstva pokrývající vnitřní povrch oční koule. Při vidění je na ni promítán obraz a drážděny zde umístěné světločivné buňky – tyčinky a čípky. Signál z nich je sváděn nervovými vlákny do zrakového nervu a dále do centrální nervové soustavy. Sítnice spočívá na cévnatce. Důležitými částmi sítnice jsou žlutá a slepá skvrna, ty však nemají pro účely tohoto textu význam.

Nitro většiny oka vyplňuje sklivec – gelovitá tekutina, asi čtyřikrát viskóznější než voda. Tyto vlastnosti jí propůjčuje hyaluronová kyselina a systém kolagenových vláken. Dále je zde přítomna komorová voda (nitrooční tekutina), která zajišťuje regulaci nitroočního tlaku. Ta se se sklivcem ve zdravém oku nemísí a je tak přítomná především v anteriorní komoře. Sklivec neregeneruje a při odstranění je nahrazen komorovou vodou.

2.2 Odchlípení sítnice

Odchlípení sítnice je stav, kdy se světločivná vrstva sítnice přestane držet pigmentové (poslední vnější) a získá volnost pohybu. Jelikož je pro dobré vidění kritická správná geometrie sítnice jakožto svého druhu detekční matice, způsobuje to četné zrakové problémy. Dále je při tomto stavu narušena výživa sítnice, což vede k jejímu postupnému odumírání a tím i ztrátě zraku [1]. Pokud dojde k zasažení žluté skvrny, hovoříme o makulární díře.



Obr. 1: Odchlípení sítnice

Příčiny odchlípení lze rozdělit do tří skupin.

1. **Rhegmatogenní** – v sítnici vzniklou trhlinou proniká nitrooční tekutina a odtlačuje ji. Trhlina může vzniknout nárazem či upnutím sklivce na sítnici. Jde o nejběžnější příčinu.
2. **Exsudativní** – sítnici odtlačuje tekutina vzniklá pod ní. Příčinou je na rozdíl od předchozího případu zánět či nádor.
3. **Trakční** – vazivo vzniklé v důsledku cukrovky tahá za sítnici a odtahuje ji od podkladu.

Dále může teoreticky jít o komplikaci po očních operacích – pokud je odstraněn sklivec přidržující sítnici z jedné strany. Jak bude dále vidět, operace se tak mohou několikrát opakovat s tímtéž cílem.

Symptomem blížícího se odchlípení sítnice jsou pozorované jiskry či blesky, způsobené mechanickým drážděním nervových zakončení, které působí na sítnici, po samotném odchlípení také zkreslené vidění (zakřivení rovných čar).

2.3 Chirurgická léčba

Jelikož je odchlípení sítnice spíše mechanická závada, současná medicína zná pouze metody léčby sestávající z chirurgických zásahů. Podle komplikovanosti odchlípení a stupně poškození sítnice je možno volit ze tří metod, jejichž kořeny sahají i několik desítek let do minulosti [2]. Při jednoduchých odchlípeních není třeba do oka zasahovat, většinou je však bezpečnější pracovat se sítnicí přímo.

2.3.1 Pars plana vitrektomie (PPV)

V současnosti nejčastěji používanou metodou léčby odchlípení sítnice je PPV, při níž je do nitra oka proniknuto zepředu skrz řasnaté těleso, odstraněna část sklivce a následně takto uvolněný prostor využit pro přístup k sítnici. Tento postup je využíván i při odstraňování chorobných částí sklivce či předmětů v něm plovoucím (krev), které znesnadňují vidění [3].

Při nápravě odchlípené sítnice je použito např. „svaření“ sítnice s podkladem v několika bodech laserem.

Odebraný objem sklivce je po operaci nutno nahradit tamponádou která přitlačuje sítnici na místo, což je možné dvěma způsoby. Buď je použit plyn (více druhů), který má výhodu v biologické odbouratelnosti, takže již není zapotřebí do oka dále zasahovat. Nevýhodu naopak představuje velmi špatné vidění okem s plynovou bublinou a nutnost udržování hlavy v určité dané pozici.

Druhou možností je čirý silikonový olej, který sice nezhoršuje vidění, ale nevstřebá se a je nutno jej později z oka odsát. Na druhou stranu je však olejová výplň tolerantní ke změnám pozice, takže je vhodná i pro děti [2].

Oba způsoby výplně zvyšují riziko poškození rohovky a vzniku glaukomu nebo kataraktu.

Jelikož je odstraněn sklivec podporující částečně sítnici, může v krajně nepříznivých případech dojít k opětovnému odchlípení a nutnosti další operace.

2.3.2 Episklerální plomba

Tato metoda je vhodná při nekomplikovaných odchlípeních. Její hlavní výhodou je vnější přístup, kdy není zapotřebí oko otevřít. Stěna oka je na místech odchlípení zmrazena a zvenčí našita silikonová plomba, protlačující do oční koule jamku a tak přitlačující sítnici zpět k podkladu. Jelikož je přitlačení vytvořeno plombou zvnějšku připevněnou na oko, pozice hlavy není příliš důležitá, spíše vyvarování se nárazů [5].

2.3.3 Pneumatická retinopexe

Metoda se skládá ze vpuštění plynové bubliny do oka a teprve poté, co je takto sítnice přiložena zpět, je kryopekticky či laserem připevněna. Udržování správné pozice hlavy je zde nadmíru důležité po delší dobu.

2.4 Společné rysy požadavků na pacienty

Pro dvě ze tří zmíněných metod je nutno udržovat danou pozici hlavy, při níž bublina přitlačuje sítnici zpět a narovnává ji. V případě plynu bublina tlačí směrem nahoru. Díky výhodnějším vlastnostem plynu a jelikož důležitá část sítnice se nalézá především na zadní části oka, obvykle vyžadovaná pozice je „pohled dolů“, kdy je zadní část oka nejvýše [4]. Naopak zaujmutí opačné pozice má za následek výše zmíněné problémy (glaukom, zákal).

V současnosti je pro usnadnění plnění takovýchto požadavků pacienty použito speciálních postelí, podušek a židlí. Navrhovaný systém má umožnit pacientům větší volnost díky okamžité zpětné vazbě mezi jejich pozicí a splněním požadovaného náklonu a tak zvýšit jejich komfort. Další funkcí systému je kontrola ze strany lékaře, důležitá zejména při pooperačních komplikacích.

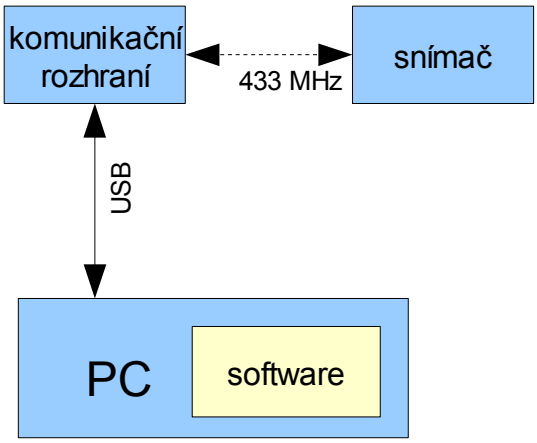
Obr. 2: Židle pro udržování předepsané pozice hlavy [6]

3. Popis externích částí systému

Systém se v celém rozsahu skládá ze tří částí:

1. Snímače polohy, umístěného na pacientově hlavě,
2. Bezdrátového komunikačního rozhraní, připojeného k PC,
3. Software pro práci s tímto systémem.

Popis poslední položky zároveň činí převážnou část této práce a proto není součástí této kapitoly.



Obr. 3: Schéma celého systému

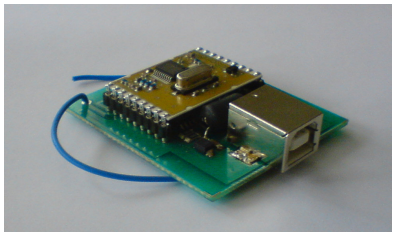
3.1 Bezdrátové komunikační rozhraní

Tato součást je v podstatě prostředníkem mezi rozhraním počítače a rozhraním snímače polohy. Jejím jediným úkolem je konverze signálů mezi formáty na obou stranách.

Rozhraní na straně počítače je USB port. K připojení je využit integrovaný obvod FT 232 firmy Future Technology Devices International Ltd. [11]. Tento obvod při použití příslušného softwarového ovladače rozšiřuje počítač o další sériový port. Takovéto uspořádání umožňuje snadno hardwarově realizovat komunikaci s dále připojenými obvody, již po sběrnici RS-232. Kromě toho je dále možno využít funkcí ovladače pro identifikaci připojeného zařízení – tento obvod může obsahovat textový popisek, kterým lze určit identitu připojeného zařízení.

Takto vzniklý sériový port je propojen s bezdrátovým modulem nRF401 firmy Nordic Semiconductor [10]. Tento modul je kompletní transceiver v pásmu 433 MHz, schopný jednosměrné (poloduplexní) komunikace [10]. Pro přepínání směru komunikace modulu jsou využity signály RTS a DTR sběrnice RS-232: při aktivním RTS modul přijímá, při DTR vysílá, ostatní stavy komunikaci blokují.

Směr komunikace signalizují dvě červené LED diody.



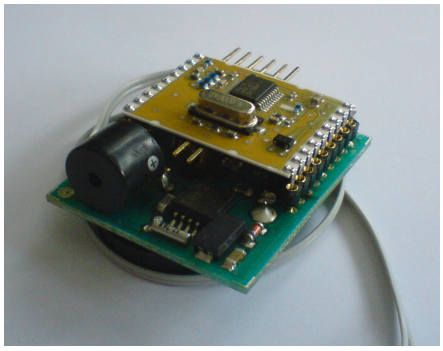
Obr. 4: Bezdrátové komunikační rozhraní

3.2 Snímač polohy hlavy

Samotný snímač je poháněn 8-bitovým mikrokontrolérem ATMEGA16L. Pro ukládání disponuje pamětí typu 24LC512, obsahující paměťový prostor o velikosti 64 KiB. Měření náklonu je realizováno miniaturním tříosým akcelerometrem. Dále snímač obsahuje bezdrátový modul totožný s oním umístěným na komunikačním rozhraní, jehož pomocí probíhá výměna dat s ním a počítačem. Pro akustickou signalizaci je připojen piezoměnič. Kromě toho je součástí snímače také kontakt pro inicializaci spojení a krátká anténka připojená k bezdrátovému modulu.

Snímač je v současné době ve formě prototypu a je napájen ze zdroje o napětí 3,6 V. To odpovídá původnímu záměru využít k napájení knoflíkové baterie CR2032 či podobného typu. Praktické testy komunikace však ukazují, že při odesílání většího množství dat dochází k proudové zátěži v řádu stovek mA a proto bude zapotřebí využít jiného zdroje.

Rozměry celého snímače činí 4 × 3,5 × 2,5 cm.



Obr. 5: Snímač polohy hlavy

3.2.1 Obsah a organizace pamětí snímače

Jak již bylo řečeno, snímač obsahuje mikrokontrolér a externí paměť. Tyto dvě složky obsahují data různého druhu. Zjednodušeně je možno říci, že v interní paměti se nalézají informace důležité pro samotnou činnost snímače, zatímco v externí se nalézají data nepotřebná při měření. Do externí paměti rovněž probíhá zápis naměřených dat.

Pro interní paměť se konkrétně jedná o následující položky: počitadlo probuzení (interval pro záznam, viz část „běžná činnost“), ukazatel na místo v externí paměti k zápisu, hodnota maximální povolené odchylky, interval ukládání měřených hodnot, časové konstanty pro vyhodnocení vstupu do nepovoleného náklonu a jeho ukončení, kalibrační hodnoty výchozího vektoru náklonu, hodnotu pro srovnání s napětím baterie.

0x0000	0x0001	0x0002	0x0003	0x0004	0x0005
status	počitadlo probuzení [-]		adresa posledního zápisu měř.		max. náklon [°]

0x0006	0x0007	0x0008	0x0009	0x000A	0x000B
perioda ukládání měření [s]		mrtvá doba [ms]		alarm minimálně po [ms]	

0x000C	0x000D	0x000E	0x000F	0x0010	0x0011
kalibrace X		kalibrace Y		kalibrace Z	

0x0012	0x0013	0x0014	0x0015	0x0016	0x0017
100% hodnota baterie		verze firmware			verze hw...

0x0018	0x0019	0x001A	0x001B	0x001C	0x001D
verze hardware (pokračování)					

Obr. 6: Obsah paměti mikrokontroléru (interní)

Barevně jsou odlišeny proměnné, jenž je vhodné měnit vnějšími zásahy. Oranžově podbarvený maximální povolený náklon je fundamentální hodnotou, jejíž hodnota přímo ovlivňuje celé měření. Pomocí adresy posledního zápisu (zelená) je zjišťováno umístění záznamu a vhodným nastavením (0x0024) lze provést jeho „vymazání“ z paměti. Žluté hodnoty ovlivňují časování poplachu při nepovoleném náklonu a frekvenci ukládání dat.

Za adresou 0x0019 následují oblasti, v nichž se nenalézají žádné hodnoty u nichž by byly žádoucí modifikace z vnějšku.

Externí paměť obsahuje – kopie nejvyšší hodnoty povoleného náklonu, jméno a příjmení pacienta, datum a čas operace, datum a čas zahájení záznamu. Dále jsou v externí paměti uloženy měřené hodnoty, obsahující vždy čas a hodnotu odchylky náklonu.

Na rozdíl od interní paměti je externí v celé velikosti strukturována podle specifikace; na začátku se nejprve nalézají informace o měření, nevyžadované k výpočtům. Zbytek adresního prostoru za nimi může být zcela využit pro záznam měření.

0x0000	0x0001	0x0002	0x0003	0x0004	0x0005
max. náklon [°]	jméno a příjmení				
0x0006	0x0007	0x0008	0x0009	0x000A	0x000B
jméno a příjmení					
0x000C	0x000D	0x000E	0x000F	0x0010	0x0011
jméno a příjmení					
0x0012	0x0013	0x0014	0x0015	0x0016	0x0017
jméno a příjmení			den	měsíc	rok (sta)
0x0018	0x0019	0x001A	0x001B	0x001C	0x001D
rok (jednotky)	hodina	minuta	den	měsíc	rok (sta)
0x001E	0x001F	0x0020	0x0021	0x0022	0x0023
rok (jednotky)	hodina	minuta			
0x0024	0x0025	0x0026	0x0027	0x0028	0x0029
měření			měření		
0x002A	0x002B	0x002C	0x002D	0x002E	0x002F
měření			měření		

Obr. 7: Obsah externí paměti

Barevné kódování obr. 7 je následující: Modrý rozsah paměti o délce 20 byte je vyhrazen pro jméno a příjmení pacienta. Za předpokladu záznamu bez diakritiky tato velikost odpovídá 20 znakům. Otázka kódování textu zde není nijak vyřešena – paměť zde slouží pouze pro „odložení“ dat během měření, proto zařízení nemusí nijak zkoumat její obsah a tato otázka je přenechána systémům provádějícím zápis a čtení tohoto bloku. Žluté bloky jsou informace o datu a čase operace (bledě žlutá) a začátku záznamu (sytě žlutá). Stojí za zmínku, že rok je kódován jako dvě části dekadicky zapsaného letopočtu, nikoliv jako 16-bitová hodnota. Oranžové pole s maximálním náklonem na začátku je pouze kopií stejného pole v interní paměti. Zelené bloky představují jednotlivé datové body měření, bitově kódované do 3 bytů; podrobný pohled viz obr. 8.

byte	0								1								2							
bit v byte	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
bit celk.	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
bit v prom.	6	5	4	3	2	1	0	4	3	2	1	0	5	4	3	2	1	0	5	4	3	2	1	0
proměnná	úhel [°]							hodiny					minuty				sekundy							

Obr. 8: Bitová mapa měřených dat

3.2.2 Běžná činnost snímače

Při běžném režimu provozu je po většinu času snímač v režimu měření, kdy je vypnut bezdrátový modul. Mikrokontrolér se aktivuje periodicky každou sekundu a měří vektor náklonu pomocí akcelerometru. Získaný vektor je srovnán s kalibračním, je zjištěna jejich odchylka a srovnána s nastavenou maximální povolenou hodnotou.

V případě, že je povolený úhel náklonu překročen, pokračuje měření po stanovenou dobu (v řádu stovek milisekund) a při trvajícím překročení povoleného úhlu je aktivován alarm piezoměniče a zaznamenán náklon. Zařízení dále pokračuje v měření i alarmu až do té doby, než náklon klesne zpět do povolených hodnot a zůstane tak alespoň po nastavenou dobu. Při opuštění nepovoleného náklonu je nová (povolená) hodnota uložena.

Pokud povolený náklon překročen není, zařízení zkontroluje, zda již vypršel nastavený interval pro záznam náklonu a pokud ano, zaznamená současnou hodnotu výchylky.

Takto je získán záznam obsahující jednak přibližně v daném intervalu běžná měření, ale i nepravidelně umístěné hodnoty při alarmech.

3.2.3 Režim komunikace

Pokud je po dobu několika sekund podržen kontakt, snímač opouští režim měření a přechází do režimu komunikace. Tehdy je aktivován bezdrátový modul, který zvyšuje spotřebu zařízení a způsobuje tak velkou zátěž baterie. Proto je žádoucí, aby byl režim komunikace co nejkratší.

Při režimu komunikace se zařízení snaží navázat kontakt s počítačem vybaveným bezdrátovým komunikačním rozhraním. Základní prvek celé komunikace představuje jistý druh smyčky, v níž se celá komunikace odehrává a která je udávána zařízením.

Zařízení odesílá po deset sekund postupně deset výzev k udržení komunikace. Během této doby přijímá jakýkoliv požadavek. Pokud ani po desáté výzvě nedojde k odpovědi, režim komunikace končí vypršením času. Je-li však v této době přijata odpověď žádající pokračování komunikace, zařízení odešle krátký popis svého stavu obsahující napětí baterie a aktuální výchylku, načež opět zahajuje cyklus desíti výzev.

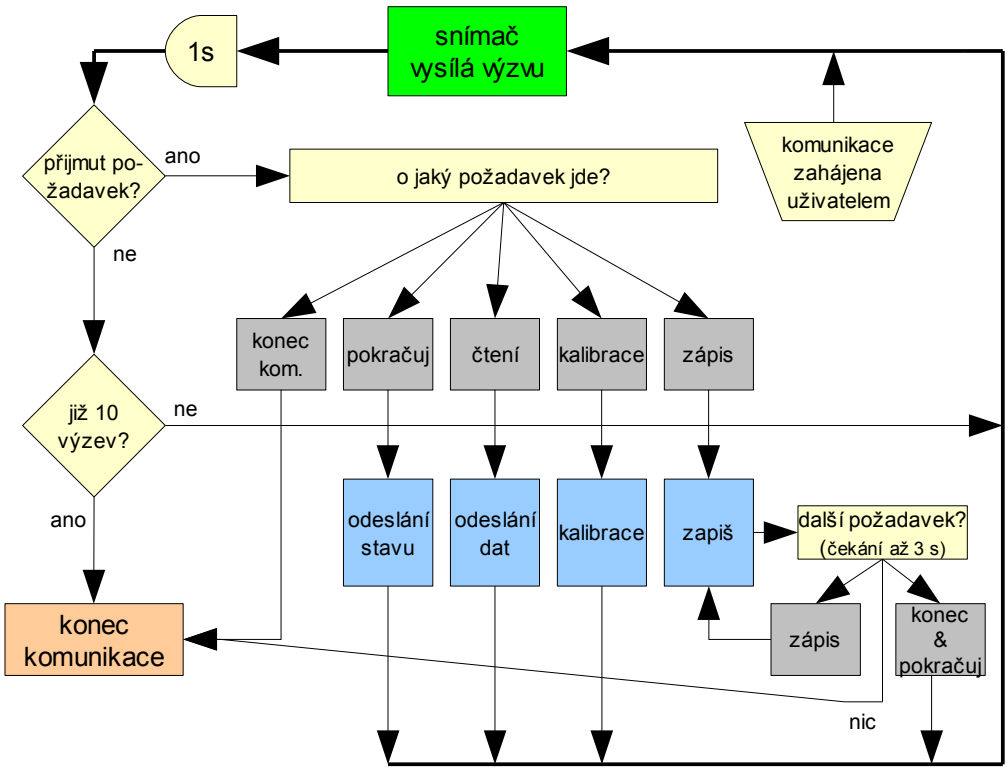
3.3 Schéma komunikace

Zkrácený popis komunikačního protokolu je již obsažen o odstavec výše. Základem je svého druhu smyčka, v níž snímač odpočítává deset výzev k udržení komunikace. Po každé výzvě je k dispozici 1 sekunda, v níž snímač naslouchá a může být odeslána odpověď či zahájena složitější fáze komunikace.

Celkově je komunikační schéma navrženo tak, aby snímač udával rytmus celé komunikace pomocí opakovaných výzev, zatímco počítač ji řídí a kontroluje její správnost. To je dáno tím, že protokol neobsahuje explicitní informace o datovém obsahu a snímač sám o sobě ji také nemá zabudovanou v takovém druhu, aby byl schopen popisovat obsah své paměti.

Komunikaci vždy iniciuje snímač, který má v běžném stavu RF modul vypnut. Ukončení komunikace naopak vydává počítač buď pasivně, když přestane odpovídat na výzvy, či aktivně při odeslání příkazu pro ukončení komunikace.

Celkový pohled na možné vývoje komunikace (včetně vypršení času na odpověď) nabízí obr. 9; šedě jsou zde zvýrazněny požadavky a modře reakce zařízení.



Obr. 9: Vývojový diagram komunikačního schématu

(zvýrazněna je společná část hlavní pseudo-smyčky)

Existují tři druhy požadavků, které může počítač zařízení odeslat a na něž dostane bezprostředně datovou odpověď nějakého druhu: pokračování komunikace, odeslání dat z interní a externí paměti.

Dále může zařízení přijmout čtyři druhy požadavků, na které neexistuje žádný druh datové odpovědi či potvrzení: nastavení interní či externí paměti, kalibrace a ukončení komunikace.

3.4 Komunikační protokol

Použité moduly pro bezdrátovou komunikaci obsahují kompletní logiku pro veškeré činnosti potřebné k navázání spojení a přenosu. Jejich rozhraní se pro ostatní hardware jeví jako černá skřínka, umožňující přepínat režim příjmu/vysílání a odesílat či přijímat souvislé proudy 8-bitových bloků – čili byte, tak jak je interně užíván PC i mikrokontrolérem.

Protokol tedy neobsahuje žádné synchronizační či ostatní nízkoúrovňová opatření a staví na schopnosti přenášet kompletní posloupnosti dat. Jedinými problémy, jenž musí řešit obě koncová zařízení softwarově, je příjem „falešných“ dat následkem rušení a časování přepínání příjmu či vysílání.

Protokol komunikace je založen na paketech o délce 15 byte. Základními položkami jsou informace o odesílateli a příjemci, číslo paketu, kontrolní součet a zakončovací značka. Celkově tyto informace zabírají 6 byte z 15, pro samotný datový obsah tedy zbývá 9 byte. Rozvržení této struktury znázorňuje obrázek 10.

0x00 0	0x01 1	0x02 2	0x03 3	0x04 4	0x05 5	0x06 6	0x07 7	0x08 8	0x09 9	0x0A 10	0x0B 11	0x0C 12	0x0D 13	0x0E 14
od	pro	pořadí	data									CRC16		konec

Obr. 10: Základní struktura paketu

V současné době je možná komunikace pouze mezi jedním PC a jedním snímačem, proto pole „od“ a „pro“ obsahují pouze hexadecimálně `0x11` pro zařízení a `0x55` pro PC.

Pole „pořadí“ obsahuje pořadové číslo paketu v sekvenci tvořící jednolitý datový proud. Konstruktivního využití však dochází pouze v případě že je odesílána sekvence paketů a příjemce chápe význam jejich pořadí. V ostatních případech je vždy ignorováno. Pořadí začíná vždy jedničkou, nikoliv nulou.

Paket je zakončen kontrolním součtem vytvořeným z prvních 12 byte paketu za použití algoritmu CRC 16. Zakončovací značka je vždy konstantní, `0x4E`.

Druh paketu je odvozován z různých kombinací jeho časového umístění, směru, nebo datového obsahu – pro tuto informaci není vyhrazeno samostatné pole.

3.4.1 Pakety výzvy snímače

Tyto pakety obsahují pouze informaci o stavu odpočtu smyčky a dávají najevo přítomnost snímače naslouchajícího v režimu komunikace. Jejich příjem tak mimo jiné signalizuje PC, že může vyslat další příkaz.

od	pro	pořadí	výzva	výplň								CRC16		konec
0x11	0x55	0x01		0x00	0x01	0x00	0x01	0x00	0x01					0x4E

Obr. 11: Obsah paketu výzvy

Na obr. 11 je znázorněn datový obsah tohoto druhu paketu. Zeleně je podbarvena proměnná informace, žlutě konstantní. Jak je vidět, tento druh paketu obsahuje pouze jedno datové pole, udávající pořadí výzvy. Nabývá hodnot od 1 po 10. Ostatní data, představující jakousi výplň, umožňují jednoznačně určit že se jedná o paket výzvy.

3.4.2 Paket pro udržení spojení

Tento paket je odesílán PC v případě, že se počet výzev blíží konečnému počtu desíti či v jakémkoliv dřívějším okamžiku. Signalizuje snímači, že má pokračovat další vyčkávací smyčkou.

Zařízení na příjem tohoto paketu reaguje restartem počítání výzev a odesláním stavově informačního paketu (viz níže).

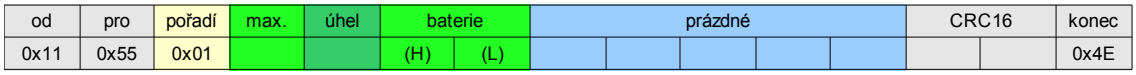
od	pro	pořadí	udrž.	prázdné								CRC16		konec
0x55	0x11	0x01	0x05											0x4E

Obr. 12: Obsah paketu pro udržení spojení

Obr. 12 ukazuje obsah. Barevné kódování: žlutá konstantní obsah, modře libovolná data. Jelikož jde o paket odesílaný PC, dodržuje jistou nepsanou konvenci, že druh příkazu je obsažen v prvním byte datového obsahu (zvýrazněno). Veškerá data jsou konstantní či libovolná.

3.4.3 Stavový paket snímače

Tento paket je snímačem dvakrát odeslán po každém požadavku na udržení spojení, jako svého druhu vedlejší efekt. Informuje pouze o aktuálním stavu zařízení, což zahrnuje současný náklon a napětí baterie.



Obr. 13: Obsah stavového paketu

Obsah viz obr. 13. Zeleně jsou proměnná data, modře libovolná výplň volného místa. Data obsahují maximální povolenou hodnotu výchylky, současnou výchylku a napětí baterie odečtené interním A/D převodníkem mikrokontroléru snímače. Napětí je odečteno jako 12-bitová hodnota a proto je rozloženo do dvou byte.

3.4.4 Paket požadavku na blok paměti

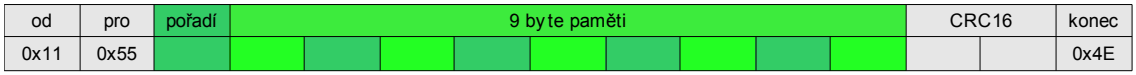
Tento paket signalizuje snímači, že má odeslat sekvenci paketů obsahující daný blok paměti. Jak již bylo dříve řečeno, snímač obsahuje dva druhy paměti, interní a externí. Požadavek je proto zcela popsán třemi informacemi: druhem paměti, počátkem a koncem bloku.

Obr. 14: Obsah paketu požadavku na paměťový blok

Obrázek 14 opět znázorňuje datový obsah. Druh je nastaven na `0x01` pro interní a `0x02` pro externí paměť. Adresy v paměti jsou odesílány jako 32-bitové proměnné, ačkoliv snímač používá pro adresování pouze 16 bitů.

3.4.5 Paket s datovou odezvou

Tento druh paketu je odpovědí na předchozí požadavek. Tvoří sekvence a proto je u něj skutečně využito pole „číslo paketu“ (zatím zřejmě v jediném případě v rámci celého komunikačního protokolu).



Obr. 15: Obsah paketu s datovou odezvou

Každý takovýto odpovědní paket obsahuje data z devíti po sobě následujících buněk paměti, čímž je zcela využita dostupná kapacita polí paketu. Ve kterém druhu paměti snímače a kde se tato data nacházejí je nutno odvodit z předchozího požadavku, na nějž je tento paket

odpovědí. První paket sekvence (pořadí je rovno jedné) začíná bytem v paměti, ležícím na první požadované adrese (tedy offset 0)!

3.4.6 Paket s požadavkem zápisu

Tento paket je odesílán PC při nastavování paměti snímače. Na rozdíl od příjmu dat je součástí každého paketu informace o adrese i datech, takže nevzniká nutně souvislá sekvence umožňující číslování. Zápisové pakety tedy inkrementálně číslovány sice být mohou, ale nemusejí.

od	pro	pořadí	zápis	adresa 1			data 1	adresa 2			data 2	CRC16		konec
0x55	0x11			(H)		(L)		(H)		(L)				0x4E

Obr. 16: Obsah paketu s požadavkem zápisu

Obsah, uspořádaný dle obr. 16, zahrnuje kromě pořadí paketu dále druh zápisu – `0x03` pro interní a `0x04` pro externí paměť. Samotný datový „náklad“ představuje pouhé dva byte, jelikož jsou s nimi zároveň odesílány i jejich adresy, uložené zde jako 24 bitů (opět je zbytečně každou využit navíc 1 byte).

Na tento požadavek nepodává zařízení žádnou odpověď, takže je správnost uložení nutno zkontrolovat buď následným čtením zapisované paměti, nebo zajistit mnohonásobným odesláním.

Po odeslání paketů požadavku zápisu je nutno potvrdit konec zápisu odesláním paketu konce komunikace a ihned poté reaktivovat spojení paketem pro udržení spojení.

3.4.7 Kalibrační paket

Tento paket je vysílán PC v případě, že má být zařízení kalibrováno, tzn. uložena současná poloha jako výchozí. Na tuto akci není podána žádná odpověď.

od	pro	pořadí	kalibruj	prázdné								CRC16		konec
0x55	0x11	0x01	0x06											0x4E

Obr. 17: Obsah kalibračního paketu

3.4.8 Paket konce komunikace

Tento paket má dvě funkce. Za prvé slouží při odeslání v hlavní smyčce komunikace jako oznámení konce spojení, po němž má snímač za úkol odpojit RF modul a přejít opět do stavu měření. Za druhé ukončuje sekvenci zápisů (viz výše).

od	pro	pořadí	konec	prázdné								CRC16		konec
0x55	0x11	0x01	0xFF											0x4E

Obr. 18: Obsah paketu konce komunikace

Reakce zařízení záleží podle kontextu příjmu tohoto paketu, jak již bylo řečeno dříve. Toto jej činí jednou ze slabších částí komunikačního protokolu.

3.4.9 Shrnutí formátu paketů a návrhu protokolu

Jak lze vidět, struktura paketů odesílaných snímačem se značně liší. Předpokladem je, že počítač zpracovávající pakety má dostatek výkonu na složitější analýzu jejich obsahu. Pakety posílané snímači naproti tomu obsahují vždy na 4. pozici číslo udávající druh požadavku, takže snímači stačí prozkoumat pouze toto jedno pole pro rozhodnutí o druhu reakce.

Z hlediska unifikace by však pravděpodobně bylo žádoucí, aby pakety všech druhů obsahovaly o jedno pole více, kde by byl vždy uložen druh paketu, bez ohledu na odesílatele a příjemce. Slabé místa protokolu dále představují stavový paket snímače a paket pro ukončení komunikace.

První z nich není dokonale určen svým obsahem a lze jej splést s datovým paketem. Mohl by tedy obsahovat nějaký druh jednoznačného označení a dále rovněž i maximální hodnotu napětí baterie, která je konfigurovatelná.

Paket ukončení komunikace je svým dvojím naprosto odlišným použitím rovněž značně neurčitý. Kvůli rušení je vhodné pakety odesílat mnohonásobně (typicky 3-5 krát), což u tohoto paketu znamená potenciální nebezpečí nechtěného ukončení komunikace. Při jediném odeslání zde však hrozí poněkud ironicky opět ukončení komunikace, tentokrát vypršením časového limitu při neukončeném zápisu nezjištěním tohoto paketu.

3.4.10 Algoritmus CRC 16

CRC 16 náleží do rodiny cyklických redundantních součtů (cyclic redundancy check). Ty jsou často používány pro jednoduchý matematický základ a snadnou implementaci na hardwarové i softwarové úrovni. CRC umožňují pouze detekci chyb, nikoliv jejich opravy.

CRC jsou založeny na zpracování řetězce bitů pomocí klíče – dalšího řetězce bitů. Při interpretaci obou řetězců jako polynomů o bázi 2 a koeficientech odpovídajících hodnotám jednotlivých bitů (možno tedy pouze 1 a 0!) je možno provedení CRC zapsat jako zbytek po dělení polynomu představujícího původní zprávu polynomem klíče, přičemž veškeré koeficienty jsou zkráceny na poslední binární číslici. Výsledný (zbytkový) polynom má tudíž opět koeficienty rovny jedné či nule a lze jej také opět převést na binární řetězec. Za předpokladu shodné konstantní délky zprávy a klíče bude i délka výsledku konstantní a o jedna menší než délka vstupů (lze ji proto rozšířit na délku vstupu).

Hardwarová realizace je založena na posuvném registru prokládaném hradly XOR, propojenými s koncem registru a umístěnými na pozicích odpovídajících přítomným koeficientům klíče [12]. Softwarová realizace může být založena buď v pomalejší verzi na imitaci funkce takovéhoto posuvného registru, nebo ve (většinou) rychlejší verzi na kombinaci operace XOR s tabulkovým vyhledáváním.

Vzhledem k možné variabilitě délky a zejména obsahu klíče zahrnuje rodina CRC množství variant pod různými názvy. Zejména pro 16 bitů existuje celá škála variací podle klíče, standardizovaných pro různé druhy využití např. v telekomunikacích, sběrnicích nebo discích. Varianta používaná zde má klíč ve tvaru $x^{16}+x^{15}+x^2+1$.

4. Detaily implementace programu

Tato část obsahuje popis kódu pohánějícího aplikaci, tedy pohled „zevnitř“. Objasňuje různé volby předcházející samotnému převedení do programového kódu a vztah mezi jednotlivými komponentami (míněno především celými formuláři) uživatelského rozhraní. Dále popisuje použité formáty pro reprezentaci dat a způsoby manipulace s nimi.

4.1 Použitý jazyk a knihovny

Program je vyvíjen v IDE prostředí C++ Builderu, které je vhodné z několika hledisek.

Především, umožňuje velmi pohodlnou tvorbu grafického rozhraní pomocí ucelené knihovny ovládacích prvků, předzpracovaných vazeb a spolupráce mezi nimi. Návrh rozhraní probíhá editací prvků ve WYSIWYG formě, nezbytný kód pro zajištění jejich propojení a funkce na úrovni grafické vrstvy operačního systému je zcela skryt a „viditelné“ je pro programátora pouze rozhraní umožňující snadnou manipulaci s vlastnostmi jako pozice, popisky atd. Na programátorovi je již pouze tvorba kódu popisující skutečné interakce dodávající grafickému rozhraní zamýšlenou funkcionalitu.

Za druhé, použitím jazyka C++ je zaručena možnost propojení či znovuvyužití množství již existujícího kódu, často volně dostupného.

Posledním argumentem mluvícím ve prospěch tohoto IDE je jeho volná dostupnost pro nekomerční účely, úhlavní nevýhodu však představuje stáří této verze tohoto programu a veškerých knihoven s ním dodávaných; přesto však i dnes stále není obtížné je využít.

Pro ukládání a načítání dat formulářů je využita knihovna, umožňující práci s konfiguračními soubory formátu INI. Tato je dostupná ve formě zdrojového kódu.

K práci s bezdrátovým komunikačním rozhraním je využita externí knihovna, dodávaná výrobcem čipu v něm použitém. Tuto knihovnu výrobce poskytuje ve formě DLL souborů a hlaviček pro jejich import.

Program již dále nepoužívá žádné další (rozměrné) externí knihovny a je tak plně replikovatelný ze zdrojového kódu a příslušných knihoven pomocí volně šiřitelné verze IDE. Tento stav je nanejvýše žádoucí zachovat i během dalšího vývoje.

4.2 Charakter dat

Snímač zaznamenává náklon tří os, srovnává s kalibrační hodnotou a poté přepočítává na odchylku v absolutní hodnotě, tedy teoreticky maximálně 180°. Záznam je prováděn podle potřeby, obecně je možno předpokládat, že snímač ve snaze uspořit paměť bude zapisovat pouze význačné změny v určité toleranci a hodnoty mimo povolený rozsah.

Udávané časové rozlišení činí přibližně jednu sekundu. Vstupem je tedy datová řada o rozsahu 0–180° s neekvidistantním vzorkováním.

4.3 Plán řešení

Program musí být schopen předat smysluplně informaci o průběhu velikosti výkyvů v čase; z hlediska uživatele důležitou součástí tedy bude graf s časovou osou a znázorněním velikosti výkyvu.

Zároveň by měl program přehledně zobrazit informace o situaci v komunikačním řetězci, která zahrnuje minimálně informaci o tom, zda probíhá komunikace se snímačem či nikoliv. Samozřejmostí je dále řízení komunikace – uživatel musí mít umožněno jednoduše snímač požádat o data, nastavit jej a kalibrovat.

Data mají pokud možno co nejpřesněji popisovat již proběhlé události – není potřeba je nijak upravovat a měnit. Jedinou součástí, kterou lze dodatečně upravit, jsou metadata – popis pacienta, doby záznamu a nastavení zařízení (především povolená odchylka náklonu).

Pro snadné přizpůsobení rozhraní podle individuálních preferencí lze využít technologie MDI („okno v okně“) s ukládáním pozic a velikostí jednotlivých pod-oken.

Jelikož je samotná komunikace se snímačem poměrně intenzivní vstupně–výstupní operace s notnou dávkou čekání, probíhá v samostatném vlákně. K výzvám synchronizace se snímačem se tedy přidává další, synchronizace vláken uživatelského rozhraní a pracovního.

4.4 Rozvržení odpovědností a činností do tříd

Pro usnadnění manipulace s kódem a lepší organizaci byla celá aplikace rozdělena do několika hlavních tříd, obalujících příslušnou funkcionalitu a komunikujících vzájemně pouze přes své vnější rozhraní. Tím je umožněn oddělený vývoj a ladění jednotlivých částí, stejně jako snadnější orientace v kódu.

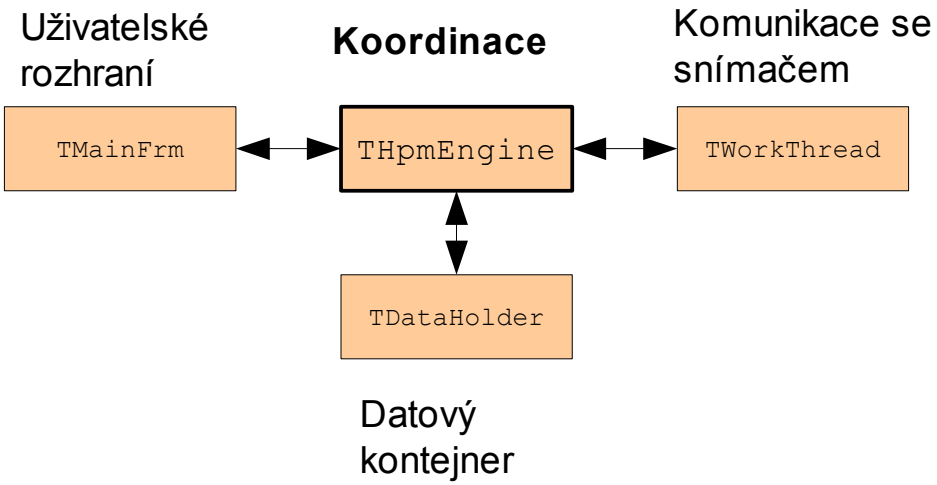
Ústředním uzlem celého systému je třída `THpmEngine`, jejíž úlohou je především propojení jednotlivých subsystémů a zajištění jejich vzájemné spolupráce. S ní komunikují dvě další části systému.

Za prvé je s ní propojen hlavní formulář, který zároveň ovládá většinu uživatelského rozhraní a reprezentuje tak grafický výstup a interaktivní vstup pro uživatele do celého systému. Zde je propojení realizováno pomocí událostí (callback).

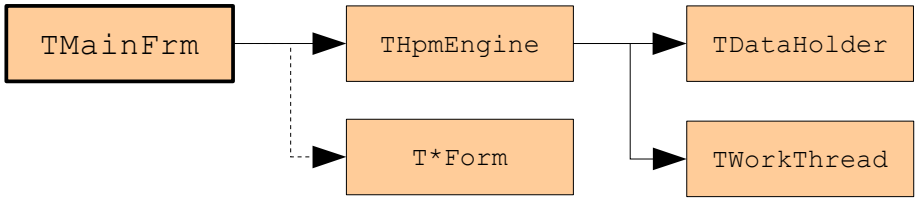
Za druhé je to třída `TWorkThread`, která zajišťuje komunikaci s bezdrátovým rozhraním a snímačem, a to s pomocí několika dalších vlastních tříd. Jelikož je kód této třídy vykonáván ve vlastním vlákně, propojení probíhá pomocí zpráv systému Windows (messages).

Konečně třetí, již spíše pouze pasivní, součástí centrálního systému je třída `TDataHolder`, která plní úlohu datového kontejneru.

Z hlediska vlastnictví, tedy která třída je pouze položkou jiné třídy, je situace odlišná; Vlastníkem celého systému je hlavní formulář `TMainFrm`, jenž vlastní `THpmEngine`, jejímiž podřízenými jsou následně `TDataHolder` a `TWorkThread`. Hlavní formulář také z určitého hlediska vlastní veškeré ostatní části uživatelského rozhraní, neboť je rodičem MDI.



Obr. 19: Vztahy mezi hlavními součástmi programu z hlediska interakcí



Obr. 20: Vztahy mezi hlavními součástmi programu z hlediska vlastnictví

4.5 Datový kontejner

Funkcionalitu práce s daty zcela abstrahuje třída `TDataHolder`, která má prostředky pro vstup, uchování a rešerše nad daty, tzn. sekvencí `TDataItem`. Dále je zde uložena hlavní kopie nastavení typu `TPatientData`.

Základním prvkem je seznam typu `TList` nazvaný prostě `Data`. V něm se nachází sekvence vzorků. Vzhledem k systému zacházení s daty není k těmto položkám jiný přístup než pouze pro čtení, a to dvěma způsoby.

Prvním je přístup ke konkrétnímu vzorku v pořadí podle jeho indexu; tato funkce je v podstatě pouhým předáním indexu do spodní vrstvy (`Data`).

Druhou formu přístupu představuje zjištění prvního indexu po určitém čase. Příslušná funkce se jmenuje `FirstIndexFor`. Toto vyhledávání pracuje na principu bisekce seznamu vzorků podle jejich času. Binární vyhledávání je použito zejména kvůli algoritmické třídě O(log n), čímž je pohodlně eliminována potřeba další optimalizace při překreslování grafu (viz dále). Pro urychlení konvergence je navíc nejprve provedena interpolace podle časového rozmezí prvního a posledního vzorku. Zkušební kód v Matlabu prokázal, že se tak sníží počet iterací bisekce přibližně o pět, ovšem pouze na velkých datových objemech vyžadujících zhruba 15 kroků pro nalezení cíle. Použití interpolace dále předpokládá střednědobý průměr časového kroku víceméně konstantní, což však lze ve většině případů předpokládat a není kritickou podmínkou. Z hlediska analýzy celkového chování algoritmu se prvotní interpolace i v nejhorším případě stává pouze jedním méně efektivním krokem. Jelikož nelze nijak zaručit existenci vzorku přesně v dotazovaném čase, funkce vrací první viditelný vzorek v či po zadaném čase.

Načítání dat je rozvrženo s ohledem na import ze snímače, jehož možné implementace by teoreticky mohly být i asynchronní. Samotná akce je proto rozdělena do tří: inicializace, přidání jednotlivých datových bodů a finalizace. Při inicializaci jsou nejprve odstraněna stará data, následně jsou po jednom přidány nové vzorky a při finalizaci jsou aktualizovány cachované informace získané z celého průběhu dat, jako např. časové rozmezí.

`TDataHolder` dále obsahuje veškeré nastavení a metadata – informace o pacientovi – stejně jako metody manipulace s nimi, které jsou zcela přímočaré, bez jakéhokoliv složitějšího obsloužení. Zde platí, že za korektní manipulaci s těmito daty odpovídá přistupující kód.

Pro pohodlnější použití jsou přímo v tomto datovém kontejneru k dispozici i dvě funkce, podávající informaci o přítomnosti dat a stavu načítání.

4.5.1 Reprezentace dat

Jak již bylo zmíněno dříve, jednotlivé vzorky dat jsou reprezentovány hodnotou udávající náklon, přičemž je zapotřebí dále vědět, kdy k události došlo. Na základě neekvidistantního rozmístění vzorků v čase je nutno vyloučit datové schéma, ve kterém je samotná pozice (index) vzorku informací o jeho časovém umístění. Proto je jeden vzorek reprezentován dvojicí hodnot: časem od začátku měření v sekundách a náklonem vzhledem ke kalibrační poloze. Deklarace příslušného prvku tedy viz ukázka 1.

```
class TDataItem {
  public:
    int t;
    float value;
    TDataItem() {};
    TDataItem(int time, float val)
      {t = time; value = val; };
    ~TDataItem() {};
};
```

Ukázka 1: Deklarace elementárního datového vzorku

Celočíselný 32-bitový (se znaménkem) čas v sekundách umožňuje nejzazší uloženou hodnotu 2147483647, odpovídající přibližně 68 rokům. Nejbližší úspornější varianta je 16-bitový bezznaménkový typ, kde je maximální hodnota 65535 odpovídající o něco málo více než 18 hodinám. V kontextu těchto dvou informací je zřejmé, proč zde nelze více šetřit pamětí a proč je jednodušší poté ponechat znaménko či spíše irelevantní se jím již dále zabývat. Vzhledem ke kapacitě paměti mikrokontroléru ve snímači lze očekávat záznamy trvající nanejvýše několik týdnů.

Pro uložení hodnoty náklonu byl zvolen nejmenší datový typ s plovoucí desetinnou čárkou, taktéž 32bitový, takže celková velikost jednoho datového bodu je v paměti teoreticky 8 byte. V praxi ovšem záleží na zarovnání členů struktur podle kompilátoru, běžně tedy i dvakrát tolik. Dalším faktorem mohou být režijní náklady na objektový model. Zaručit tedy lze pouze, že data budou v paměti zabírat nejméně 8 byte na vzorek.

4.5.2 Nastavení

Pod tento jednoduchý pojem jsou shrnuta jak metadata, tak nastavení zařízení. Manipulace s nimi probíhá na vyšších úrovních sjednoceně – pro zjednodušení činností v uživatelském rozhraní. Při pohledu zpět do kapitoly o organizaci paměti snímače lze vyčíst, že jde o: povolenou odchylku, interval záznamu, mrtvou dobu, minimální délku alarmu, jméno pacienta a datum a čas operace a začátku záznamu. Deklarace záznamového typu shrnujícího tyto položky viz ukázka 2:

```
typedef struct {
  unsigned char max_angle;
  unsigned char * name; // max length 20 !
  IntTime operation, monitoring;
  unsigned long save_every_s, eval_false_ms, eval_true_ms;
} TPatientData;
```

Ukázka 2: Deklarace datového typu pro nastavení

Poznámka: názvy jednotlivých položek vycházejí z historického vývoje programu nebo starší dokumentace a proto neodpovídají terminologii použité zde v textu.

S výjimkou textové informace lze všechny ostatní položky přímo kopírovat. Pro Snadnější manipulaci jsou dále definovány funkce, které značně abstrahují většinu činností prováděných s tímto datovým typem.

- `InitPatient` – inicializuje záznam na vhodné výchozí hodnoty a vynuluje ukazatel na jméno. Hodnoty jsou 1.1.1900 0:00:00 pro časy, interval ukládání dat 30 s, mrtvá doba 1000 ms a min. doba alarmu 500 ms.
- `AssignPatient` – kopíruje data z jednoho záznamu do druhého, přičemž blok paměti se jménem a příjmením je duplikován, nikoli referencován podruhé.
- `PatientSetName` – nastaví jméno a příjmení včetně bezpečné likvidace jejich předchozího obsahu.
- `ClearPatient` – připraví záznam pro zrušení; uvolní paměť se jménem a příjmením.

Ostatní operace s nastavením, zejména čtení textové informace a kopírování do prvků uživatelského rozhraní, mohou s výhodou využít implicitních typových konverzí C++ Builderu.

4.6 Koordinační objekt

Úlohu koordinátora celého programu sehrává třída `THpmEngine`. Z programátorského hlediska jde o jakýsi adaptér, který se nachází mezi třemi ostatními subsystémy, s nimiž interaguje různými způsoby a jejichž různé komunikační druhy (synchronní a asynchronní) všechny zvládá.

Datový kontejner sám nespouští žádné akce, je pouze řízen ostatním kódem – při příjmu dat ze snímače právě koordinátorem, při načítání z disku přímo grafickým rozhraním. Koordinátor poskytuje referenci na tento podřízený objekt jakožto jednu z jeho vlastností.

Uživatelské rozhraní interaguje s koordinátorem při vydávání požadavků přímým voláním funkcí tohoto objektu. Při opačném směru pohybu informace, tedy vycházejícím od koordinátora, je využito událostí (callback) – procedurálních proměnných, které koordinátoru umožňují zpětně volat funkce jiných objektů, zde hlavního formuláře.

Funkce poskytnuté uživ. rozhraní zahrnují požadavek na nastavení, odeslání nastavení, kalibraci, restart záznamu, požadavek na data a odpojení. Dostupné události jsou především při příjmu dat a nastavení, avšak většina jich informuje o stavu komunikace se snímačem – jde o informaci o stavu komunikačního řetězce, odpočtu výzev, stavu snímače, změně směru vysílání/příjmu a statistice komunikace.

Komunikace s vláknem obsluhujícím bezdrátové komunikační rozhraní musí být asynchronní, poněvadž je jeho kód vykonáván nezávisle na zbytku aplikace. Z druhů mezivláknově bezpečné komunikace dostupných v systému Windows (trubky, zprávy, COM, sdílená paměť, sokety) padla volba jednoznačně na zprávy, kvůli jejich jednoduchému zpracování. Pro zjednodušení má koordinační objekt vytvořeno vlastní okno pro příjem zpráv (okno je zde nutno chápat ve smyslu interní terminologie Windows, nikoliv uživatelského rozhraní), vlákno pak přijímá zprávy vlastní frontou.

Zprávy vyměňované s vláknem odpovídají víceméně funkcím a událostem poskytovaným uživatelskému rozhraní.

Jak datový kontejner, tak vlákno jsou koordinátorem vlastněny, tzn. koordinátor odpovídá za jejich vznik a zánik v pravou chvíli v životním cyklu aplikace.

4.6.1 Práce se zprávami Windows

Zprávy (messages) představují jeden ze základních prvků organizace systému Windows, především v oblasti grafického rozhraní, ale nacházejí využití i jinde. Východiskem pro činnost programů s grafickým rozhraním je hlavní smyčka, v níž kontrolují svou frontu zpráv a tyto zprávy následně dále rozesílají jednotlivým prvkům. Většinu zpráv v tomto případě generuje samotný systém Windows a informuje jimi programy o vstupech z myši, klávesnice a dalších zařízení pro vstup. Kromě toho však lze zprávy posílat jakémukoliv prvku, jenž má alokováno handle („rukověť“) s požadavkem příjmu zpráv, či mezi jednotlivými procesy, vlákny i do celého systému.

Zprávy se skládají ze čtyř položek: cílový handle, číselný kód zprávy a dvě datové položky označované jako `WParam` a `LParam`. Všechny tyto položky mají shodnou velikost 32 bitů. Cílový handle je určen pro API systému Windows, ostatní části určují obsah zprávy. Kromě těchto položek bývá v souvislosti se zprávami zmiňována také jako pátá návratová hodnota; ta však není součástí zprávy samotné a také nemusí být vždy určena. Hodnota čísla zprávy může nabývat mnoha hodnot s předdefinovaným významem, takže je nutno při tvorbě vlastního systému využívajícího zpráv brát v potaz, zda bude dokonale izolován od veškerého ostatního kódu běžícího v celém spuštěném operačním systému, případně jakým způsobem určit čísla zpráv volná k použití. Předdefinované číselné hodnoty jsou většinou k dispozici ve formě pojmenovaných konstant či maker, jejich předpona určuje oblast k níž se daná zpráva váže.

Jeden způsob jak získat číslo zprávy zaručeně jedinečné v celém systému je volání funkce API `RegisterWindowMessage`, která pro daný jedinečný textový řetězec vygeneruje jedinečné číslo zprávy. Při opakovaném použití řetězce je vraceno totéž číslo; tento postup je tedy vhodný zejména při komunikaci mezi samostatnými programy. Vzhledem k tomu, že získaná hodnota není předem známa, však takový postup není vhodný z hlediska jazykové složitosti konstrukcí zpracovávajících zprávy. Pro použití v rámci jednoho programu je schůdnější cestou využití číselných rozsahů určených pro volné použití, kde je možno definovat čísla zpráv jako konstanty.

Práce se zprávami je z hlediska API v zásadě dvojího druhu – odesílání a příjem. Odesílání je možno provádět především funkcemi `SendMessage` a `PostMessage` [13]. Rozdíl je ve způsobu zpracování, který využívá faktu, že zprávy přenáší třetí strana – operační systém. Zatímco první verze při volání čeká na zpracování zprávy cílovým oknem a navrací hodnotu podle jeho výsledku, druhá ihned pokračuje dále a výsledek příjmu zprávy nezjišťuje. Dále existuje celá řada úžeji specializovaných variant, sledujících však vždy jeden z těchto dvou vzorů chování. Pro zde popisovanou aplikaci má význam především

`PostThreadMessage`, jenž umožňuje odesílat zprávy vláknu programu (v případě že ono vlákno vytvořilo svou vlastní frontu zpráv).

Příjem zpráv realizuje podobně dvojice funkcí `GetMessage` a `PeekMessage`. Rozdíl je opět v čekání. První varianta vybírá existující zprávy z fronty či čeká na příchozí zprávy a je tudíž vhodná zejména pro vlákna s uživatelským rozhraním (primární), kde je vhodné, aby vlákno při nečinnosti zbytečně „necyklovalo“. Naproti tomu `PeekMessage`, jak již název napovídá, v případě vyprázdnění fronty signalizuje tento stav a umožňuje pokračování provádění kódu. Proto je vhodnější v situaci, kdy je prvořadým úkolem kódu intenzivní zpracování a zprávy pouze usměrňují tuto činnost. Obě funkce také při prvním volání zakládají pro vlákno frontu zpráv, pokud nebyla dosud vytvořena.

4.7 Reprezentace času

Jelikož je aritmetické použití výchozích časových typů C++ Builderu nadmíru složité, pro potřeby této aplikace bylo snazší vytvořit soustavu vlastních, s přesně definovanými vztahy a rozsahy platnosti, odsouvající stranou problémy přestupných roků, délky měsíců atd. Příslušné definice jsou v souboru `TimeHolder.h`. Základním východiskem prolínajícím se celou koncepcí je měření času v celých sekundách.

K dispozici jsou dva datové typy – `IntTime` a `LocalTime`. První je „globální“, který zahrnuje datum i čas odpovídající současně používanému Gregoriánskému kalendáři. Druhý je lokální, vztahující se pouze k době záznamu dat. Vzniká tak jakýsi hybridní druh časové reprezentace, odvozený od počátku dne v němž bylo započato měření. Hypoteticky lze uvažovat i třetí, implicitní typ, skládající se pouze z celočíselné reprezentace sekund od začátku měření, avšak zde nelze mluvit o žádné struktuře a tak nemá smysl jej popisovat.

Čas a datum počátku měření je vzhledem k potřebě kompletní informace uložen jako `IntTime`. Pro konverzi `IntTime` na `LocalTime` je zde funkce `GlobalToLocalTime`, jenž ponechá informace o času a zahodí všechny informace o datu vyjma dne, který nastaví na první.

`LocalTime` se uplatňuje při výpočtech časové osy a jejího vztahu k reálnému času. Nejdůležitější je přitom funkce `MinimizeLocalTime`, která ve své delší verzi bezpečně přičítá k proměnné typu `LocalTime` libovolné množství sekund a rozpočítává je dále na dny, hodiny a minuty.

Vzhledem k omezením daným hardwarem snímače mají data z něj přicházející čas zaznamenán ve formě tří proměnných popisujících hodinu, minutu a sekundu záznamu. Tato reprezentace je pro další práci nevhodná a ihned při načítání je konvertována do formátu, kdy je každý datový bod popsán počtem sekund uplynulých od začátku měření. Doplňující informace o dni je získána analýzou sekvence měření, poté je zjištěn rozdíl od začátku měření a převeden na sekundy.

Celkový životní cyklus časové informace tedy lze popsat takto: Informace získaná z datového souboru či uživatelského rozhraní typu `TDateTime` nebo interního formátu snímače je konvertována na prosté číslo a uložena pro daný běh dat. Data jsou uložena s časem v sekundách od začátku měření. Při vykreslování je poté nejprve konvertován počáteční čas na `LocalTime`, poté přičten potřebný počet sekund a výsledná informace je zobrazena.

Jak je zřejmé, většina z těchto konverzí je zapotřebí pro popis časové osy grafu a přechod mezi časem absolutním a relativním vzhledem k počátku měření.

Dále je v tomto souboru umístěna dvojice funkcí pro převod roku z formátu počítače do formátu používaného pro ukládání v paměti snímače. Jejich názvy jsou `YearToDevice` a `YearFromDevice`. V paměti počítače je rok uložen jako prosté binární číslo, snímač však využívá formát dvou oddělených dekadických částí letopočtu ve dvou byte (viz také kapitola „Obsah a organizace pamětí snímače“) Konverze mezi zmíněnými dvěma formáty tedy zahrnuje bitové posuny, maskování, násobení a dělení.

Teoreticky zde může být zahrnuto i téma měřítka grafu, avšak to se nedotýká problematiky samotné reprezentace času a tak bude popsáno ve spojitosti s grafem, nikoliv zde.

4.8 Komunikační vlákno

Komunikační vlákno je z hlediska složitosti a relevance k zadání nejdůležitějším objektem celého programu. Kromě toho, že je jeho programový kód nejobsáhlejším souborem celého projektu, dále odděluje části své funkcionality do dalších objektů či jednotek.

4.8.1 Pomocné funkce pro práci s pakety

Pakety samotné jsou prostými ukazateli na paměťové bloky typu `unsigned char`. Práce s nimi je částečně abstrahována od jejich implementace, aby bylo možno eventuálně aplikaci snadno rozšiřovat a kvůli složitosti kódu.

V souborech `Packets.*` jsou tedy k dispozici následující funkce:

- `CreatePacket` – vytvoří nový paket a nastaví jeho pevně dané položky, tzn. odesílatele, adresáta a koncovou značku.
- `DestroyPacket` – provede úklid a likvidaci paketu (v současnosti není nutný žádný úklid).
- `PacketAddChecksum` – spočítá kontrolní součet paketu a uloží jej do příslušných polí.
- `PacketMakeRequest` – nastaví pole paketu tak, aby byl požadavkem na blok paměti. Vstupní parametry jsou v typech používaných programem, nikoliv binární nebo zkrácené jako v obsahu paketu.
- `PacketMakeSetter` – nastaví pole paketu tak, aby byl požadavkem na uložení dat. Vstupní parametry jsou brány totožně jako u předchozí funkce.

- `PacketMakeEndOfComm` – učiní z paketu požadavek na konec komunikace.
- `PacketMakeKeepAlive` – učiní z paketu požadavek na pokračování komunikace.
- `PacketMakeCalibration` – učiní z paketu požadavek na kalibraci snímače.
- `IsKeepAlivePacket` – vrací pravdu, pokud je zkoumaný paket výzvou ke komunikaci.
- `IsStatusPacket` – vrací pravdu, pokud je zkoumaný paket informací o stavu snímače.

Ačkoliv je délka paketu pevně dána na 15 byte, toto číslo není nikde v kódu implicitně vyžadováno. Namísto toho všechna místa pracující s pakety na nízké úrovni zaručují, že bude manipulace prováděna korektně při jakékoliv délce, ale zároveň že bude při odesílání a příjmu uvažována pouze část prvních 15 byte s definovaným významem.

4.8.2 Fronta pro detekci paketů

Tento objekt nazvaný `TByteQueue` je oddělen do souborů `ByteQueue.*` a realizuje imitaci posuvného registru na úrovni celých byte, potřebného pro detekci paketů.

Při příjmu dat z bezdrátového komunikačního rozhraní dochází k tomu, že v nepřítomnosti signálu vyslaného snímačem je přijímán šum. V tomto šumu se eventuálně nacházejí platné pakety, jejichž platnost lze ověřit správnými hodnotami adresáta, odesílatele, koncové značky a kontrolního součtu.

Bloky dat přijaté kom. rozhraním tedy procházejí, byte po byte, touto frontou a ta pro každý posuv vyhodnocuje zda se jedná o platný paket. Pro urychlení zpracování jsou postupně kontrolována všechna kritéria tak, že v případě dřívějšího vyloučení platnosti již není počítán kontrolní součet.

Rozhraní tohoto objektu je poměrně jednoduché, obsahuje pouhé tři položky:

- funkce `Push` – posune frontu a vloží nový byte,
- funkce `IsValid` – určuje, zda je současný obsah fronty platným paketem,
- vlastnost `PacketBuffer` – ukazatel na současný obsah fronty.

Interní implementace této „fronty“ využívá ve skutečnosti namísto pravé fronty dvou bufferů, mezi nimiž je obsah kopírován s posunutím.

4.8.3 Synchronizační paměť

Vzhledem k tomu, že přenos dat bezdrátově není nikdy zcela spolehlivý, je nutno kontrolovat konzistenci dat. Kontrolní součty obsažené v paketech zaručují s vysokou spolehlivostí, že nebude nikdy přijata špatná informace. Nezaručují však již, že informace přijata skutečně bude. Proto je zapotřebí přijatá data průběžně ukládat a v případě nekompletního příjmu žádat pouze nepřenesené části.

Roli správce přijatých dat a minimalizace požadavků plní objekt typu `TSyncMem`. Obsahuje dva paměťové bloky, představující kopie interní a externí paměti snímače. Každý byte tohoto bloku má kromě samotného obsahu přidruženy ještě další dvě informace – zda je

žádán a zda byl přijat. Byte paměti je tedy platný a použitelný pouze tehdy, když je nastavena i informace o tom, že již byl přijat.

Jelikož má tento objekt k dispozici úplnou „mapu“ pamětí a nakládání s nimi, kromě jejich pouhého ukládání dále plní i funkci jistého řízení požadavků, alespoň na úrovni výběru žádané paměti. Hlavní rozhodování, kdy požadavky odesílat, zajišťuje logika v kódu samotného vlákna. Možné stavy pro každý byte a výsledné zacházení popisuje tabulka na obr. 21:

žádán	uložen	připravenost	požadavek
0	0	ne	ne
0	1	ano	ne
1	0	ne	ano
1	1	ano	ne (!)

Obr. 21: Tabulka možných stavů jednotlivých bytů kopií paměti

Při dotazu na návrh požadavku je nejprve kontrolována kopie interní a poté externí paměti. Toto pořadí je dáno tím, že většina přístupů do externí paměti závisí na zjištění informací z paměti interní a proto by bylo zbytečné kontrolovat bezvýsledně nejprve celou externí paměť.

Samotné vyhledávání v rámci jednotlivých pamětí probíhá tak, že je nejprve nalezen první byte, pro nějž je nutno vydat požadavek, a jeho pozice považována za začátek žádaného bloku. Následně je vzat blok maximální povolené délky a zpětně od jeho konce hledán poslední byte, pro nějž je třeba vydat požadavek. Takto je tedy snížena velikost požadavku na žádoucí velikost. Možné „mezery“ již přijatých dat uvnitř takto nalezeného rozsahu nejsou uvažovány.

Stojí za zmínku, že příjem dat nemění stav jejich žádanosti; Proto lze pouhým vypínáním stavu uložení zahájit opětovné stažení.

Programové rozhraní třídy `TSyncMem` tvoří následující položky:

- funkce `ClearAll`, sloužící k celkovému vynulování obsahu obou pamětí,
- funkce `Wanted`, která nastavuje daný rozsah paměti jako žádaný,
- funkce `Invalidate`, která nastavuje daný rozsah paměti jako nepřijatý,
- funkce `RangeReady`, zjišťující připravenost daného rozsahu paměti,
- funkce `AddPacket`, začleňující obsah datového paketu do pamětí,
- funkce `SuggestRequest`, vracející návrh na požadavek k odeslání,
- funkce `Get2` a `Get3`, vracející již ve formě korektní 32-bitové proměnné 16-ti a 24-bitové hodnoty uložené v paměti,
- vlastnosti `InternalData` a `ExternalData` umožňující přímý přístup ke kopiím paměti jako paměťovým blokům,
- vlastnosti `InternalSize` a `ExternalSize` podávající informaci o velikostech

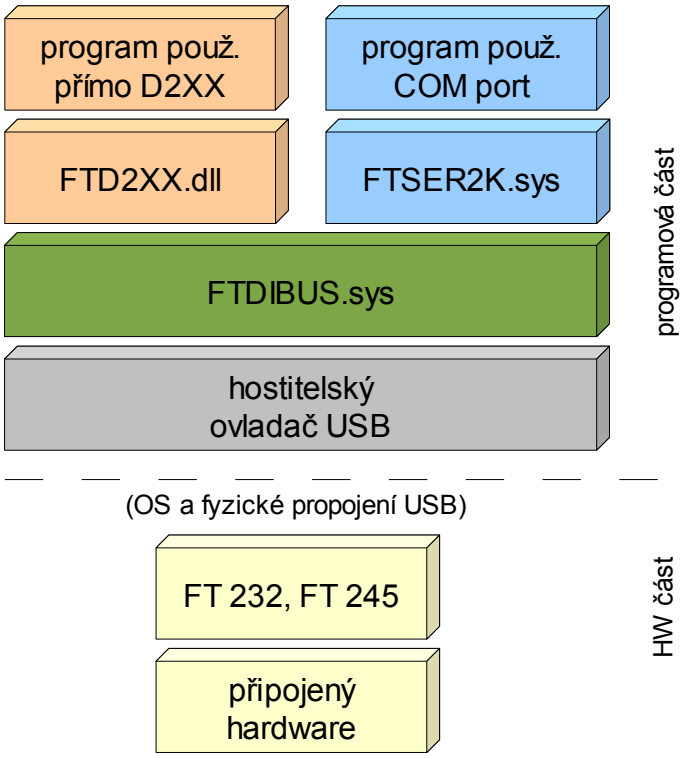
kopií pamětí, potřebnou při používání dvou předchozích vlastností,

- vlastnost `AllReady` shrnující připravenost veškerého obsahu obou pamětí.

Vlastnost `AllReady` je kódem vlákna intenzivně využívána a proto její vnitřní implementace zahrnuje cachování, umožňující v klidovém stavu podstatně rychlejší zpracování dotazů.

4.8.4 Práce s knihovnou FTDI pro USB rozhraní

Knihovna poskytovaná firmou FTDI pro snadnou komunikaci s hardware připojeným k převodníkům FT 232 a FT 245 se skládá z několika souborů DLL a hlavičkového souboru v jazyce C. „Vstupním bodem“ do knihovny je soubor `ftd2xx.dll`, s nímž musí být program staticky či dynamicky sestaven. Funkce dostupné v této knihovně mají předpony `FT_` [11].



Obr. 22: Architektura ovladačů FTDI a vztah součástí systému k ní [11]

Základy práce s jedním „virtuálním portem COM“ (VCP), jak jej původní dokumentace nazývá, se skládají z jeho otevření, čtení, zápisu a uzavření. Tyto činnosti provádějí funkce `FT_Open`, `FT_Read`, `FT_Write` a `FT_Close`. Pro otevírání je však k dispozici i funkce `FT_OpenEx`, která dokáže otevřít port nikoliv podle interního číselného identifikátoru (ten je nutno zjistit sekvencí volání různých funkcí jako např. `FT_ListDevices` ap.), ale přímo podle textového názvu připojeného k čipu převodníku.

Kromě toho lze s VCP pracovat stejně jako s klasickým portem COM, tedy je nutno nastavovat různé příznaky atd. Prvořadým úkolem je nastavení přenosové rychlosti portu funkcí `FT_SetBaudRate`. Pro tento konkrétní případ je užita hodnota 4800 Bd, která vychází z rychlosti mikrokontroléru ve snímači, jenž musí být schopen zpracovat veškerá příchozí data. Dále je po otevření vhodné explicitně nastavit délku slova, počet stop bitů a paritu portu funkcí `FT_SetDataCharacteristics`.

Jelikož je RF modul schopný pracovat pouze v poloduplexním režimu, směr pohybu dat přepínají kontrolní signály portu RTS a DTR, vyvedené z převodníku jako samostatné piny. Tyto příznaky lze zapínat a vypínat funkcemi `FT_Set*` a `FT_Clr*`.

Poslední zajímavá dvojice funkcí obsahuje jednak `FT_GetStatus`, zjišťující množství dat připravených ke čtení z portu a čekajících na odeslání, druhak `FT_Purge`, umožňující smazat z vyrovnávacích front tato dosud „nevyřízená“ data.

Všechny tyto funkce vracejí číselný kód výsledku akce; nejzajímavější hodnotou je `FT_OK` při úspěchu.

4.8.5 Používání vláken v prostředí VCL C++ Builderu

API systému Windows umožňují vytvářet vlákna z vlastních funkcí pomocí funkce `BeginThread`. Sem předaná funkce je opakovaně volána, dokud sama nezavolá funkci `EndThread`. Tento přístup je poměrně nízkoúrovňový a vyhovuje zajisté psaní programů v jazycích jako C nebo strojový kód.

Programové prostředí C+ Builderu však obaluje chování systému do objektů s cílem zjednodušit kód a umožnit programátorovi vyhnout se složité práci s API, které často vyžaduje pro „jednoduché“ operace několik parametrů.

Součástí knihovny VCL, kterou C++ Builder obsahuje, je i základní objekt pro vytváření vláken, nazvaný prostě `TThread`. Obsahuje již kompletní obsloužení základních akcí nutných pro vytvoření, běh a zrušení vlákna. Programátor od této třídy může odvodit vlastní třídu, která zdědí veškeré tyto výhody a přitom bude v podstatě „běžným“ objektem z hlediska tvorby kódu [15].

Nejcennější aspekty představuje především to, že je vlákno programátorovi předkládáno jako standardní objekt s lineárním kódem vytváření (konstruktor), likvidace (destruktor) a spuštění vlastního činného kódu. Je tím eliminován značný rozdíl mezi lineárním vykonáváním kódu, které je vnímáno jako běžné, a cyklickým, které vnucuje přímé použití API. Další výhodou je možnost přiřadit vláknu množství různých lokálních proměnných a funkcí jakožto soukromých (private) položek jeho objektu. Tyto položky by jinak musely být pracně uloženy externě, s obtížným přístupem k nim.

4.8.6 Přehled proměnných komunikačního vlákna

Pracovní / komunikační vlákno vlastní řadu soukromých proměnných, využívaných při jeho činnosti všemi jeho funkcemi. Uchovávají především informace o jakémsi „rozloženém stavu“ komunikace. Podle zvyklostí programování s produkty firmy Borland mají tyto soukromé proměnné vždy v názvu za předponu písmeno „`f`“ (odvozeno od „field“ – položka).

- `fWantPatient`, `fWantData`, `fSendPatient`, `fResetCounter`, `fCalibrate`, `fDisconnect` – booleovské proměnné určující požadavky přijaté vláknem od koordinačního objektu.
- `fPatient` – lokální kopie nastavení.
- `fLastActivity`, `fLastDataActivity` – proměnné typu `TDateTime`, používané pro správné časování požadavků.
- `fByteQueue` – podřízený objekt fronty pro detekci paketů.
- `fMemSyncer` – podřízený objekt synchronizační paměti.
- `fPacketStorage` – objekt typu fronta pro dočasné ukládání přijatých paketů před zpracováním. Zde je jako v jediném místě programu použita standardní knihovna STL; příslušná deklarace je zde `std::deque<unsigned char*>`.
- `fConnection`, `fResult` – proměnné pro práci s portem.
- `fStatusTarget`, `fMessage` – proměnné pro práci se zprávami Windows.
- `fStatus` – významná proměnná, popisuje celkový stav komunikace – port otevřen / neotevřen, zařízení nalezeno atd. Může nabývat čtyř hodnot, viz ukázku č. 3.

```
enum thread_status {ts_no_antenna = 0,
                    ts_scanning,
                    ts_connected,
                    ts_shutdown};
```

Ukázka 3: Deklarace datového typu pro nastavení

4.8.7 Přehled funkcí komunikačního vlákna

Kód celého vlákna je pro přehlednost rozdělen do množství funkcí (celkem 24), které lze zhruba rozdělit do dvou skupin – nízko a vysokoúrovňové. Toto dělení nelze uplatnit zcela důsledně, neboť část funkcí původně zde umístěných je oddělena do jednotky pro práci s pakety nebo samostatných objektů – avšak pro hrubé rozdělení postačuje. Druhou možnost klasifikace představuje dělení dle účelu funkcí. Tyto dva přístupy však logicky neposkytují zcela ortogonální pohled, neboť například funkce pro práci s portem se pochopitelně nacházejí na nižší úrovni než obecné zpracování požadavků. Faktem však je, že nakonec všechny funkce tvoří poněkud amorfní masu.

První skupinu tvoří funkce zabývající se portem.

- `CheckConnection` – pomocí `FT_GetStatus` se snaží zjistit, zda je port otevřen; premisou je, že při neotevřeném portu volání selže a nebude navrácen kód úspěšného vykonání. Pokud k tomuto dojde, nastaví `fStatus` na `ts_no_antenna`.

- `CloseConnection` – typicky poslední práce s portem před ukončením programu. Provede úklid portu, tedy smazání nedokončených přenosů, a zavře jej. Chyby nebere v úvahu.
- `TryOpenAntenna` – pokusí se otevřít a konfigurovat port. Podle výsledku nastavuje `fStatus` buď na `ts_scanning` nebo `ts_no_antenna` a výsledek také dále zprávou sděluje koordinátoru.
- `SwitchToReceiving`, `SwitchToSending` – přepínají stavové signály portu a řídí tak směr RF modulu. Kromě toho sdělují koordinátoru zprávou nový směr toku dat.
- `WaitForSending` – při odesílání čeká, dokud nejsou všechny data odeslána, pomocí smyčky s voláním `FT_GetStatus`.
- `SendPacketLow` – zapíše do portu daný paket. Komunikační rozhraní již musí být předem přepnuto do režimu vysílání.

Druhá skupina sestává z funkcí komponujících pakety a odesílající je, přičemž směr toku dat již musí být přepnut na odesílání.

- `SendCommEnd` – vyšle příkaz ukončení komunikace.
- `SendData` – odešle blok dat na určenou pozici.
- `SendKeepAlive` – odešle příkaz k pokračování komunikace.

Třetí skupinu tvoří funkce obalující sekvenci volání funkcí z předchozí skupiny spolu s prací s pakety a odesílající příkazy snímači. Typicky se jedná o vytvoření jednoho či více paketů, jejich kompozice do podoby zamýšleného příkazu, přepnutí kom. rozhraní do režimu odesílání, odeslání paketů, vyčkání na odeslání a přepnutí zpět na příjem.

- `SendRequest` – odešle požadavek na blok paměti.
- `SendCounterReset` – odešle na příslušné adresy hodnoty způsobující restart zápisu měřených hodnot.
- `SendCommEnd` – vyšle příkaz ukončení komunikace.
- `SendPatientToDevice` – odešle do snímače nastavení.
- `DoCalibration` – odešle snímači požadavek na kalibraci.

Čtvrtou a zároveň poslední skupinu představují funkce řídící chod komunikace a obsluhující konverze dat v rámci vlákna, tedy typicky mezi třemi možnými reprezentacemi: paketovou, paměťovou (v kopiích pamětí) a strukturovanou.

- `TryReceiveData` – přečte z portu dostupný blok dat, filtruje skrze detekční frontu `fByteQueue` a nalezené pakety ukládá pro další zpracování do fronty `fPacketStorage`.
- `CanSend` – zjistí ze stavu komunikace `fStatus` a časových údajů `fLastActivity` a `fLastDataActivity`, zda je možno v danou chvíli odesílat pakety či je třeba vyčkat.
- `EmptyQueue` – vyprázdní paketovou frontu `fPacketStorage`, například při odpojení.
- `DispatchPackets` – odebírá pakety z fronty `fPacketStorage` a dekóduje na místě spolu s předáním informace v nich obsažené koordinátoru, nebo v případě paketů s příchozími daty tyto předává do synchronizační paměti `fMemSyncer`.
- `PatientReceived` – při dokončeném příjmu nastavení jej konvertuje ze

synchronizační paměti, ukládá do `fPatient` a odesílá koordinátoru.

- `DataReceived` – totéž pro zaznamenaná data měření, avšak bez lokálního uložení.
- `StepRequests` – při aktivních požadavcích na nastavení či data zkoumá obsah synchronizační paměti a podle potřeby buď upravuje žádané rozsahy nebo kompletní data nepřímo odesílá pomocí volání `PatientReceived` a `DataReceived`.
- `NotifyPacketCounters` – odesílá koordinátoru informace o počtu přijatých a odeslaných paketů.
- `ProcessMessages` – provádí příjem zpráv od koordinátoru a převádí je na stav proměnných požadavků.

4.8.8 Hlavní smyčka komunikačního vlákna

Název hlavní smyčka je poněkud zavádějící, neboť samotná smyčka je ještě obklopena na začátku i konci drobnými částmi kódu prováděnými při startu a ukončení činnosti, které nemohou být součástí konstruktoru a destruktoru. Jde o vytvoření fronty zpráv pro vlákno provedené prvním voláním `PeekMessage` a o uzavření portu pomocí `CloseConnection`.

Samotná smyčka je typu `while`, prováděná do okamžiku signalizace vhodného ukončení vlákna. To je zajištěno čtením vlastností `Terminated`, kterou nastavuje externí zavolání funkce `Terminate`, kterýžto mechanismus je součástí bázového objektu vláken ve VCL.

Při začátku každé iterace jsou nejprve přijaty příkazy od koordinátoru a zkontrolován stav portu. Poté je při stavu připojení (`fStatus == ts_connected`) zjištěno časování komunikace a provedena kontrola vypršení času pro aktivitu. Nato již je definitivně stav znám a dochází k odeslání zprávy o něm koordinátoru.

Následuje rozlišení podle stavu. Pokud není port otevřen, dojde k pokusu o jeho otevření. V ostatních případech jsou přijata data z portu, zpracovány pakety a aktualizován stav požadavků. Pokud byla přijata výzva a její pořadové číslo je vyšší než 8, dojde k odeslání příkazu k udržování spojení. Pokud ne a zároveň je možno odesílat ostatní druhy požadavků, je zkontrolováno, zda jsou aktivní některé požadavky a případně odeslán ten s nejvyšší prioritou. Pořadí priorit je: odpojení, získání bloku paměti, kalibrace, reset záznamu, odeslání nastavení.

Před koncem iterace smyčky je odeslán stav počitadel paketů. Posledním krokem je uspání běhu vlákna; při připojeném snímači je žádoucí rychlá odezva, proto činí prodleva pouze 10 ms; při odpojeném snímači je to 100 ms.

4.9 MDI a společné vlastnosti formulářů

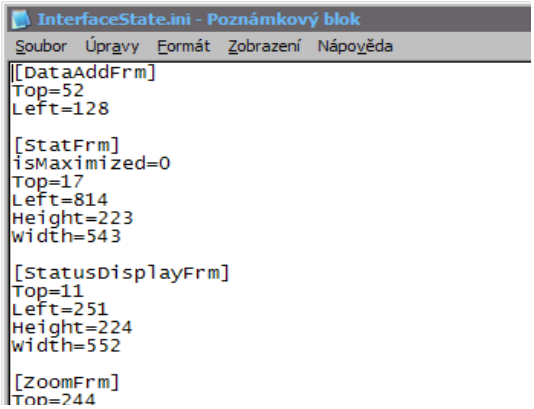
Všechny formuláře si „pamatují“ poslední stav, v němž se nacházely a tak umožňují přizpůsobení rozložení aplikace podle uživatelových potřeb. Informace o pozici, velikosti a maximalizaci formulářů je ukládána do INI souboru v adresáři se spustitelným souborem

programu. Název tohoto souboru je pevnou součástí programu, určenou při kompilaci a sestavení; v tuto chvíli jím je `InterfaceState.ini`.

Veškeré související činnosti obstarává obsah souboru `FormDataHandler.cpp`, z jehož rozhraní jsou vidět funkce `SaveForm` a `LoadForm`, beroucí za parametr referenci formuláře. Ty díky bohatým vlastnostem knihovny grafického rozhraní VCL dokáží zjistit interní název formuláře a použít jej při určení sekcí v souboru pro ukládání a načítání. Všechny formuláře tak mohou volat při vytvoření tentýž příkaz `LoadForm(this)` pro obnovení svého předchozího stavu, stejně jako při likvidaci `SaveForm(this)`.

Tato dvojice funkcí může být řízena specifikací druhého parametru, obsahujícího bitové pole popisující položky k ukládání a načítání. Lze aktivovat zpracování velikosti a stavu maximalizace. Výchozí hodnotou je vše.

Formát tohoto souboru je prostý – každá tzv. sekce je nazvána podle interního názvu příslušného formuláře a obsahuje několik položek nazvaných totožně s jejich názvy v kódu programu (de facto jejich názvy ve VCL).



Obr. 23: Struktura informací uložených v konfiguračním souboru

S pochopitelnou výjimkou hlavního formuláře, který je rodičovským prvkem MDI, a dále grafu, jenž je speciálním, mají některé dětské formuláře zakázánu změnu velikosti a skryta systémová tlačítka vyjma zavření. Důvod je prostý – pokud se na nich nachází pouze několik prvků neměnné velikosti, změna geometrie by neměla žádný smysl.

Dále je zde jistá zvláštnost implementace MDI v C++ Builderu. Dětská okna nelze skrýt, a při pokusu o zavření jsou místo toho minimalizována. Tak jako tak, toto chování není zcela nevýhodné, protože umožňuje radikálně odstranit většinu systémových tlačítek a ponechat pouze jediné, s jedinou možnou funkcí. Nemožnost skrývat formuláře je v jistém bizarním smyslu také výhodou. Jelikož má dynamická tvorba formulářů ve spojení s MDI množství zádrhelů, je jednodušší ponechat je v existenci po celou dobu a nechat je v souladu s možnými akcemi pouze minimalizovat.

Okno grafu je ústředním prvkem celého programu, a tak je umožněna změna jeho velikosti, včetně maximalizace, zatímco „zavření“ (minimalizace) nikoliv.

4.10 Konfigurace při překladu programu

Soubor `appsharedconf.h` obsahuje množství definic maker, prostupujících celým programem. Jsou zde definovány textové identifikátory bezdrátového komunikačního rozhraní a souboru pro ukládání stavu rozhraní, konstantní části paketů, velikosti synchronizačních pamětí a konečně také čísla všech zpráv vyměňovaných mezi koordinátorem a komunikačním vláknem.

4.11 Hlavní formulář

Hlavní formulář je z hlediska vlastnictví nejvýše postaveným objektem celé aplikace. Při svém vytvoření a likvidaci zajišťuje vznik a likvidaci koordinačního objektu, který dále takto zajišťuje ostatní části. Tímto „vlastnickým řetězcem“ je zajištěno, že ostatní části programu budou vznikat a zanikat společně s hlavním formulářem bez další námahy. Ukazatel na koordinátor je zároveň k dispozici pro ostatní části uživatelského rozhraní, aby mohly manipulovat s daty podle své vlastní potřeby.

Pro vazbu s koordinátorem definuje formulář řadu funkcí, volaných jako události koordinátorem. Podrobný výčet lze snadno odhadnout i z dřívějšího textu a byl by pouze opakováním.

4.12 Časová osa

Vykreslování grafu se děje na plátně komponenty `TPaintBox`, která je pro tento účel ideální. Namísto kreslení na bitmapu a teprve poté kopírování na obrazovku, zde dochází k vykreslování přímo na obrazovku. Tím je uspořena trocha času, která při překreslování tak velké plochy umožňuje rychlejší činnost a tím i lepší odezvu.

Základem celého vykreslování je rozměření výšky plátna. Kromě vertikálních okrajů je nutno odpočíst prostor pro textové popisky; takto získaná velikost je poté rozdělena na 12 dílků – základních jednotek. Konkrétně jde o dělení rozsahu datové stopy (0° až 180°) na dílky po 15° (180 / 15 = 12). Jakmile je známa tato základní jednotka výšky, lze postoupit dále k samotnému vykreslování.

Při vykreslování je nejprve smazána celá plocha barvou pozadí. Poté je zjištěna hodnota povoleného náklonu a plátno přebarveno do odpovídající výše barvou pozadí pro povolený rozsah. Pozadí je černá, posunutá lehce k červené či kaštanové, povolený rozsah naopak o něco světlejší do zelena. Následně je zjištěno, zda je co vykreslovat. Pokud ne, zde činnost končí. Pokud ano, pokračuje kód dále.

Dotazem na data je zjištěn první viditelný vzorek, načež je couvnuto o jeden zpět. Důvodem je vytvoření zdání pokračování grafu – vykreslené vzorky vyplňují celý interval až do okamžiku dalšího vzorku. Pokud by byl předchozí vynechán, na začátku grafu by zůstalo tmavé místo a při posunu zpět náhle zaplněno, jakmile by se tento vzorek dostal do zobrazovaného časového rozsahu na obrazovku.

Vzorek je v grafu reprezentován vyplněným obdélníkem příslušné výšky, dosahujícím až k dalšímu vzorku. Tato reprezentace vzorků typu „sample & hold“ je nejlogičtějším řešením, protože její účinnost neklesá při nevhodném měřítku – pokud by byly použity čáry, může při příliš blízkém pohledu dojít k „rozředění“ grafu, který by obsahoval několik málo čar ve větších vzdálenostech.

Vzorky jsou tedy vykreslovány od před-prvního viditelného až po poslední viditelný. Zde není třeba pokračování o jeden dále, neboť tento poslední viditelný vzorek pokračuje až k dalšímu, již neviditelnému. Speciální případ na konci naopak představuje poslední vzorek celého záznamu, který má de facto nulovou délku, neboť za ním nenásleduje další. Pro tento případ je použita délka odpovídající nastavenému intervalu ukládání dat.

V tuto chvíli jsou vykreslena data, avšak schází mřížka grafu a popisky. V závislosti na výšce je rozhodnuto, zda vykreslit pouze vodící čáry mezí a úroveň 90°, či také násobky 45 stupňů, případně pro velmi velké výškové rozměry po 15 stupních. Jednotlivé čáry jsou vykresleny podle důležitosti tenčí čí tlustší čarou, násobky 15 tečkovaně.

Poté je provedena konverze času (viz dříve) do počáteční polohy mřížky osy x (časové), stejně jako u dat, i zde pro začátek před zobrazeným rozsahem. Následně dochází k periodickému zvyšování časového počitadla a přepočtu na hybridní čas, s vykreslováním mřížky osy y (datové) a popisků. Takto je tedy vykreslena celá mřížka a kreslení dokončeno.

Zvláštní roli hraje při vykreslování kurzor; je vykreslen ještě před daty, aby je nezakrýval, vždy s konstantní šířkou 4 px, a jeho pozice odpovídá nejbližšímu předchozímu vzorku. Nad ním v horním okraji grafu je zobrazena hodnota náklonu číselně. Vzorek je zároveň zvýrazněn použitím další odlišné barvy.

Při pohybu myši nad grafem je průběžně aktualizována pozice kurzoru, překreslován graf a zjištěný úhel také zvlášť zobrazován na popisku mimo graf kvůli čitelnosti.

Pouhé vykreslování dat je překvapivě nenáročné na procesorový čas a funguje bez jakýchkoliv obtíží i při miliónu vzorků. Hlavní spotřebu času vykazuje vykreslování popisků, tedy rendering textu. Blikání obrazovky proto musí zabraňovat aktivace double-bufferingu pro oblast formuláře s grafem.

4.13 Formát souboru na disku

Pokud jsou data uložena na disk, využívají vlastní binární formát, odvozený od formátu používaného C++ Builderem pro ukládání formulářů. Důvodem je nadmíru snadné použití – pro tento formát je k dispozici třída obalující operace načítání a ukládání,

odstraňující veškeré problémy s ukládáním délek řetězců, nízkoúrovňovým kopírováním paměti atd. Po otevření souboru je k dispozici triviální rozhraní typu „zapiš xyz / načti xyz“. Rozložení dat viz obr. 11.

	položka	typ	jednotka	poznámka
hlavička	maximální náklon	int	stupně	ve stupních
	jméno pacienta	AnsiString		lib. délka
	rok měsíc den hodina minuta sekunda	int int int int int int	dle názvu položky	čas operace
	rok měsíc den hodina minuta sekunda	int int int int int int	dle názvu položky	čas začátku měření
	interval záznamu mrtvá doba alarm minimálně	int int int	ms ms ms	časování
	počet vzorků	int		nezávazný
data	čas	int	sekundy	poč. od startu
	náklon	float	stupně	

Obr. 24: Organizace dat uložených na disku

Počet vzorků je nezávazný – při načítání se jím program neřídí v žádné kritické části, pouze tuto informaci využije při prealokaci paměti pro urychlení přidávání dat. Pokud tato hodnota neodpovídá skutečnosti, k pádu programu ani chybovému hlášení nedojde; vyžadované místo je přidáno podle potřeby v průběhu čtení.

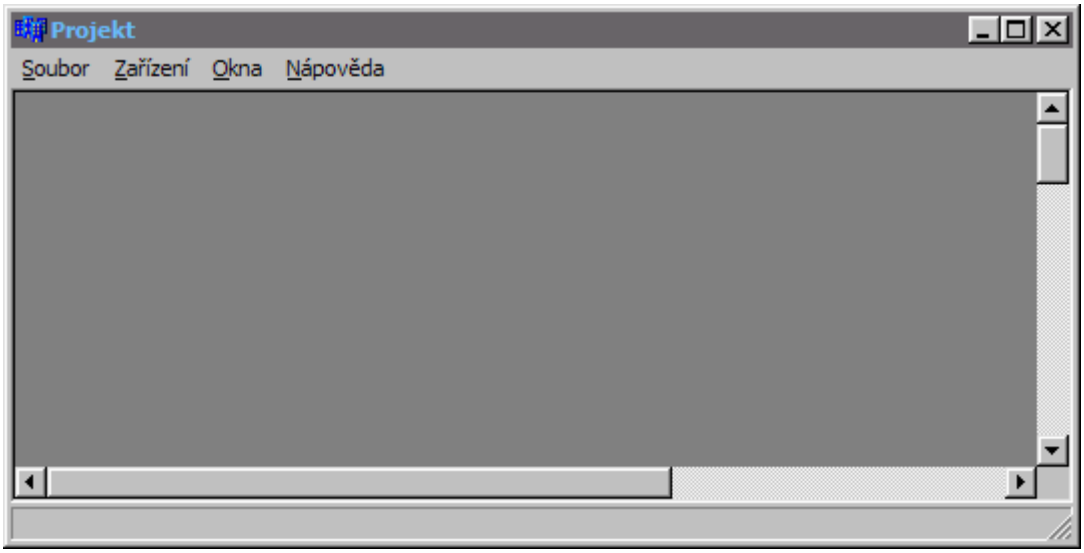
Je pravděpodobně vhodné ještě jednou zdůraznit, že samotná struktura souborových binárních dat sice odpovídá zde zobrazenému rozložení, ale navíc obsahuje značky používané třídami `TReader` a `TWriter` [9] pro kontrolu typů a integrity dat.

5. Popis uživatelského rozhraní aplikace a jeho použití

Tato část textu je víceméně uživatelským manuálem. Popisuje možnosti a prvky, které má k dispozici uživatel aplikace, aniž by podrobněji zkoumala plány v celé šíři či vnitřní příčiny omezení.

5.1 Hlavní formulář

Základním či spíše rámcovým prvkem uživatelského rozhraní je hlavní formulář (obr. 25). Je rodičovským elementem MDI a stává se tak lokální pracovní plochou pro všechny ostatní formuláře (neustále zobrazeny). Kromě toho obsahuje hlavní menu aplikace (obr. 26) zahrnující jednak příkazy pro načítání a ukládání souborů, druhak správu podřízených oken.



Obr. 25: Hlavní formulář aplikace

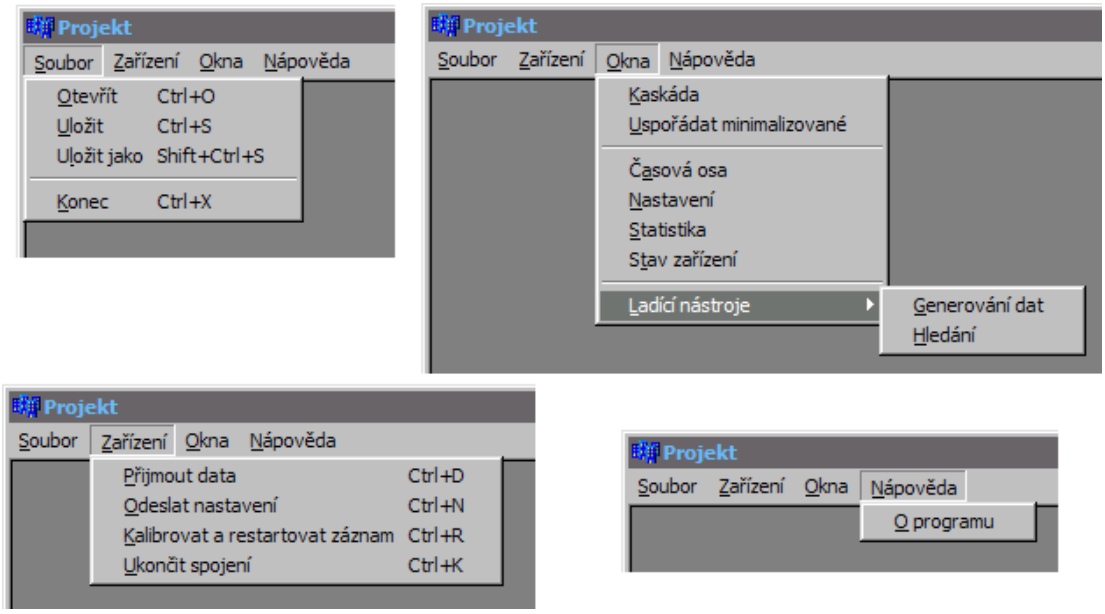
Nabídka soubor umožňuje uložit zobrazovanou datovou sadu (v případě změn metadat), otevřít již uloženou a zcela ukončit aplikaci.

Nabídka zařízení umožňuje ovládat připojený snímač, tedy přijímat z něj data a vydávat mu další příkazy. Tato nabídka je aktivní pouze tehdy, je-li skutečně komunikující snímač připojen. Volba „kalibrovat a restartovat záznam“ je pro popis pravděpodobně nejzajímavější částí, neboť kromě činností ve svém názvu také nastavuje čas začátku měření podle hodin počítače.

Nabídka okna poskytuje po řadě přístup k možnostem MDI pro automatické uspořádání oken (první tři volby): Dlaždice rozmístí okna do pravidelné mřížky v rodičovském okně, Kaskáda uspořádá všechna okna levým horním rohem na diagonálu

vycházející z levého horního rohu rodičovského okna a Uspořádat minimalizované seskupí minimalizovaná dětská okna na spodní stranu rodičovského okna.

Dále je zde série čtyř voleb přenášejících do popředí korespondující dětská okna. Konečně podmenu „ladící nástroje“ umožňuje zobrazení podoken dovolujících v tuto chvíli pouze generování a zkoumání dat. Toto podmenu je zobrazeno pouze v případě že je aplikace spuštěna s parametrem `-debug`.



Obr. 26: Položky hlavního menu aplikace

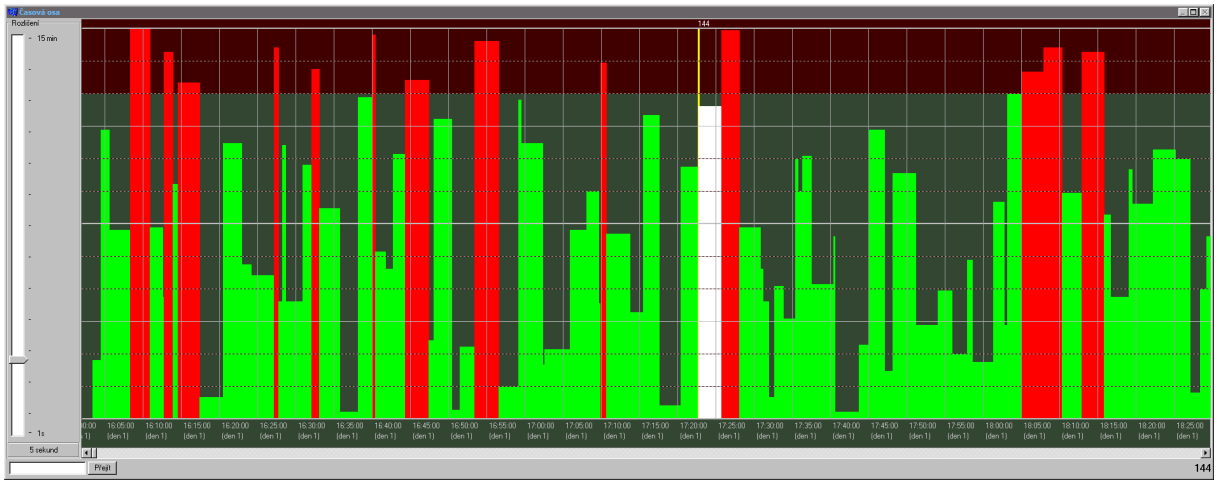
5.2 Časová osa

Formulář časové osy je nejdůležitějším prvkem celé aplikace. Obsahuje především graf a posuvník pro pohyb v čase pro prezentaci dat v přímé formě a podává tak zřejmě nejvíce informací ze všech formulářů.

Data jsou zobrazena ve formě sloupců o délce odpovídající intervalu mezi měřením současné a další hodnoty. Zelená barva signalizuje hodnotu v povoleném rozsahu, červená mimo něj. Povolený pás je mimo to vyznačen pozadím lehce odlišné (zelenavé) barvy. V ose x (časové) jsou vyznačeny pravidelné intervaly vhodné délky, odpovídající měřítku zobrazených dat. Přesný čas začátku jednotlivých intervalů ukazují popisky ve spodní části grafu. V ose ypsilon, popisující náklon, jsou vždy přítomny vodící čáry popisující pozice mezních hodnot (0° a 180°) a středu (90°). Podle rozměrů celého okna jsou dále zobrazeny čáry mřížky pro ±45°, případně pro velmi velké rozměry i všech celých násobků patnácti stupňů.

Stěžejním prvkem umožňujícím posun v čase je posuvník; pro plné využití aplikace je vhodné znát veškeré možnosti, jenž takovýto prvek nabízí. Pro účely tohoto použití jako

hlavní ovládací prvek mu bylo umožněno přijímat fokus, takže po zvýraznění táhla lze místo myši používat i klávesnici. Šipkové klávesy umožňují posun grafu po jednotce času (sekundě), klávesy Page Up / Page Down posouvají o celou délku zobrazeného grafu, Home a End přesouvají okamžitě na začátek a konec celé časové osy.



Obr. 27: Okno časové osy

Při přejíždění grafu myší se objevuje kurzor – žlutá svislá čára, zvýrazňující pozici dat pod kurzorem. Pozice tohoto kurzoru zůstává uchována při změně měřítka zobrazení či posunu grafu, dokud není opět najeto myší nad graf. Zároveň je zvýrazněn jinou barvou i samotný datový bod (viz obr. 27).

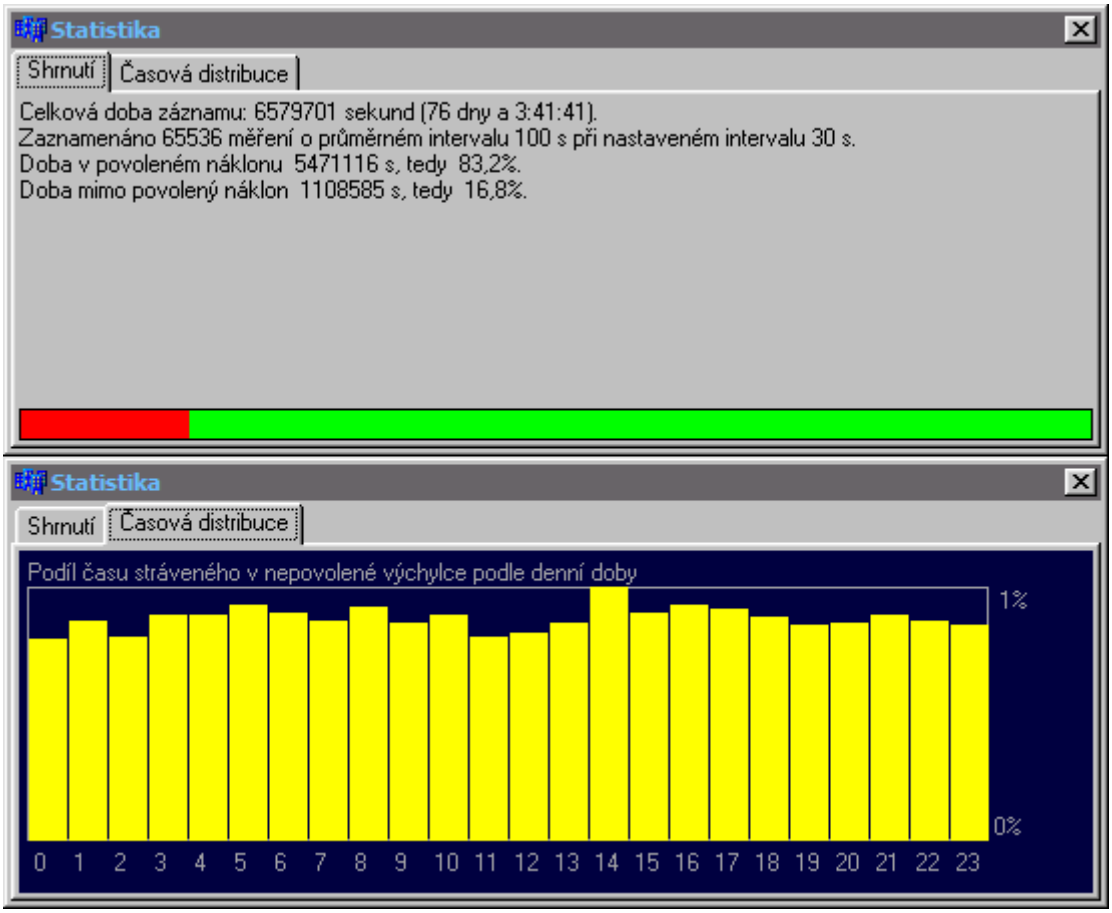
Dále je součástí formuláře vlevo umístěné táhlo ovládající rozlišení grafu, a to v rozmezí 1 sekunda až 15 minut na pixel, pokrývajícím takto široký rozsah, od nejmenších teoreticky zachytitelných detailů (záškuby?) při přesném zobrazení všech sekund zvlášť, až po zobrazení více než celého týdne. Kvůli tomuto velkému rozsahu není poloha táhla přepočítána na rozlišení lineárně, ale logaritmicky, což lépe odpovídá potřebám drobných úprav ve všech oblastech měřítka. Při změně rozlišení je zachován začátek pohledu na graf – první viditelný datový bod zůstane prvním.

Poslední částí formuláře je spodní lišta s prvky pro rychlý pohyb v čase. Zde se nachází textové pole pro vstup času a tlačítko pro přesun na zadanou pozici. Zcela vpravo je číselný popisek duplikující hodnotu na vrcholu kurzoru.

5.3 Statistika

Tento formulář obsahuje statistické zpracování naměřených dat. Ta jsou dvojího druhu: jednak shrnutí podstatných údajů o celém běhu měření, druhak distribuce nepovolené výchylky v závislosti na denní době. Proto je obsah formuláře rozdělen na dvě samostatné záložky dle obr. 28.

První záložka obsahuje shrnutí údajů o měření. To zahrnuje počet datových bodů, celkovou dobu měření a především dobu strávenou v povoleném a nepovoleném náklonu, a to v absolutní i relativní míře. Podíl těchto dvou na celkovém čase lze rychle zjistit pomocí barevného proužku, který znázorňuje poměr těchto dvou absolutních časů. Jinak řečeno, pro jeho dvě různě zbarvené části platí, že podíl jejich jejich ploch odpovídá podílu celkových délek bloků příslušné barvy v celém grafu.

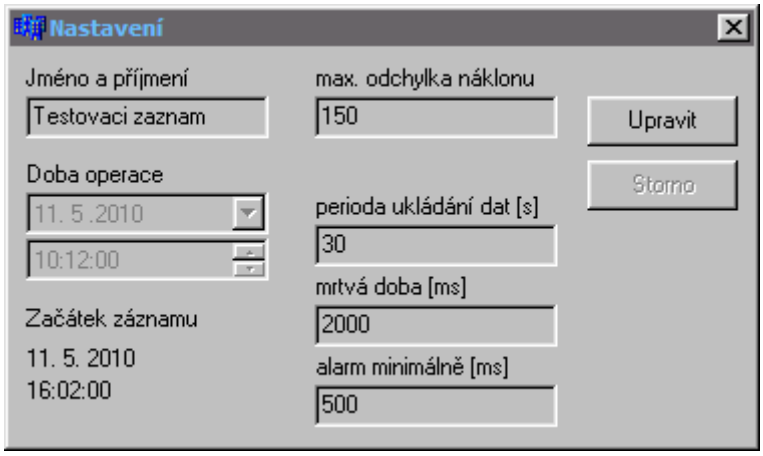


Obr. 28: Formulář se statistickými údaji, obě záložky

Druhá záložka obsahuje histogram doby nepovolené výchylky, roztříděné podle hodiny v níž byly tyto výchylky zjištěny. Pomocí kontextového menu lze graf přepínat mezi zobrazením podílu založeného na srovnání s celou dobou měření nebo s daty měřenými pouze v dané denní hodině.

Dále je v kontextovém menu i možnost kopírovat zobrazený graf do schránky.

5.4 Nastavení



Obr. 29: Formulář pro úpravu nastavení

Zde je možno upravovat nastavení – jde o tři skupiny údajů: o pacientovi (jméno a doba operace), o datech (pouze mezní náklon) a o časování – nastavení rychlosti záznamu a přechodů mezi alarmem a klidem. Informace o době začátku záznamu úmyslně není pro uživatele zpřístupněna k úpravám.

Vzhledem k zobrazení dat v textových polích, do jisté míry „vyzývavých“ co se týče nechtěných změn, pracuje tento formulář ve dvou režimech – zobrazení a úprav. Po většinu času se nachází právě v režimu zobrazení: textová pole jsou uzamčena pro úpravy a indikují tento stav barvou pozadí všeobecně srozumitelnou pro neaktivní prvek tohoto druhu (šedou). Při stisku tlačítka upravit dojde k přechodu do režimu úprav a barva pozadí se změní na běžnou (bílou). Nyní lze tato data změnit a úpravy buď uložit opětovným stiskem tlačítka, jehož popisek se změnil, nebo zrušit tlačítkem storno.

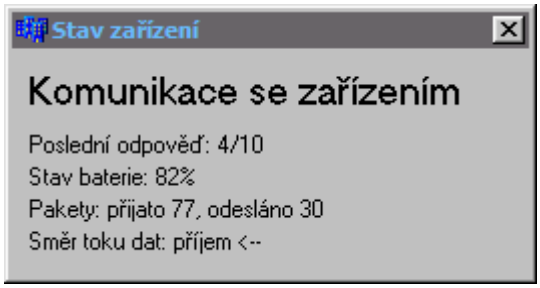
Data nacházející se v prvcích uživatelského rozhraní jsou pouze místní kopií, skutečná verze se nachází v datovém kontejneru.

5.5 Stav zařízení

Tento formulář je zcela pasivní. Podává informace o komunikaci se snímačem. Především je velkým textem udán současný stav. Další informace zahrnují pořadové číslo poslední výzvy vyslané snímačem, stav baterie zjištěný ze stavového paketu a ukazatel směru toku dat, tedy přepnutí směru portu. Poslední údaj je počet odeslaných a přijatých paketů.

Možné zobrazované stavy zahrnují „Komunikační rozhraní odpojeno“, „Čekám na příchozí spojení“ a „Komunikace se zařízením“.

Při odpojení snímače a konci komunikace jsou všechny informace zde zobrazené opět smazány.



Obr. 30: Formulář informující o stavu zařízení

5.6 Ladící formuláře

Existují dva ladící formuláře, dostupné při spuštění aplikace s parametrem `-debug`. Prvním z nich je formulář pro podrobnou analýzu dat (obr. 31).

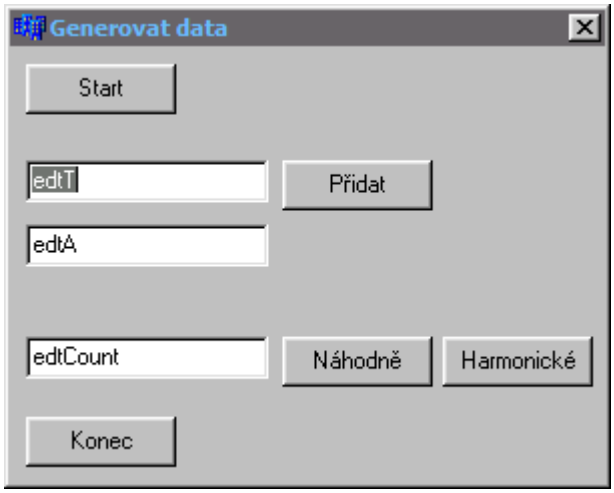
Obr. 31: Formulář pro hledání

Pomocí tohoto formuláře se lze okamžitě přenést k jakémukoliv bodu v čase, spadající do rozmezí pokrytého daty. Pokud je vyžádána pozice mimo něj, buď je odpovězeno posunem na konec grafu či žádnou reakcí. Pokud je vyžádaný časový bod platný, je nalezen příslušný datový bod (začínající v tomto okamžiku nebo dříve) a graf posunut tak, aby byl tento datový bod zobrazen na začátku grafu.

Kromě toho je výsledkem kompletní informace o datovém bodu včetně jeho indexu, času a hodnoty náklonu.

Druhým ladícím formulářem je formulář pro poloautomatické či ruční generování dat (obr. 32). Jeho uspořádání vychází částečně z kroků v kódu při načítání dat: První tlačítko přepíná datový kontejner do režimu načítání a poslední jej naopak ukončuje. Datové body je možno přidávat buď jednotlivě (první dvě textová pole s tlačítkem), nebo vygenerovat

automaticky ve větším množství jako náhodné hodnoty či harmonický průběh ve volitelném množství.



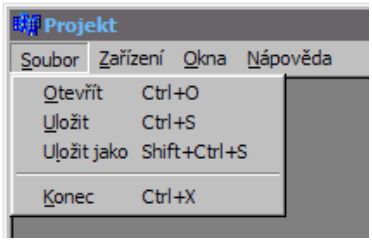
Obr. 32: Formulář pro přidávání dat

6. Stručný uživatelský manuál aplikace

Minimální požadavky pro používání aplikace a ostatních částí systému jsou: počítač s operačním systémem Windows verze 2000 (nebo novější) a volným USB portem.

Při využívání aplikace je prvním krokem instalace ovladačů fy. FTDI, aby byl hardware komunikačního rozhraní vůbec počítačem rozeznán. Ty lze získat na jejích stránkách, konkrétně http://www.ftdichip.com/Drivers/D2XX.htm. Současná verze (2010) je zabalena v archivu formátu ZIP; instalace proběhne kliknutím druhým tlačítkem myši na soubory INF obsažené v archivu a volbě „instalovat“. Dalším krokem je pochopitelně připojení kom. rozhraní. V tuto chvíli je již počítač připraven na veškerou další činnost. Pokud aplikace selhává z důvodu nenalezení DLL souborů s názvy začínajícími „ft“, je zapotřebí je získat, nejlépe z již staženého archivu s ovladači.

Po spuštění aplikace jsou okna grafu, statistiky a nastavení prázdná, což odráží skutečnost, že nejsou načtena žádná data. Data lze načítat ze souborů a opět do nich ukládat – viz obsah hlavního menu pod položkou soubor (obr. 33).



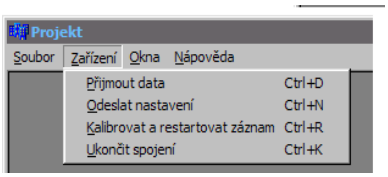
Obr. 33: Obsah menu „Soubor“.

Při přítomnosti dat lze s grafem posouvat, měnit měřítko atd. – viz podkapitola 5.2 „Časová osa“. Takto lze prohlížet vývoj náklonu v čase. Rovněž okno statistik se při načtení naplní údaji.

Okno se stavem zařízení popisuje současnou situaci ohledně komunikace. nejdůležitějším ukazatelem je velký popisný text. V případě, že z nějakého důvodu nelze pracovat s komunikačním rozhraním, je zde zobrazeno „Komunikační rozhraní odpojeno“. Možné příčiny tohoto jsou: nepřítomnost kom. rozhraní, vadný či nepřipojený kabel, otevření portu jinou aplikací. Poslední případ se vztahuje i na násilná ukončení samotné aplikace, která port neuzavře, a následné znovuspuštění. V takových případech je vhodné kom. rozhraní odpojit fyzicky a znovu připojit.

Pokud je kom. rozhraní použitelné, text se změní na „čekám na příchozí spojení“. Aplikace je v tuto chvíli plně funkční, skenuje rádiový provoz a čeká na připojení snímače. To také signalizuje červená kontrolka na kom. rozhraní, ukazující že probíhá čtení. Snímač lze připojit delším stlačením jeho kontaktu (cca 5 sekund), přičemž se i na něm rozsvítí kontrolka. Přitom začíná snímač vysílat výzvy k odpovědi. Jakmile je první výzva aplikací

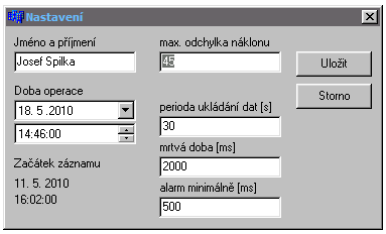
zachycena, nastane změna textu stavu zařízení na „Komunikace se zařízením“. Počitadla výzev a paketů by se měla periodicky měnit. Rovněž se zde po chvíli objeví stav baterie.



Obr. 34: Položky menu „Zařízení“

Komunikace se snímačem umožní aktivaci položek menu pod „Zařízení“. Pokud je cílem získat data snímačem naměřená, volba „Přijmout data“ vydá příslušné požadavky a pokusí se data ze snímače stáhnout. Postupně jsou přijata data nastavení a měření, což se projeví změnou obsahu formuláře „Nastavení“. Přijatá data jsou zaručeně totožná s obsahem ve snímači. Ihned po přijetí je vhodné data uložit!

Pokud je úmyslem obsluhy snímač konfigurovat, je třeba upravit obsah formuláře „Nastavení“ na požadované hodnoty (viz kapitola 5.4) a tato data následně odeslat snímači pomocí volby „Odeslat nastavení“ (stále v menu „Zařízení“). Při dobrém stavu baterie je vhodné odeslání opakovat po několika sekundách (cca 20), aby byla minimalizována možnost chybného příjmu. Samotná jedna volba odeslání data odesílá mnohonásobně, avšak ani takto není zcela zaručeno jejich uložení.



Obr. 35: Formulář s nastavením při úpravě obsahu

Pokud má být snímač připojen na nového pacienta, musí být kalibrován a restartován. Toto vykoná položka „Kalibrovat a restartovat záznam“ v menu. V okamžiku stlačení a cca 5 s poté musí být snímač v kalibrační poloze, jinak nedojde ke správné kalibraci!

Pokud není v plánu další práce se snímačem, je vhodné jej co nejdříve volbou „Ukončit spojení“ v tomtéž menu odpojit; komunikace značně zatěžuje jeho baterii a v krajních případech by mohlo dojít k jejímu vybití během komunikace.

Pozice a velikosti podoken jsou při korektním ukončení běhu aplikace uloženy a při dalším spuštění obnoveny. Tyto údaje jsou uloženy v souboru `InterfaceState.ini`, uloženém ve složce s aplikací. Pozor při kopírování celé složky na počítač s menší obrazovkou!

7. Závěr

Při chirurgických zákrocích v zadním segmentu oka (vitrektomie) je odstraněna část sklivce a dočasně nahrazena tamponádou, která má obecně jinou hustotu než nitrooční kapalina. Aby tamponáda plnila svou funkci, musí přitlačovat určité místo a pacientova hlava proto musí být udržována v určité poloze.

Současné metody udržení polohy se zakládají na fixaci hlavy. Vyvíjený systém toto nahrazuje nepřetržitým sledováním náklonu hlavy a upozorňováním při nepovolené poloze. Výrazně je tak zlepšen komfort pacienta.

Výstupem této diplomové práce je koncový článek zmiňovaného systému pro pooperační sledování pacientů po vitrektomických zákrocích. Jde o softwarovou aplikaci pro stahování a prohlížení dat získaných ze snímače náklonu a jeho ovládání a běžnou uživatelskou údržbu. Zvolenou cílovou platformou je operační systém Windows verze XP a programovým prostředím je C++ Builder verze 6.

Aplikace je vyvíjena jako součást systému již částečně realizovaného, s existujícími prototypy hardware i firmware ostatních potřebných zařízení a specifikacemi komunikačního protokolu. Je schopna zobrazovat data s dostatečnou dávkou uživatelské přívětivosti jako graf včetně možnosti změny měřítka a rychlého posunu v čase. Dále umožňuje i generovat jednoduché statistické přehledy. Celé rozhraní lze uspořádat dle preferencí uživatele.

Z hlediska logického rozvržení aplikace obsahuje dvě částečně oddělené funkcionality, za prvé analytickou / prohlížecí a za druhé schopnost zacházet s bezdrátově připojeným snímačem náklonu, který představuje první článek celého systému. Snímač je zdrojem dat v aplikaci prezentovaných uživateli (zdravotníkovi), zároveň je však aplikace nástrojem umožňujícím uživateli snímač před sběrem dat konfigurovat dle požadavků.

Z hlediska technického, tedy programátorského, se aplikace skládá z několika vzájemně interagujících podsystémů realizovaných objekty, a to: uživatelského rozhraní, koordinačního objektu, datového kontejneru a komunikačního vlákna. Tyto objekty vzájemně komunikují buď přímo nebo prostřednictvím zpráv systému Windows. Komunikace s hardwarem probíhá přes rozhraní USB.

Celkově je možno říci, že cíle kladené v zadání byly splněny beze zbytku: aplikace je schopna data načítat, vizualizovat a statisticky analyzovat; kromě toho s ní lze snímač i ovládat a není tedy třeba dalších programů pro tyto činnosti.

Literatura

[1] Vitreous–Retina–Macula Consultants of New York. *Vitreous–Retina–Macula Consultants of New York : Patient Education* [online]. 2006, 2009 [cit. 2010-05-18]. Retinal Tears and Retinal Detachment. Dostupné z WWW: < http://vrmny.com/pe/rtrd.html > .

[2] Vitreous–Retina–Macula Consultants of New York. *Vitreous–Retina–Macula Consultants of New York : Education* [online]. 2006, 2009 [cit. 2010-05-18]. Vitrectomy. Dostupné z WWW: < http://vrmny.com/pe/vitrectomy.html > .

[3] St. Luke's Cataract & Laser Institute. *St. Luke's Cataract & Laser Institute : Surgical* [online]. 2008, 2010 [cit. 2010-05-18]. Vitrectomy Surgery. Dostupné z WWW: < http://www.stlukeseye.com/surgical/vitrectomy.html >.

[4] Vitrectomy Solutions, Inc. *Vitrectomy Solutions : Facedown Recovery for a Brighter Tomorrow* [online]. 2006 [cit. 2010-05-18]. Vitrectomy Surgery. Dostupné z WWW: < http://vitrectomysolutions.com/surgery.asp >.

[5] LékařiOnline.CZ s.r.o. *Lékaři-online.cz - Informace půl zdraví* [online]. 2008 [cit. 2010-05-18]. Chirurgická léčba chorob sítnice a sklivce. Dostupné z WWW: < http://vitrectomysolutions.com/surgery.asp >.

[6] DOOBAL Medical & Health Furniture. *Dubal Medical & Health Furniture* [online]. 2006 [cit. 2010-05-18]. Chair for rehabilitation of post operative "Vitrectomy" patients. Dostupné z WWW: < http://www.doobal.com/english/vitrectormy_chair.html >.

[7] Microsoft Corporation. *MSDN* [online]. 5/4/2010. 2009, 2010 [cit. 2010-05-18]. PlgBlt Function. Dostupné z WWW: < http://msdn.microsoft.com/en-us/library/dd162804(VS.85).aspx >.

[8] LEUNEN, Michel. *Leunen.com* [online]. 2007 [cit. 2010-05-18]. Rotating a bitmap. Dostupné z WWW: < http://www.leunen.com/cbuilder/rotbmp.html >.

[9] *Free Pascal Community* [online]. 2009 [cit. 2010-05-18]. TWriter. Dostupné z WWW: < http://community.freepascal.org:10000/docs-html/rtl/classes/twriter.html >.

[10] Nordic Semiconductor ASA. *Nordic Semiconductor* [online]. 2010 [cit. 2010-05-18]. NRF401 Transceiver. Dostupné z WWW: < http://www.nordicsemi.com/index.cfm?obj=product&act=display&pro=56 >

[11] Future Technology Devices International Ltd. *D2XX Programmer's Guide* [online]. [s.l.] : [s.n.], 2008 [cit. 2010-05-18]. Dostupné z WWW: < http://www.ftdichip.com/Documents/ProgramGuides/D2XX_Programmer's_Guide (FT_000071).pdf >

[12] FAITHURST, Godred. *Research Information : Prof. Godred Fairhurst* [online]. 2001 [cit. 2010-05-18]. Cyclic Redundancy Check. Dostupné z WWW: < http://www.erg.abdn.ac.uk/users/gorry/eg3567/dl-pages/crc.html >.

[13] Microsoft Corporation. *MSDN* [online]. 5/17/2010. 2010 [cit. 2010-05-18]. About Messages and Message Queues. Dostupné z WWW: < http://msdn.microsoft.com/en-us/library/ms644927 (VS.85).aspx >.

[14] VIRIUS, Miroslav. *Jazyky C a C++ : Kompletní průvodce programátora*. První vydání. Praha : Grada Publishing, 2006. 520 s. ISBN 80-247-1494-9.

[15] TEIXEIRA, Steve; PACHECO, Xavier. *Borland Delphi : průvodce vývojáře*. První vydání. Brno : UNIS Publishing s.r.o., 1999. Tvorba aplikací s více thready, s. 61-114. ISBN 80-86097-36-6.

[16] ČÍŽEK, Martin, et al. Electronic Monitoring of Head Position after Vitrectomy. In SLOTEN, Jos Vander, et al. *4th European Conference of the International Federation for Medical and Biological Engineering*. Heidelberg : Springer Berlin, 2009. Volume 22, Part 11. ISBN 978-3-540-89207-6.

[17] ČÍŽEK, Martin, et al. Long-Term Head Posture Monitoring Reduces Complications Following Vitrectomy. In DÖSSEL, Olaf; SCHLEGEL C., Wolfgang. *World Congress on Medical Physics and Biomedical Engineering, September 7 - 12, 2009, Munich, Germany : Vol. 25/11 Biomedical Engineering for Audiology, Ophthalmology, Emergency & Dental Medicine*. Heidelberg : Springer Berlin, 2009. s. 134-136. ISBN 978-3-642-03890-7

Seznam zkratek

(abecední řazení)

API	Application programming interface (rozhraní programování aplikací)
COM	Component object model (komponentově objektový model)
CRC	Cyclic redundancy check (cyklický kontrolní součet)
DLL	Dynamic-link library (dynamicky linkovaná knihovna)
DTR	Data terminal ready (terminál připraven) – signál v rámci RS-232
FTDI	Future Technology Devices International (název firmy)
GUI	Graphical user interface (grafické uživatelské rozhraní)
IDE	Integrated development environment (integrované vývojové prostředí)
MDI	Multiple document interface (rozhraní více dokumentů)
PC	Personal computer (osobní počítač)
PPV	Pars plana vitrektomie
RF module	Radio frequency module (modul rádiové frekvence – pro bezdrátovou komunikaci)
RS-232	Recommended standard 232 (doporučený standard 232 – označení rozšířené sériové sběrnice počítače)
RTS	Request to send (požadavek odeslání) – signál v rámci RS-232
URL	Uniform resource locator (uniformní lokátor zdroje)
USB	Universal serial bus (univerzální sériová sběrnice)
VCL	Visual component library (knihovna vizuálních komponent)
VCP	Virtual COM port (virtuální port COM – pozor, „COM“ zde je pouze označením, nikoliv zkratkou)
WYSIWYG	„What you see is what you get“ (co vidíte, to dostanete)

Přílohy

A. Kompletní uživatelské rozhraní aplikace