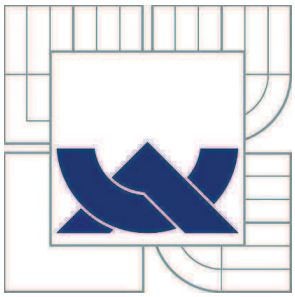
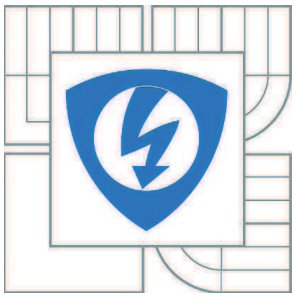


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF TELECOMMUNICATIONS

NOUZOVÉ TLAČÍTKO PRO SENIORY

EMERGENCY BUTTON FOR SENIORS

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

AUTOR PRÁCE — Bc. JOSEF VERBÍK

AUTHOR

VEDOUCÍ PRÁCE — Ing. IVO STRAŠIL

SUPERVISOR

BRNO 2011

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Diplomová práce

magisterský navazující studijní obor
Telekomunikační a informační technika

Student:	Bc. Josef Verbík	***ID:***	83155
Ročník:	2	***Akademický rok:***	2010/2011

NÁZEV TÉMATU:

Nouzové tlačítko pro seniory

POKYNY PRO VYPRACOVÁNÍ:

Cílem diplomové práce je vyvinout firmware pro programovatelné hodinky - vývojový kit Texas Instruments EZ430-CHRONOS. Hodinky jsou vybaveny rozhraním pro bezdrátovou komunikaci a akcelerometrem.
Firmware přemění hodinky na inteligentní nouzové tlačítko, které upozorní pomocí bezdrátového spojení, internetu a příslušné SW aplikace na PC pečovatelskou službu na nutnost zásahu. Požadavek na provedení zásahu bude vyvolán v případě, že nositel hodinek stiskne příslušné tlačítko nebo nebude akcelerometrem detekován pohyb déle než 12 hodin. Je možné rovněž pokusně doplnit detekci záchvatu pomocí akcelerometru.
Výsledkem práce bude rovněž funkční software pro PC resp. nadřazený systém.

DOPORUČENÁ LITERATURA:

[1] EZ430-Chronos Development Tool User's Guide [online]. 2. vydání. [s.l.] : Texas Instruments, 2009, červen 2010 [cit. 2010-09-26]. Dostupné z WWW: <http://focus.ti.com/lit/ug/slau292b/slau292b.pdf>.
[2] HOROWITZ, Paul, WINFIELD, Hill. The Art of Electronics. 2nd enl. edition. Cambridge : Cambridge University Books, 1989. 1152 s. ISBN 0521370957.

Termín zadání:	7.2.2011	***Termín odevzdání:***	26.5.2011

Vedoucí práce: Ing. Ivo Strašil

prof. Ing. Kamil Vrba, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

ABSTRAKT

Diplomová práce Nouzové tlačítko pro seniory se zabývá návrhem a realizací systému pohotovostní signalizace. Systém tvoří osobní programovatelné digitální hodinky s bezdrátovým rozhraním a PC se softwarem zpracovávající signály. Oznámení stavu je tvořena pomocí SMS zpráv a zobrazením dialogu na monitoru.

K realizaci práce je použitý svobodný software. Operační systém na PC běží Linux (distribuce Ubuntu 11.04). Pro vytvoření grafického rozhraní PC aplikace je použit multiplatformní framework QT. K tvorbě firmwaru hodinek slouží C kompilátor MSP430-GCC, MSP debuger v0.15 a firmware OpenChronos.

KLÍČOVÁ SLOVA

Signalizace, Nízko-spotřebová zařízení, Bezdrátová technologie, Detekce pohybu, Pohotovost

ABSTRACT

This Thesis „Emergency Button for Seniors“ is about suggestion of realization of system emergency signalization. The System consists of programmable digital watch with wireless interface and PC with software processing signals. Status signalization is created by phone message and figuration of dialog on monitor.

There is used free software for realization. Operation system on PC is Linux (distribution Ubontu 11.04). For graphic transitional plane PC application is used cross-platform framework QT. For watch firmware is used C compiler MSP430-GCC, MSP debug v0,15 and firmware OpenChronos.

KEYWORDS

Signaling, Low power Device, Wireless Technology, Motion Detection, Emergency

Bibliografická citace práce:

VERBÍK, J. *Nouzové tlačítko pro seniory.* Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2011. 42 stran, 3 přílohy. Vedoucí diplomové práce Ing. Ivo Strašil.

Prohlášení

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

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

V Brně dne …… .....

podpis autora

PODĚKOVÁNÍ

Děkuji vedoucímu diplomové práce Ing. Ivo Strašilovi za velmi užitečnou metodickou pomoc a cenné rady při celkové zpracování diplomové práce.

V Brně dne …… .....

podpis autora

Obsah

1. ÚVOD

V dnešních moderních průmyslových zemích se stále zvyšuje počet seniorů [1]. Pro společnost se zvyšuje náročnost péče o ně. Usnadnit by to mohly nové postupy a technologie. Tato diplomová práce se takovou možnou technologií zabývá. Nese jméno - Nouzové tlačítko pro seniory.

Cílem této diplomové práce je seznámit se s vývojovým kitem eZ430-Chronos-433 od firmy Texas Instruments a realizovat nouzovou signalizaci pro seniory.

Úvodní část práce popisuje samotný vývojový kit eZ430-Chronos a jeho součásti. Nejvíce popsané jsou programovatelné sportovní hodinky Chronos, které nesou celou řadu senzorů s ultra nízkou spotřebou. Se zdařilým hardwarem drží krok taky jeho bohatá softwarová výbava, která je v práci taky popsána.

V další části je uveden rozbor teoretického nasazení v praxi. Je zde popsán koncept nasazení v reálných podmínkách. Koncept se odvíjí od požadavku co nejmenší energetické spotřeby hodinek. Z tohoto popisu vychází návrh zjednodušeného konceptu, který je realizován.

V poslední části je rozebrána samostatná realizace projektu. K vývoji PC aplikace i firmwaru hodinek se používají výhradně aplikace a knihovny pod licencí GNU. Popsány jsou pouze nejdůležitější zdrojové kódy firmwaru hodinek a PC aplikace.

2. EZ430-CHRONOS

Vývojový kit Texas Instrument eZ430-Chronos-433 je vysoce integrovaný [2], přenosný vývojový systém založený na mikrokontroléru CC430. Může se použít jako osobní hodinky, osobní displej pro monitorování sítě nebo jako bezdrátový uzel pro sběr dat. Integrované radiové rozhraní umožňuje sběr dat z blízkých bezdrátových senzorů (např. krokoměr, monitor srdečního tepu).

Hodinky eZ430-Chronos jdou přeprogramovat vlastními aplikacemi pomocí dodávaného softwaru.

Vývojový kit Texas Instrument eZ430-Chronos-433 obsahuje:

- hodinky Chronos,
- programovací USB rozhraní eZ430,
- bezdrátový přístupový bod CC1111 (433 MHz),
- křížový šroubovák,
- dva náhradní šroubky,
- článkovou lithiovou baterii CR2032,
- CD s bohatou dokumentací a vývojovým softwarem.

Obr. 1: Hodinky Chronos, programovací rozhraní a bezdrátový přístupový bod [2]

Firma Texas Instruments nabízí tři varianty svého vývojového kitu eZ430-Chronos. Liší se pouze frekvencí bezdrátového rozhraní. Frekvence jsou 433, 868 a 915 MHz. Kit s 433 MHz se nabízí celosvětově, 868 MHz se prodává v Evropě a Indii a 915 MHz v severní a jižní Americe. Rozdílné frekvence jsou dány různými bezplatnými frekvenčními pásmy v daných regionech. Všechny tři varianty jsou nabízeny za jednotnou cenu 49.00 $.

2.1 HODINKY CHRONOS

Hodinky Chronos (Obr. 1) jsou dodány jako plně funkční sportovní digitální hodinky. Lze k nim dokoupit sportovní snímač srdečního tepu s bezdrátovým rozhraním (BM-CS5). K prodeji jsou dvě varianty – první s krátkým dosahem (10 m) a druhá s velmi dlouhým dosahem (400 m).

Hodinky Chronos se skládají z:

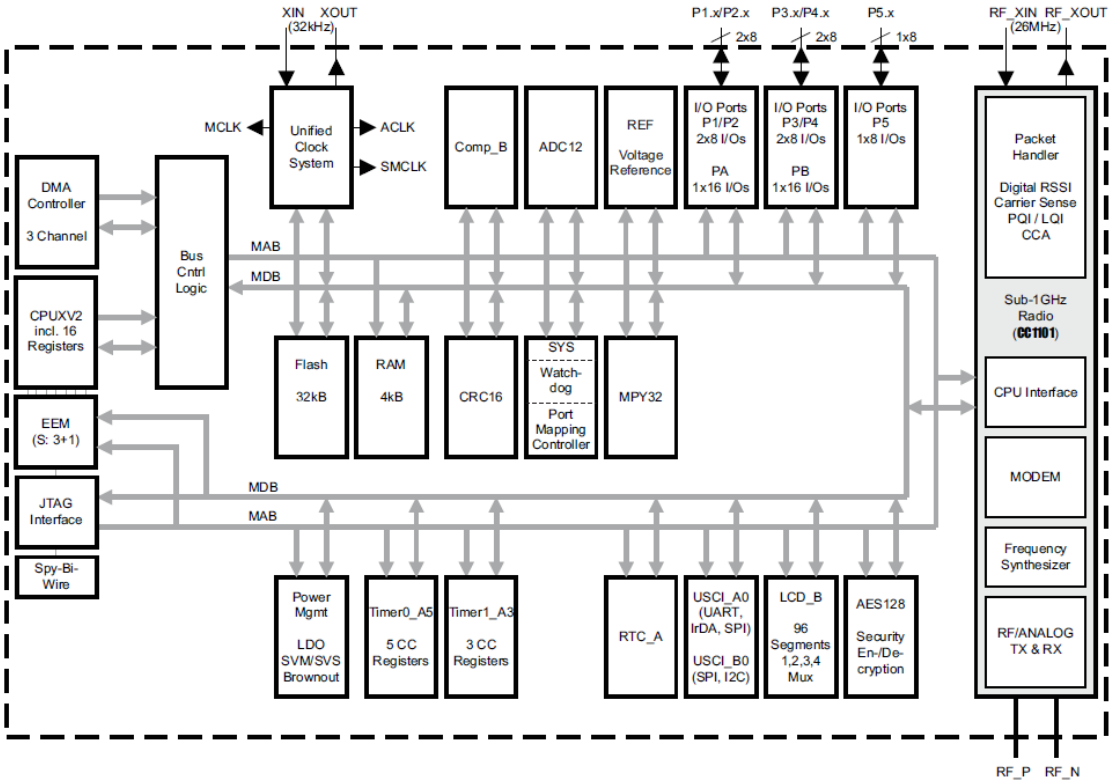
- mikrokontroléru MSP430 s integrovaným bezdrátovým rozhraním,
- akcelerometr CMA3000,
- tlakového senzoru SCP1000,
- teplotního senzoru (integrovaný v mikrokontroléru CC430F6137),
- senzoru napětí (integrovaný v CC430F6137),
- 96ti segmentového LCD displeje.

2.1.1 Mikrokontrolér MSP430

Mikrokontrolér MSP430 je 16-ti bitový RISC procesor [3]. Byl navržen speciálně pro zařízení s ultra nízkou spotřebou. Má v sobě integrované velké množství analogových a digitálních periferií. Tyto periferie byly navrženy tak, aby byla zachována jejich maximální funkčnost s minimálním odběrem elektrické energie. Mnoho periferií může fungovat autonomně, čímž se ušetří čas procesoru v aktivním režimu.

V hodinkách Chronos je použit mikrokontrolér MSP430 s integrovaným bezdrátovým rozhraním. Tento mikrokontrolér je označován jako CC430F6137. Běží na frekvenci 20 MHz, operační paměť RAM má velikost 4 kB a paměť FLASH 32 kB.

Má mnoho integrovaných periferií, které nejlépe vystihne jeho blokové schéma (Obr. 2).



Obr. 2: Blokové schéma CC430F6137 převzato z [3]

Mikrokontrolér CC430F6137 disponuje osmi operačními režimy označovanými LPM0 až LPM4.5. V aktivním režimu pracují všechny součásti. V dalších režimech se postupně jednotlivé periferie přepínají do pohotovostního režimu, až nakonec bude celý obvod neaktivní, do aktivního režimu ho může přivést už jen externí signál.

CPU v aktivním režimu má spotřebu 160 µA/MHz, v pohotovostním režimu 2 µA a ve vypnutém stavu 1 µA.

2.1.2 Akcelerometr CMA3000

Akcelerometr CMA3000 dokáže snímat pohyb ve všech třech osách – výška, šířka a hloubka [4]. Má dva měřící rozsahy mezi kterýma lze přepínat. Tíhové zrychlení *g* udává výrobce VTI Technologies v rozsahu ±2 g a ±8 g. Rozlišení snímače při rozsahu ±2 g je 17 mg. Při rozsahu ±8 g je 67 mg. Spotřeba se pohybuje od 7 do 70 µA.

2.1.3 Senzor tlaku SCP1000

Senzor tlaku SCP1000 je stejně jako akcelerometr CMA3000 vyráběn firmou VTI Technologies [5]. SCP1000 je digitální senzor absolutního tlaku, který se skládá ze tří kapacitních snímačů. Senzor pracuje v pásmu 30 až 120 kPa při teplotách -20 až 70 °C. Spotřebu výrobce udává 3,5 µA.

2.1.4 Spotřeba hodinek

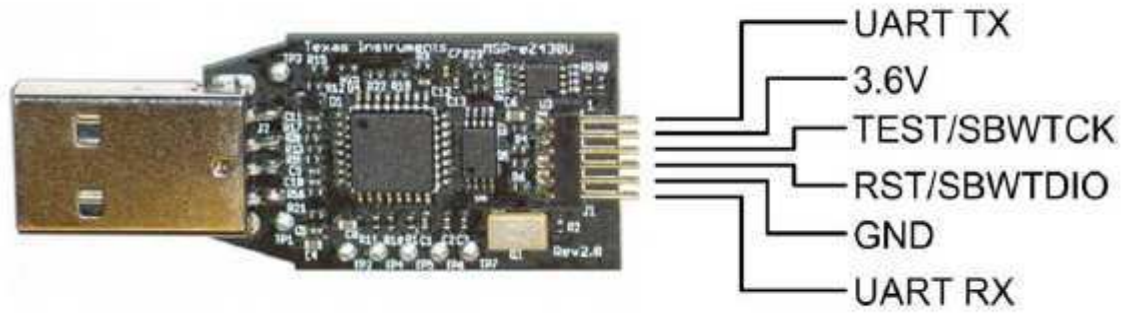
Výrobce hodinek Chronos s dodávaným softwarem pro sportovní účely udává odhadovanou výdrž baterie 12 měsíců. Podrobnější informace o spotřebě jsou uvedeny v Tab. 1.

Tab. 1: Průměrná spotřeba jednotlivých částí hodinek Chronos

Mód	**spotřeba**	**životnost baterie**
úsporný režim	2,7 µA	92 měsíců
úvodní obrazovka	8,9 µA	28 měsíců
hodiny	9 µA	27,7 měsíců
měření teploty	10 µA	25 měsíců
měření tlaku a nadmořské výšky	18 µA	13, 8 měsiců
senzor pohybu	166 µA	1,5 měsíce
příjem radiového signálu	0,9 mA	8 dní
vysílání radiového signálu	3,7 mA	2 dny

2.2 EZ430-CHRONOS DEBUG INTERFACE

Kit eZ430-Chronos Debug Interface (Obr. 3) obsahuje USB emulátor s programovací a ladící funkcí pomocí dvoudrátového rozhraní.

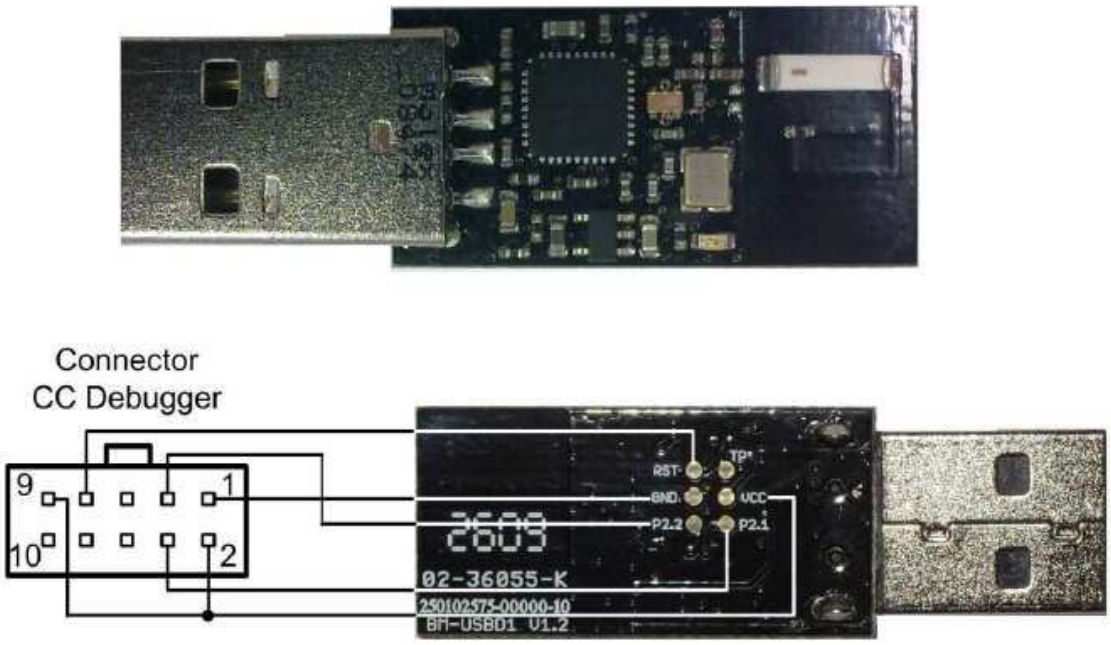


Obr. 3: eZ430-Chronos Debug Interface [2]

Pomocí tohoto zařízení se spojí hodinky Chronos (respektive mikrokontrolér MSP430) s počítačem (PC). Pak lze nahrát nový program do hodinek a případně ladit program k dokonalosti.

2.3 EZ430-CHRONOSRF ACCESS POINT

Zařízení eZ430-Chronos RF Access Point (Obr. 4) slouží k bezdrátovému propojení počítače a hodinek. K PC se připojuje pomocí USB rozhraní. S dodaným softwarem slouží k nastaveni času, nadmořské výšky a teploty, k synchronizaci informací (např. z měřiče tepu či krokoměru) a k ovládání PC pomocí (kurzor myši lze ovládat pomocí pohybu hodinek snímaných akcelerometrem).



Obr. 4: eZ430-Chronos RF Access Point [2]

Srdcem zařízení je integrovaný obvod CC1111 RF. Je schopen komunikovat pomocí USB 2.0 a to maximální přenosovou rychlostí 12 Mb/s, což je vzhledem k velikosti paměti v hodinkách zcela dostatečné.

2.3.1 Komunikační protokol

Zařízení emuluje USB připojení jako sériový port (COM). Komunikace mezi Access pointem (AP) a PC probíhá obousměrně pomocí sériového portu. PC se zeptá a

AP odpovídá. Komunikaci mezi hodinkami a PC zprostředkovává AP. Komunikační protokol mezi AP a hodinkami tedy nepotřebujeme znát. V prvním kroku si PC zjistí vysláním kódu o třech bytech (FF 01 03) jestli je v dosahu jiné zařízení. AP to zjistí a odpoví taky kódem o třech bytech (FF 06 03). Po této inicializaci si PC říká o požadovaná data. Požadavek má minimální délku 3B, odpověď 4B. PC si vždycky žádá jako první o data z hodinek. Odpověď se nahrává do bufferu, který je součástí AP.

Posloupnost komunikace vypadá takto:

1. Inicializace: - žádost PC
   - odpověď AP
2. Spuštění přenosu dat: - žádost PC
   - odpověď AP
   - uložení odpovědi do bufferu
   - poslání obsahu bufferu
   - smazání bufferu
3. Ukončení přenosu dat: - žádost PC
   - odpověď AP
4. Ukončení spojení: - žádost PC
   - odpověďAP

Krok číslo dva se opakuje dokud se nepřenese celý požadovaný blok dat.

2.4 SOFTWARE

Na přiloženém CD se nachází bohatá softwarová výbava a podrobná technická dokumentace k hodinkám a jednotlivým součástem.

Softwarová výbava se skládá z:

- programu (firmware) pro sportovní hodinky Chronos,
- balíčku aplikací a ovladačů pro MS Windows i Linux,
- vývojového prostředí Code Composer Studio v4.1 Core Edition,

- vývojového prostředí IAR Embedded Workbench Kickstart for MSP430 5.10.4.

Program hodinek Chronos má řadu funkcí, které jsou nadstandardní a pro většinu lidí i nepoužitelné. Základní funkce jsou: hodiny, datum, stopky, alarm, teploměr, měřič napětí na baterii a synchronizace s PC. Při dokoupení měřiče srdečního tepu umí hodinky po spárování zobrazit i údaje z tohoto měřiče. Další, dle mého názoru, nadstandardní funkce jsou:

- ukazatel nadmořské výšky,
- ovládání PC (pomocí tlačítek, akcelerometru a aplikací na PC),
- zobrazení pohybu na PC v osách X, Y a Z.

Dodaný programový vývojový kit obsahuje všechny potřebné knihovny a zdrojové kódy pro tvorbu firmware hodinek a aplikací na PC. Veškerý software je založený na myšlence co nejmenší energetické zátěže hardware. Tj. např. přenos dat mezi hodinkami Chronos a PC je založen na protokolech BlueRobin a SimpliciTI. Protokol BlueRobin je duchovní vlastnictví německé firmy BM wireless Ltd.&Co.KG a protokol SipliciTI je, jak název napovídá, ve vlastnictví Texas Instruments.

Obě dodávaná vývojová prostředí IAR Embedded Workbench Kickstart a Code Composer Studio v4.1 Core Edition jsou však omezeny na velikost zdrojového kódu. První jmenováný má limit nastaven na 8 kB. Code Composer Studio má limit o 2kB větší, tj. 10kB. Pro využití celé paměti (32kB) je tedy potřeba koupit vyšší verze těchto edicí, které však začínají na cenách 500$. Z tohoto důvodu použiji komunitní kompilátor MSP430-GCC, který je zdarma a nemá žádné limity.

2.5 ALTERNATIVNÍ SYSTÉMY

Autorovi se nepodařilo zjistit, že by byl komerčně nasazen podobný produkt jako je vývojový kit eZ430-Chronos od firmy Texas Instruments. Sportovních hodinek se prodává velké množství, ale oproti Chronosu „nic“ neumí a jsou předražené. Konkurence si pod pojmem programovatelné hodinky představuje nastavení funkce (alarm, stopky, apod) na tlačítko.

Jediné podobné zařízení, které se už na našem trhu pár let prodává je mobilní telefon pro seniory s velkým dobře čitelným displejem a velkými tlačítky. Tento telefon začala u nás prodávat jako první firma Aligator (dnes je jich víc). Nyní má v nabídce několik modelů pro seniory. Všechny mají nouzové tlačítko, které po zmáčknutí automaticky začne postupně vytáčet zvolená čísla. Když se telefon v požadované době s nikým nespojí, tak automaticky rozešle krátké textové zprávy na tyto čísla.

Z vlastní zkušenosti autora jsou však tyto telefony pro seniory nepraktické, protože jsou pro ně moc velké (mobilní telefony obecně) a drahé. Proto je nechávají ležet někde na stole nebo poličce. Moje babička má tento telefon a nikdy ho s sebou nenosí. V případě pádu či jiné nehody by byl pro ni nedostupný. Pro seniory je tedy vhodné něco menšího a něco na co jsou zvyklí. V tomto případě jsou hodinky ideální volbou.

Na asijských trzích se dají koupit i hodinky s GSM modulem. Ceny s poštovným začínají kolem 1500 Kč. Výrobců je celá řada, ale pro evropana jsou zcela neznámí. Nevýhodou těchto hodinek je velmi malá výdrž baterie a pochybná kvalita produktu.

30. června 2011 mají být uvedeny do prodeje dva nové modely hodinek firmy Texas Instrument MSP-WDS430BT1000AD [6] a MSP-WDS430BT2000D [7]. Cena je stanovena na 199.00 $. Funkcionalitou se tyto modely neliší. První jmenovaný model má analogové zobrazení času (tj. ručičky) a dva malé digitální displeje. Druhý jmenovaný má jeden velký digitální displej. Tyto nové vylepšené modely mají navíc oproti eZ430 podporu bluetooth (technologie bezdrátové komunikace mezi dvěma a více zařízeními).

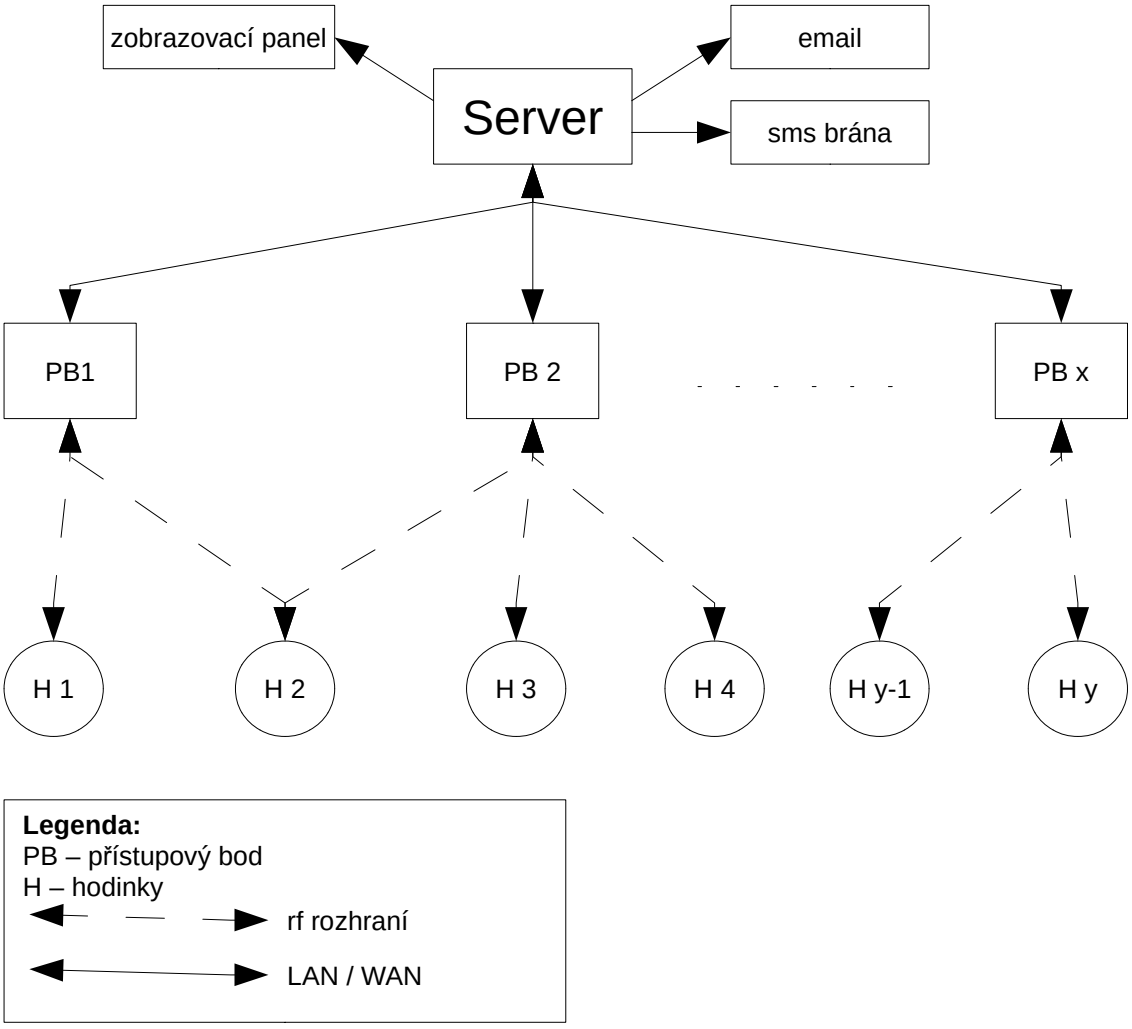
Díky bluetooth by byl systém nouzové tlačítko pro seniory založený na těchto hodinkách výhodnější, protože tato technologie je dnes implementována téměř v každém mobilním telefonu a notebooku. Nevýhodou je vysoká pořizovací cena hodinek, která je čtyřikrát vyšší než cena eZ430 a též zatím neznámá výdrž baterie.

Obr. 5: Hodinky MSP-WDS430BT1000AD a MSP-WDS430BT2000D [8]

3. NÁVRH REALIZACE

3.1 TEORETICKÉ NASAZENÍ

Můj návrh realizace projektu vychází z předpokladu, že uživatelé hodinek budou pacienti nemocnic nebo obyvatelé domů pro seniory. Obecné schéma zapojení v takové budově je zobrazen na Obr. 5.



Obr. 6: Schéma zapojení systému

Server je počítač s potřebnou aplikací, který bude shromažďovat a analyzovat data z hodinek. V případě potřeby server upozorní personál pomocí sms, emailu nebo zobrazením zprávy na monitoru. Jednotlivé hodinky jsou v dosahu bezdrátových

přístupových bodů (PB). Jeden PB by zpravidla pokrýval jednu místnost, na větší místnost (halu či chodbu) by se použilo více PB. Přístupové body by se propojily se serverem pomocí sítě LAN. V dnešní době už je mnoho budov pokryto počítačovou sítí (WAN, LAN) a při realizaci tohoto systému by se snížily finanční náklady použitím této infrastruktury. Elektrické napájení PB by se provedlo pomocí LAN.

Z hlediska provozu a údržby je nutné, aby hodinky vydržely fungovat co nejdéle. Tím pádem se musí všechny zbytečné funkce a periferie vypnout nebo aspoň přepnout do pohotovostního režimu a všechny potřebné výpočty a analýzy nechat vykonávat server.

3.1.1 Komunikační protokol

Přenos dat ze strany hodinek musí být omezeny na co nejmenší úroveň kvůli energetické náročnosti bezdrátového přenosu dat. Proto je vhodné posílat co nejmenší objem dat.

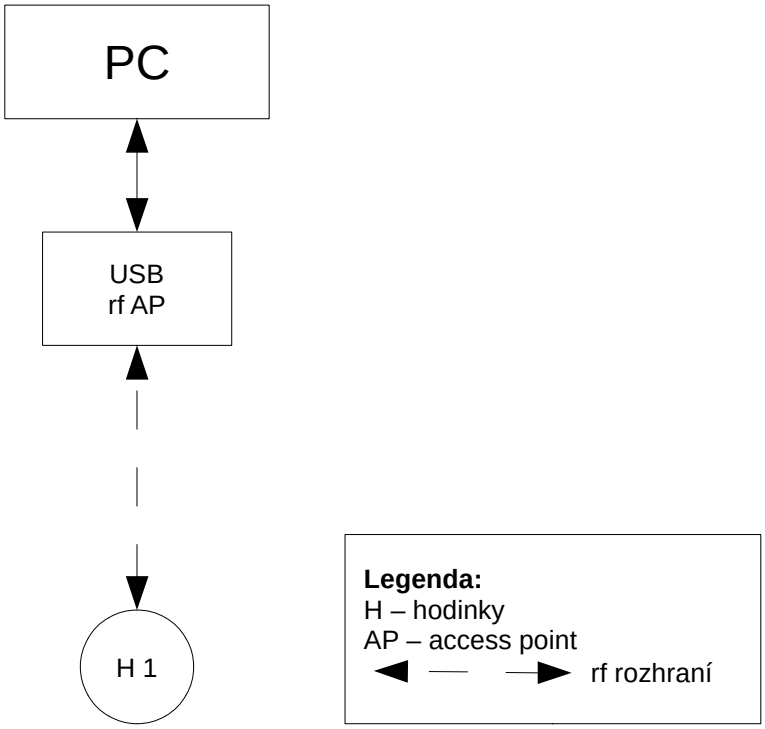
Každé hodinky by měli jedinečné identifikační číslo (HID), ke kterému by se v databázi přiřadil sledovaný uživatel. Při synchronizaci hodinek se serverem by hodinky vždy poslaly svoje HID. Další přenášenou informací z hodinek by bylo identifikační číslo signálu (SOSID). Každý přístupový bod by měl taky jedinečné identifikační číslo (PBID), ke kterému by se přiřadila místnost pro snadnou lokalizaci. HID by měl mít aspoň 1 byte (255 zařízení) a PBID taktéž. Pro SOSID by stačily 3 byty, které by byly rozděleny na tři části. První část by měla čtyři bity, které by byly určeny pro určení druhu signálu (záchvat, stisknuté tlačítko, běžné hlášení). Druhá část, taky čtyř bitová, by signalizovala stav baterie (slabá, v pořádku). Na poslední třetí část tedy zbývají 2 byty. Ty budou sloužit pro počet vteřin, které jsou hodinky bez pohybu. Hodinky by tím pádem odeslali jen 4 bytový kód ve tvaru [HID|SOSID]. PB, který zachytí signál, přidá svůj PBID a odešle serveru kód ve tvaru [PBID|HID|SOSID]. Tento kód (Obr. 7) server zpracuje a upozorní o případném problému. Podle kódu pozná kde a kdo má nesnáze.

<table>
<tr><td>PBID</td><td>HID</td><td colspan="3">SOSID</td></tr>
<tr><td colspan="2"></td><td>Stav</td><td>Baterie</td><td>Čas</td></tr>
</table>

Obr. 7: Paket příchozího signálu

3.2 VLASTNÍ NASAZENÍ

Předcházející popis teoretického nasazení je složitější na realizaci kvůli množství prvků. Proto při realizaci mého projektu modelovou situaci poněkud zjednoduším (Obr. 8).



Obr. 8: Zjednodušené schéma zapojení systému

Navrhovaný systém bude obsahovat pouze jedny hodinky, které mi zapůjčil vedoucí projektu pan Ing. Ivo Strašil. Jako server použiji běžný domácí počítač nebo notebook. K bezdrátové komunikaci mezi PC a hodinkami Chronos použiji dodaný eZ430 access point. Mezi hodinkami Chronos a tímto access pointem je dosah asi 4 metry.

3.2.1 Funkcionality systému

Hodinky vyšlou každých 15 minut signál HIDSOSID (Tab. 2). Po přijmutí signálu si server zaznamená HID a čas přijetí. Pokud se nebudou hlásit dobu delší jak hodinu, server upozorní, že je dotyčná osoba mimo dosah signálu.

Signál HIDSOSID vyšlou hodinky i v případě předem určeného tlačítka nebo

při detekci silnějšího třesu (podezření na záchvat). Tyto signály bude aplikace na serveru zaznamenávat a v případě definovaných kritérií vyvolá poplach. Z Tab. 2 je patrné, že celkový počet všech možných stavů je šest.

Hodinky budou měřit napětí na baterii jednou denně. Pokud zjistí nižší napětí než přípustných 2,6 V, bude odesílat požadovaný SOSID dokud se baterie nevymění či napětí na baterii nepoklesne pod 2,5 V, kdy se podle údaje výrobce už nezapne bezdrátové rozhraní hodinek.

Tab. 2: Významy signálu SOSID v bitech

SOSID					
STAV	popis	Baterie	popis	Čas bez pohybu	popis
0000	pravidelné hlášení	0010	dobrá	0000000000000000	0 sekund
0010	stisknuté tlačítko	0100	slabá	0000000000000001	1 sekunda
0100	třes			⋮	
				1111111111111111	65535 sekund

3.3 POŽADAVKY NA FIRMWARE

3.3.1 Úsporná energetická opatření

Všechny nepotřebné periferie hodinek budou deaktivované. Zapnuté zůstanou jen nezbytné součásti. Tj. hodinky (datum, čas a budík), některé digitální periferie (tlačítka k ovládání hodinek) a v daných intervalech bude zapnutý akcelerometr a bezdrátové rozhraní.

Akcelerometr je velmi náročný na spotřebu elektrické energie. Proto bude akcelerometr snímat pohyb jen čtyřikrát za sekundu. Jinak by se baterie v hodinkách asi po měsíci a půl nepřetrženého provozu vybila (Tab. 1). Tímto se výdrž baterie významně prodlouží.

Napětí na baterii se bude měřit jen jednou denně. Sice má měření velmi malou spotřebu, ale i tak je potřeba ušetřit co možná nejvíce energie. Pokud bude napětí nad danou mez, tak v SOSID bude informace, že je baterie stále nabitá (Tab. 2). Pokud měření napětí bude nevyhovující, projeví se v kódu SOSID informace, že je baterie vybitá.

3.3.2 Funkce hodinek

Hlavní funkcí hodinek Chronos bude vyslání signálu pečovatelské službě při:

- stisknutí příslušného tlačítka,
- zjištění nezvyklého třesu osoby nesoucí hodinky.

Zjištění nezvyklého třesu vyhodnotí hodinky z údajů dodávající akcelerometry.

Hodinky budou mít doplňkovou funkci, která bude ukládat číslo znamenající čas v sekundách. Toto číslo se tedy bude každou sekundu navyšovat o 1, za předpokladu, že hodinky nezaznamenají pohyb. Číslo se vynuluje při odeslání signálu nebo při detekci pohybu. To znamená, že číslo může nabýt maximálně hodnoty 900 (15 minut = 900 sekund).

3.4 POŽADAVKY NA APLIKACI SERVERU

Aplikace má být uživatelsky přivětivá, přehledná a jednoduchá. Psát ji budu v textovém editoru jazykem C++. Díky využití frameworku Qt by měla být i multiplatformní.

3.4.1 Funkce aplikace

Aplikace musí umět komunikovat s USB access pointem. Údaje (HIDSOSID) z access pointu bude aplikace zaznamenávat a zpracovávat. Z těchto záznamů bude aplikace tvořit statistiky a bude hlídat, zda některé údaje nepřekročí stanovenou mez. Aplikace dokáže podporovat více hodinek a ke kódu HID dokáže přiřadit jméno osoby. Bude taky zobrazovat přehledně online záznam činnosti hodinek.

Ze zpracovávaného kódu HIDSOSID dokáže určit zda napětí na baterce v hodinkách nekleslo pod požadovanou mez, zda není pacient mimo dosah radiového signálu a či jsou hodinky dlouho bez pohybu.

Při událostech „stisknuté nouzové tlačítko“ nebo „záchvat“ vyskočí v aplikaci nové okno s oznámením o nouzové situaci a současně by měla aplikace odeslat zprávu ve formě SMS nebo emailu.

Pokud aplikace zaznamená, že je uživatel dvanáct hodin bez zaznamenaného

pohybu, tak upozorní operátora vyskakovací okno.

Při nedostupnosti signálu delším jako dvě hodiny (procházka sledované osoby) aplikace opět upozorní operátora vyskakovacím oknem.

4. PC APLIKACE

4.1 VÝVOJOVÉ PROSTŘEDÍ

PC aplikace je vytvořena pod Linuxem [10] pomocí volně šiřitelných programů. Linux je operační systém unixového typu, který vznikl v roce 1991. S Unixem nemá shodné zdrojové kódy, ale má stejnou filosofii a aplikační rozhraní. Linux je kompatibilní s Unixem na úrovni zdrojových kódů. Vyvinul ho finský vysokoškolský student Linus Torvalds. Nyní se na vývoji Linuxu podílí vývojáři z celého světa. Linux je vyvíjen pod licencí GNU, tj. zdrojový kód je volně k dispozici.

Linux podporuje současný běh více uživatelů. Každý uživatel může spouštět neomezený počet programů.

Open source je software, který má otevřené zdrojové kódy. K softwaru jsou k dispozici zdrojové kódy, jenž mohou být za určitých podmínek prohlíženy a upravovány.

4.1.1 Zvolená distribuce

Pro realizaci byla vybrána 32 bitová distribuce Ubuntu [11] ve verzi 11.04 s kódovým označením Natty Narwhal. Ubuntu je jihoafrické slovo, které se přibližně překládá jako „lidskost ostatním“. Ubuntu vyvíjí firma Canonical Limited. Díky GNU licenci Linuxu je tato distribuce zdarma.

Práce na Ubuntu je dána filozofií svobodného softwaru. Jádrem filozofie jsou tyto myšlenky:

- Každý uživatel by měl mít svobodu stahovat, kopírovat, studovat, měnit, sdílet, distribuovat a vylepšovat svůj software k jakýmkoli účelům, aniž by musel platit licenční poplatky.
- Každý uživatel by měl mít možnost užívat software v jazyce podle svého výběru.
- Každý uživatel by měl dostat možnost používat software co nejsnáze, i když je jeho práce omezována nějakým postižením.

Tato distribuce je vhodná pro notebooky, osobní počítače i servery. Je to moderní operační systém podporující širokou škálu hardwaru. Je založen na distribuci Debian, ze které převzal velké množství softwaru.

4.1.2 Linuxový ovladač CDC_ACM

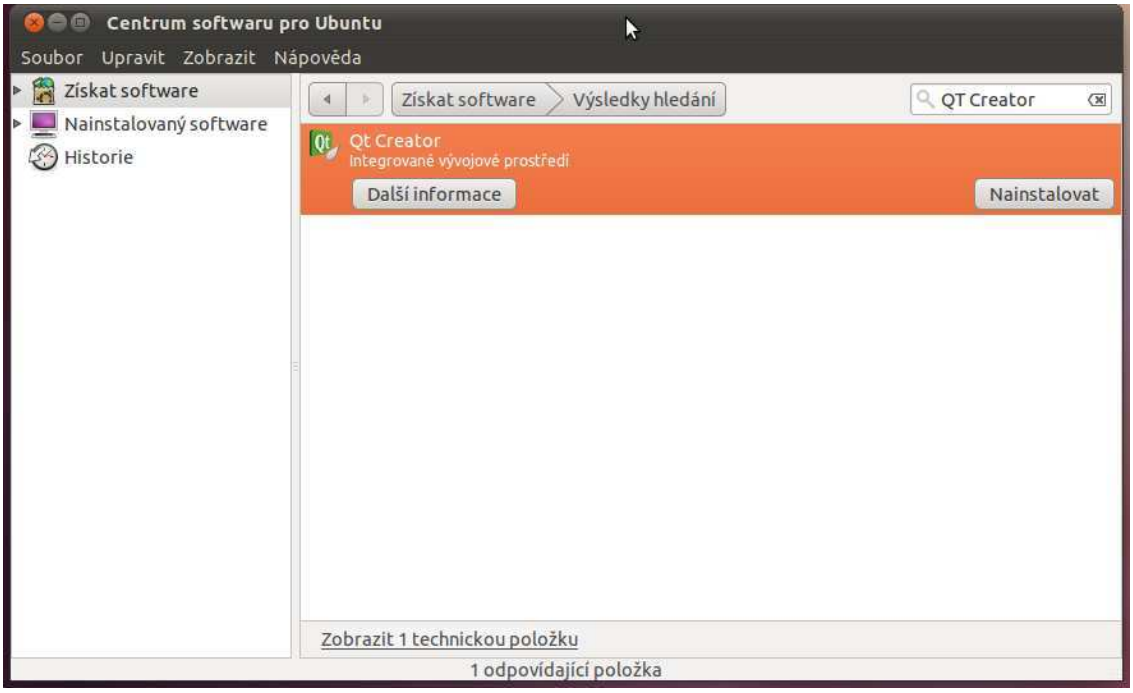
K použití bezdrátového USB accesss pointu pod Linuxem je potřeba doinstalovat ovladač CDC_ACM v0.16 vytvořený Vojtěchem Pavlíkem. Ten se naštěstí nainstaluje automaticky po připojení access pointu do USB. Jen je potřeba, aby byl počítač připojen k internetu, odkud si stáhne požadovaný ovladač.

Ovladač CDC_ACM je licencován jako GNU GPL. Ovladač je určen pro USB modemy a USB ISDN adaptéry.

4.1.3 Framework Qt Creator

Framework Qt Creator je multiplatformní vývojové prostředí. Tzn. aplikace vytvořená pomocí Qt lze používat pod různými platformami (MS Windows, Linux – jak PC tak i chytré telefony). Podporuje jazyk C++ a JavaScript. Integruje v sobě nástroj Qt Designer, který slouží k návrhu grafického rozhraní. Qt Creator je vyvíjen firmou Qt Development Frameworks, kterou vlastní známá společnost Nokia.

Qt Creator není součástí Ubuntu, proto je nutné jej nainstalovat. K nejjednodušší instalaci je potřeba internetového připojení a využití služeb integrované aplikace Centrum softwaru pro Ubuntu (Obr. 9).



Obr. 9: Centrum softwaru pro Ubuntu

Ve vyhledávacím okně stačí zadat „Qt Creator“ a aplikace zobrazí požadovaný výsledek. Kliknutím na položku Qt Creator se zobrazí tlačítko Nainstalovat. Po kliknutí na toto tlačítko se aplikace nainstaluje je připravena k použití.

4.2 APLIKACE

PC aplikaci jsem psal v textovém editoru vim [14] jazykem C++ [15]. Jazyk C++ je objektově orientovaný programovací jazyk, který vyvinul Bjarne Stroustrup s kolegy z Bellových laboratoří. Patří mezi nejrozšířenější jazyky. Zdrojový kód je pouze ve dvou souborech – *aplikace.cpp* a *aplikace.h*.

Pro kompilaci je potřeba zadat v terminálu příkazy:

```
qmake -project
qmake
make
```

První příkaz vygeneruje soubor *aplikace.pro*. Druhý příkaz z tohoto souboru vyrobí makefile (utilita pro automatický překlad zdrojových kódů). Poslední příkaz vytvoří spustitelný soubor, který se spouští samotnou aplikaci.

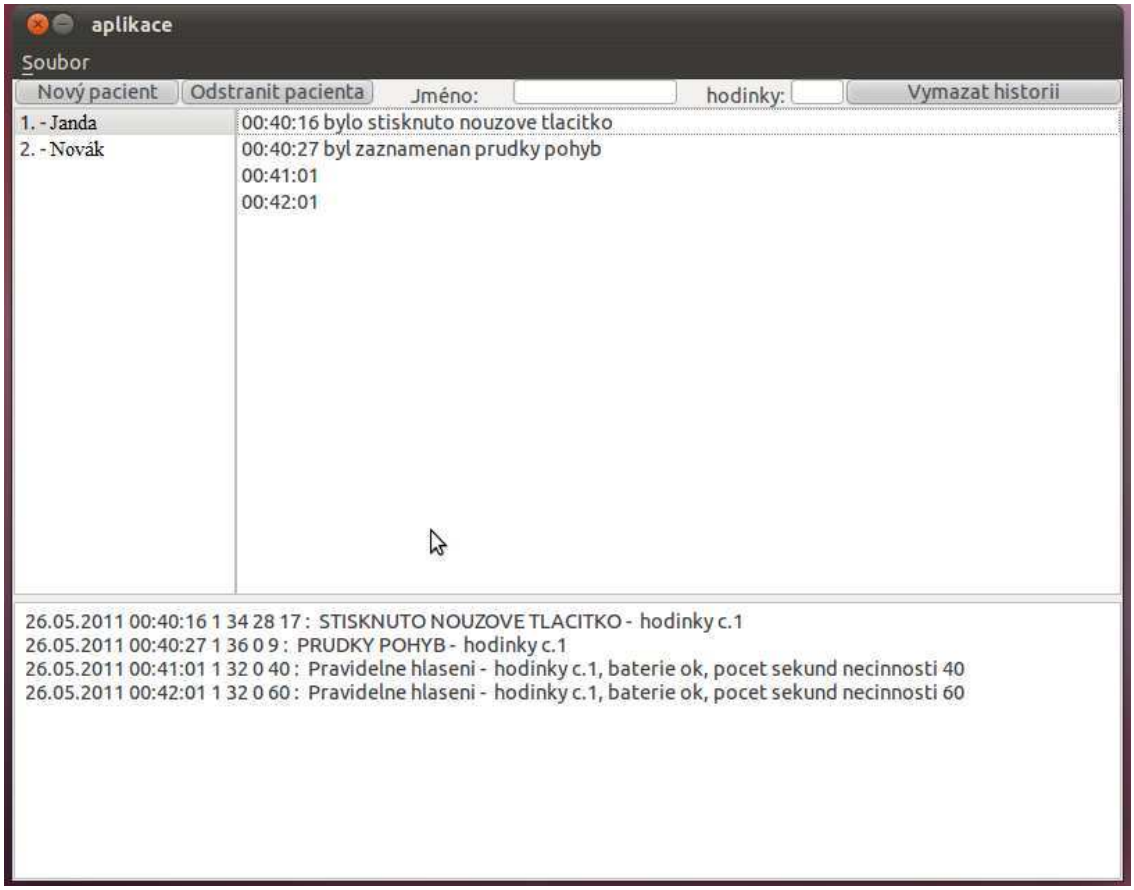
4.2.1 Popis aplikace

Před spuštěním aplikace je potřeba připojit k počítači USB access point. Po spuštění aplikace se otevře okno (Obr. 10). Aplikace neustále posílá každých 100 ms požadavky pro access point a čeká na jeho odpověď (viz kap. 2.3.1). Access point odpovídá na každý požadavek. Komunikace aplikace a access pointem vypadá takto:

```
; spuštění zařízení
PC        >   ff 01 03
přijímač  >   ff 06 03

; spuštění přijímače
PC        >   ff 07 03
přijímač  >   ff 06 03

; požadavek pro data (každých 100 ms)
PC        >   ff 08 07 00 00 00 00
; odpověď na požadavek (žádná data nejsou k dispozici)
přijímač  >   ff 06 07 ff 00 00 00
; odpověď na požadavek (k dispozici je pravidelné hlášení hodinek č.1)
přijímač  >   ff 06 07 01 00 01 FF
```



Obr. 10: Hlavní okno aplikace

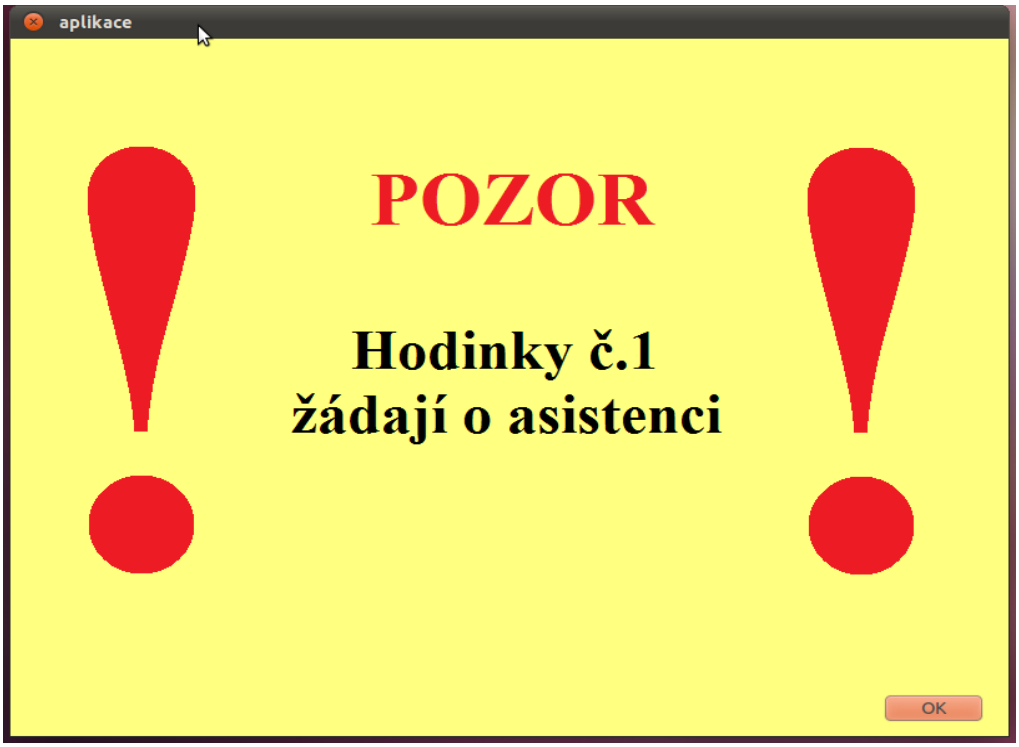
V menu Soubor se ukrývají dvě položky. První nese název O programu. Při kliknutí na něj se otevře nové okno, ve kterém se zobrazí zadání diplomové práce a jméno autora. Druhá položka Quit ukončí spuštěnou aplikaci. Ukončení aplikace lze dosáhnout též klávesovou zkratkou Ctrl+Q či kliknutím levým tlačítkem myši na červené kolečko s křížkem v levém horním rohu aplikace.

Tlačítko Nový pacient přiřazuje jméno k číslu hodinek (HID). Jméno a číslo se první musí vepsat do příslušných polí, až poté lze kliknout na tlačítko Nový pacient. Není ošetřeno přidávání různých jmen na stejné číslo hodinek. Tlačítko Odstranit pacienta toto přiřazení zruší. V největší části okna se zobrazují základní výpisy událostí. Ve spodní části je výpis podrobnější. Zobrazí se tam datum a čas příchozího signálu, tip události, stav baterie a dobu trvání nehybnosti hodinek. Tlačítko Vymazat historii tyto záznamy smaže.

Při detekci stisknutí nouzového tlačítka se otevře velké výrazné okno s výstražným textem (Obr. 11). Toto okno lze zavřít kliknutím na tlačítko OK nebo na červeně kolečko s křížkem v levém horním rohu. Při detekci záchvatu vyskočí obdobné okno, ale s jiným zněním textu: „Byl zaznamenán nezvyklý pohyb hodinek č. 1“.

Při zjištění slabé baterie hodinek vyskočí nové okno s textem: „Hodinky č. 1 potřebují vyměnit baterii“. Když se hodinky dvě hodiny nehlásí, tak se otevře nové okno s dialogem: „Hodinky č. 1 jsou už dlouho mimo dosah přijímače!“. Po zjištění, že jsou hodinky dvanáct hodin bez pohybu, tak se zobrazí nové okno s textem: „Hodinky č. 1 jsou už dlouho bez pohybu!“. Všechny informativní či varovná okna jsou ve skutečnosti jen obrázky. Použil sem je protože mám pouze jedny hodinky a obsah oken se proto nemění.

Nepodařilo se mi však aplikovat funkce odeslání SMS zprávy či emailu při stisknutí nouzového tlačítka či detekci záchvatu.



Obr. 11: Výstražné vyskakovací okno

4.2.2 Nejdůležitější zdrojové kódy

Hlavičkový soubor *aplikace.h* obsahuje jen definici knihovny Qt framevorku, její třídy a několik funkcí (viz příloha C).

Nejdůležitější částí zdrojového souboru *aplikace.cpp* je funkce send_command(), která má na starost komunikaci mezi aplikací a access pointem.

```
// vymazání bufferu
buffer_in_len=0;
reset(buffer_in,buffer_in_len+1);
buffer_out_len=0;
reset(buffer_out,buffer_out_len+1);
```

První část funkce smaže buffer access pointu.

```
// odeslání požadavku na port
portfile_out = fopen("/dev/ttyACM0","w");
for(a = 0; a < command_len; a++){
    fputc(command[a],portfile_out);
}
fclose(portfile_out);
```

Druhá část funkce posílá požadavek byte za bytem na port, který je značený v systému jako ttyACM0.

Poslední nejdelší část čte data z portu.

```
// čtení z portu

      b = 0;
      do{
            buffer_in_len = read(portfile_in,buffer_in,1024);
            usleep(200);
      }while( buffer_in_len <= 0 );
      if( buffer_in[3] > 0 && buffer_in_len == 7){
      if(  buffer_in[3] != 255){
          if(memcmp(buffer_in,last_buffer_in,1023) != 0){
            sprintf(response, "%i %i %i %i : ",
            buffer_in[3],buffer_in[4],buffer_in[5],buffer_in[6]);
            strcat(global_log, response);
            global_log_len+=strlen(response);
```

Zde se kontroluje, zda jsou příchozí data v pořádku. Jestli jsou, tak se uloží 4 byty signálu HIDSOSID k dalšímu zpracování. Pokud data nesplňují podmínku, tak jsou zahozena.

```
            switch(buffer_in[4] & 0x0F){

                  default:case 0:
                   if(((buffer_in[5]*256)+buffer_in[6]) < (20*60)){
                   if(buffer_in[4] & 64)
                     sprintf(response, " Pravidelne hlaseni - SLABA
BATERIE - hodinky c.%i, pocet sekund necinnosti %i", buffer_in[3],
((buffer_in[5]*256)+buffer_in[6]));
                    else
                     sprintf(response, " Pravidelne hlaseni -  hodinky
c.%i, baterie ok, pocet sekund necinnosti %i", buffer_in[3],
((buffer_in[5]*256)+buffer_in[6]));
                    }
                  break;

                  case 2:
                   if(buffer_in[4] & 64)
                     sprintf(response, " STISKNUTO NOUZOVE TLACITKO -
SLABA BATERIE - hodinky c.%i", buffer_in[3]);
                    else
                     sprintf(response, " STISKNUTO NOUZOVE TLACITKO -
hodinky c.%i", buffer_in[3]);
                  break;

                  case 4:
                   if(buffer_in[4] & 64)
                     sprintf(response, " PRUDKY POHYB - SLABA BATERIE
- hodinky c.%i", buffer_in[3]);
                    else
                     sprintf(response, " PRUDKY POHYB -  hodinky c.
%i", buffer_in[3]);
                  break;
}
```

Příkaz switch slouží k vyhodnocení stavu příchozích bytů.

Data hodinek jsou ukládána pomocí struktury:

```
typedef struct{
      char name[1024];
      int watch_id;
      FILE *logfile;
      int log_line_count;
      char log_lines[1024][1024];
      int inactivity_counter;
      int no_signal_counter;
      char battery;
} pacient;
```

Proměnná name typu char slouží k přiřazení jména k identifikačnímu číslu hodinek HID, které je ve formátu integer. Dále jsou tam čítače nedostupnosti signálu a nepohyblivosti hodinek.

Funkce MainWindow::update() se stará a aktualizaci zobrazeného okna.

5. FIRMWARE HODINEK

5.1 VÝVOJOVÉ PROSTŘEDÍ

Tak jako aplikace tak i firmware je vytvořený pomocí programů pod licencí GNU. K psaní zdrojových kódů firmwaru jsem opět využil textového editoru vim. Jako základ firmwaru hodinek jsem využil firmwaru OpenChronos. Ke kompilaci posloužil nástroj MSP430 GCC a nahrání binárních dat do hodinek MSP Debug.

5.1.1 MSP Debug

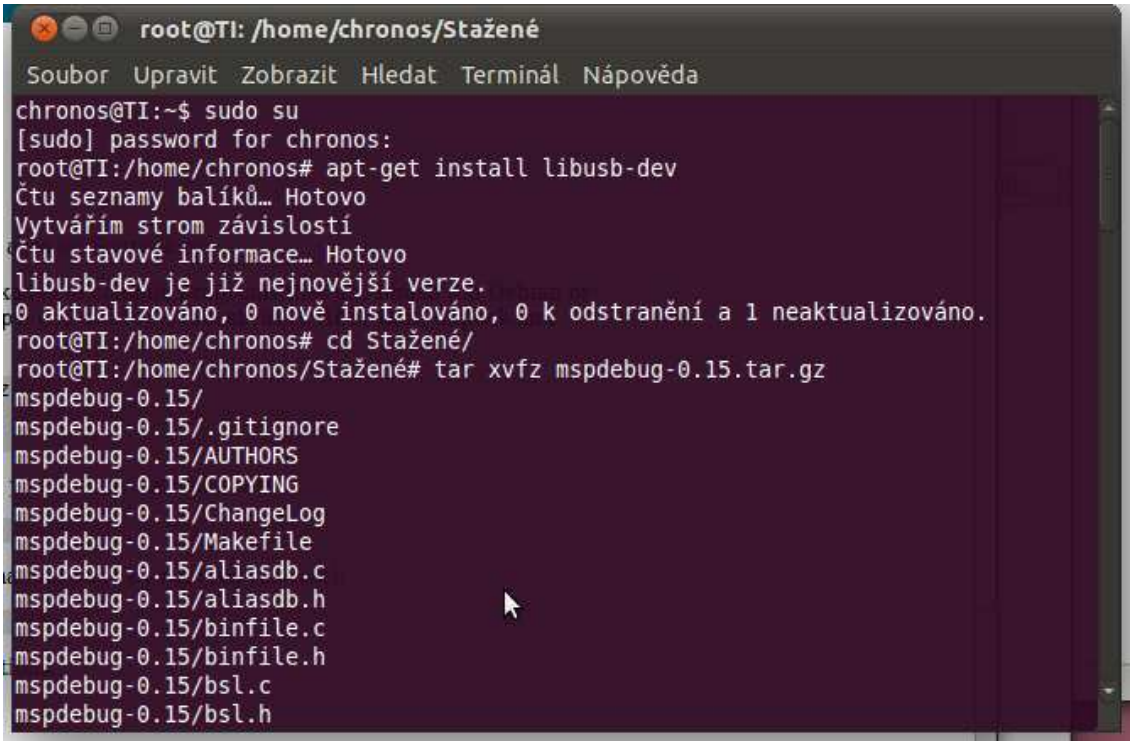
MSP Debug [13] je volně šiřitelný ladící program pro MCU MSP430. Podporuje eZ430, FET430UIF, RF2500 a Olimex MSP-JTAG-TINY. Jednou z mnoha jeho funkcí je i schopnost nahrání firmwaru do hodinek. Aktuální verze je 0.15.

Postup instalace je následnovný:

1. Z internetových stránek projektu [12] se stáhne archiv se zdrojovými kódy.
2. Spustí se konzole - Terminál (Obr. 12) a do ní vypisují následující příkazy:

```
sudo su
apt-get install libusb-dev
cd Stažené/
tar xvfz mspdebug-0.15.ta.gz
cd mspdebug-0.15
make WITHOUT_READLINE=1
make install
```

Příkaz „sudo su“ zapíná administrátorský režim, je požadováno heslo. Další příkaz nainstaluje balíček knihoven potřebných pro MSP Debug. Následující příkaz „cd Stažené/“ přepne pracovní do adresáře Stažené kam se stáhl archiv MSP Debug. Následujícím příkazem se obsah archívu extrahuje do adresáře mspdebug-0.15. Pak je zase nutné se přepnout do vytvořeného adresáře. Příkaz „make“ přeloží zdrojové kódy do spustitelných binárních souborů. Poslední příkaz tyto soubory nainstaluje.



Obr. 12: Konzole Linuxu - Terminál

5.1.2 MSP430 GCC

MSP430 GCC [12] je vývojový nástroj podporující rodinu čipů MSP430. Je šířen pod licencí GNU. Obsahuje v sobě několik utilit – kompilátor jazyka C do assembleru (GCC), linker (binutils) a ladící utilitu (GDB). Tyto nástroje mohou být použity v operačních systémech Windows (98SE a vyšší), Linux (jádro 2.4 a vyšší), BSD a většině odnoží Unixu. Poslední verze MSPGCC nese označení 3.2.3.

Instalace probíhá obdobným způsobem jako MSP Debug.

5.1.3 Firmware OpenChronos

OpenChronos [9] je firmware vytvořený pod licencí Open source. Firmware vytvořila komunita zajímající se o architekturu MSP430. Má více funkcí než originální firmware od Texas Instruments. Je více optimalizován a je efektivnější při práci s pamětí. Nemá podporu protokolu BlueRobin, protože to je uzavřený standard.

Na internetových stránkách projektu [9] jsou volně ke stažení zpřístupněny zdrojové kódy funkcí (soubor.c) a jejich hlavičky (soubor.h). Dle přípon je patrné, že

jsou tyto kódy psány v jazyku C. Při tvoření firmwaru hodinek jsem využil těchto zdrojových kódů.

5.1.4 Postup vytvoření firmware

Z celého balíku OpenChronos jsem více či méně upravil jen tyto zdrojové soubory: ezchronos.c, acceleration.c, clock.c, date.c a rfsimpliciti.c.

Z hlavičkových souborů jsem upravil následující: config.h, project.h, clock.h a fsimpliciti.h

V celém balíku OpenChronos se nachází 74 hlavičkových souborů a 59 zdrojových souborů.

Kompilace firmware se opět provádí pomocí terminálu. Postupně se zadají tyto příkazy:

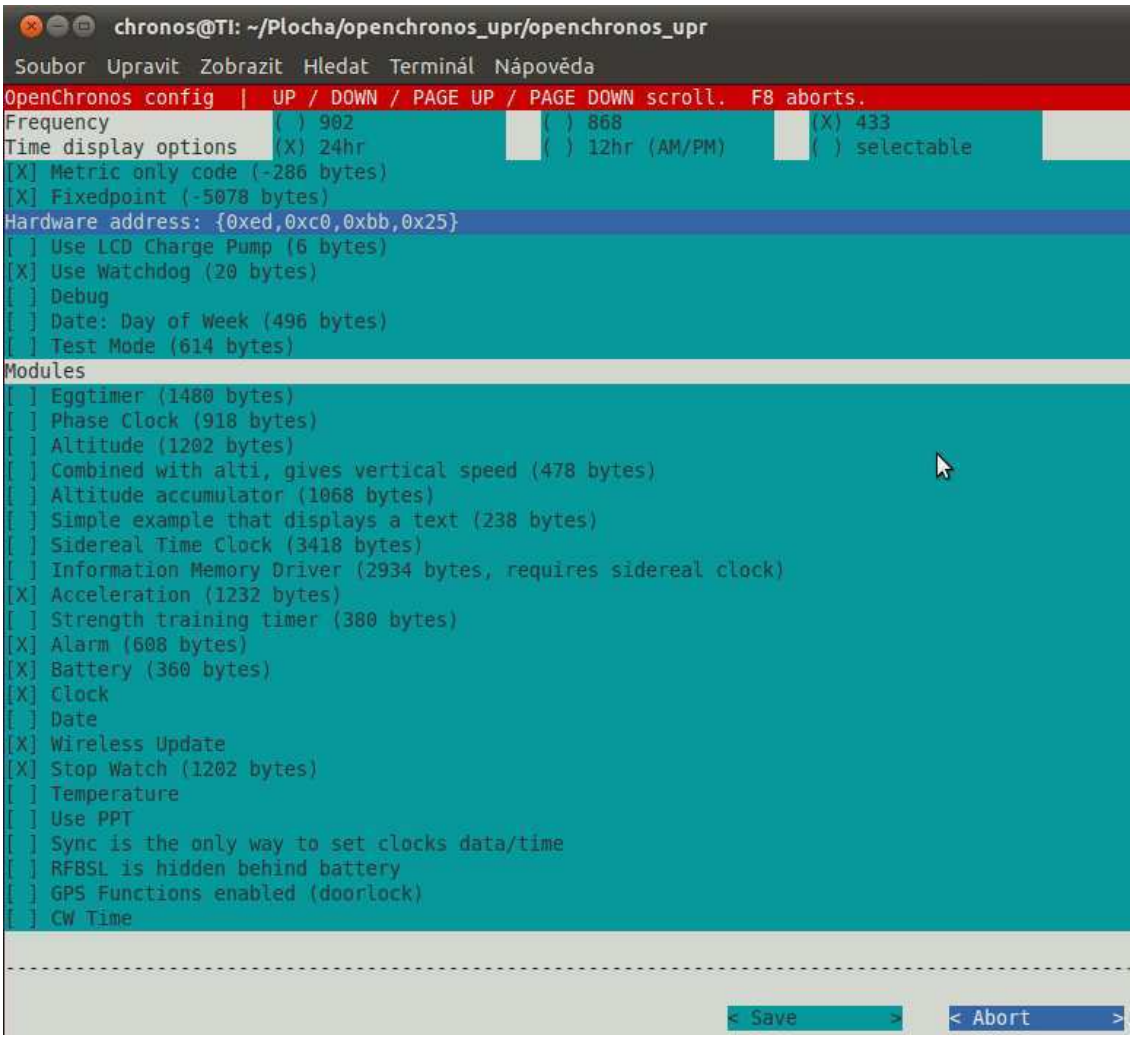
```
make config
make
./mspdebug rf2500 -reset
./mspdebug rf2500 -erase
./mspdebug rf2500 -load build\ezchronos.txt
./mspdebug rf2500 -run
./mspdebug rf2500 -exit
```

První řádek otevře nabídku (Obr. 13), ve které se zvolí základní parametry jako frekvence bezdrátového rozhraní nebo časový režim (12h nebo 24h). Také lze zapnout či vypnout jednotlivé funkce hodinek (měření a zobrazení teploty, stopky, budík, měření napětí na baterii, apod.).

Druhý řádek vytvoří binární soubor, který se později nahraje do hodinek. Cesta k tomuto souboru je build/ezchronos.txt.

Před vykonáním třetího příkazu je potřeba nejdříve připojit do USB počítače eZ430 Chronos debug interface (kap. 2.2) a k němu fyzicky připojit hodinky. Až teď se může vykonat třetí příkaz, který vyresetuje hodinky. Další příkaz smaže flash paměť hodinek. Následující příkaz nahraje vytvořený binární soubor ezchronos.txt do hodinek.

Předposlední příkaz zapne hodinky v ladícím a testovacím režimu. Po vykonání posledního příkazu se můžou hodinky od Chronos debug interface odpojit. V tuto chvíli jsou hodinky připraveny k použití.



Obr. 13: Hlavní nastavení knihoven OpenChronos

5.2 POPIS FIRMWARU HODINEK

Hodinky čtyřikrát za sekundu kontrolují jejich pohyb. Celé měření tvá asi 25ms. Pokud se během sekundy nezaznamená žádný pohyb tak se přičte do proměnné sTime.acc_delay_secs_a jednička. Tato hodnota se vynuluje vždy, když je zaznamenán pohyb nebo při odeslání signálu. Záchvat se detekuje porovnáním dvou posledních měření. Pokud je změna významně větší, tak se tento stav vyhodnotí jako záchvat a okamžitě se odešle požadovaný signál.

Měření napětí na baterii probíhá jednou denně. Pravidelné zasílání signálu se provádí po patnácti minutách. Vysílání probíhá na všech profilech krom Sync (slouží k synchronizaci času mezi hodinkami a PC) a acc (zobrazuje pohyb podle os X, Y a Z).

5.2.1 Nejdůležitější funkce firmwaru

Velmi důležitou funkcí je funkce send_signal(`u8 which`) nacházející se v souboru *ezchronos.c*.

```
void send_signal(u8 which){
      if((&menu_L1_Acceleration != ptrMenu_L1)
         && (&menu_L2_Rf != ptrMenu_L2) && (&menu_L2_Sync !=
ptrMenu_L2)   ){

      // Prepare radio for RF communication
      open_radio();

      // Set SimpliciTI mode
      sRFsmpl.mode = SIMPLICITI_SEND_MESSAGE;

      // Set SimpliciTI timeout to save battery power
      sRFsmpl.timeout = SIMPLICITI_TIMEOUT;

      // Start SimpliciTI stack. Try to link to access point.
      // Exit with timeout or by a button DOWN press.
      if (simpliciti_link())
      {
            #ifdef FEATURE_PROVIDE_ACCEL
            // Start acceleration sensor
            //as_start();
            #endif
            simpliciti_data[0]  = 0x01;                // id hodinek

            if(sys.flag.low_battery)
//baterie je slaba
              simpliciti_data[1] = 64;
            else                                  // baterie ok
              simpliciti_data[1] = 32;

              simpliciti_data[1] |= which;             // id stavu
```

```
            simpliciti_data[2]  = sTime.acc_delay_secs_a;
            simpliciti_data[3]  = sTime.acc_delay_secs_b;
            //simpliciti_data[2] = sAccel.xyz[0];
            //simpliciti_data[3] = sAccel.xyz[2];
          simpliciti_main_tx_only();
          sTime.acc_delay_secs_a = 0;
          sTime.acc_delay_secs_b = 0;
          sTime.move_count = 0;
     }

     // Set SimpliciTI state to OFF
     sRFsmpl.mode = SIMPLICITI_OFF;

     #ifdef FEATURE_PROVIDE_ACCEL
     // Stop acceleration sensor
     //as_stop();
     #endif

     // Powerdown radio
     close_radio();
     // Reload display
     display.flag.full_update = 1;
     }
}
```

Pokud je zapnutý požadovaný profil (krom Sync a acc) je povoleno zasílání dat. Zapne se bezdrátové rozhraní, načtou se požadovaná data do simpliciti_data[X] a tyto data se odešlou.

Další významnou funkcí je accel_time_diff(), která se vyskytuje v *clock.c* – viz příloha B. Tato funkce vyhodnocuje jestli je pohyb dost výrazný na to, aby se dal vyhodnotit jako záchvat.

6. ZÁVĚR

Cílem této diplomové práce bylo navrhnout a zrealizovat signalizační systém tvořený vývojovým kitem eZ430-Chronos-433 firmy Texas Instruments a osobním počítačem.

V práci jsem popsal vývojový kit se všemi jeho součástmi a uvedl jsem přehled spotřeby jednotlivých periferií. Na tomto základě jsem navrhl konkrétní řešení omezení spotřeby periferií hodinek Chronos a zpracovávat všechna získaná data pomocí aplikace na počítači. Navrhl jsem přenášet malé objemy dat jednou za 15 minut, ve kterých bude přenášená informace jen 4 byty dlouhá. Takto malý tok dat významně ušetří elektrickou energii z baterie. V případě stavu nouze se data odešlou okamžitě.

K realizaci projektu jsem využil jen programů a zdrojových kódů, které jsou šířeny volně pod licencí GNU.

V praktické části jsem uvedl podrobný postup vytvoření PC aplikace a firmwaru hodinek. Tento postup zahrnuje popis instalací potřebných programů, utilit a knihoven. Dále je uveden postup zadávání potřebných příkazů ke kompilaci zdrojových kódů do binárního souboru a jeho následujícího nahrání do hodinek.

Popsal jsem princip fungování PC aplikace i firmwaru hodinek. Uvedl jsem přehled základních funkcí a jejich zdrojových kódů. Podařilo se mi zrealizovat téměř vše dle zadání. Jen se mi nepodařilo uvést do funkční podoby zasílání upozornění pomocí SMS zpráv a emailu.

7. LITERATURA

[1] Populační prognóza ČR do r. 2050 [online] : Český statistický úřad, 11.6. 2004 Dostupné z WWW: <http://www.czso.cz/csu/2004edicniplan.nsf/t/B0001D6145/$File/4025rra.pdf>.

[2] EZ430-Chronos Development Tool User's Guide [online]. 2. vydání. [s.l.] : Texas Instruments, 2009, červen 2010 [cit. 2010-09-26]. Dostupné z WWW: <http://focus.ti.com/lit/ug/slau292b/slau292b.pdf>.

[3] CC430 Family User's Guide [online]. 2. vydání. [s.l.] : Texas Instruments, 2009, červen 2010 [cit. 2010-10-26]. Dostupné z WWW: <http://focus.ti.com/lit/ug/slau259b/slau259b.pdf>.

[4] CMA3000 Series Data Sheet [online]. 1. vydání. [s.l.] : VTI Technologies [cit. 2010-10-26]. Dostupné z WWW: <http://www.vti.fi/midcom-serveattachmentguid-1df5c00f2c48b705c0011df90f6fbf9599777447744/cma3000_d01_datasheet_8277800a.02.pdf>.

[5] SCP1000 Series Data Sheet [online]. 1. vydání. [s.l.] : VTI Technologies [cit. 2010-10-26]. Dostupné z WWW: <http://www.vti.fi/midcom-serveattachmentguid-9cbae6a382efd245cb62354a54ff62c7/scp1000-d01_-d11_pressure_sensor_datasheet_28-08-2007.pdf>.

[6] Texas Instruments [online]. 2011 [cit. 2011-05-20]. MSP-WDS430BT1000AD - Bluetooth Wearable Watch development system with Analog & Digital display. Dostupné z WWW: <https://estore.ti.com/MSP-WDS430BT1000AD-Bluetooth-Wearable-Watch-development-system-with-Analog-Digital-display-P2446.aspx>

[7] Texas Instruments [online]. 2011 [cit. 2011-05-20]. MSP-WDS430BT2000D - Bluetooth Wearable Watch development system with Digital display. Dostupné z WWW: <https://estore.ti.com/MSP-WDS430BT2000D-Bluetooth-Wearable-Watch-development-system-with-Digital-display-P2447.aspx>

[8] Metawatch [online]. 2011 [cit. 2011-05-20]. Watch Specifications. Dostupné z WWW: <http://www.metawatch.org/hardware.html>

[9] Github [online]. 2011 [cit. 2011-05-20]. Open Chronos. Dostupné z WWW: <https://github.com/poelzi/OpenChronos#readme>

[10] Wikipedia [online]. 2011 [cit. 2011-05-20]. Linux. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Linux>

[11] Ubuntu [online]. 2011 [cit. 2011-05-20]. Ubuntu. Dostupné z WWW: <http://www.ubuntu.cz/>

[12] Daniel Beer [online]. 2011 [cit. 2011-05-20]. The GCC toolchain for the Texas Instruments MSP430 MCUs. Dostupné z WWW: <http://mspgcc.sourceforge.net/>

[13] Daniel Beer [online]. 2011 [cit. 2011-05-20]. MSPDebug. Dostupné z WWW: <http://mspdebug.sourceforge.net/index.html>

[14] Wikipedia [online]. 2011 [cit 2011-05-21]. Vim. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Vim>

[15] Wikipedia [online]. 2011 [cit. 2011-05-21]. C++. Dostupné z WWW: <http://cs.wikipedia.org/wiki/C%2B%2B>

8. SEZNAM OBRÁZKŮ A TABULEK

Seznam použitých zkratek

BSD (Berkeley Software Distribution)	– odvozenina Unixu
CD (Compact Disc)	– kompaktní disk
COM	– sériový port
GNU (**G**NU's **B**ot **U**nix)	– projekt zaměřený na svobodný software
GPL (Genral Public License)	– licence pro svobodný software
GSM (Groupe Spécial Mobile)	– globální systém pro mobilní komunikaci
ISDN (Integral Services Digital Network)	– digitální síť integrovaných služeb
MS (Microsoft)	– společnost vyvíjející software
MCU (Microcontroller unit)	– mikrokontrolér
LAN (Local Area Nerwork)	– lokální síť
LPM (Low Power Mode)	– úsporný režim
MS (Microsoft)	– softwarová firma
PC (Personal Computer)	– osobní počítač
RISC (Reduced Instruction Set Computer)	– procesor s redukovanou instrukční sadou
SMS (Short message service)	– služba krátkých textových zpráv
SQL (Structured Query Language)	– strukturovaný dotazovací jazyk
UNIX	– obchodní značka
USB (Universal Serial Bus)	– univerzální sériová sběrnice

Seznam použitých symbolů

g	– tíhové zrychlení
V	– volt
$ (dollar)	– měna

Seznam příloh

A Uživatelský manuál

Hodinky mají pět tlačítek označených jako - * hvězdička, # křížek, ↑ šipka nahoru, ↓ šipka dolů a ☼ „sluníčko“. Tlačítko hvězdička má stříbrnou barvu. Všechny ostatní tlačítka černou.

Šipky slouží k přepínaní funkcí hodinek. Vrchní šipka ovládá vrchní část displeje a spodní tlačítko ovládá spodní část. Křížek a hvězdička slouží k aktivaci či změně nastavení funkce. Při všech profilech (krom Sync a acc) funguje automatické odesílání signálu.
Při stisku hvězdičky delším jak 4 sekundy odešlou hodinky nouzový signál.

B Rozdíly zdrojového kódu firmware oproti OpenChronos

význam znaků na začátku řádku:
! - řádek byl upraven
+ - řádek byl přidán

```
diff -cBr ./openchronos_orig/config.h ./openchronos_upr/config.h
*** ./openchronos_orig/config.h    2011-05-18 06:41:10.000000000 +0200
--- ./openchronos_upr/config.h     2011-05-23 01:13:55.000000000 +0200
***************
*** 3,31 ****
  #ifndef _CONFIG_H_
  #define _CONFIG_H_

! #define CONFIG_FREQUENCY 902
  #define OPTION_TIME_DISPLAY 0
! #define ONFIG_METRIC_ONLY 1
  #ifndef THIS_DEVICE_ADDRESS
  #define THIS_DEVICE_ADDRESS {0xed,0xc0,0xbb,0x25}
  #endif // THIS_DEVICE_ADDRESS
  // USE_LCD_CHARGE_PUMP is not set
  #define USE_WATCHDOG
  // DEBUG is not set
! #define CONFIG_DAY_OF_WEEK
! #define CONFIG_TEST
  // CONFIG_EGGTIMER is not set
  // CONFIG_PHASE_CLOCK is not set
! #define CONFIG_ALTITUDE
  // CONFIG_VARIO is not set
  // CONFIG_PROUT is not set
  #define CONFIG_ACCEL
  #define CONFIG_ALARM
  #define CONFIG_BATTERY
  #define CONFIG_CLOCK
! #define CONFIG_DATE
  #define CONFIG_RFBSL
  #define CONFIG_STOP_WATCH
! #define CONFIG_TEMP

  #endif // _CONFIG_H_
--- 3,41 ----
  #ifndef _CONFIG_H_
  #define _CONFIG_H_

! #define CONFIG_FREQUENCY 433
  #define OPTION_TIME_DISPLAY 0
! #define CONFIG_METRIC_ONLY
! #define FIXEDPOINT
  #ifndef THIS_DEVICE_ADDRESS
  #define THIS_DEVICE_ADDRESS {0xed,0xc0,0xbb,0x25}
  #endif // THIS_DEVICE_ADDRESS
  // USE_LCD_CHARGE_PUMP is not set
  #define USE_WATCHDOG
  // DEBUG is not set
! // CONFIG_DAY_OF_WEEK is not set
! // CONFIG_TEST is not set
```

```
  // CONFIG_EGGTIMER is not set
  // CONFIG_PHASE_CLOCK is not set
! // CONFIG_ALTITUDE is not set
  // CONFIG_VARIO is not set
+ // CONFIG_ALTI_ACCUMULATOR is not set
  // CONFIG_PROUT is not set
+ // CONFIG_SIDEREAL is not set
+ // CONFIG_INFOMEM is not set
  #define CONFIG_ACCEL
+ // CONFIG_STRENGTH is not set
  #define CONFIG_ALARM
  #define CONFIG_BATTERY
  #define CONFIG_CLOCK
! // CONFIG_DATE is not set
  #define CONFIG_RFBSL
  #define CONFIG_STOP_WATCH
! // CONFIG_TEMP is not set
! // CONFIG_USEPPT is not set
! // CONFIG_USE_SYNC_TOSET_TIME is not set
! // CONFIG_USE_DISCRET_RFBSL is not set
! // CONFIG_USE_GPS is not set
! // CONFIG_CW_TIME is not set

  #endif // _CONFIG_H_
Pouze v ./openchronos_upr/: .config.h.swp
Pouze v ./openchronos_upr/driver: .buzzer.c.swp
Binární soubory ./openchronos_orig/driver/ports.o a ./openchronos_upr/driver/ports.o jsou různé
Pouze v ./openchronos_upr/driver: .radio.c.swp
Binární soubory ./openchronos_orig/driver/timer.o a ./openchronos_upr/driver/timer.o jsou různé
Pouze v ./openchronos_upr/driver: vti_as.c~
Pouze v ./openchronos_upr/driver: .vti_as.c.swp
Pouze v ./openchronos_upr/driver: .vti_as.h.swp
Binární soubory ./openchronos_orig/driver/vti_ps.o a ./openchronos_upr/driver/vti_ps.o jsou různé
diff -cBr ./openchronos_orig/ezchronos.c ./openchronos_upr/ezchronos.c
*** ./openchronos_orig/ezchronos.c            2011-03-28 02:19:39.000000000 +0200
--- ./openchronos_upr/ezchronos.c  2011-05-25 12:48:46.000000000 +0200
***************
*** 433,438 ****
--- 433,498 ----
  }


+
+ void send_signal(u8 which){
+
+         if((&menu_L1_Acceleration != ptrMenu_L1)
+           && (&menu_L2_Rf != ptrMenu_L2) && (&menu_L2_Sync != ptrMenu_L2)   ){
+
+
+         // Prepare radio for RF communication
+         open_radio();
+
+         // Set SimpliciTI mode
+         sRFsmpl.mode = SIMPLICITI_SEND_MESSAGE;
+
+         // Set SimpliciTI timeout to save battery power
+         sRFsmpl.timeout = SIMPLICITI_TIMEOUT;
```

```
+
+       // Start SimpliciTI stack. Try to link to access point.
+       // Exit with timeout or by a button DOWN press.
+       if (simpliciti_link())
+       {
+               #ifdef FEATURE_PROVIDE_ACCEL
+               // Start acceleration sensor
+               //as_start();
+               #endif
+               simpliciti_data[0]  = 0x01;                     // id hodinek
+
+               if(sys.flag.low_battery)                //baterie je slaba
+                 simpliciti_data[1] = 64;
+               else                                                    // baterie ok
+                 simpliciti_data[1] = 32;
+
+           simpliciti_data[1] |= which;                        // id stavu
+           simpliciti_data[2]  = sTime.acc_delay_secs_a;
+           simpliciti_data[3]  = sTime.acc_delay_secs_b;
+           //simpliciti_data[2] = sAccel.xyz[0];
+           //simpliciti_data[3] = sAccel.xyz[2];
+               simpliciti_main_tx_only();
+               sTime.acc_delay_secs_a = 0;
+               sTime.acc_delay_secs_b = 0;
+               sTime.move_count = 0;
+       }
+
+       // Set SimpliciTI state to OFF
+       sRFsmpl.mode = SIMPLICITI_OFF;
+
+       #ifdef FEATURE_PROVIDE_ACCEL
+       // Stop acceleration sensor
+       //as_stop();
+       #endif
+
+       // Powerdown radio
+       close_radio();
+       // Reload display
+       display.flag.full_update = 1;
+       }
+ }
+
+
+
+
 //
*****************************************************************************************
*************
 // @fn          wakeup_event
 // @brief       Process external / internal wakeup events.
****************
*** 464,470 ****
                button.flag.star_long = 0;

                // Call sub menu function
!               ptrMenu_L1->mx_function(LINE1);
```

```
                    // Set display update flag
                    display.flag.full_update = 1;
--- 525,532 ----
                    button.flag.star_long = 0;

                    // Call sub menu function
!                   //ptrMenu_L1->mx_function(LINE1);
!                   send_signal(2);

                    // Set display update flag
                    display.flag.full_update = 1;
Pouze v ./openchronos_upr/: ezchronos.c~
Pouze v ./openchronos_upr/: .ezchronos.c.swp
Binární soubory ./openchronos_orig/ezchronos.o a ./openchronos_upr/ezchronos.o jsou různé
diff -cBr ./openchronos_orig/include/project.h ./openchronos_upr/include/project.h
*** ./openchronos_orig/include/project.h     2011-03-28 02:19:39.000000000 +0200
--- ./openchronos_upr/include/project.h      2011-05-23 03:41:40.000000000 +0200
***************
*** 75,80 ****
--- 75,87 ----
  #define CONV_MS_TO_TICKS(msec)                                  (((msec) * 32768) / 1000)


+
+ //
*****************************************************************************************
**************
+ //
+ // Prototypes section
+ extern void send_signal(u8 which);
+
+
  //
*****************************************************************************************
**************
  // Typedef section

Pouze v ./openchronos_upr/include: project.h~
Pouze v ./openchronos_upr/include: .project.h.swp
diff -cBr ./openchronos_orig/logic/acceleration.c ./openchronos_upr/logic/acceleration.c
*** ./openchronos_orig/logic/acceleration.c 2011-03-28 02:19:39.000000000 +0200
--- ./openchronos_upr/logic/acceleration.c    2011-05-25 14:17:21.000000000 +0200
***************
*** 279,282 ****
                    }
            }
  }
! #endif
\ Chybí znak konce řádku na konci souboru
--- 279,282 ----
                    }
            }
  }
! #endif
Pouze v ./openchronos_upr/logic: acceleration.c~
Pouze v ./openchronos_upr/logic: .acceleration.c.swo
Pouze v ./openchronos_upr/logic: .acceleration.c.swp
```

```
diff -cBr ./openchronos_orig/logic/acceleration.h ./openchronos_upr/logic/acceleration.h
*** ./openchronos_orig/logic/acceleration.h2011-03-28 02:19:39.000000000 +0200
--- ./openchronos_upr/logic/acceleration.h    2011-05-25 10:40:25.000000000 +0200
***************
*** 68,73 ****
--- 68,80 ----
          // Sensor raw data
          u8                          xyz[3];

+         // Sensor additional raw data
+         u8                          xyz_old[3];
+
          // Acceleration data in 10 * mgrav
          u16                         data;

Pouze v ./openchronos_upr/logic: acceleration.h~
Pouze v ./openchronos_upr/logic: .acceleration.h.swp
Binární soubory ./openchronos_orig/logic/acceleration.o a ./openchronos_upr/logic/acceleration.o jsou
různé
Binární soubory ./openchronos_orig/logic/altitude.o a ./openchronos_upr/logic/altitude.o jsou různé
Pouze v ./openchronos_upr/logic: .battery.c.swp
diff -cBr ./openchronos_orig/logic/clock.c ./openchronos_upr/logic/clock.c
*** ./openchronos_orig/logic/clock.c          2011-03-28 02:19:39.000000000 +0200
--- ./openchronos_upr/logic/clock.c2011-05-25 14:17:24.000000000 +0200
***************
*** 51,56 ****
--- 51,58 ----
  #include "menu.h"
  #include "clock.h"
  #include "user.h"
+ #include "acceleration.h"
+ #include "battery.h"

  //pfs
  #ifndef ELIMINATE_BLUEROBIN
***************
*** 108,117 ****
          sTime.system_time = 0;

          // Set main 24H time to start value
!         sTime.hour   = 4;
!         sTime.minute = 30;
          sTime.second = 0;

          // Display style of both lines is default (HH:MM)
          sTime.line1ViewStyle = DISPLAY_DEFAULT_VIEW;
          sTime.line2ViewStyle = DISPLAY_DEFAULT_VIEW;
--- 110,123 ----
          sTime.system_time = 0;

          // Set main 24H time to start value
!         sTime.hour   = 0;
!         sTime.minute = 0;
          sTime.second = 0;

+         sTime.acc_delay_secs_a = 0;
+         sTime.acc_delay_secs_b = 0;
```

```
+           sTime.tx_delay_minutes = 0;
+
            // Display style of both lines is default (HH:MM)
            sTime.line1ViewStyle = DISPLAY_DEFAULT_VIEW;
            sTime.line2ViewStyle = DISPLAY_DEFAULT_VIEW;
***************
*** 124,129 ****
--- 130,165 ----
            #endif
  }

+ void accel_time_diff() {
+
+           u8 axis;
+
+           sAccel.xyz_old[0] = sAccel.xyz[0];
+           sAccel.xyz_old[1] = sAccel.xyz[1];
+           sAccel.xyz_old[2] = sAccel.xyz[2];
+
+           as_get_data(sAccel.xyz);
+
+           for(axis=0;axis<3;axis++){
+
+             if(sAccel.xyz_old[axis] > sAccel.xyz[axis])
+               sTime.move[axis] = sAccel.xyz_old[axis] - sAccel.xyz[axis];
+             else
+               sTime.move[axis] = sAccel.xyz[axis] - sAccel.xyz_old[axis];
+
+             if(sTime.move[axis] > 0x7F){
+
+               if( (sAccel.xyz[axis] > 0x7f) && (sAccel.xyz_old[axis] < 0x7f)){
+                       sTime.move[axis] = sAccel.xyz_old[axis] + (0xff - sAccel.xyz[axis]);
+               }
+
+               if( (sAccel.xyz[axis] < 0x7f) && (sAccel.xyz_old[axis] > 0x7f)){
+                       sTime.move[axis] = sAccel.xyz[axis] + (0xff - sAccel.xyz_old[axis]);
+               }
+
+           }}
+ }
+


  //
*****************************************************************************************
*************
  // @fn          clock_tick
***************
*** 144,159 ****

            // Add 1 second
            sTime.second++;

            // Add 1 minute
!           if (sTime.second == 60)
            {
                    sTime.second = 0;
                    sTime.minute++;
```

```
                    sTime.drawFlag++;

                    // Add 1 hour
!                   if (sTime.minute == 60)
                    {
                            sTime.minute = 0;
                            sTime.hour++;
--- 180,257 ----

          // Add 1 second
          sTime.second++;
+         sTime.seconds_elapsed++;
+
+         if((&menu_L1_Acceleration
 != ptrMenu_L1)
+           && (&menu_L2_Rf != ptrMenu_L2) && (&menu_L2_Sync != ptrMenu_L2)   ){
+
+
+         // spusteni akcelerometru 4x za sebou
+         as_start();
+         sAccel.timeout = ACCEL_MEASUREMENT_TIMEOUT;
+         sAccel.mode = ACCEL_MODE_ON;
+
+         sTime.move_count = 0;
+
+         Timer0_A4_Delay(CONV_MS_TO_TICKS(20));
+         accel_time_diff();
+
+         if(sTime.move[0] > 25) sTime.move_count++;
+         Timer0_A4_Delay(CONV_MS_TO_TICKS(25));
+         accel_time_diff();
+
+         if(sTime.move[0] > 25) sTime.move_count++;
+         Timer0_A4_Delay(CONV_MS_TO_TICKS(25));
+         accel_time_diff();
+
+         if(sTime.move[0] > 25) sTime.move_count++;
+         Timer0_A4_Delay(CONV_MS_TO_TICKS(25));
+         accel_time_diff();
+
+         as_stop();
+
+
+         if(sTime.move[0] > 25) sTime.move_count++;
+         if(sTime.move_count > 2) send_signal(4);  // pohyb je prilis casty, zachvat?
+       sTime.move_count = 0;
+
+
+         if(sTime.move[0] < 50){
+
+                    sTime.move_count = 0;
+
+                    if(sTime.acc_delay_secs_b == 255 && sTime.acc_delay_secs_a < 254){
+                       sTime.acc_delay_secs_a++;
+                       sTime.acc_delay_secs_b = 0;
+                    }
+
```

```
+               sTime.acc_delay_secs_b++;
+
+       }else{
+               // aa konecne pohyb
+         sTime.acc_delay_secs_a = 0;
+         sTime.acc_delay_secs_b = 0;
+
+        }
+       }

        // Add 1 minute
!       if (sTime.second >= 60)
        {
                sTime.second = 0;
                sTime.minute++;
                sTime.drawFlag++;

+               sTime.tx_delay_minutes++;
+
+               if(sTime.tx_delay_minutes >= 15 ){
+
+                  send_signal(0);
+                  sTime.tx_delay_minutes = 0;
+               }
+
                // Add 1 hour
!               if (sTime.minute >= 60)
                {
                        sTime.minute = 0;
                        sTime.hour++;
**********
*** 163,168 ****
--- 261,267 ----
                        if (sTime.hour == 24)
                        {
                                sTime.hour = 0;
+                               battery_measurement();
                                add_day();
                        }
                }
Pouze v ./openchronos_upr/logic: clock.c~
Pouze v ./openchronos_upr/logic: .clock.c.swp
diff -cBr ./openchronos_orig/logic/clock.h ./openchronos_upr/logic/clock.h
*** ./openchronos_orig/logic/clock.h         2011-03-28 02:19:39.000000000 +0200
--- ./openchronos_upr/logic/clock.h         2011-05-25 14:09:10.000000000 +0200
**********
*** 74,80 ****
        u8              hour;
        u8              minute;
        u8              second;
!
        // Inactivity detection (exits set_value() function)
        u32     last_activity;
        #ifdef CONFIG_SIDEREAL
--- 74,94 ----
        u8              hour;
        u8              minute;
```

```
        u8              second;
!
!       // Pocet sekund od posledni zmeny polohy
!       u8              acc_delay_secs_a;
!       u8              acc_delay_secs_b;
!
!       // Pocet minut od posledni aktualizace
!       u8              tx_delay_minutes;
!
!       // Pocet pohybu nasledujicich za sebou
!       u8              move_count;
!       u8              move[3];
!
!       // Pocitadlo pro doserizovani hodinek
!       u8              seconds_elapsed;
!
        // Inactivity detection (exits set_value() function)
        u32     last_activity;
        #ifdef CONFIG_SIDEREAL
Pouze v ./openchronos_upr/logic: clock.h~
Pouze v ./openchronos_upr/logic: .clock.h.swp
Binární soubory ./openchronos_orig/logic/clock.o a ./openchronos_upr/logic/clock.o jsou různé
diff -cBr ./openchronos_orig/logic/date.c ./openchronos_upr/logic/date.c
*** ./openchronos_orig/logic/date.c          2011-03-28 02:19:39.000000000 +0200
--- ./openchronos_upr/logic/date.c  2011-05-22 21:19:55.000000000 +0200
***************
*** 87,93 ****
  void reset_date(void)
  {
        // Set date
!       sDate.year  = 2009;
        sDate.month = 8;
        sDate.day       = 1;

--- 87,93 ----
  void reset_date(void)
  {
        // Set date
!       sDate.year  = 2011;
        sDate.month = 8;
        sDate.day       = 1;

Pouze v ./openchronos_upr/logic: date.c~
Pouze v ./openchronos_upr/logic: .date.c.swp
Binární soubory ./openchronos_orig/logic/date.o a ./openchronos_upr/logic/date.o jsou různé
Pouze v ./openchronos_upr/logic: .menu.c.swp
Binární soubory ./openchronos_orig/logic/menu.o a ./openchronos_upr/logic/menu.o jsou různé
Pouze v ./openchronos_upr/logic: .MYsimpliciti.c.swp
diff -cBr ./openchronos_orig/logic/rfsimpliciti.c ./openchronos_upr/logic/rfsimpliciti.c
*** ./openchronos_orig/logic/rfsimpliciti.c  2011-03-28 02:19:39.000000000 +0200
--- ./openchronos_upr/logic/rfsimpliciti.c      2011-05-25 11:21:36.000000000 +0200
***************
*** 248,256 ****
        {
                simpliciti_data[0] = SIMPLICITI_KEY_EVENTS;
        }
!       simpliciti_data[1] = 0;
```

```
!           simpliciti_data[2] = 0;
!           simpliciti_data[3] = 0;

            // Turn on beeper icon to show activity
            display_symbol(LCD_ICON_BEEPER1, SEG_ON_BLINK_ON);
--- 248,263 ----
            {
                        simpliciti_data[0] = SIMPLICITI_KEY_EVENTS;
            }
!           else if (mode == SIMPLICITI_SEND_MESSAGE)
!           {
!                       simpliciti_data[0] = SIMPLICITI_MESSAGE_EVENTS;
!           }
!
!           if(mode != SIMPLICITI_SEND_MESSAGE){
!             simpliciti_data[1] = 0;
!             simpliciti_data[2] = 0;
!             simpliciti_data[3] = 0;
!           }

            // Turn on beeper icon to show activity
            display_symbol(LCD_ICON_BEEPER1, SEG_ON_BLINK_ON);
****************
*** 556,562 ****
                                    __no_operation();
                        }
            }
!
            // Update clock every 1/1 second
            if (display.flag.update_time)
            {
--- 563,603 ----
                                    __no_operation();
                        }
            }
+
+
+           if (sRFsmpl.mode == SIMPLICITI_SEND_MESSAGE) // send one-time message
+           {
+                       // New button event is stored in data
+                       if ((packet_counter == 0) && (simpliciti_data[0] & 0x0F) != 0)
+                       {
+                                   packet_counter = 5;
+                       }
+
+                       // Send packet several times
+                       if (packet_counter > 0)
+                       {
+                                   // Clear button event when sending last packet
+                                   if (--packet_counter == 0)
+                                   {
+                                               simpliciti_data[0] = 0;
+                                               simpliciti_flag |= SIMPLICITI_TRIGGER_STOP;
+                                   }
+                                   else
+                                   {
+                                               // Trigger packet sending in regular intervals
```

```
+                                       Timer0_A4_Delay(CONV_MS_TO_TICKS(30));
+                                       simpliciti_flag |= SIMPLICITI_TRIGGER_SEND_DATA;
+                               }
+                       }
+                       else
+                       {
+                               // Wait in LPM3 for next button press
+                               _BIS_SR(LPM3_bits + GIE);
+                               __no_operation();
+                       }
+               }
+
+
                // Update clock every 1/1 second
                if (display.flag.update_time)
                {
Pouze v ./openchronos_upr/logic: rfsimpliciti.c~
Pouze v ./openchronos_upr/logic: .rfsimpliciti.c.swo
Pouze v ./openchronos_upr/logic: .rfsimpliciti.c.swp
diff -cBr ./openchronos_orig/logic/rfsimpliciti.h ./openchronos_upr/logic/rfsimpliciti.h
*** ./openchronos_orig/logic/rfsimpliciti.h  2011-03-28 02:19:39.000000000 +0200
--- ./openchronos_upr/logic/rfsimpliciti.h     2011-05-23 02:15:58.000000000 +0200
***************
*** 64,69 ****
--- 64,70 ----
  #endif
    SIMPLICITI_BUTTONS,                 // Transmitting button events
    SIMPLICITI_SYNC,                    // Syncing
+   SIMPLICITI_SEND_MESSAGE,            // Send one-time signal
  #ifdef CONFIG_PHASE_CLOCK
    SIMPLICITI_PHASE_CLOCK_START, // Start new phase
    SIMPLICITI_PHASE_CLOCK,             // Phase Clock is running
***************
*** 79,88 ****
  #define SIMPLICITI_BUTTON_UP                          (0x30)

  // SimpliciTI mode flag
! #define SIMPLICITI_MOUSE_EVENTS           (0x01)
! #define SIMPLICITI_KEY_EVENTS             (0x02)
  #define SIMPLICITI_PHASE_CLOCK_EVENTS             (0x03)
  #define SIMPLICITI_PHASE_CLOCK_START_EVENTS (0x04)

  // notify the ap that sync mode started
  #define SIMPLICITI_SYNC_STARTED_EVENTS     (0x10)
--- 80,90 ----
  #define SIMPLICITI_BUTTON_UP                          (0x30)

  // SimpliciTI mode flag
! #define SIMPLICITI_MOUSE_EVENTS               (0x01)
! #define SIMPLICITI_KEY_EVENTS             (0x02)
  #define SIMPLICITI_PHASE_CLOCK_EVENTS             (0x03)
  #define SIMPLICITI_PHASE_CLOCK_START_EVENTS (0x04)
+ #define SIMPLICITI_MESSAGE_EVENTS             (0x05)

  // notify the ap that sync mode started
  #define SIMPLICITI_SYNC_STARTED_EVENTS     (0x10)
***************
```

C Zdrojové kódy PC aplikace

aplikace.cpp

```
// / / / / / / / / / / / / / /
//                          //
//    Hlavni aplikace       //
//                          //
// / / / / / / / / / / / / / /
//  autor : Bc. Josef Verbík
// / / / / / / / / / / / / / /

// Global
#include <pthread.h>

#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>

typedef struct{

        char name[1024];
        int watch_id;

        FILE *logfile;

        int log_line_count;
        char log_lines[1024][1024];

        int inactivity_counter;
        int no_signal_counter;

        char battery;

} pacient;

int number_of_pacients;
pacient *pacients[512];

typedef struct{

        int state;
        int batt;
        int seconds;

} watch_response;

watch_response *watch_responses[256][256];
int watch_response_count;


char global_log[32768];
int global_log_len;
char global_error_state;
```

```
pthread_t serialthread,networkthread;
pthread_mutex_t exclusion;
pthread_attr_t nastavenivlakna[2];

// GUI
#include "aplikace.h"

#include <QApplication>
#include <QMenuBar>
#include <QToolBar>
#include <QAction>
#include <QFileDialog>
#include <QDir>
#include <QFileInfo>
#include <QMessageBox>
#include <QListWidget>

#include <QPushButton>
#include <QMessageBox>
#include <QTextEdit>
#include <QPainter>
#include <QLabel>
#include <QLineEdit>

#include <QTimer>
#include <QSettings>
#include <QDateTime>

//#include <smtp.h>
//#include <smtp.cpp>

QTextEdit *logbox;
QLineEdit *pacient_name;
QLineEdit *watch_id;
QListWidget *pacients_list_box;
QListWidget *activity_list_box;

QSettings *settings;

int input_queue_len;
char input_queue[2048];

int output_queue_len;
char output_queue[2048];

// Serial communication

int portfile_in;
FILE *portfile_out;

int command_len;
char command[1024];

int response_len;
char response[1024];

int buffer_in_len;
```

```
unsigned char buffer_in[1024];
unsigned char last_buffer_in[1024];

int buffer_out_len;
unsigned char buffer_out[1024];

int a,b,c;
unsigned char lineinput[1024];
unsigned char charinput[128];

char minibuf[128];

char reset(char *input,int len) {
	int fdt;
	for (fdt = 0;fdt < len;fdt++) input[fdt] = 0;
	return 1;
}

char ureset(unsigned char *input,int len) {
	int fdt;
	for (fdt = 0;fdt < len;fdt++) input[fdt] = 0;
	return 1;
}

MainWindow::MainWindow() {

	QMenuBar* menuBar = new QMenuBar(this);
	QMenu* menuFile = new QMenu(tr("&Soubor"));

	QAction* actionQuit = new QAction(tr("&Quit"), this);
	QAction* actionAbout = new QAction(tr("O programu"), this);

	connect(actionQuit, SIGNAL(triggered()), this, SLOT(quit()));
	connect(actionAbout, SIGNAL(triggered()), this, SLOT(aboutProgram()));

	actionQuit->setShortcut(QKeySequence("Ctrl+Q"));

	menuFile->addAction(actionAbout);
	menuFile->addAction(actionQuit);
	menuBar->addMenu(menuFile);

	setMenuBar(menuBar);

	logbox = new QTextEdit(this);
	logbox->setFixedSize(798,199);
	logbox->move(1,400);

	QPushButton *add_pacient_button = new QPushButton("Nový pacient",this);
	add_pacient_button->setFixedSize(120,20);
	add_pacient_button->move(1,24);
	QPushButton *remove_pacient_button = new QPushButton("Odstranit pacienta",this);
	remove_pacient_button->setFixedSize(140,20);
	remove_pacient_button->move(121,24);

	QLabel *text1 = new QLabel("Jméno:", this);
	text1->move(287,22);
```

```
        pacient_name = new QLineEdit(this);
        pacient_name->setFixedSize(120,20);
        pacient_name->move(360,24);

        QLabel *text2 = new QLabel("hodinky:", this);
        text2->move(500,22);

        watch_id = new QLineEdit(this);
        watch_id->setFixedSize(40,20);
        watch_id->move(560,24);

        QPushButton *clear_history_button = new QPushButton("Vymazat historii",this);
        clear_history_button->setFixedSize(200,20);
        clear_history_button->move(600,24);

        pacients_list_box = new QListWidget(this);
        pacients_list_box->setFixedSize(160,350);
        pacients_list_box->move(1,45);

        activity_list_box = new QListWidget(this);
        activity_list_box->setFixedSize(638,350);
        activity_list_box->move(161,45);

        connect(add_pacient_button, SIGNAL(clicked()), this, SLOT(add_pacient()));
        connect(remove_pacient_button, SIGNAL(clicked()), this, SLOT(remove_pacient()));
        connect(clear_history_button, SIGNAL(clicked()), this, SLOT(clear_history()));
        connect(pacients_list_box, SIGNAL(itemSelectionChanged()), this, SLOT(pacient_click()));

        settings = new QSettings("chronos","hospital",this);

        number_of_pacients = settings->value("number_of_pacients").toInt();
        for(a = 0; a < number_of_pacients;a++){

            pacients[a] = (pacient *)malloc(sizeof(pacient));

       sprintf(minibuf,"pacient_%i_watch_id",a);
       pacients[a]->watch_id = settings->value(minibuf).toInt();

       sprintf(minibuf,"pacient_%i_name",a);
            strcpy(pacients[a]->name, settings->value(minibuf).toByteArray());

            printf("pridavam pacienta : '%s', hodinky %i . \n", pacients[a]->name, pacients[a]-
>watch_id);
            sprintf(minibuf, "%i. - %s ", pacients[a]->watch_id, pacients[a]->name);
      if(strlen(pacients[a]->name))pacients_list_box->addItem( minibuf );
        }

        QTimer *timer = new QTimer(this);
        connect(timer, SIGNAL(timeout()), this, SLOT(update()));
        timer->start(100);

        setFixedSize(800,600);
}

void MainWindow::add_pacient(){

    if( pacient_name->text().length() > 0 && watch_id->text().length()  ){
```

```
            pthread_mutex_lock(&exclusion);

            pacients[number_of_pacients] = (pacient *)malloc(sizeof(pacient));
        pacients[number_of_pacients]->watch_id = atoi(watch_id->text().toAscii());
            strcpy(pacients[number_of_pacients]->name, pacient_name->text().toAscii());
            printf("pridavam pacienta : '%s', hodinky %i . \n", pacients[number_of_pacients]->name,
pacients[number_of_pacients]->watch_id);

            number_of_pacients++;
            pthread_mutex_unlock(&exclusion);
        pacients_list_box->addItem( watch_id->text() + ". - " + pacient_name->text() );
    }
}

void MainWindow::remove_pacient(){
    if(pacients_list_box->currentItem()){
        pacients_list_box->removeItemWidget( pacients_list_box->currentItem() );
    }
}

void MainWindow::clear_history(){
            logbox->clear();
            activity_list_box->clear();
}

void MainWindow::pacient_click(){

}

void MainWindow::update(){

            int a,b;
            QDialog *alertDialog;
            QPushButton *close_dialog_button;
            //Smtp *mailer;
            QDateTime *cas;

            pthread_mutex_lock(&exclusion);

            for(a = 0; a < number_of_pacients; a++){
               pacients[a]->no_signal_counter++;

               if(pacients[a]->no_signal_counter > 72000){
                        alertDialog = new QDialog();
                        alertDialog->setFixedSize(800,600);
                        close_dialog_button = new QPushButton("OK",alertDialog);
                        close_dialog_button->setFixedSize(80,24);
                        close_dialog_button->move(700,564);
                        close_dialog_button->isDefault();
                        connect(close_dialog_button, SIGNAL(clicked()), alertDialog, SLOT(accept()));
                        alertDialog->setStyleSheet("QDialog { background-image :
url(Hlaska_dosah.png); }");
                        alertDialog->exec();
                        strcpy(minibuf,cas->currentDateTime().toString("hh:mm:ss").toAscii());
                        strcat(minibuf," pacient je prilis dlouho bez signalu");
                        activity_list_box->addItem( QString(minibuf) );
               }
```

```
            for(b = 0; b < watch_response_count;b++){
                    c = pacients[a]->watch_id;
                    if(watch_responses[c][b] != NULL){
                       pacients[a]->no_signal_counter = 0;
                       switch(watch_responses[c][b]->state){

                                case 0:
                                cas = new QDateTime();
                                cas->addSecs(watch_responses[c][b]->seconds);
                                activity_list_box->addItem(cas->currentDateTime().toString("hh:mm:ss"));

                                if(watch_responses[c][b]->seconds >= 15*60)
                                        pacients[a]->inactivity_counter++;
                                else
                                        pacients[a]->inactivity_counter = 0;

                                if(pacients[a]->inactivity_counter > 48){
                                alertDialog = new QDialog();
                                alertDialog->setFixedSize(800,600);
                                close_dialog_button = new QPushButton("OK",alertDialog);
                                close_dialog_button->setFixedSize(80,24);
                                close_dialog_button->move(700,564);
                                close_dialog_button->isDefault();
                                connect(close_dialog_button, SIGNAL(clicked()), alertDialog,
SLOT(accept()));
                                alertDialog->setStyleSheet("QDialog { background-image :
url(Hlaska_bez_pohybu.png); }");
                                alertDialog->exec();
                                strcpy(minibuf,cas->currentDateTime().toString("hh:mm:ss").toAscii());
                                strcat(minibuf," pacient je prilis dlouho bez pohybu");
                                activity_list_box->addItem( QString(minibuf) );
                                }
                                delete(cas);
                                break;

                                case 2:
                              //mailer = new
Smtp("josef.verbik@studentagency.cz","verbikj@seznam.cz","Hodinky","problem");
                                //delete mailer;
                                cas = new QDateTime();
                                alertDialog = new QDialog();
                                alertDialog->setFixedSize(800,600);
                                close_dialog_button = new QPushButton("OK",alertDialog);
                                close_dialog_button->setFixedSize(80,24);
                                close_dialog_button->move(700,564);
                                close_dialog_button->isDefault();
                                connect(close_dialog_button, SIGNAL(clicked()), alertDialog,
SLOT(accept()));
                                alertDialog->setStyleSheet("QDialog { background-image :
url(Hlaska_tlacitko.png); }");
                                alertDialog->exec();
                                strcpy(minibuf,cas->currentDateTime().toString("hh:mm:ss").toAscii());
                                strcat(minibuf," bylo stisknuto nouzove tlacitko");
                                activity_list_box->addItem( QString(minibuf) );
                                delete(cas);
                                break;
```

```
                        case 4:
                    //mailer = new
Smtp("josef.verbik@studentagency.cz","verbikj@seznam.cz","Hodinky","tres");
                        //delete mailer;
                        alertDialog = new QDialog();
                        cas = new QDateTime();
                        alertDialog->setFixedSize(800,600);
                        close_dialog_button = new QPushButton("OK",alertDialog);
                        close_dialog_button->setFixedSize(80,24);
                        close_dialog_button->move(700,564);
                        close_dialog_button->isDefault();
                        connect(close_dialog_button, SIGNAL(clicked()), alertDialog,
SLOT(accept()));
                        alertDialog->setStyleSheet("QDialog { background-image :
url(Hlaska_tres.png); }");
                        alertDialog->exec();
                        strcpy(minibuf,cas->currentDateTime().toString("hh:mm:ss").toAscii());
                        strcat(minibuf," byl zaznamenan prudky pohyb");
                        activity_list_box->addItem( QString(minibuf) );
                        delete(cas);
                        break;
                    }
                }}
            }

            watch_response_count = 0;

            if(global_log_len){
              cas = new QDateTime();
              logbox->append( cas->currentDateTime().toString(" dd.MM.yyyy hh:mm:ss ") +
QString(global_log) );
              reset(global_log, global_log_len);
              global_log_len = 0;
              delete(cas);
            }
            if(global_error_state < 0){

               QMessageBox msg;
               msg.setText("Nemohu najit USB prijimac!");
               msg.exec();
               exit(0);
            }
            pthread_mutex_unlock(&exclusion);
}
void MainWindow::aboutProgram(){

            QDialog aboutDialog;

            aboutDialog.setFixedSize(800,600);

            QPushButton *close_dialog_button = new QPushButton("OK",&aboutDialog);
            close_dialog_button->setFixedSize(80,24);
            close_dialog_button->move(700,564);
            close_dialog_button->isDefault();
            connect(close_dialog_button, SIGNAL(clicked()), &aboutDialog, SLOT(accept()));
```

```
	aboutDialog.setStyleSheet("QDialog { background-image : url(zadani.png); }");

	aboutDialog.exec();
}

void send_command(){

	// vymazu buffery
	buffer_in_len = 0;
	ureset(buffer_in,buffer_in_len+1);
	buffer_out_len = 0;
	ureset(buffer_out,buffer_out_len+1);

	// odeslani na port
	portfile_out = fopen("/dev/ttyACM0","w");
	for(a = 0; a < command_len; a++){
	   fputc(command[a],portfile_out);
	}
	fclose(portfile_out);

	// cteni z portu

	b = 0;
	do{
		buffer_in_len = read(portfile_in,buffer_in,1024);
		usleep(200);

	}while( buffer_in_len <= 0 );

	if( buffer_in[3] > 0 && buffer_in_len == 7){
	if(  buffer_in[3] != 255){
	   if(memcmp(buffer_in,last_buffer_in,1023) != 0){
		sprintf(response, "%i %i %i %i : ", buffer_in[3],buffer_in[4],buffer_in[5],buffer_in[6]);
		strcat(global_log, response);
		global_log_len+=strlen(response);

		switch(buffer_in[4] & 0x0F){

			default:case 0:
			 if(((buffer_in[5]*256)+buffer_in[6]) < (20*60)){
			 if(buffer_in[4] & 64)
			   sprintf(response, " Pravidelne hlaseni - SLABA BATERIE - hodinky c.%i,
pocet sekund necinnosti %i", buffer_in[3],((buffer_in[5]*256)+buffer_in[6]));
			 else
			   sprintf(response, " Pravidelne hlaseni -  hodinky c.%i, baterie ok, pocet
sekund necinnosti %i", buffer_in[3],((buffer_in[5]*256)+buffer_in[6]));
			 }
			break;

			case 2:
			 if(buffer_in[4] & 64)
			   sprintf(response, " STISKNUTO NOUZOVE TLACITKO - SLABA
BATERIE - hodinky c.%i", buffer_in[3]);
			 else
			   sprintf(response, " STISKNUTO NOUZOVE TLACITKO -  hodinky c.%i",
buffer_in[3]);
			break;
```

```
                        case 4:
                         if(buffer_in[4] & 64)
                           sprintf(response, " PRUDKY POHYB - SLABA BATERIE - hodinky c.%i",
buffer_in[3]);
                         else
                           sprintf(response, " PRUDKY POHYB -  hodinky c.%i", buffer_in[3]);
                        break;
                }
                //printf("%s",response);
                pthread_mutex_lock(&exclusion);
                if(watch_responses[buffer_in[3]][watch_response_count] == NULL)
                    watch_responses[buffer_in[3]][watch_response_count] = (watch_response
*)malloc(sizeof(watch_response));

                watch_responses[buffer_in[3]][watch_response_count]->state = (buffer_in[4] &
0x0F) ;
                watch_responses[buffer_in[3]][watch_response_count]->batt = (buffer_in[4] & 0xF0) ;
                watch_responses[buffer_in[3]][watch_response_count]->seconds =
((buffer_in[5]*256)+buffer_in[6]) ;
                watch_response_count++;

                strcat(global_log, response);
                global_log_len+=strlen(response);
                pthread_mutex_unlock(&exclusion);
            }else ureset(last_buffer_in,1023);
            memcpy(last_buffer_in,buffer_in,1023);


        }else{
          ureset(buffer_in,1024);
        }}
}

void *serial_main(void *params){


        usleep(1000000);
        portfile_in = open("/dev/ttyACM0", O_NOCTTY | O_NONBLOCK);

        if(!(portfile_in > 0)) {
           printf("Bez seriove linky nic neudelam.\n");
           global_error_state = -1;
           return NULL;
        }

        do{
           strcpy(command,"\xff\x01\x03");
           command_len = 3;
           send_command();

           strcpy(command,"\xff\x07\x03");
           command_len = 3;
           send_command();

           if(buffer_in_len && buffer_in[1] == 6 && buffer_in[2] == 3){
                pthread_mutex_lock(&exclusion);
```

```
                strcat(global_log,"Prijimac pripojen.\n");
                global_log_len = strlen(global_log);
                pthread_mutex_unlock(&exclusion);
            }else{
                global_error_state = -2;
                return NULL;
            }

            while(1){
                strcpy(command,"\xff\x08\x07\x00\x00\x00\x00");
                command_len = 7;
                send_command();
                usleep(8000);
            }


        }while(buffer_out_len);


        close(portfile_in);
        fclose(portfile_out);
        return NULL;
}

void MainWindow::quit(){

        int a,b;
        char minibuf[128];

        strcpy(command,"\xff\x09\x03");
        command_len = 3;
        send_command();

        usleep(10000);

        strcpy(command,"\xff\x09\x03");
        command_len = 3;
        send_command();

        usleep(10000);


        settings->setValue("number_of_pacients", number_of_pacients);
        for(a = 0; a < number_of_pacients; a++){
          sprintf(minibuf,"pacient_%i_watch_id",a);
          settings->setValue(minibuf, pacients[a]->watch_id);
          sprintf(minibuf,"pacient_%i_name",a);
          settings->setValue(minibuf, pacients[a]->name);
        }

        return qApp->quit();
}

int main(int argc, char** argv) {
        QApplication app(argc, argv);

        MainWindow w;
```

```
	w.show();

	sprintf(global_log, "Chronos Aplikace spustena.\n");
	global_log_len = strlen(global_log);

	pthread_create(&serialthread, &nastavenivlakna[0], serial_main, (void*)NULL);

	return qApp->exec();
}
```

aplikace.h

```
#include <QMainWindow>

class QDir;
class QMessageBox;
class QListWidget;
class QListWidgetItem;

class MainWindow : public QMainWindow
{
	Q_OBJECT

public:
	MainWindow();

private:
	QDir* m_dir;
	QListWidget* m_list;
	QString m_dName;

private slots:
	void add_pacient();
	void remove_pacient();
	void clear_history();
	void pacient_click();
	void update();
	void aboutProgram();
	void quit();
};
```