ZÁPADOČESKÁ UNIVERZITA V PLZNI
FAKULTA ELEKTROTECHNICKÁ

KATEDRA TECHNOLOGIÍ A MĚŘENÍ

BAKALÁŘSKÁ PRÁCE

Vzdálený monitoring životních funkcí pacienta s využitím GSM modulů na platformě Java ME – IMP-NG

Tomáš Brynda **2013**

Originál (kopie) zadání BP/DP

Abstrakt

Tato bakalářská práce je zaměřena na monitoring životních funkcí pacienta s využitím GSM modulů na platformě Java ME – IMP-NG.

Klíčová slova

GSM modul, Cinterion, TC65, Java, monitoring životních funkcí, saturace krve, teplota, puls, GPRS, MySQL, PHP, Javascript, HTML…

Abstract

This bachelor's thesis is focused on monitoring patient's life functions using GSM modules on platform Java ME – IMP-NG.

Key words

GSM module, Cinterion, TC65, Java, monitoring life functions, blood saturation, temperature, puls, GPRS, MySQL, PHP, Javascript, HTML…

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně s použitím odborné literatury a pramenů uvedených v seznamu, který je součástí této bakalářské práce.

Dále prohlašuji, že veškerý software, použitý při řešení této bakalářské práce, je legální.

.....

podpis

V Plzni dne 4.6.2013 Tomáš Brynda

Poděkování

Tímto bych rád poděkoval vedoucímu bakalářské práce Ing. Petru Kropíkovi, Ph.D. za cenné rady, připomínky a pomoc při řešení problémů.

Obsah

Seznam symbolů a zkratek

ME ..... Micro Edition

SE..... Standart Edition

GSM..... Globální systém pro mobilní komunikaci

DC..... Stejnosměrné napětí

ADC..... Analog Digital Converter

SIM ..... Subscriber Identity Module

ARM ..... Acorn RISC Machine

RAM ..... Random Access Memory

EEPROM ..... Electrically Erasable Programmable Read-Only Memory

TCP/IP ..... Transmission Control Protocol/Internet Protocol

PKI..... Public Key Infrastructure

HTTP ..... Hypertext Transfer Protocol

HTTPS ..... Hypertext Transfer Protocol Secure

UDP ..... User Datagram Protocol

FTP..... File Transfer Protocol

IMP ..... Information Module Profile

CLDC..... Connected Limited Device Configuration

Bd..... Baud, jednotka modulační rychlosti

Úvod

Tato bakalářská práce měla být původně řešena trochu jiným směrem. Naplánováno bylo vytvoření javovské aplikace, která bude snímat v pravidelných intervalech naměřená data z měřicí desky a kontrolovat stav pacienta, případně odesílat textové zprávy či prozvonit předem zvolené číslo při nestandardních naměřených parametrech.

Měřicí deska však bohužel od samého počátku nebyla funkční, byly na ní patrné některé zkraty mezi vodivými cestami, kapky cínu, odtržené součástky atd. Navíc při návrhu této desky nebyly dodrženy všechny potřebné technické postupy (objevovaly se zde např. vodivé cesty pod součástkami s integrovanými obvody).

S vedoucím práce, p. Ing. Kropíkem, jsme se dohodli, že by bylo lepší celkově pozměnit plošný návrh desky a případně ještě zmenšit její rozměry a vytvořit tak desku novou a spolehlivější, to však již bylo nad rámec rozsahu bakalářské práce. Zároveň se nám nedařilo sehnat potřebná čidla na měření oxidace krve a pulsu pacienta.

Rozhodl jsem se proto rozšířit tuto práci o server, aby byla data z modulu přístupná online. Naučil jsem se základy PHP, dynamického HTML a databází MySQL. Vytvořil jsem server, kam modul v pravidelných intervalech posílá naměřené hodnoty prostřednictvím GPRS, kde jsou uloženy do databáze, a také formuláře pro přihlášení uživatelů.

Naměřené hodnoty jsou online k dispozici příslušnému personálu nemocnice nebo i rodině pacienta.

Po přihlášení na internetové stránky může vidět uživatel grafy pacientovy teploty a pulsu. Grafy jsou generovány pomocí Google Chart Tools a zobrazovány pomocí Javascriptu.

1 mHealth

mHealth (neboli také mobile health) je oblastí elektronického zdravotnictví (eHealth) a jde o poskytování zdravotnických služeb a informací pomocí mobilních technologií (GSM moduly, mobilní telefony, PDA, ...). Využívá se zde řada bezdrátových technologií jako Bluetooth, GPRS/3G, nebo WiFi. V řešení, kterým se tato práce zabývá, je k přenosu dat využívána technologie GPRS.

Mobilní technologie jsou pro účely monitoringu stavu pacienta velmi vhodné. Díky současnému technologickému pokroku mají tato zařízení malé rozměry a přitom jsou dostatečně rychlá. Již není potřeba kabelů k přenosu informací ani k napájení. Zařízení jsou bez problémů přenosná a manipulace s nimi je značně jednodušší a úspornější z hlediska času.

Avšak i tato vyspělá zařízení mohou za určitých podmínek selhat. Jedním z problémů, které mohou nastat, je výpadek nebo přetížení sítě operátora. Modul pak není schopen podávat informace o stavu pacienta. Další problémy mohou nastat např. při mechanickém či elektrickém poškození zařízení.

2 Technická specifikace modulu Cinterion TC65

Cinterion TC65 podporuje 4 GSM pásma: 850/ 900/ 1800/ 1900 MHz. Napájecí napětí musí být v rozmezí od +8V do +30V DC. Při nečinnosti se modul přepne do tzv. Power save módu a šetří tak energii. Teplotní rozmezí při provozu je -30 až +65 °C, při +75 °C se modul automaticky vypne. Skladovací teplota je v rozmezí -40 až +85 °C. [1]

Modul je vybaven procesorem z řady ARM7, pamětí 400 kB RAM a 1,7 MB flash pro ukládání dat a aplikací. Nechybí ani zásobník TCP/ IP přístupný přes AT příkazy a podpora zabezpečeného protokolu HTTPS a PKI. Podporovány jsou také protokoly UDP, HTTP, FTP a emailové SMTP a POP3. Zajímavá je funkce OTAP, která umožňuje aktualizaci softwaru over-the-air neboli na dálku. [1]

Díky integrovanému slotu na SIM kartu lze také z modulu zahajovat nebo přijímat hovory a odesílat SMS. Po zaregistrování do sítě operátora lze také využívat připojení GPRS.

Pro přenos zvuku je možno použít GSM-FR (Full Rate), GSM-HR (Half Rate), GSM-EFR (Enhanced Full Rate) a zvukový kodek AMR. Je zde také možnost jednoduchého hands-free režimu a potlačení ozvěn a šumu. [1]

3 Jazyk Java

3.1 Historie jazyka Java

Jazyk Java dostal svoje jméno údajně podle jávské kávy, kterou programátoři často popíjeli při práci. Java byla vytvořena společností Sun Microsystems. Viceprezident této společnosti, Bill Joy, dal podnět k vytvoření nového jazyka, který by umožňoval stručný a efektivní zápis programů. V roce 1990 vznikl Green Team vedený Jamesem Goslingem. Tato malá vývojová skupina společnosti Sun si vytyčila za svůj cíl vytvoření systému pro domácí spotřebiče. Po dobu dvou let pracovala odděleně od zbytku světa na něčem, co by mohlo vnést nový život do světa informačních technologií. [2]

Green Team nejprve používal programovací jazyk C++. Ale ani on, ani jeho modifikace však nebyly shledány vhodnými. Stejná situace nastala při pokusu o „oživení“ jazyka Pascal, jehož autorem je Niklaus Wirth (USCD Pascal). Nakonec James Gosling navrhl nový jazyk, který pojmenoval Oak podle stromu, jež rostl před okny jeho kanceláře, a ze kterého se postupně vyvinula Java, jak ji známe dnes. Za přímého předchůdce jazyka Oak můžeme považovat C++. [2]

Koncová zařízení se osazují různými procesory podle požadovaného výkonu, a proto byl nově navržený jazyk interpretovaný, tj. nezávislý na konkrétním hardware. Požadavky na paměť pro základní interpret jsou 40 kB a pro standardní knihovny s podporou více vláken 175 kB. [2]

3.2 Java výbava modulu TC65

Co se týče Java výbavy, k dispozici máme CLDC 1.1 HI, J2ME s IMP 2.0.

CLDC je určen především pro mobilní telefony či PDA s malou pamětí. Aplikace v CLDC konfiguraci se nazývají MIDlety (hlavní třída je potomkem abstraktní třídy javax.microedition.midlet.MIDlet). V současné době drtivé množství výrobců mobilních zařízení implementuje CLDC ve svých výrobcích. Paměťová náročnost je 160 kB ROM a 32 kB RAM. [3]

CLDC 1.1 obsahuje tyto balíčky [3]:

- *java.io - poskytuje vstupní a výstupní datové proudy*
- *java.lang - poskytuje třídy nutné pro návrh v programovacím jazyce Java*
- *java.lang.ref – podpora slabých referencí*
- *java.util - obsahuje framework kolekcí, např. nástroje pro usnadnění práce s datem a časem*
- *javax.microedition.io - třídy pro správu všeobecného spojení*

IMP spadá pod CLDC konfiguraci, je to profil pro jednoduchá zařízení, buď s jednoduchým displejem nebo bez displeje, s omezenou možností připojení k síti, většinou prodejní automaty nebo jednoduché bezpečnostní systémy. [3]

Profil IMP je úzce spojen s profilem MIDP (jak 1.0, tak i následným 2.0). Je to v podstatě jeho podmnožina, omezená o možnost tvorby uživatelského rozhraní (především o balíček javax.microedition.lcdui). [3]

IMP přidává k CLDC ještě tyto balíky [3]:

- *javax.microedition.rms - správa persistence dat*
- *javax.microedition.midlet - třída MIDlet, která je základní třídou MIDP profilu*
- *javax.microedition.io - zejména třída HttpConnection*

4 Tvorba aplikace a její nahrání do modulu

4.1 Instalace Javy a vývojového prostředí

Abychom vůbec mohli vytvořit nějakou aplikaci, potřebujeme do našeho počítače nainstalovat nezbytné součásti.

Prvním klíčovým prvkem je nainstalování Java Development Kitu (JDK). V nabídce je JDK v nejrůznějších variantách, jako například JDK SE (Standart Edition), ME (Micro Edition), která je vhodná právě pro programování embedded modulů a mobilních aplikací, a ještě několik dalších. Jde o balík knihoven a ostatních souborů, které jsou nezbytné pro vývoj aplikací.

Abychom mohli tyto aplikace spouštět a testovat, potřebujeme k tomu další nástroj - Java Runtime Environment (JRE). Pokud máme úspěšně nainstalovány tyto dva klíčové prvky v našem systému, je potřeba si vybrat vývojové prostředí, ve kterém budeme aplikace psát. Mezi nejrozšířenější patří Eclipse a NetBeans, které jsem při programování používal já.

4.2 Nastavení prostředí a přidání platformy pro Cinterion

Poté, co máme nainstalované všechny komponenty včetně vývojového prostředí, chybí už jen Cinterion Mobility Toolkit (CMTK). Jeho součástí je Wireless Toolkit (WTK) pro TC65 a ovladač pro modul. Vše je zahrnuto na instalačním disku. [4]

Náš TC65 se pak objeví v nabídce *Tento počítač* jako nová disková jednotka A. Poklepáním se otevře a naskytne se nám možnost nahrát sem naši budoucí aplikaci.

Dále musíme v prostředí NetBeans nebo Eclipse přidat novou platformu pro náš modul. K tomu se dostaneme přes menu *Tools/ Java Platforms* a poté klikneme v dialogovém okně na tlačítko *Add Platform*. Zde stačí zvolit Custom Java ME MIDP Platform Emulator a pokračujeme dál. Jako Platform Home najdeme pomocí průvodce složku s WTK. Jako Platform Name si můžeme zvolit jakýkoli název, např. Cinterion. To samé platí i o políčku Device Name. Proklikáme se zbytkem Platform Managera a nová platforma by měla být vidět pod složkou J2ME. [4]

Nyní máme vše připravené a můžeme se dát do vývoje první aplikace. Kliknutím na *File*/ *New Project* v hlavní nabídce založíme nový projekt a jako platformu vybereme náš Cinterion.

Hlavní třída našeho projektu musí být potomkem třídy MIDlet a musí mít nadeklarovány metody startApp(), pauseApp() a destroyApp().

Když máme aplikaci hotovou, zkompilujeme ji volbou Build Project. Poté do modulu zkopírujeme soubory s příponou .jad a .jar, které se nachází v adresáři naší aplikace ve složce dist.

Soubor .jar je vlastní zkompilovaná aplikace, která se následně spouští.

Soubor .jad je něco jako informační soubor k souboru .jar. Obsahuje tyto údaje:

- *název projektu*
- *název aplikace*
- *název hlavní třídy*
- *velikost aplikace v bajtech*
- *jméno autora*
- *číslo verze*
- *MicroEdition konfigurace (např. CLDC- 1.1)*
- *MicroEdition profil (např. MIDP- 2.0)*

Aplikaci nahranou v modulu spustíme pomocí AT příkazu AT^SJRA, kde jako parametr nastavíme jméno zkompilované aplikace.

Další možností je využít funkce AutoStart, která spouští aplikaci po uplynutí určitého časového intervalu po inicializaci modulu. Zde je třeba nastavit jméno aplikace a prodlevu spuštění v desetinách sekundy (je lepší dát větší kvůli případným problémům). Více o AT příkazech viz následující část nebo oficiální dokumentace k AT příkazům.

4.3 AT příkazy a jejich struktura

AT příkazy slouží k nastavení modulu, zapínání/ vypínání jeho funkcí, nastavení parametrů a také k obsluze SIM karty.

Každý AT příkaz začíná písmeny AT, dále obsahuje většinou jeden ze znaků +, &, nebo ^, podle typu příkazu. Za těmito znaky pak následuje samotný příkaz (např. AT^SCFG, AT+IPR, AT&F). Přikazy se dají otestovat přidáním znaků =? (AT^SCFG=?). Odpověď modulu pak bude obsahovat výpis použitelných parametrů a jejich atributů následovanými potvrzením o správném zpracování příkazu OK. V případě chybné syntaxe nebo jiného důvodu pro selhání příkazu je vrácen řetězec ERROR. Chybové hlášky lze vypisovat detailněji pomocí příkazu AT+CMEE=2. Modul pak přidá do odpovědi popisek chybové hlášky nebo v případě nastavení hodnoty 1 kód chyby.

Když zrovna nepotřebujeme příkaz testovat, můžeme přečíst současné nastavení vynecháním znaku = (např. AT^SCFG?). Pro zápis se použije znak = následovaný parametry oddělenými čárkou. Textové řetězce by se měly zadávat ve dvojitých uvozovkách. Chceme-li povolit automatické spuštění programu po zapnutí terminálu, použijeme příkaz AT^SCFG=”Userware/Autostart”,””,1. Prázdné uvozovky jsou tam v případě, že není nastavené bezpečnostní heslo. Jednička znamená stav zapnuto, nula stav vypnuto.

AT příkazy se dají provádět přímo z javovské aplikace. Vytvoříme novou instanci typu ATCommand a použijeme metodu send(), která má jako parametr řetězec s příkazem. Zde jsem narazil na problém, že se žádný příkaz nevykonal. Bylo to způsobeno tím, že jsem na konec příkazového řetězce nepřidal ukončovací znak \r. Je dobré si napsat jednoduchou metodu na odesílání AT příkazů a automaticky tento znak přidávat na konec řetězce (viz Třída AT).

Další variantou je použití obslužného terminálu, např. putty. Je třeba si uvědomit, že pokud komunikujeme s modulem přes terminál, nemůžeme přistupovat k nahraným souborům v modulu, terminál obsadí sériovou linku. Další poměrně důležitá věc je, že pokud napíšeme příkaz do terminálu špatně a smažeme pár znaků užitím klávesy BackSpace, příkaz vrátí

ERROR i v případě následně správné syntaxe. Je to tím, že se i tento znak odešle přes sériovou linku a způsobí pak chybu v syntaxi.

Na *Obr. 1.1* je příklad komunikace přes terminál putty. Je zde vidět použití příkazu AT+IPR, který se používá na nastavení rychlosti komunikace mezi modulem a druhou stranou (v tomto případě stolní počítač). Testovací příkaz vrátí výčet dostupných módů přenosových rychlostí. Dále následuje ukázka čtecího příkazu, který vrátí aktuální nastavenou hodnotu. Poslední je přiřazovací příkaz, nastavena byla rychlost 9600 Bd.

```
AT+IPR=?
+IPR: (1200,2400,4800,9600,14400,19200,28800,38400,57600,115200,230400,460800),(
0,300,600,1200,2400,4800,9600,14400,19200,28800,38400,57600,115200,230400,460800
)

OK
AT+IPR?
+IPR: 115200

OK
AT+IPR=9600
OK
```

Obr. 1.1 Příklad komunikace přes terminál putty

4.4 Nahrání aplikace do modulu

Poté, co máme hotový projekt, je třeba ho zkompilovat. Kompilace se v NetBeans provádí tlačítkem Build v liště hlavní nabídky. Poté jsou do složky dist, která se nachází v adresáři projektu, vytvořeny soubory s příponou .jar a .jad. Tyto soubory jsou nezbytné ke spuštění aplikace v samotném modulu. Tyto soubory zkopírujeme do modulu (jednotka A:) prostřednictvím Windows Exploreru nebo jiného správce souborů. Aby se aplikace spustila, je nutné buď otevřít terminál a odeslat AT příkaz AT^SJRA se jménem souboru jako parametrem, nebo nastavit funkci Autostart a zde také uvést název souboru.

5 Měřicí deska

Hlavní logickou jednotkou je mikrokontrolér ATmega8L. Jedná se o CMOS 8-bitový nízko příkonový mikrokontrolér založený na AVR RISC architektuře. Tento mikrokontrolér, pomocí spouštění výkonných instrukcí v jednom hodinovém cyklu, dosahuje rychlosti 1MIPS/MHZ. [5]

Vlastnosti ATmega8L: [5]

- *130 výkonných instrukcí*
- *32x8 registrů*
- *Až 16MIPS při 16MHz*
- *8 KB In-System Self-Programmable flash paměti*
- *512 B EEPROM*
- *1 KB vnitřní SRAM*
- *Zápisové/Mazací cykly*
- *10 000 u paměti Flash*
- *100 000 u paměti EEPROM*

Uchování dat:

- *20 let při 85ºC*
- *100 let při 25ºC*

Periférie:

- *Dva 8bitové časovače/čítače*
- *Jeden 16bitový časovač/čítač*
- *3 PWM kanály*
- *8mi kanálový ADC s 10bitovým rozsahem*
- *Programovatelný USART*
- *Master/Slave SPI*

Deska obsahuje také několik čidel- akcelerometr, teploměr, čidlo tepu a saturace kyslíkem.

Použitý akcelometr je od firmy Freescale Semiconductors. Jedná se o kapacitní mikromechanický akcelerometr s možností nastavení citlivostí a tepelnou kompenzací [5]. Tento akcelerometr může snímat pohyb ve všech třech osách. Nám bude stačit pro registraci

pádu pouze osa z. Je zde však potřeba, aby zařízení bylo stále ve stejné poloze. Vyřešit se to dá například uchycením páskem k pasu pacienta.

Poté je důležité odlišit stav, kdy si člověk pouze sedne a kdy se již jedná o pád. Stoprocentně to odlišit nejde, proto je použita metoda „více falešných poplachů“, kdy je mez nastavena níže s tím, že občas dochází k zareagování i na falešný impulz, ale představuje jistotu zachycení případného pádu. Z tohoto důvodu je také připojeno tlačítko pro reset zařízení. Falešný impulz může vzniknout například rychlým sednutím pacienta. [5]

Čidlo tepu a saturace je ve formě náprstku. Jde o neinvazivní metodu (nedochází k průniku kůží). Senzor se umístí na úzkou část na těle – prst nebo ušní lalůček. Je vysíláno záření o dvou vlnových délkách (první v pásmu 600 – 750 nm, druhé v infračervené oblasti 850 – 1000 nm) a na druhé straně je přijímáno fotodetektorem. Změna absorpce každé vlnové délky je měřena – z toho údaje je vyhodnocena pulzace arteriální krve. Pokud je hladina kyslíku v krvi nízká, absorbuje více světla (v nižším pásmu), ale prochází více infračerveného záření. Po průchodu signálu je spočten poměr záření a výsledek je poté vybrán z implementované tabulky (většinou si dělá každý výrobce sám na základě měření). Typické hodnoty jsou od 95 do 99 % (pro lidi s chronickou obstrukční plicní nemocí jsou hodnoty od 88 do 95 %). Přídavné absorpce jako kůže, kosti, žíly jsou stále přítomny a jejich hodnota se nemění, proto se dají odečíst. V případě tepu srdce se na okamžik zvýší objem krve v místě měření a to se projeví ve změně absorpce. Jediný faktor ovlivňující měření je podchlazení – pokud je člověk podchlazen, nemusí být vůbec možné měření provést (z důvodu nedostatečného prokrvení periferií). Zařízení komunikuje po rozhraní UART a posílá data do modulu jednou za sekundu. [5, 7]

Jako teploměr byl vybrán termistor NTC640-4k7. Jak je již vidět z názvu, jedná se o termistor s negativní teplotní závislostí, což znamená, že se stoupající teplotou klesá odpor. Jedná se o termistor od společnosti Vishay BCcomponents. [5]

Parametry: [5]

- *Teplotní rozsah: -40ºC až 125ºC*
- *Odpor při 25ºC: 4700 Ω*
- *Tolerance: ±3%*
- *Rozptylový faktor: 7mW/K*
- *Negativní závislost odporu na teplotě*

Měřicí deska komunikuje hlavně přes sériový port, ale v případě poruchy se dá komunikace přesměrovat na SPI piny. Data jsou odesílána každou sekundu. Jsou ohraničena závorkami <, > a obsažená data jsou oddělena čárkami. V případě správné komunikace by měl být datový řetězec dlouhý 18 až 19 znaků a měl by splňovat následující formát: <tep,saturace,teplota,pád>. Hodnoty saturace a teploty jsou dvouciferná čísla s jedním desetinným místem. Hodnota tepu může přesáhnout hodnotu 100, proto může být i trojciferná s jedním desetinným místem. Signalizace pádu má délku jeden znak, a to buď P v případě pádu nebo N v případě klidového stavu.

Datový řetězec je po přijetí modulem zkontrolován a odeslán na internetový server, kde je rozložen na jednotlivé naměřené hodnoty a uložen do databáze.

6 Aplikace monitorující stav pacienta

V této sekci jsou popsány jednotlivé třídy aplikace a jejich funkce. Při vývoji monitorovacího softwaru byly použity zdroje [5, 9, 10]. Veškeré zdrojové kódy jsou přiloženy na CD.

6.1 Hlavní MIDlet

Hlavní MIDlet je umístěn v souboru Main.java. V této třídě jsou vytvářeny potřebné instance tříd Log, Comm, AT a GPRS, se kterými se poté pracuje. Jsou zde také dvě lokální proměnné typu boolean- rpe a pc. Proměnná rpe slouží k nastavení povolení výpisů odpovědí modulu na AT příkazy (true= povoleno, false= vypnuto). Proměnná pc je zde kvůli nastavení COM portu a přenosové rychlosti. Pokud je nastavena na hodnotu true, aktivuje se režim komunikace s počítačem (např. pro testování softwaru). Hodnota false indikuje komunikaci s měřicí deskou.

V metodě startApp() je nastaven AutoStart na požadovanou dobu. Metoda destroyApp() se stará o řádné ukončení aplikace.

6.2 Třída Log

Tato třída slouží k zaznamenávání všech použitých AT příkazů a následných odpovědí modulu do textového souboru a také jsou zde vypsány kontrolní body aplikace, případně některé chybové hlášky. Soubor je vytvořen pod názvem log.txt a slouží ke zpětné kontrole chodu celé aplikace. Vytváří se v konstruktoru třídy a zápis se provádí zavoláním metody print(), kde se jako parametr předá proměnná typu String. Tato metoda automaticky odřádkuje každý zápis, takže není třeba za řetězec přidávat ukončovací znaky. Při restartu modulu se původní log vynuluje a otevře pro nový zápis.

6.3 Třída Comm

Tato třída slouží ke komunikaci přes sériový port. Při volání konstruktoru jsou předány tyto parametry: instance třídy Log a proměnná typu boolean k rozlišení, zda se jedná o komunikaci s počítačem nebo měřicí deskou.

Hned z počátku jsem narazil na problém při tisku výpisů přes sériový kanál do terminálu. Chyba byla v tom, že jsem používal klasickou javovskou metodu System.out.println(). Tato metoda v případě Cinterionu nefunguje. Pokud chceme tisknout, musíme vytvořit OutputStream a do něj zapisovat data. To se provádí příkazem write(), kde jako parametr předáme pole bytů.

Pro zjednodušení jsem zde vytvořil metodu print(), která pracuje s metodou write() a zjednoduší nám práci. Parametrem je proměnná typu String, takže nemusíme při každém výpisu převádět řetězec na byty. Na konec je opět přidán ukončovací znak, takže není třeba ani řetězec předem formátovat. Metoda ještě ověří, zda je připraven logovací soubor, a případně vytiskne řetězec i tam pro možnost následné kontroly.

Komunikace probíhá přes port COM0, rychlost přenosu je v případě komunikace s počítačem 115200 Bd/s a v případě komunikace s měřicí deskou 4800 Bd/s. Rychlost komunikace musí být nastavena na stejnou hodnotu jak v modulu, tak v obslužném terminálu. V případě rozdílných rychlostí nebudou příkazy fungovat a výpis do terminálu může zobrazit nesmyslné znaky. Při komunikaci s deskou je nastaven ještě CommConnectionControlLinesListener, který indikuje změnu stavu na pinu DTR (Data Terminal Ready). Tento pin signalizuje, zda-li je terminál připraven a může dojít ke spuštění komunikačního kanálu [6]. Poté jsou pomocí cyklu while načítána data z InputStreamu do proměnné buffer a následně jsou otestována z hlediska správnosti formátu a z hlediska stavu pacienta.

V případě vyhodnocení negativního stavu pacienta je odeslána varovná SMS zpráva s popisem problému a prozvoněno zvolené telefonní číslo. Za negativní stav pacienta se považuje pokles saturace krve kyslíkem pod 90 %, pokles tepu pod 60 úderů za minutu (Bradykardie) nebo naopak zvýšení tepu nad 90 úderů za minutu (Tachykardie), pád z lůžka, pokles teploty pod 35,5°C (Hypotermie) nebo nárůst teploty nad hranici 38°C (horečka, neboli Febris). [7, 8]

6.4 Třída AT

Tato třída slouží k obsluze AT příkazů. Je zde definována metoda SIMCode(), která zkontroluje, zda je třeba zadat PIN kód a případně ho automaticky nastaví. Zpočátku se mi

nedařilo připojit k síti operátora právě kvůli nezadanému PIN kódu. Proto jsem volání metody SIMCode() přidal do metody register(), kde se zajistí registrace telefonního čísla v síti operátora, aby bylo možné využívat funkce GSM.

Dále je zde metoda SyncTime() na sychronizaci času mezi modulem a operátorem, metoda na nastavení AutoStartu, na nastavení funkce WatchDog, metoda Send(String) pro odesílání AT příkazů do modulu a metoda getTime(), která vrací String s aktuálním časem.

Metoda Send() přidává na konec předaného řetězce ukončovací znak \r, což je mnohem příjemnější, než tento znak přidávat při každém příkazu manuálně.

Důležité jsou také metody sendSMS(String, String) a Call(String). Tyto metody slouží k odeslání textové zprávy a prozvonění požadovaného telefonního čísla. U metody sendSMS() je prvním parametrem telefonní číslo, druhým parametrem pak text zprávy.

6.5 Třída GPRS

Tato třída slouží k odesílání hodnot stavu pacienta na internetový server. Aby bylo možné tyto hodnoty uložit do databáze, vyžaduje se z bezpečnostních důvodů identifikační číslo pacienta a heslo. Hodnoty jsou pak pod číslem pacienta uloženy do databáze, kde k nim mají přístup pověřené osoby.

Pro připojení k síti GPRS jsou nezbytné následující kroky:

- *vložení SIM karty s dostupným kreditem či platným tarifem do modulu*
- *zadání správného PIN kódu*
- *registrace v síti GSM příslušného operátora*
- *vytvoření profilu pro automatický přístup k síti GPRS vytáčeným připojením (v našem případě pomocí AT^SJNET=\"gprs\", \"internet\", \"\", \"\")*

Následující kód je ukázkou metody httpSend() pro odeslání řetězce s naměřenými hodnotami na internetový server. Adresa serveru je specifikována v proměnné serverURL typu String. K řetězci je následně připojeno ID pacienta a pacientovo heslo v systému. K uložení dat dojde jen v případě shody těchto hodnot s hodnotami v databázi. Zajímavé je, že se tyto parametry pro PHP skript na serveru předávají přímo. Nepoužívá se zde znak ? jako v případě předávání parametrů přes URL.

```
public void httpSend(String data) {
      HttpConnection conn = null;
      InputStream is = null;
      OutputStream os = null;
      String parameters = "data=" + data + "&id=1&password=1234";

      try {                                                          // Test HTTP připojení
                                                                     // Příprava požadavku POST
         conn = (HttpConnection) Connector.open(serverURL);
         conn.setRequestMethod( HttpConnection.POST );
         conn.setRequestProperty( "Content-Type", "application/x-www-form-urlencoded" );
         os = conn.openOutputStream();
         os.write(parameters.getBytes() );
                                                                     // Požadavek na odpověď serveru
         Print( "Response code : " + conn.getResponseCode() );
                                                                     // Zobrazení obsahu
         is = conn.openInputStream();
         int ch;
         String r="";
         while ( (ch = is.read()) != -1 ){
            r += (char) ch;
         }
         com0.print( "Output : " + r);
      } catch ( Exception ex ) {
         Print( "Http : ex : " + ex.getClass() + " : " + ex.getMessage() );
         ex.printStackTrace();
      }                                                              // Zavřeme vše
      finally {
         try {
            if ( conn != null )
               conn.close();
            if ( is != null )
               is.close();
            if ( os != null )
               os.close();
         } catch ( Exception ex ) {
            Print( "Http : ex2 : " + ex.getClass() + " : " + ex.getMessage() );
            ex.printStackTrace();
         }
      }
   }
```

7 Server

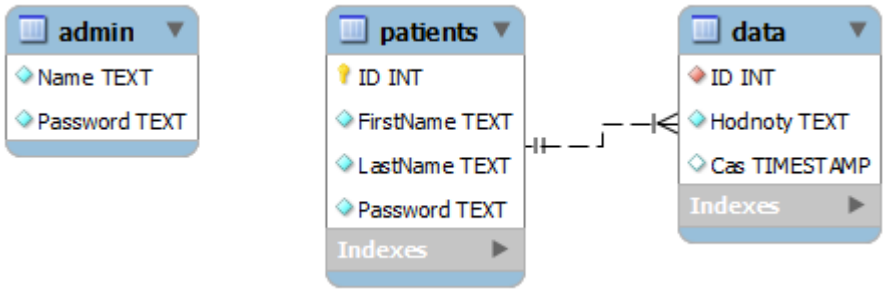
Databáze a webové stránky pro přihlášení personálu nemocnice a rodiny pacienta byly vytvořeny na serveru ic.cz. Adresa na tyto stránky je http://cinterion.ic.cz. Web využívá dynamického HTML kódu, PHP skriptu a Javascriptu. Dynamický HTML se trochu podobá jazyku Java, existuje zde např. i objekt typu „this.“ Javascript jsem použil hlavně pro zobrazení grafů, které jsou generovány pomocí nástrojů Google Charts.

Při studiu programovacího jazyka PHP, databáze MySQL a syntaxe HTML 5 jsem využíval zdroje [11-15]. Při vývoji a testování PHP skriptů a databáze jsem používal program EasyPHP 12.1, který obsahuje MySQL databázi a nástroje pro její administraci, PHP hypertextový preprocessor a server Apache.

7.1 Databáze

Jak již bylo zmíněno, databáze je realizována pomocí MySQL. Je zde vytvořena tabulka pacientů, administrátora a tabulka přijatých dat. Tabulka admin slouží pro přihlášení správce, obsahuje pouze položky přihlašovacího jména a hesla. Tabulka pacientů obsahuje jméno, příjmení, heslo a ještě identifikační číslo pacienta (ID), podle kterého jsou následně přiřazovány nasnímané hodnoty. Tabulka data obsahuje číslo ID, řetězec hodnot a datum a čas přijetí, který je typu TIMESTAMP a generuje se automaticky při ukládání. Model databáze vytvořený pomocí programu MySQL WorkBench viz *Obr. 2.1*.

Aby nedošlo k zobrazení obsahu databáze, popřípadě k jejímu narušení neautorizovanou osobou, použil jsem ochranu proti SQL injekcím kódu. Každý zadaný řetězec je chráněn funkcí mysql_real_escape_string() a každé číslo funkcí intval(). Tyto funkce provedou analýzu použitých znaků a v případě nebezpečí útoku je předán pouze prázdný řetězec.



Obr. 2.1 Model databáze

7.2 Webové stránky

Webové stránky se skládají z celkem pěti PHP souborů, které využívají dynamického HTML, PHP, Javascriptu a připojení k MySQL databázi.

7.2.1 Index.php

Tato stránka obsahuje hlavní menu a formuláře pro přihlášení administrátora a osob, kterým byly poskytnuty přihlašovací údaje pro zobrazení stavu pacienta, testovací formulář pro manuální odeslání hodnot a formulář pro registraci nového pacienta.

Formuláře se zobrazují najetím kurzorem myši na příslušný odkaz (událost onmouseover). Při najetí kurzoru myši na určitý odkaz se zavolá Javascriptová funkce, která vždy skryje ostatní formuláře a vybraný formulář nastaví jako viditelný.

Po stisknutí tlačítka “Submit Query” se obsah formuláře předá jako parametr příslušnému PHP souboru. K proměnným z daného formuláře se v PHP přistupuje pomocí proměnné $POST[‘název proměnné’]. Název proměnné je shodný se jménem prvku nastaveným v HTML kódu formuláře.

7.2.2 Patient.php

Tento soubor slouží k zobrazení stavu pacienta. Vytvoří se připojení k databázi pomocí příkazu mysqli_connect(), kde se jako parametr uvede: IP adresa serveru, název uživatele, heslo a název databáze.

```
$con = mysqli_connect("127.0.0.1","root","","my_db");      //údaje pro přihlášení do lokální testovací databáze
```

Poté je z databáze vybrán pacient a jeho data uložena do proměnné $result.

```
$result = $con -> query("SELECT * FROM Patients WHERE patients.FirstName='"
.mysql_real_escape_string($_POST["firstname"]). "' AND patients.LastName='"
.mysql_real_escape_string($_POST["lastname"]). "'AND patients.Password='"
.mysql_real_escape_string($_POST["password"]). "'");
```

Poté je zobrazeno jméno, příjmení, ID pacienta a vypsány naměřené hodnoty, které pacientovi přísluší. Všechny hodnoty jsou uloženy do pole $values[]. Pak je potřeba předat obsah této proměnné do Javascriptu, aby se mohly zobrazit grafy teploty a pulsu. To se dá zařídit v Javascriptu následujícím kódem:

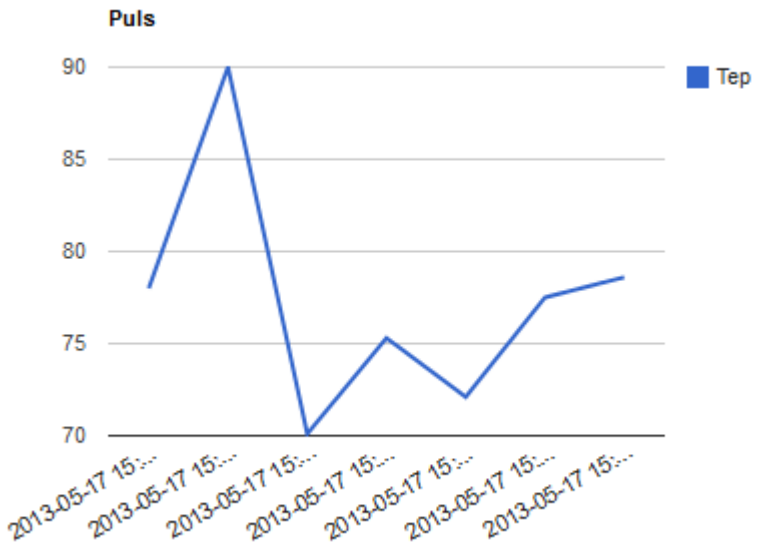
```
var myvar = <?php echo json_encode($values); ?>;
```

Řetězce jsou pak rozděleny na jednotlivé hodnoty pulsu, teploty, saturace krve a času uložení do databáze.

```
var datum = new Array();
var tep = new Array();
var saturace = new Array();
var teplota = new Array();
var pom;

for(var a=0;a<myvar.length;a++){
	pom = myvar[a].split(",");
	tep.push(pom[0].substring(1));
	saturace.push(pom[1]);
	teplota.push(pom[2]);
	datum.push(pom[4]);
}
```

Poté jsou předány jako hodnoty pro grafy. Na ose *x* je vždy zobrazeno datum a čas, na ose *y* dané hodnoty. V případě většího počtu zobrazovaných hodnot nemusí být vidět přesný čas měření. Stačí však najet na danou hodnotu kurzorem myši a zobrazí se jak datum, tak celý čas. Ukázka průběhu grafu pacientova pulsu viz *Obr. 3.1*.



Obr. 3.1 Graf pacientova pulsu

7.2.3 Insert.php

Tento soubor slouží ke vložení nového pacienta do databáze. Opět dojde k připojení do databáze a poté je v tabulce patients vytvořena nová položka s údaji o pacientovi. Ukázka načtení parametrů a vložení do databáze:

```
$fn=mysql_real_escape_string($_POST['firstname']);
$ln=mysql_real_escape_string($_POST['lastname']);
$pw=mysql_real_escape_string($_POST['password']);
$sql="INSERT INTO patients (FirstName, LastName, Password) VALUES ('$fn','$ln','$pw')";

if(!mysqli_query($con,$sql))
{
  die('Error: ' . mysqli_error());
}
echo "1 record added";
```

7.2.4 Print.php

Po odeslání obsahu formuláře pro přihlášení administrátora zobrazí tento skript tabulku zobrazující údaje i hesla všech pacientů v databázi. Ukázka vytvoření tabulky:

```
echo “<table border=’0’>
    <tr>
    <th>Firstname</th>
    <th>Lastname</th>
    <th>Password</th>
    </tr>";
```

```
$result = $con -> query("SELECT * FROM patients");
 if(!$result) echo "chyba\n";
    while($row = $result -> fetch_assoc())
    {
      echo "<tr>";
      echo "<td>" . $row['FirstName'] . "</td>";
      echo "<td>" . $row['LastName'] . "</td>";
      echo "<td>" . $row['Password'] . "</td>";
      echo "</tr>";
    }
    echo "</table>";
```

7.2.5 Upload.php

Upload.php je určen ke kontrole a ukládání naměřených hodnot do databáze. GSM modul prostřednictvím GPRS předá tomuto skriptu následující údaje: ID pacienta, heslo a data. K uložení dojde pouze za předpokladu, že se shoduje ID a heslo pacienta s údaji v databázi. Ukázka kódu:

```
$values= explode(",", $_POST["data"]);

if(!count($values)==4) echo "\nNesprávný formát dat.\n";
else {
    $con=mysqli_connect("127.0.0.1","root","","my_db");
                                                                                    // kontrola připojení
    if (mysqli_connect_errno()) echo "Failed to connect to MySQL: " . mysqli_connect_error();

    $sql="SELECT Password FROM patients WHERE ID=".intval($_POST["id"]);
    if (!($result = mysqli_query($con, $sql))) die('Error: '.mysqli_error($con))."\n";

    $dta = $result -> fetch_array();
                                                                                    //format:  tep-saturace-teplota
    $towrite= substr($values[0],1,3)."-".$values[1]."-".substr($values[2],-4,4)."\r\n";
    $pass= $dta[0];
    $id= intval($_POST["id"]);
    $data= mysql_real_escape_string($_POST["data"]);
    if($pass==mysql_real_escape_string($_POST["password"])){
          $sql="INSERT INTO data (ID, Hodnoty) VALUES ('$id','$data')";
        if (!mysqli_query($con,$sql))
          die('Error: ' . mysqli_error($con))."\n";
        else {
          echo "Data updated.\n";
        }
    }
    else echo "Wrong password!\n";
}
```

Závěr

Realizace serveru s databází a webovými stránkami proběhla úspěšně. Komunikace mezi modulem a serverem probíhá bez problémů. V aplikaci monitorující stav pacienta jsem vytvořil vlákno, které simuluje příchozí hodnoty z měřicí desky. Až bude hotový návrh nové desky, stačí jen toto vlákno odstranit a přijímat hodnoty přímo.

Oprávnění uživatelé můžou kdykoli zjistit aktuální i historický stav pacienta. Stačí mít pouze přístup k internetu a znát potřebné přihlašovací údaje. V rámci zabezpečení dat a celé databáze jsem skripty ošetřil proti možným SQL injekcím kódu.

V budoucnu, až bude vyrobena nová měřicí deska, bude tento systém monitoringu díky vytvořenému serveru a GPRS komunikaci mnohem účelnější, než tomu bylo doposud. Lékařský personál může mít díky moderním technologiím data pacientů vždy po ruce, i když nebude zrovna v dosahu. Na internetové stránky bude možno přistupovat např. pomocí mobilního telefonu nebo tabletu.

Seznam literatury a informačních zdrojů

[1] SIEMENS. *TC65 Terminal datasheet* [pdf]. 2010 [cit. 2013-04-01].

[2] NOVOTNÝ, Luděk. *Historie a vývoj jazyka Java*. [online]. 2003 [cit. 2013-04-03]. Dostupné z: http://www.fi.muni.cz/usr/jkucera/pv109/2003p/xnovotn8.htm

[3] *Java ME*. [online]. 2013 [cit. 2013-04-03]. Dostupné z: http://cs.wikipedia.org/wiki/Java_ME

[4] KROPÍK, Petr. *Práce s GSM moduly Cinterion/Siemens TC65* [pdf]. 2010 [cit. 2013-05-02]. Dostupné z: http://faraday.fel.zcu.cz/wordpress/wp-content/uploads/2011/01/GSM_moduly.pdf

[5] CINERT, Tomáš. *Využití embedded modulů pro monitoring životních funkcí pacienta* [online]. Plzeň, 2012 [cit. 2013-05-10]. Dostupné z: https://portal.zcu.cz/wps/PA_StagPortletsJSR168/KvalifPraceDownloadServlet?typ=1&adipidno=40709. Diplomová práce. ZČU.

[6] DTR. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2013 [cit. 2013-05-10]. Dostupné z: http://en.wikipedia.org/wiki/DTR

[7] Transport kyslíku krví. In: *Wikiskripta.eu* [online]. 2007 [cit. 2013-05-15]. Dostupné z: http://www.wikiskripta.eu/index.php/Transport_kysl%C3%ADku_krv%C3%AD

[8] Fyziologické funkce. In: *Wikiskripta.eu* [online]. 2008 [cit. 2013-05-15]. Dostupné z: http://www.wikiskripta.eu/index.php/Fyziologick%C3%A9_funkce

[9] JavaCint. CLAIRAMBAULT, Florent. *TC65Dev* [online]. 2013 [cit. 2013-05-15]. Dostupné z: http://www.javacint.com/TC65Dev

[10] JavaCint. CLAIRAMBAULT, Florent. *FAQ* [online]. 2013 [cit. 2013-05-15]. Dostupné z: http://www.javacint.com/FAQ

[11] Dynamické HTML: jednoduché základy. *Jakpsatweb.cz* [online]. 2013 [cit. 2013-05-20]. Dostupné z: http://www.jakpsatweb.cz/dhtml.html

[12] *Mysql_connect* [online]. 2013 [cit. 2013-05-20]. Dostupné z: http://php.net/manual/en/function.mysql-connect.php

[13] MySQL, SQL, PHP. *Tvorba-webu.cz* [online]. 2008 [cit. 2013-05-20]. Dostupné z: http://www.tvorba-webu.cz/php/mysql.php

[14] Jak začít s PHP: zejména instalace PHP. *Jakpsatweb.cz* [online]. 2012 [cit. 2013-05-20]. Dostupné z: http://www.jakpsatweb.cz/php/jak-zacit.html

[15] PHP triky: Obrana proti SQL Injection. *Php.vrana.cz* [online]. 2005 [cit. 2013-05-20]. Dostupné z: http://php.vrana.cz/obrana-proti-sql-injection.php

Přílohy

Všechny zdrojové kódy jsou k práci přiloženy na CD.