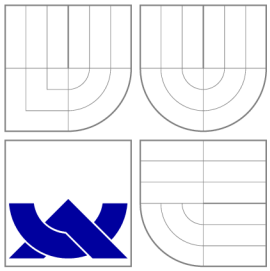
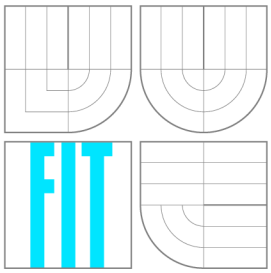


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

GSM MODUL PRO BEZDRÁTOVÉ SENZOROVÉ SÍTĚ

GSM MODULE FOR WIRELESS SENSOR NETWORKS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — JAN MAYER
AUTHOR

VEDOUCÍ PRÁCE — Ing. JAN HORÁČEK
SUPERVISOR

BRNO 2010

Abstrakt

Projekt si klade za cíl vytvoření komunikačního kanálu mezi uživatelem a vzdálenou bezdrátovou senzorovou sítí prostřednictvím GSM technologie. Toho docílíme spojením senzoru Iris firmy Crossbow s GSM modulem firmy Teltonika. Samotná vzdálená komunikace bude provedena pomocí posílání/přijímaní SMS zpráv a připojením platformy ke vzdálenému serveru pomoci GPRS technologie.

Abstract

Main goal of this project is to achieve communication between user and distant wireless sensor network using GSM technology. This task will be accomplished by connecting Crossbow Iris wireless sensor node and Teltonika GSM module device. Communication itself will be provided by sending/receiving SMS and connecting platform to remote server over the internet using GPRS technology.

Klíčová slova

TinyOS, bezdrátová sít, GPRS, GSM, SMS, RS-232, Iris, ATmega1281

Keywords

TinyOS, wireless network, GPRS, GSM, SMS, RS-232, Iris, ATmega1281

Citace

Jan Mayer: GSM modul pro bezdrátové senzorové sítě, bakalářská práce, Brno, FIT VUT v Brně, 2010

GSM modul pro bezdrátové senzorové sítě

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Jana Horáčka

.....
Jan Mayer
18. května 2010

Poděkování

Děkuji vedoucímu bakalářské práce Ing. Janu Horáčkovi za výborné vedení a chotu poradit a účastnit se konstuktivního dialogu během řešení vzniklých problémů.

Obsah

Kapitola 1

Úvod

Bezdrátové senzorové sítě sestávají z velkého počtu malých senzorů, které jsou spolu schopny komunikovat na vzdálenosti řádově stovek metrů. Tyto sítě mají v dnešní době významné uplatnění při monitorování rozsáhlých systémů. Jako příklad si můžeme uvést detekci vlhkosti půdy na velkých obdělávaných polích, nebo detekci požárů v lese.

Při práci s těmito autonomními systémy je samozřejmě žádoucí mít se sítí komunikační kanál, pomocí kterého můžeme zpracovávat sesbíraná data nebo upozorňovat na vzniklé události. Protože jsou sítě často rozsety po terénu, nabízí se jako vynikající spojující prostředek právě technologie GSM. V dnešní době má vynikající pokrytí po celém světě a nabízí hned několik možností, jak komunikovat na velké vzdálenosti. Přenos paketů přes internet pomocí GPRS navíc poskytuje dostatečnou rychlost pro odesílání sledovaných dat a vzdálenou správu sítě.

Z výše zmiňovaných potřeb vznikla tato práce. Podrobně zde popíšeme způsob spojení jednoho senzoru bezdrátové sítě s GSM modulem, který nám zajistí praktický komunikační kanál se vzdáleným uživatelem.

Dokument je rozdělen na dvě hlavní části. V první části, která je tvořena druhou až čtvrtou kapitolou, se obecně seznámíme s použitými platformami a rozebereme podrobně využívané principy. V druhé kapitole se blíže seznámíme se senzorovými sítěmi a námi použitou platformou. Třetí kapitola pojednává o způsobu psaní aplikací pro tyto platformy. Ve čtvrté kapitole se pokusíme shrnout technologii GSM a popsat námi použitý modem Teltonika.

V druhé části se budeme zabývat potřebnými úpravami a samotnou implementací projektu. Pátá kapitola rozebírá problémy spojené s návrhem komunikačního rozhraní dvou odlišných platforem — senzoru a GSM modemu. V šesté kapitole si předvedeme samotnou implementaci projektu. A v sedmé kapitole krátce popíšeme práci s vytvořenou demonstrační aplikací. Závěrečná kapitola nakonec shrne výsledky projektu.

Kapitola 2

Senzorové sítě

Kapitola rozebírá obecnou filozofii senzorových sítí a pokouší se upozornit na jejich přednosti a nedostatky. Popíšeme zde i senzory firmy Crowsbow a blíže se seznámíme s platformou Iris, kterou použijeme pro samotnou realizaci projektu.

2.1 Vývoj senzorových sítí

Počátky technologie senzorových sítí sahají až do padesátých let minulého století. Tehdy byl v rámci studené války zahájen výzkum senzorů, které by byly rozprostřeny pod mořskou hladinou a mohly zachytávat hluk plujících ponorek.[14] Od těchto prvních prototypů však senzorové sítě zaznamenaly významné technologické změny. V dnešní době zažíváme trend miniaturizace, který se odrazil i na jejich vývoji. Myšlenka však zůstala zachována. Potřebujeme rozprostřenou síť malých nenápadných jednotek, které vydrží běžet i několik let, aniž by vyžadovaly údržbu. Tyto jednotky obsahují senzory, mikrokontroler a vysílač pro komunikaci s ostatními zařízeními. Jeden senzor, jako součást sítě, je často nazýván také anglicky „mote“, i my se budeme držet tohoto označení.

2.2 Technologie ZigBee

Jedná se o poměrně novou bezdrátovou komunikační technologii vystavenou na standardu IEEE 802.15.4. Ten specifikuje fyzickou vrstvu, zatímco ZigBee specifikuje síťovou a aplikační vrstvu. [7] Tyto dva standardy dohromady vytváří technologii, která byla vyvinuta přímo s ohledem na požadavky bezdrátových sítí.

Podobně jako Bluetooth se ZigBee používá pro spojení nízkovýkonových zařízení v sítích PAN (Personal Area Network). Rádio může při přímé viditelnosti komunikovat až na vzdálenosti stovek metrů a díky multiskokovému ad-hoc směrování spolu mohou komunikovat i zařízení, která nejsou v přímé rádiové viditelnosti. Vysílání se provádí v bezlicenčních pásmech 868 MHz, 902-298 MHz a 2,4 GHz. Přenosová rychlost činí 20, 40 a 250 kbit/s.

2.3 Porovnání ZigBee a Bluetooth

Technologie Bluetooth je definovaná standardem IEEE 802.15.1. Rychlost přenosu dle specifikace Bluetooth 2.0 EDR (Enhanced Data-Rate) dosahuje až 3 Mbit/s. Podobně jako ZigBee se řadí mezi energeticky méně náročné. Přesto však nesplňuje požadavky pro potřeby

senzorových sítí. Jeden z hlavních nedostatků je to, že spolu může komunikovat maximálně osm zařízení. My však potřebujeme sítě o stovkách senzorů.

Dalším nedostatkem je relativně dlouhá doba, kterou trvá než spolu zařízení navážou kontakt. Pro práci se senzorovými sítěmi se spíše hodí, aby zařízení v pravidelných cyklech rychle navázala kontakt, odeslala data a okamžitě vypnula rádio, aby spořila energii.

V neposlední řadě má Bluetooth oproti ZigBee stále daleko větší spotřebu energie i během režimu spánku.

	ZigBee	Bluetooth
Rychlost přenosu	250 kb/s	3 Mb/s
Počet zařízení	2^{16}	8
Zabezpečení	PIN, 64 bit, 128 bit	128 bit, AES
Připojení nového zařízení	30 ms	3s (typicky 20s)
Aktivace spícího zařízení	15 ms	3s (typicky)

Tabulka 2.1: Srovnání Bluetooth a ZigBee

Z tabulky 2.1 jasně vyplývají rozdíly mezi technologiemi. ZigBee je ideální pro pravidelné připojování, přenášení menšího objemu dat a přepínání do režimu spánku. Díky tomu vydrží jeho baterie až 100x déle než Bluetooth při podobném používání. [1]

2.4 Platformy firmy Crossbow

Platforem, které pracují s technologií ZigBee, vzniklo od jejího vydání velké množství. My se zaměříme pouze na pár nejznámějších a snadno dostupných.

TelosB je základní OpenSource platforma firmy Crossbow založená na mikrokontroléru MSP430 firmy Texas Instruments. Mezi její největší přednosti patří USB konektor, který je přímo součástí desky. Toto řešení přináší nespornou výhodu při častém přeprogramovávání přístroje.

Imote2 je velmi výkonná platforma obsahující procesor Intel PXA271 XScale, který je schopný běžet na frekvenci až 416MHz. Imote2 nabízí 32 MB FLASH paměť, 32 MB SDRAM a možnost připojení velkého množství periferií pomocí rozličných rozhraní (např. UART, SPI, I2C). Jedná se o platformu pro velmi náročné aplikace.

Oblíbenou platformou je platforma MicaZ. Vychází ze starší platformy Mica2. Obě tyto platformy jsou založeny na mikrokontroléru ATMega128L, který nabízí 128 KB paměti na uložení programu a 4 KB vnitřní RAM paměti. [11]

Z platformy MicaZ vychází i námi použitá platforma Iris, které se budeme podrobněji věnovat v následující podkapitole.

2.5 Platforma Iris

Pro náš projekt použijeme platformu Iris. [10] Ta se příliš neliší od dříve zmiňované platformy MicaZ. Je pouze osazena výkonnějším mikrokontrolérem XM2110CA, který vychází z mikrokontroléru ATMega1281 firmy Atmel.

Obrázek 2.1: Platforma Iris

2.5.1 Napájení desky

Celá platforma je napájena dvěmi tužkovými bateriemi AA a měla by vydržet přes dva roky provozu. Důležité je zmínit, že dvě tužkové baterie poskytují napětí 3 V pouze v ideálním případě. Zpravidla dodávají napětí nižší a navíc postupným vybíjením baterie napětí časem klesá. Z toho vyplývá, že celá deska nepracuje se standardní 3,3 V logikou, ani s ideální 3 V logikou, ale její logická „1“ se pohybuje někde kolem 2,7 V. Toto musíme mít na paměti při připojování vlastních komponent k senzoru.

2.5.2 Mikrokontrolér

Iris je založena na mikrokontroléru ATMega1281, který je taktován na 8 MHz a nabízí nám (oproti svému předchůdci ATMega128L v MicaZ) 8 KB RAM. Dále můžeme zmínit až čtyři UART rozhraní, SPI rozhraní a v neposlední řadě desetibitový A/D převodník. Hlavním důvodem, proč je kontrolér používán pro senzorové sítě je samozřejmě jeho spotřeba. Při frekvenci 8 MHz může být napájen už 2,7 V a jeho spotřeba se pohybuje okolo 8 mA. V režimu spánku si však vystačí s pouhými 8 uA.

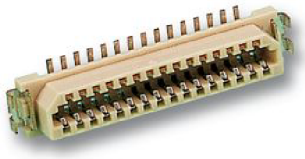
2.5.3 Rádio

Jedná se o výkonný vysílač/příjmač Atmel RF230 pracující na frekvenci 2.4 GHz, který samozřejmě odpovídá standardům 802.15.4 a ZigBee. Dle tohoto standardu je definovaná rychlost přenosu dat na 250 kbps. Dosah rádia by měl při spotřebě 16 mA dosahovat minimálně 50 metrů v uzavřených prostorách a na otevřeném prostranství až 300 metrů.

2.5.4 Rozšiřující konektor

K platformě je možno připojit pomocí 51pinového konektoru senzorovou nebo jinou přídavnou desku. I samotné nahrávání software do mote probíhá připojením takzvané base station do tohoto konektoru. Base station zprostředkovává spojení mezi počítačem a Iris. Pro programování používáme base station MIB520, kterážto je obdařena USB konektorem. Za zmínku ještě stojí base station MIB510, jež umožňuje programovat mote přes RS-232 nebo zprostředkovává přímé připojení k jednomu z UART rozhraní mikrokontroléru.

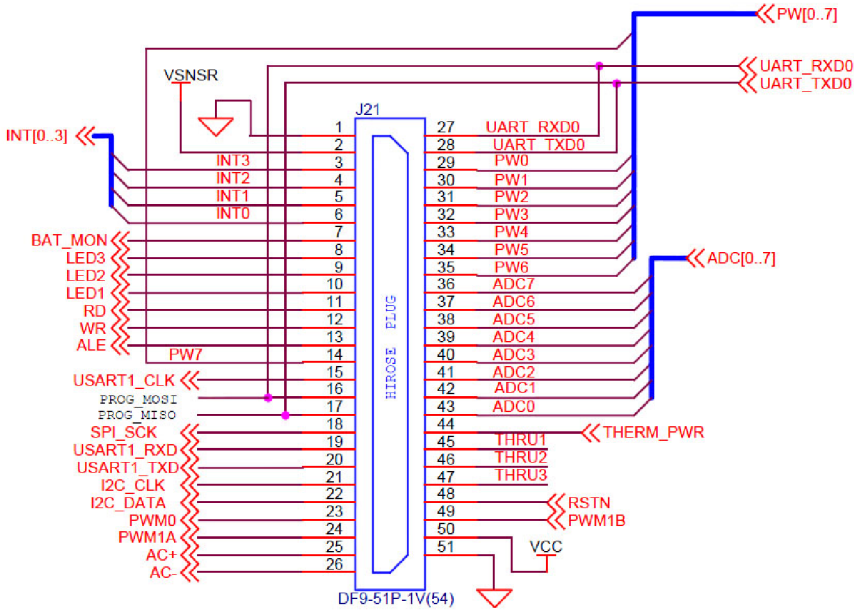
Samotný konektor je vyráběný japonskou firmou Hirose a nese označení DF9-51(P/S). Bohužel je však v Evropě nesehnatelný a je potřeba ho objednávat z Ameriky, což přináší dlouhé dodací lhůty a přirážky. Naštěstí se nám po delším pátrání podařilo najít ekvivalentní



Obrázek 2.2: Rozšiřující konektor

konektor vyráběný firmou Harwin, nesoucí označení M40-6205146, který je do Evropy běžně dodáván.

Na tomto konektoru jsou vyvedeny piny z kontroléru, jak znázorňuje obrázek 2.3 Pro naše další potřeby jsou významné piny 19, 20, 27, 28, 50, 51. Tedy přímé spojení s oběma UART rozhraními kontroléru a vývody napájení.



Obrázek 2.3: Schéma zapojení konektoru

Kapitola 3

TinyOS

Pro programování jednotlivých uzlů se používá open-source operační systém TinyOS. Jedná se o jednoduché, na systémové prostředky velice nenáročné, prostředí. Systém je navržen tak, aby si vystačil i s prostředky těch nejjednodušších senzorů. Hravě si dokáže poradit i s 8 KB paměti pro uložení programu a 512 bytů RAM. [12] TinyOS proto přímo neposkytuje metody pro dynamickou správu paměti, ani další komfortní metody známé z komplexnějších operačních systémů.

Síla tohoto prostředí je ukryta v jeho jednoduchém konceptu. Kritickým místem u vestavěných i běžných počítačů jsou zřejmě interakce programu s periferiemi. Procesor zadá periferii požadavek na jistou operaci. Počká, než periferie operaci dokončí a pochlubí se výsledkem, a po té může procesor pokračovat v další činnosti. V běžných počítačích, které mají dostatečné hardwarové prostředky, se čekání na periferie řeší pomocí systému přepínání a uspávání jednotlivých nezávislých programových vláken. Díky tomu dostáváme zdánlivý dojem synchronního provádění programu. O něco požádáme a přijde nám, že se výsledky dostaví hned. To, že během čekání na periferii naše vlákno nějakou dobu nebylo aktivní a procesor dával prostor vláknům jiným, se nás nijak netýká.

Tento systém si v senzorech s omezenými prostředky dovolit nemůžeme. Pamatovat si například obsah zásobníku a stav registrů kontroléru pro každý proces v systému, který může operovat s pouhými 512 B RAM, je naprosto nemyslitelné. Proto se TinyOS vydal jiným směrem. Namísto snahy vytvořit v softwaru dojem synchronního provádění úkolů dělí řešení úkolů do dvou fází (tzv. split-phase). Aby došlo k provedení operace, musíme jednak operaci zavolat a jednak zvlášť definovat kód, který se má provádět po (ať již úspěšném, či neúspěšném) dokončení operace. Jak tento postup vypadá v praxi si ukážeme v podkapitole věnované programovacímu jazyku nesC, ve kterém je TinyOS napsán.

Druhou neméně významnou vlastností TinyOS, pomocí které dochází k ušetření velkého množství systémových prostředků, je statická práce s funkcemi a proměnnými. V běžných modulárních programovacích jazycích musí být za běhu v RAM uloženy ukazatele na funkce a proměnné, které mohou být používány v rámci každého jmenného prostoru (anglicky naming scope). Díky této vlastnosti mohou být proměnné a funkce sdíleny mezi jednotlivými moduly. TinyOS tento problém radikálně zjednodušuje. Proměnné nemohou být sdílené mezi moduly a u každého modulu musí být dopředu stanoveno, se kterými funkcemi z externích modulů bude pracovat. Díky tomuto opatření je navíc graf volání funkcí znám již v době překladu a kompilátor může výborně optimalizovat.

Všechna tato omezení a opatření si může systém dovolit pouze díky tomu, že program je jednou zkompilován a nahrán na platformu. Pro každou platformu je tedy jen jeden program a nedochází k paralelnímu spouštění programů a dalším situacím, se kterými se

setkáváme na běžných počítačích.

Mluvit tedy o TinyOS jako o operačním systému je velice nadnesené. Jedná se spíše o sadu knihoven a jakýsi malý kernel, který nám zajišťuje správné volání naplánovaných událostí.

3.1 Jazyk nesC

Tento programovací jazyk je derivátem známého jazyka C. Jeho převážná část byla vytvořena na univerzitě v Berkley a je plně open-source. Od začátku byl zamýšlen pro použití v senzorových sítích a je v něm napsán celý TinyOS. Při jeho popisu se budeme zabývat převážně záležitostmi, které budeme využívat při práci na ovladači pro GSM modul. Mezi hlavní rozdíly oproti klasickému C patří vytváření rozhraní, modulů a konfigurací, pomocí nichž můžeme budovat hierarchicky strukturované programy. Dále je to rozdělení práce do split-phase a v neposlední řadě definování asynchronního a atomického vykonávání kódu.

3.1.1 Hierarchická struktura

Každý program v nesC se skládá ze vzájemně propojených komponent. Každou komponentu můžeme rozdělit na část definiční a implementační. V první části komponenta definuje rozhraní, která jí musí být pro správnou prácí poskytnuta, a rozhraní, které sama může nabídnout ostatním. Druhou částí je samotná implementace komponenty.

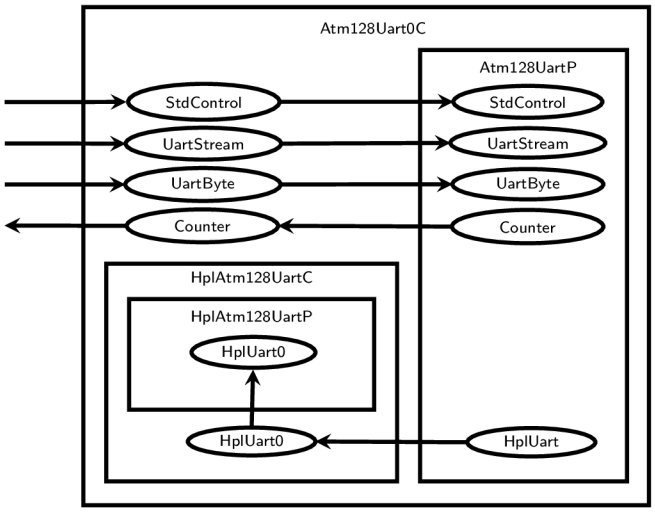
Rozhraní jsou dvousměrná, definují příkazy „commands“, které musí implementovat poskytovatel rozhraní a které může volat uživatel rozhraní, stejně jako události „events“, které jsou volány poskytovatelem a jejich implementaci obstarává uživatel rozhraní.

Ještě nám zbývá definovat si rozdíl mezi komponentou definovanou modulem a komponentou definovanou pomocí konfigurace. Moduly obsahují implementace jednoho nebo více rozhraní, zatímco konfigurace popisují pouze vzájemné propojení dalších komponent. Navenek však není vidět, (ani nás většinou nezajímá) zda-li je daná komponenta definovaná jako konfigurace nebo jako modul.

Jméno souboru	Typ obsahu
`UartLite.nc`	Rozhraní
`UartLite.h`	Hlavičkový soubor (stejně jako v C)
`UartLiteC.nc`	Veřejný soubor komponenty (je možné ho používat při spojování do větších celků)
`UartLiteP.nc`	Privátní soubor (na tento soubor by nemělo být nikde mimo tuto komponentu odkazováno)

Tabulka 3.1: Konvecne pojmenování souborů

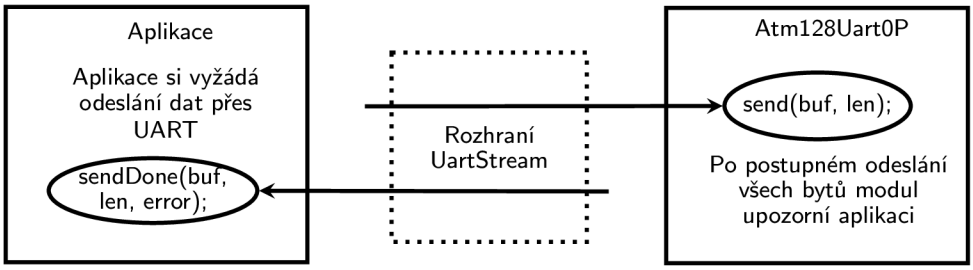
Při pojmenovávání souborů komponent a rozhraní se pro přehlednost používá jednoduchý koncept (viz. tabulka 3.1). Hierarchickou strukturu závislosti modulů a poskytování rozhraní přibližuje obrázek 3.1 Jedná se o velmi zjednodušené schéma ovladače UART0 portu.



Obrázek 3.1: Ukázka hierarchického uspořádání modulů

3.1.2 Split-phase

Výše popisované rozdělení komunikace mezi moduly do dvou fází ilustruje obrázek 3.2, na němž můžeme vidět ukázkové odesílání dat přes UART port. Aplikace používá rozhraní `UartStream`, díky tomu může volat funkci `send` na odeslání řetězce znaků přes UART a zároveň musí definovat kód funkce `sendDone`. Aplikace pak může kdykoliv zavolat funkci `send` na odeslání dat přes UART a po správném odeslání všech dat ovladačem je automaticky proveden kód funkce `sendDone`.



Obrázek 3.2: Provádění úkolů v TinyOS

3.1.3 Provádění funkcí

Na rozdíl od standardního C, kde se funkce volají a provádějí přímo, nesC při komunikaci mezi moduly pracuje s takzvanými příkazy (commands). Druhým způsobem, jak mezi sebou mohou dva moduly komunikovat je pomocí událostí (events), díky kterým jeden modul dává druhému najevo, že dokončil nějakou práci.

Dalšími důležitými jednotkami jsou úkoly (tasks), ty jsou v rámci modulu privátní. Jejich význam spočívá v tom, že jejich spuštění nenastane okamžitě, ale naplánuje se na nejbližší možný okamžik pomocí plánovače úloh (tasks scheduler). Tím je principielně jednoduchá FIFO fronta, která nově příchozí úkoly řadí na začátek a z konce fronty úkoly vybírá. Tyto úkoly postupně provádí. Jednotlivé commands, tasks a events jsou vůči sobě atomické, tedy ke spuštění nového úkolu dojde vždy až po dokončení celého úkolu předchozího.

Toto chování je zřejmě nevhodné pro práci se softwarovými nebo externími přerušeními, které je potřeba obsloužit bez čekání na dokončení jiných úkolů. V tomto případě přichází do hry asynchronní funkce. Rozdíl je v tom, že k jejich spouštění dochází nezávisle na plánovači úloh, proto je potřeba s nimi zacházet opatrně a nesetrvávat v nich příliš dlouho. Běžně používaný postup chování asynchronního kódu je následující: Pokud dojde k přerušení, asynchronní rutina provede nejnezbytnější operace a naplánuje do správce úloh spuštění synchronního úkolu. Tento úkol už může vykonávat časově náročnější operace.

Při psaní kódu v nesC je třeba dbát na to, abychom pracovali i s co nejkratšími úkoly (taks), protože TinyOS, na rozdíl od běžného operačního systému, neumí pozastavit probíhající úkol a přiřadit procesorový čas jinému úkolu, který čeká ve frontě. Na následují úkol se vždy dostane řada až po ukončení provádění úkolu předchozího. Pro plynulý běh pseudoparalerních procesů je tedy zjevně výhodnější větší počet podobně krátkých a rychle se střídajících úkolů.

3.1.4 Nahrávání programů do mote

Samotná kompilace je z uživatelského pohledu velice jednoduchá. Stačí ve složce projektu napsat:

```
make iris
```

Při každé kompilaci je potřeba specifikovat platformu, aby překladač věděl, jaké hlavičkové soubory a moduly má při překladu připojit. Pro nahrání programu do zařízení je potřeba zasunout mote do base station a tu pak připojit k počítači. Námi použitá base station MIB520 se připojuje přes USB. V adresáři `/dev/` nám vytvoří dvě nová zařízení. Zpravidla se jedná o `/dev/ttyUSB0` a `/dev/ttyUSB1`. Jak se nám již Linux snaží napovědět názvem zařízení, jedná se o emulaci dvou jednoduchých sériových linek. Soubor s nižším označením se používá pro nahrávání softwaru do mote, soubor s vyšším označením slouží pro komunikaci s UART rozhraním našeho mote. Samotné nahrání softwaru se provádí příkazem:

```
make iris install,<node_id> mib520,/dev/ttyUSB0
```

Místo `<node_id>` se dosazuje číslo, které představuje jedinečný identifikátor daného uzlu. Po úspěšném nahrání software se aplikace uloží do flash paměti na platformě a začne se automaticky vykonávat. Díky perzistenci dat ve flash paměti se začne uložený kód vykonávat pokaždé, když Iris zapneme nebo připojíme pomocí base station k počítači.

3.1.5 Kompilační proces

Nejprve si musíme vysvětlit rozdíl mezi pojmy platforma a čip. Platformou je v kontextu TinyOS myšleno kompletní zařízení, které může obsahovat několik čipů. Čipem může být například mikroprocesor (v případě Iris ATmega1281) nebo ovladač rádia.

V současné verzi TinyOS je podporovaná řada platforem a čipů, jejich příslušné konfigurační soubory a závislé moduly jsou uloženy ve stromové struktuře uvedené níže.

Při překladu projektu provádí kompilátor řadu úkonů, které jsou běžnému uživateli skryty. Nejprve vyhledá správný soubor `.platform`, který přísluší naší platformě a díky němu se dozví, jak při kompilaci postupovat. V tomto konfiguračním souboru jsou uloženy přepínače, které mají být v překladači nastaveny pro správnou kompilaci naší platformy, a seznam adresářů, ve kterých jsou uloženy moduly, jež naše platforma podporuje.

```
.tos/
  chips/
    atm128/
    atm1281/
    ds2401/
    msp430/
    rf2xx/
  interfaces/
  lib/
  platforms/
    iris/
      chips/
        at45db/
        ds2401/
        rf230/
        ...
      sim/
      .platform
      platform.h
      ...
    mica/
    mica2/
    micaz/
    telosb/
    tinynode/
    ...
  system/
  types/
  ...
```

Ukázka adresářové struktury TinyOS

Protože senzorové desky často používají stejné čipy nebo vychází ze starších modelů, je naprosto běžnou praxí, že se při kompilaci postupně prochází adresáře několika desek, platforem a čipů. Ve složce, která patří následující platformě, jsou již pak uložené pouze ty moduly, které se oproti předchozím platformám změnily. V ukázce 3.1 ze souboru `.platform`, jež náleží platformě Iris, je jasně vidět, že zařízení vychází z desek Mica a Micaz.

3.1.6 Ukázka kódu v nesC

Rozhraní

Rozhraní sestávají pouze ze seznamu funkcí, které bude daná komponenta nabízet nebo požadovat od ostatních.

V demonstračním souboru 3.2 vidíme rozhraní pro práci s telefonními hovory. Aplikace, která ho chce používat, může pomocí volání funkce `makeCall` vytáčet telefonní čísla. Musí však také definovat kód, který se má provést po úspěšném vytočení čísla (funkce `callDone`) a při příchozím hovoru (funkce `incomingCall`).

```
...
push( @includes, qw(
  %T/platforms/micaz
  %T/platforms/mica
  %T/platforms/iris/chips/rf230
  %T/chips/rf2xx/rf230
  %T/chips/rf2xx/layers
  %T/chips/rf2xx/util
  %T/platforms/iris/chips/at45db
  %T/platforms/mica2/chips/at45db
  %T/platforms/mica/chips/at45db
  %T/chips/at45db
  %T/platforms/iris/chips/ds2401
  %T/platforms/mica2/chips/ds2401
  %T/chips/ds2401
  %T/chips/atm1281
  %T/chips/atm1281/adc
  %T/chips/atm1281/timer
  %T/chips/atm128
  %T/chips/atm128/adc
  %T/chips/atm128/pins
  %T/chips/atm128/spi
  %T/chips/atm128/i2c
  %T/chips/atm128/timer
  %T/lib/timer
  %T/lib/serial
  %T/lib/power
  %T/lib/diagmsg
  ) );
...
```

Zdrojový kód 3.1: Ukázka souboru .platform TinyOS

```
interface Call {
  command error_t makeCall( uint8_t* number );
  event void callDone( error_t error );
  event void incomingCall( uint8_t* number );
}
```

Zdrojový kód 3.2: Ukázka souboru Call.ns

Konfigurace

Konfigurace bývají často velice krátké. Obsahují jen popis jak pomocí nabízených a požadovaných rozhraní dohromady propojit moduly.

Níže uvedený zdrojový kód 3.3 nám ukazuje nejvyšší komponentu jednoduché aplikace, která je schopna pracovat s naším ovladačem GSM modulu. V seznamu `components` vidíme komponenty, které entita používá a pod nimi vidíme, jak komponenty pomocí nabízených

a požadovaných rozhraní vzájemně propojuje. (Poznámka: Při napojování rozhraní pomocí `->` stačí napsat název rozhraní pouze k jedné z komponent.)

```
configuration GsmAppC {
}
implementation {
  components MainC, LedsC, GsmModuleC;
  components GsmC as App;

  App -> MainC.Boot;
  App.Leds -> LedsC;

  App -> GsmModuleC.GsmConfig;
  App -> GsmModuleC.Call;
}
```

Zdrojový kód 3.3: Ukázka souboru GsmAppC.ns

Modul

Jedná se o zdrojové kódy, které popisují samotnou implementaci. Tedy popis, jak se mají komponenty chovat, jako v konvenčních imperativních jazycích.

V ukázce kódu 3.4 můžeme vidět ukázku jednoduché aplikace, které se ihned po nabootování pokusí spojit s Gsm modulem. Tedy ve funkci `Boot.booted`, která je volána po úspěšném startu systému, dochází k volání `GsmConfig.waitUntilReady`. Po navázání spojení je automaticky volána funkce `GsmConfig.moduleIsReady`, která zajišťuje, že se uskuteční telefonní hovor na číslo +420602139922. Poslední ukázanou funkcí je `Call.incomingCall`, která definuje, že se má při příchozím telefonním hovoru rozsvítit LED dioda.

```
module GsmC {
  uses interface Boot;
  uses interface Leds;
  uses interface GsmConfig;
  uses interface Call;
}
implementation {

  event void Boot.booted() {
    call Leds.led0Off();
    call GsmConfig.waitUntilReady();
  }

  event void GsmConfig.moduleIsReady() {
    call Call.makeCall((uint8_t*)"+420602139922");
  }

  event void Call.callDone( error_t error ) {
  }

  event void Call.incomingCall( uint8_t* number ) {
    call Leds.led0On();
  }

  ...
}
```

Zdrojový kód 3.4: Ukázka souboru GsmC.ns

Kapitola 4

GSM technologie

Zkratka GSM v originále znamená Global System for Mobile Communications, původně však vznikla z francouzského Groupe Spécial Mobile. Jedná se o nejpopulárnější standard pro mobilní komunikaci na světě. Jeho navrhovatel GSM Association odhaduje, že 80% celosvětového mobilního trhu využívá právě tento standard. Není se čemu divit, když je používaný třemi miliardami lidí ve více jak 200 zemích. [8] Navíc díky mezinárodním smlouvám mezi jednotlivými operátory je pomocí roamingového systému možno spojit prakticky jakékoliv dvě zařízení na světě. GSM se odlišuje od svých předchůdců tím, že signalizační i hlasový kanál jsou přenášeny digitálně. Díky tomu se používá označení 2G nebo-li mobilní systém druhé generace.

GSM standard také přišel s levnou implementací služby pro odesílání krátkých textových zpráv SMS (short message service). Dalším přínosem je celosvětově unifikované telefonní číslo 112 pro nouzové volání.

V nové verzi standardu z roku 1997 byla do specifikace přidána podpora GPRS. General Packet Radio Service v rámci systému GSM implementuje mobilní datovou službu, která se používá pro odesílání zpráv MMS a pro připojení do sítě internet. GPRS nám poskytuje rychlostní pásma od 56 do 114 kbit/s.

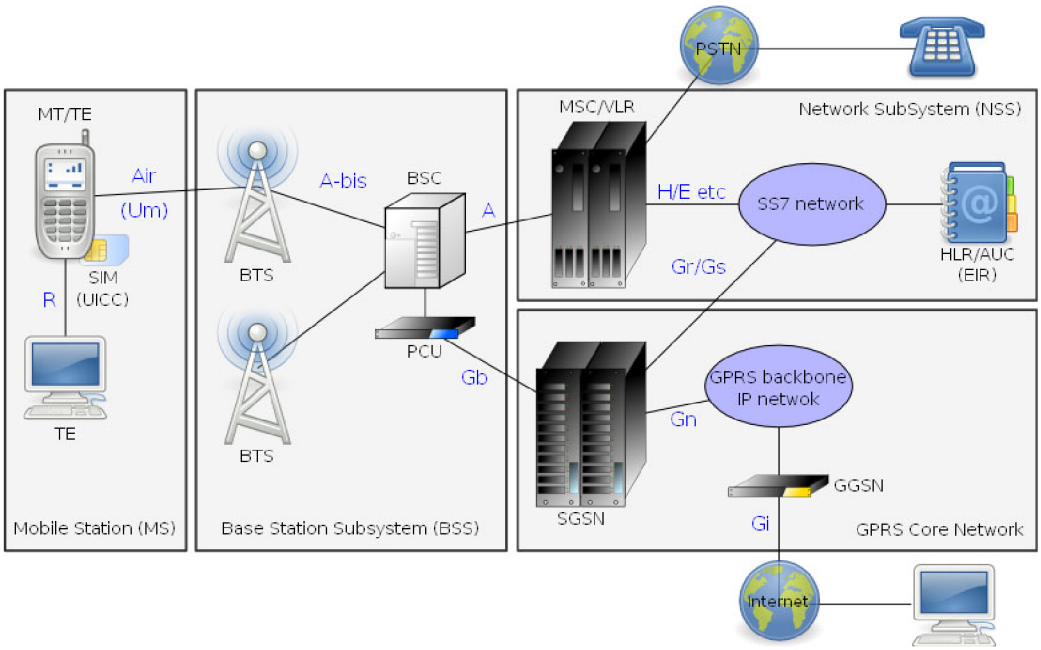
4.1 Hierarchie sítě

Jedná se o celulární sít, to znamená, že je počítáno s rozdělením území na menší celky. GSM modul se přihlašuje do sítě vyhledáním nejbližší buňky. V každé buňce se nachází vysílač/přijímač. Fungování sítě menších buněk má na starosti kontrolní středisko. Hlavní výhodou tohoto rozdělení sítě je znovupoužitelnost frekvencí — různé buňky mohou používat stejné frekvence. Telefon se sám přizpůsobí při přechodu z jedné buňky do druhé. Je několik typů buněk, které se liší dosahem, způsobem instalace a velikostí antén. GSM sítě fungují na různých frekvencích zpravidla na 900MHz a 1800MHz.

Je velmi komplikované pojmout všechny jednotky, které jsou součástí organizace sítě. Samotný zákazník o nich většinou nemá ani ponětí. Jednoduše by se daly rozdělit do několika oblastí:

- Base Station Subsystem (BSS) — sekce má na starosti spojení mobilního GSM zařízení se zbytkem sítě. Zařízení pracující v této sekci jsou: Base transciever station (BTS), který zodpovídá za příjem a odesílání signálů a Base station controller (BSC) s Packet Control Unit (PCU), které vykonávají logiku této sekce.

- Network and Switching Subsystem (NSS) — tato sekce obstarává spojování hovorů a přepojování sítí. Subsystém obsahuje Mobile switching center (MSC), jenž je základním prvkem pro přenos v GSM zodpovědný za správu volání a SMS zpráv. Mimo jiné do této skupiny patří i Authentication centre (AUC), které zodpovídá za správnou autentifikaci každé SIM karty vrámci sítě.
- GPRS Core Network — jedná se o část GSM sítě integrovanou do NSS, která dovoluje používat technologii paketově přepínaných přenosů, jež je používaná při datových přenosech.
- Operations support systém (OSS) — jedná se o počítačový systém zastávající správu sítě, automatickou konfiguraci a podobné



Obrázek 4.1: Zjednodušená struktura Gsm sítě

4.2 SIM karta

SIM je zkratka pro subscriber identity module. SIM karty obsahují čip a jsou vyjímatelné z mobilních telefonů nebo modemů, aby mohli být tato zařízení nezávislá na poskytovateli spojení. Každá karta obsahuje unikátní service-subscriber key (IMSI), aby bylo možné identifikovat uživatele v síti pomocí telefonu nebo počítače. Dále bezpečnostní a kódovací informace, dočasné informace spojené s lokální sítí, seznam služeb, ke kterým má uživatel přístup, a je blokována dvěmi hesly. Heslo PIN se používá pro běžné použití a PUK v případě zablokování karty.

Dále je možno na SIM uložit uživatelský seznam telefonních kontaktů a několik SMS zpráv. Přístup k jakémukoliv z těchto údajů je povolen až po odblokování zadáním validního PIN (personal identification nuber).

4.3 Modem Teltonika

Pro náš projekt jsme zvolili GSM modem Teltonika ModemCOM/G10. Jedná se o jednoduchý komerční modem, který nám nabízí všechny základní funkce. Dovoluje nám práci se SIM kartou, přijímat i zahajovat telefonní hovory, odesílat zprávy SMS a připojení k internetu pomocí GPRS. Dokonce pro naše pohodlí implementuje i jednoduchý TCP/IP stack. Ovládáný je pomocí AT příkazů a jako komunikační linku používá RS-232 rozhraní.

4.3.1 Spojení modemu s PC

Modem se jednoduše připojí do COM portu osobního počítače. Pro běžnou práci s modemem máme možnost nainstalovat grafickou uživatelskou aplikaci, která celou komunikaci a zadávání AT příkazů obstará za nás. Toto ovšem nebyl náš záměr a proto jsme modem ovládali i pomocí programů Hyperterminal a Minicom.

Pro úspěšnou komunikaci po lince RS-232 je potřeba znát přenosové parametry. Náš modem pracuje rychlostí 115200 bitů za sekundu, posílá 8 bitů, nepřenáší žádnou paritu a používá jeden stop bit.

4.4 AT příkazy

AT příkazy jsou anglicky také nazývané jako Hayes command set, neboť vznikly původně pro modem Hayes Smartmodem v roce 1977. Tehdy se jednalo o revoluční postup, protože pomocí nich lze modem velice prakticky ovládat. Je možné vytáčet čísla, upravovat nastavení, dokonce i měnit hlasitost reproduktoru. Není divu, že je brzy začali využívat i ostatní výrobci pro své modemy. Vznikl tak pro modem pojem „AT kompatibilní“.

AT příkazů je celá řada a zpravidla žádný modem neobsahuje všechny. Výrobci si dokonce příkazy začali upravovat a vytvářet vlastní, takže dnes se můžeme setkat s rozdělením na základní instrukční sadu a rozšířenou instrukční sadu. Jestli příkaz patří do rozšířené instrukční sady poznáme podle toho, že začíná na „AT+“.

4.4.1 Obecná pravidla

Obecně každý příkaz začíná znaky AT, což je zkratka pro anglické attention. Stejně tak každý příkaz sestává pouze ze zobrazitelných znaků. Specielní roli hraje znak `<CR>`, který slouží k ukončení příkazu. Je možné nejednou odeslat i více příkazů, tehdy je oddělujeme středníkem a sekvenci ukončíme `<CR>`.

Součástí AT příkazů může být i posílání numerických a textových hodnot. Tehdy numerické hodnoty zadáváme normálně v desítkové soustavě a textové hodnoty musí být vždy ohraničeny uvozovkami.

AT příkazy jsou čtyř druhů:

```
ATDL<CR>              <-- Běžný AT příkaz
AT+CGMI=?<CR>         <-- Dotaz, jestli je daný příkaz podporován
AT+CPIN?<CR>          <-- Dotaz na nastavenou hodnotu
AT+CPIN="9045"<CR>    <-- Nastavení příslušné hodnoty parametru
```

Na každý legitimní AT příkaz dostaneme od modemu odpověď v tomto formátu:

```
AT+CPIN?<CR>
<CR><LF>+CPIN: READY<CR><LF> <-- Odpověď na dotaz (nepovinné)
```

```
<CR><LF>OK<CR><LF>              <-- Výsledný kód operace OK/ERROR
```

Ne vždy ovšem modem pouze pasivně čeká na AT příkaz, aby mohl zaslat odpověď. Je tu jistá sada signálů nazývaná jako nevyžádané zprávy (unsolicited result codes). Těmito zprávami upozorňuje modem uživatele, že nastala zajímavá událost, jako je příchozí hovor nebo SMS. Některé modemy tímto způsobem umí signalizovat i vyjmutí SIM karty za běhu. Například příchozí hovor modem signalizuje zasíláním opakovaného textu `RING`.

4.4.2 Základní příkazy

Tabulka 4.1 obsahuje souhrn základních příkazů, které jsou obecně platné a fungují pro většinu GSM modemů. Vyzkoušeny byly pouze na našem testovacím modemu Teltonika.

Příkaz	Popis
`ATH`	Položí příchozí hovor.
`ATD+420609331121;`	Vytočí telefonní číslo. (Zde je výjimka, jedná se sice o zadání řetězce, ale není uzavřený v uvozovkách.) Pokud je hovor odmítnut, odpoví modem `BUSY`. Podaří-li se spojení navázat nedá modem žádnou odpověď. A pokud je po té hovor ukončen příjemcem, vrátí modem `NO CARRIER`.
`ATDL`	Vytočí znovu předchozí číslo.
`ATA`	Příjme příchozí hovor.
`AT+CPIN?`	Zjistíme, zda-li je potřeba zadat PIN pro odblokování karty. Odpověď `+CPIN: READY` značí, že je karta odblokována.
`AT+CPIN="9045"`	Pomocí příkazu zadáme PIN pro odblokování SIM karty.
`AT+CMGS="+420609224499«CR>Hola hola, tam Weigl!<Ctrl+z>`	Příkaz odešle SMS na dané číslo. Zadávaní SMS je ukončeno znakem substitute (Ctrl+Z, 0x1a)
`AT+CMGL`	Vypíše seznam zpráv.
`AT+CMGR=<n>`	Vypíše danou zprávu.
`AT+CMGD=<n>`	Smaže danou zprávu.
`AT+CNMI=<mode>,<mt>,<bm>,<ds>,<bfr>`	Nastaví upozornění na nově příchozí SMS pomocí nevyžádaných zpráv.
`AT+CMEE=<n>`	Nastaví detailnější vypisování chyb (0 až 2).
`ATE<n>`	0 vypne přeposílání znaků zpátky ke klientovi (takzvané echo), 1 echo zapíná

Tabulka 4.1: Základní AT příkazy

4.4.3 Formát zpráv SMS

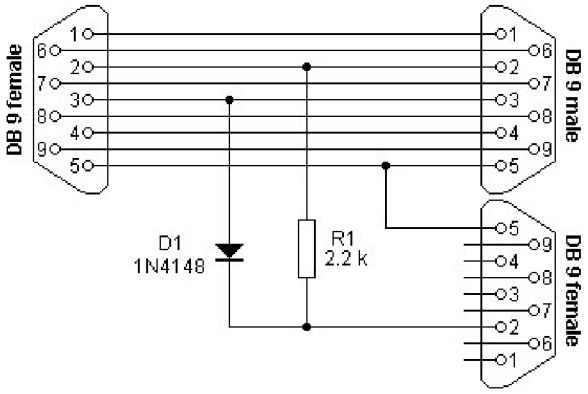
European Telecommunications Standards Institute (dále ETSI) definuje zprávy SMS pomocí PDU módu. Jedná se o kódování SMS zprávy do 8 nebo 7 bitového formátu. Při použití

7 bitového formátu dostáváme délku SMS na známých 160 znaků, při použití osmibitového kódování máme SMS o délce 140 znaků. Osmibitové kódování se většinou nepoužívá pro přenášení SMS obsahujících text, ale mobilní telefon danou zprávu zobrazí jako obrázek nebo použije jako vyzvánění. [15]

Většina modemů dovoluje nastavit, jestli chceme pracovat s SMS v přívětivém textovém formátu, nebo jestli je chceme přímo definovat pomocí binárního PDU formátu. Dále v této práci se budeme zabývat pouze prací s SMS v textovém módu.

4.4.4 Práce s GPRS

AT příkazy pro práci s GPRS patří mezi ten typ příkazů, které si výrobci definují, jak se jim zlíbí. Není definovaný žádný standard pro AT příkazy na aktivaci GPRS spojení pomocí poskytovatele nebo pro vytvoření internetového soketu a připojení k serveru. Proto jsme se v tomto směru rozhodli postupovat netypicky a sice použít dodávanou uživatelskou aplikaci pro práci s modemem a připojit se pomocí ní k internetu. Celou transakci AT příkazů jsme se rozhodli odposlouchávat pomocí třetího počítače a vyrobeného kabelu na monitorování linky RS-232, jehož schéma můžeme vidět na obrázku 4.2. [3]



Obrázek 4.2: Monitor na lince RS-232

Zjímavá část část odchycené komunikace je uvedena níže.

```
AT+CGDCONT=1,"IP","PAP:internet"
OK
ATE0V1
OK
ATATS0=0
OK
ATE0V1
OK
ATDT*99
CONNECT
~'}#Ŕ!}!}!} }8}"}&} } } } }#}Ŕ#}5}&tHZ}'}"}(}"Zn~~_}#Ŕ!}$} } "
   }-}#}&}1}$}&N}3}7}!/-ů))VC
```

Ukázka komunikace modemu s PC

Jak je vidět, modem nastaví PAP autentizaci, používanou pro protokol PPP. [13] A po

vytočení čísla pokračuje v komunikaci binárním tokem. Jedná se o málo průhledný způsob přenášení dat, který by se obtížně ladil. Proto jsme upustili od implementování této metody připojení k internetu. Mohli jsme si to dovolit, jelikož nám modemy Teltonika naštěstí nabízí ještě jednu možnost a to práce s internetem pomocí implementovaných TCP/IP AT příkazů. [5]

V kódu 4.1 je ukázka nastavení a aktivovaní GPRS profilu, vytvoření soketu, připojení k serveru a jednoduché komunikace. Modemy Teltonika bohužel nemají implementovaného DNS klienta, takže se zatím můžeme připojit pouze přímo k IP adrese.

Nastavení jako Access Point Name (APN) a jméno s heslem jsou údaje specifické pro každého operátora. Budeme předpokládat, že náš operátor vyžaduje pouze APN. (Momentálně každý operátor v ČR vyžaduje pouze APN.)

```
AT+NPSD=0,1,"internet"        // nastavit APN profilu 0 na
   internet
OK
AT+NPSDA=0,1                  // uložit profil 0 do NVM
OK
AT+NPSDA=0,3                  // aktivovat profil 0 z~NVM
OK

AT+NSOCR=6                    // vytvoření soketu 6 = TCP
+NSOCR: 0
OK

AT+NSOCO=0,"83.240.75.88",80 // otevření soketu
OK

AT+NSOWR=0,18
GET / HTTP/1.0                // zápis do soketu
+NSOWR:0,18
OK

+NUSORD: 0,1024               // nevyžádaná zpráva značící, že
   máme v~soketu 0 příchozí data o~velikosti 1024 bytů

AT+NSORD=0,15                 // čtení ze soketu
+NSORD: 0,15,"HTTP/1.0 200 OK"
AT+NSOCL=0                    // zavření soketu

+NUSOCL: 0                    // může přijít jako nevyžádaná
   zpráva, když server uzavře soket
```

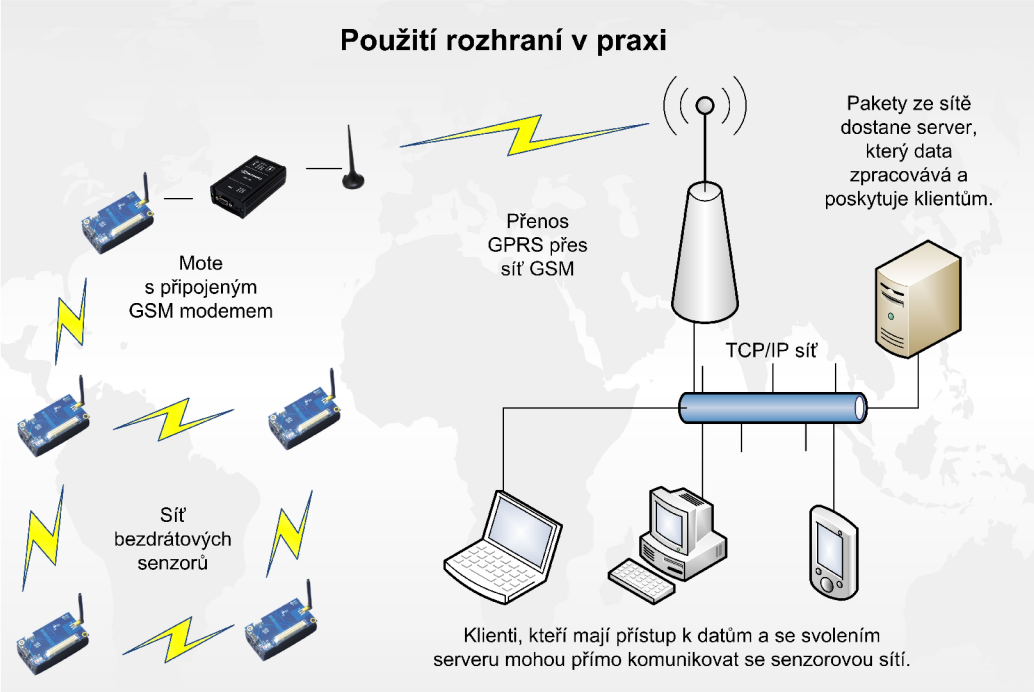
Zdrojový kód 4.1: Ukázka práce s GPRS

Kapitola 5

Návrh rozhraní

Cílem našeho projektu je dosáhnout toho, aby námi implementované rozhraní bylo schopné otevřít cestu aplikacím, které potřebují vzdáleně komunikovat se senzorovými sítěmi pomocí uživatelských multiplatformích klientů, jak to ilustruje obrázek 5.1.

V této kapitole se pokusíme osvětlit všechny problémy, které jsou spjaty s propojováním GSM modulu a bezdrátového senzoru. Samozřejmě také popíšeme cestu k jejich optimálnímu řešení.



Obrázek 5.1: Návrh potenciální aplikace

5.1 Hardwarové požadavky

Platformu Iris jsme se rozhodli spojit s modemem Teltonika pomocí rozhraní RS-232. Modem nabízí toto rozhraní přímo tak, jak je RS-232 specifikováno. Platforma Iris nabízí pouze

obecné rozhraní UART, které je ještě navíc vyvedené ne 51pinovém konektoru. Zde proto popíšeme návrh a výrobu převodního mostu mezi komponentami.

5.1.1 Rozhraní RS-232

RS-232 je rozhraní, jehož definice pochází z roku 1962. Pro svoji jednoduchost je však oblíbené do dnes. Pracuje jako jednoduchý univerzální asynchronní přijímač/vysílač (UART). Pro komunikaci principem UART nám stačí pouze tři vodiče (příjem, vysíláni, zem). Jedním odesílá jedna strana data a na druhém očekává data od strany druhé. Standard RS-232 definuje i další vodiče, které přenáší signály, jako handshake mechanismy používané při komunikaci se staršími modemy.

Standardem RS-232 je doporučeno přenášet data napěťovými úrovněmi ±12 V. Přijímač by měl rozlišovat logické úrovně již od napětí ±3 V. Časem výrobci upustili od komunikace na ±12 V (pro kterou byla garantovaná délka linky až 20 metrů) a začali vyrábět zařízení pracující na úrovni ±5,5 V. Pro tato zařízení se vžilo označení RS-232 kompatibilní.

5.1.2 Čip MAX3232

Nevýhodou RS-232 jsou právě jeho neobvyklé napěťové úrovně, protože většina zařízení pracuje na 5 V nebo 3,3 V logice. Pro převod mezi napěťovými úrovněmi byl firmou Maxim vyvinut obvod MAX232, který dokáže při jednotném napájení 5 V generovat napětí až ±7,5 V. Zvýšení napěťové úrovně je dosaženo pomocí patentovaného systému, který funguje na principu nábojových pump (charge-pumps). Jedná se o systém čtyř externích kondenzátorů, které jsou při poklesnutí napájení střídavě nabíjeny a vybíjeny. Pokračovatelem této technologie je obvod MAX3232, který dokáže pracovat i při napájení 3 V a poradí si i s vyššími přenosovými rychlostmi linky. Postupem času začali vznikat další podobné obvody, ať již firmou Maxim, nebo jinými, které upravovaly základní vlastnosti MAX3232. Jako příklad můžeme uvést převodníky MAX3233 a MAX3235, které díky zabudovaným nábojovým pumpám pro svou práci nevyžadují externí kondenzátory.

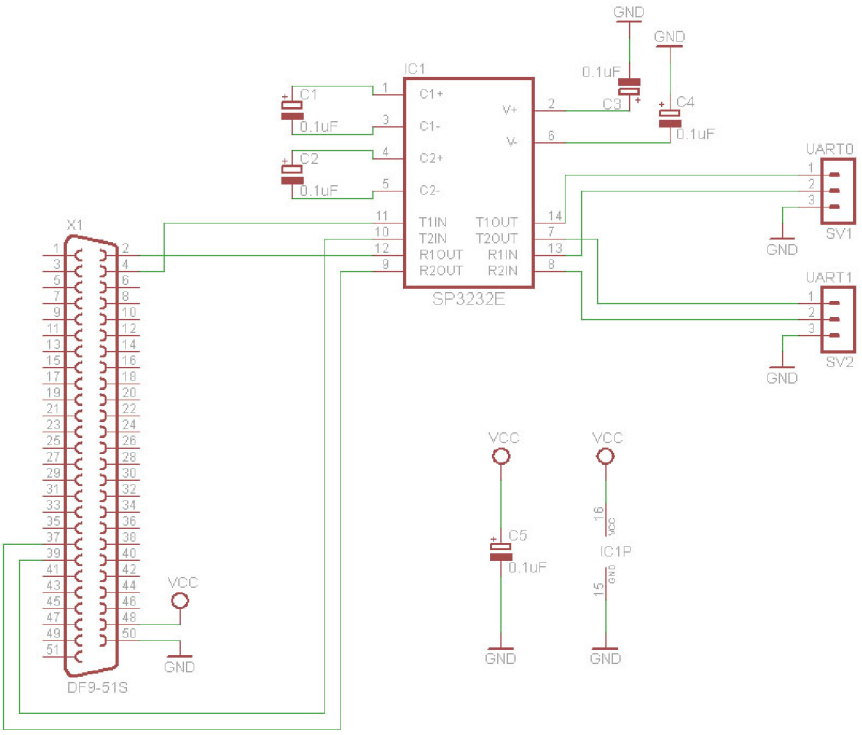
5.1.3 Čip SP3232

Jak již bylo zmiňováno dříve, celý mote je napájen pouze dvěmi tužkovými bateriemi a jejich celkové napětí nedosahuje ani 3 V, které jsou dle specifikace zapotřebí pro práci s MAX3232. [4] Naštěstí se povedlo najít obvod SP3232 dodávaný firmou Sipex, který slibuje bezchybnou práci až do napětí 2,7 V. [2] Díky tomuto obvodu můžeme vytvořit poměrně spolehlivý komunikační kanál, i když bude docházet ke snižování napětí zdroje jeho průběžným vybíjením. Tento obvod je pinově kompatibilní s obvodem MAX3232 a pro správnou práci k němu musíme přidat 5 externích kondenzátorů o velikosti 100 nF. Může se jednat o libovolný typ kondenzátorů.

Zapojením SP3232 dle schématu (viz. obrázek 5.2) dostáváme převodní most schopný pojmout dvě UART linky. Toho samozřejmě využijeme a použijeme obě UART rozhraní, které Iris z procesoru ATMega128L vyvádí pomocí 51pinového konektoru. K jednomu rozhraní připojíme GSM modul, druhé můžeme použít pro vypisování ladících informací.

5.2 Komunikace platformy a modulu

Z abstraktu práce vyplývá, jaké nároky budeme mít na GSM modul. Tyto požadavky si nyní konkretizujeme a pokusíme se odhadnout a předejít úskalím, která by mohla nastat



Obrázek 5.2: Schéma zapojení SP3232

při jejich implementaci.

5.2.1 Rozdělení ovladače

Abychom mohli v nesC navrhnout správné softwarové rozhraní pro komunikaci s GSM modulem, musíme si služby, které nám modem nabízí, rozdělit do několika nezávislých podskupin. Máme proto tyto základní skupiny, které se navzájem neovlivňují:

- Konfigurace modemu — do této podskupiny zahrneme detekci, zda-li je modem vůbec připojen, nastavení výpisů chyb, upozorňování na příchozí zprávy a vypínání echo režimu
- Práce se SIM kartou — zde implementujeme v první řadě zjišťování, jestli je karta blokována pomocí PIN a také jeho zdávání
- Práce s telefonními hovory — sem patří vytáčení libovolného telefonního čísla, upozornění na příchozí hovor a jeho ukončení nebo přijetí
- SMS rozhraní — tato skupina příkazů má na starosti základní obsluhu SMS zpráv, ta obsahuje zaslání textové zprávy libovolnému příjemci, upozornění na příchozí zprávu, přečtení uložené zprávy a mazání zpráv
- GPRS — patrně nejobsáhlejší skupina, musí zajistit správné nastavení APN hodnot pro připojení k poskytovateli, musí mít na starosti vytváření soketů, spojování soketů se servery, odesílání uživatelských dat, upozornění na nově příchozí data, čtení příchozích dat a obsluhu ukončování spojení

Jednotlivé skupiny budeme v nesC definovat jako rozhraní, abychom vytvořili univerzálně použitelný ovladač GSM modulu. Tím se pro potenciálního budoucího programátora celé použití ovladače výrazně zprůhlední. Například aplikace, která zasílá pozorovaná data na vzdálený server nebude potřebovat rozhraní pro práci s SMS. Nebo při práci s odblokovanou SIM kartou nebudeme potřebovat rozhraní pro zadávání PIN.

5.2.2 Zpracovávání příchozích AT zpráv

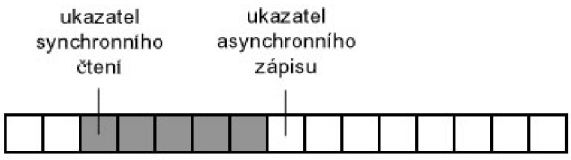
Dalším problémem, který musíme zvážit při návrhu komunikace s hardwarem v nesC, je implementace synchronního a asynchronního režimu. Ovladač pracující s UART musí pracovat v asynchronním režimu, protože každý příchozí byte je v procesoru signalizován přerušením. Rutiny pro obsluhu přerušení však musí být co možná nejkratší (řádově několik instrukcí), což nám neposkytuje dostatek času na vyhodnocování, o jakou příchozí zprávu se právě jedná.

Řešení tohoto problému by bylo elementární, pokud bychom věděli, že se modem chová naprosto stavově a zasílá nám zprávy, jen pokud si to od něj vyžádáme pomocí AT příkazů. Modem ale naopak podporuje dříve zmiňované nevyžádané zprávy, kterými nás upozorňuje na nečekané události, které nastaly. Pro nás budou významné tyto čtyři nevyžádané zprávy: příchozí telefonní hovor, příchozí zpráva, příchozí data na soketu a ukončení spojení soketu vzdáleným serverem.

V nesC je pravidlem, že vyšší modul poskytne modulu z nižší vrstvy ukazatel na místo v paměti, kam si přeje, aby byla zapsána příchozí data. Modul na nižší vrstvě tedy sám žádnou paměť nealokuje. Toto chování nemůžeme přímo dodržet, protože v době příchodu asynchronní zprávy nevíme (a nemáme dostatek času zjišťovat), zda-li se jedná o odpověď na náš dotaz nebo o nevyžádanou zprávu modemu. Při přímém zápisu do poskytnuté paměti by se tudíž mohlo stát, že uložíme něco jiného, než co si modul z vyšší vrstvy vyžádal.

Proto pro práci s příchozími byty použijeme FIFO. Na jednom konci do něj budeme v asynchronním režimu zapisovat příchozí data a na druhé straně z něj budeme v synchronním režimu číst jednotlivé znaky a dekódovat přijaté zprávy.

Velikost FIFO nemusí být příliš veliká, protože dochází k okamžitému překopírování zpráv vyšší vrstvě. Nejdelší zprávy, které nám modem bude posílat, jsou příchozí data ze socketu a tam si můžeme sami vyžádat, kolik bytů nám má poslat najednou.



Obrázek 5.3: Práce vstupního FIFO

Kapitola 6

Implementace

V prvé řadě je potřeba rozlišit implementaci hardwarové a softwarové stránky projektu. Co se týče hardwaru, jedná se o navrhnutí tištěné desky, zadání do výroby a osazení. Softwarová část projektu zahrnuje nutné zásahy do zdrojových souborů TinyOS, implementaci driveru, který nám zprostředkuje komunikaci s modemem na nejnižší úrovni, a napsání ukázkové aplikace, jež bude demonstrovat funkčnost celého systému.

6.1 Hardware

6.1.1 Návrhový program EAGLE

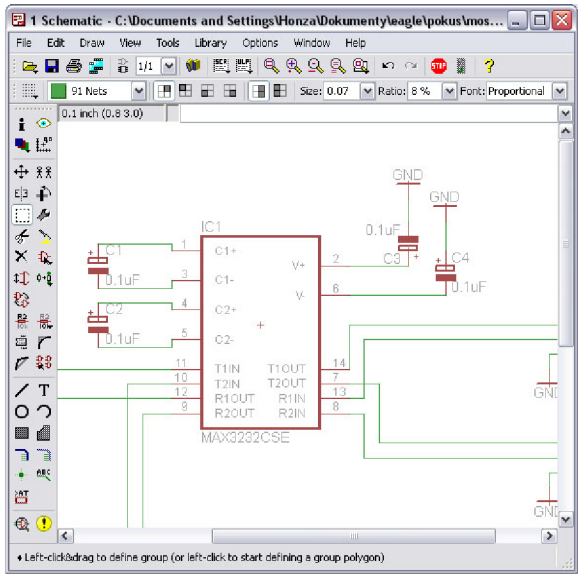
Jedná se o jeden z nejznámějších programů pro návrh obvodů a tištěných desek. Software je vyvíjen firmou CadSoft Computer z USA. Teší se velké oblibě převážně proto, že je ve své Light verzi pro nekomerční využití poskytován zdarma. Tato verze je omezena maximální velikostí navrhované desky (nesmí přesáhnout 100 x 80 mm), počtem vrstev (maximálně dvě vrstvy) a můžeme vytvořit schéma pouze na jednom listu. Díky této firemní politice je využíván řadou profesionálů i nadšenců po celém světě a na internetu je volně k dispozici velké množství knihoven, které obsahují stovky uživatelských součástek.

Práce v programu EAGLE probíhá ve dvou fázích. Nejprve je třeba navrhnout v editoru elektronické schéma. K tomuto nám pomáhá rozsáhlá knihovna nabízených součástek, které principielně pouze pospojujeme a přehledně umístíme na list.

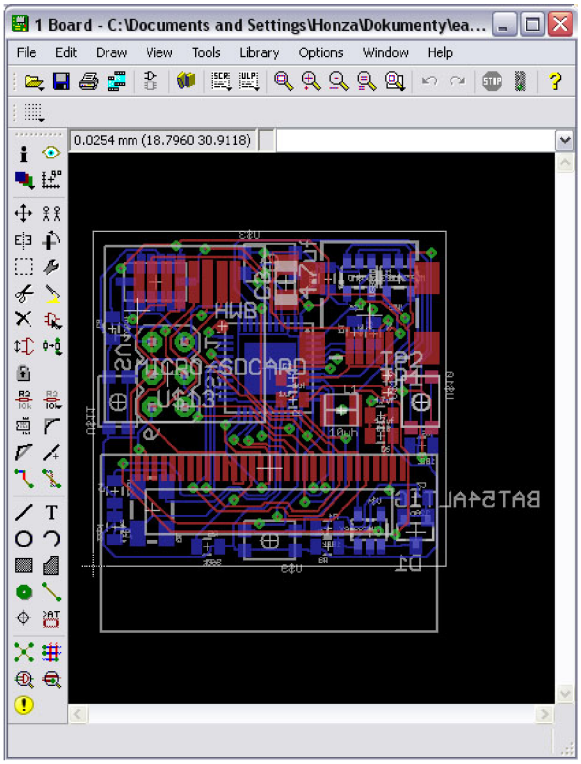
Po dokončení schématu můžeme přejít ke druhé fázi. Jedná se o samotné rozvržení součástek a spojů mezi nimi na desce. Nejprve na desku rozložíme součástky tak, aby se k sobě logicky hodili a po té můžeme spustit automatické vypočítání cest mezi nimi. Většinou se automatickému algoritmu nepovede vytvořit optimální cestu tak, aby spojil všechny body, přesto se jedná o velmi užitečného pomocníka. Nedokončené spoje je potřeba vyladit ručně posouváním naskládaných součástek nebo vytvořením drátů a přechodů.

6.1.2 Převodní můstek

Z finančních důvodů jsme se rozhodli použít pro návrh našeho rozšiřujícího převodního mostu pouze běžnou jednostrannou desku. Estetičtěji by působilo využít oboustrannou prokovenou technologii, ale ta je řádově dražší, než námi zvolená základní. Po spojení 51pinových konektorů leží desky těsně nad sebou, proto jsme rozložení součástek museli volit tak aby po přiložení desek ležely všechny součástky mimo Iris (viz. obrázek 6.3).



Obrázek 6.1: EAGLE Lite - editor schémat

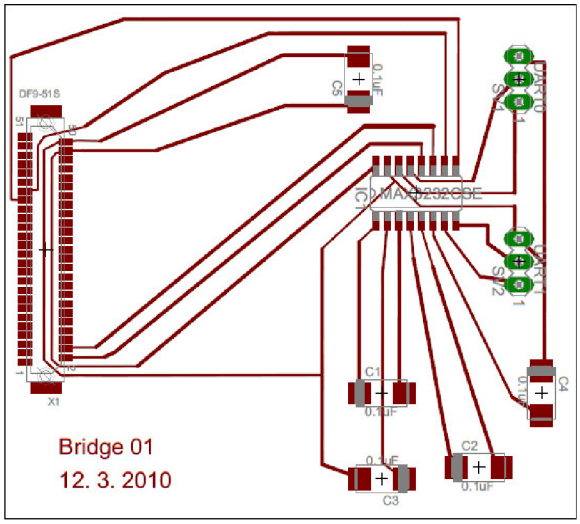


Obrázek 6.2: EAGLE Lite - editor desky

6.2 Software

6.2.1 Úpravy stávající platformy

Pro úspěšnou komunikaci naší platformy s GSM modulem přes UART rozhraní si musíme být jisti, že používáme na obou koncích stejné nastavení sériové linky. Změnit parametry



Obrázek 6.3: Deska převodního můstku

pro GSM modul není principielně možné, proto přizpůsobíme nastavení Iris. Pro modem Teltonika platí, že pracuje rychlostí 115200 bitů za sekundu, posílá 8 bitů, nepoužívá žádnou paritu a používá jeden stop bit. V základním nastavení Iris komunikuje přes UART rychlostí 57600, ostatní parametry jsou stejné. Pro korektní upravení rychlosti bude potřeba pár zásahů do souborů naší platformy.

Platforma Iris vychází z platformy MicaZ a pro komunikaci přes sériovou linku používá její nastavení. Proto změníme konstantu používané rychlosti MicaZ platformy. Správné nastavení registrů UART už za nás udělá TinyOS sám. Budeme upravovat soubor, jak je uvedeno ve zdrojovém kódu 6.1.

```
enum {
  // PLATFORM_BAUDRATE = 57600L
  PLATFORM_BAUDRATE = 115200L
};
```

Zdrojový kód 6.1: Úprava ./tos/platforms/micaz/hardware.h

Pro korektní chování nastavíme i správný přepočet hodinových cyklů časovače. Jedná se o práci s UART přes rozhraní UartByte, které používá blokovací režim při přijímaní bytů. Sice ho nebudeme používat, ale přesto budeme pracovat čistě. Přepočet je třeba nastavit v souboru dle kódu 6.2.

Za předpokladu, že bychom někdy v budoucnu potřebovali změnit další parametry sériové linky, je vhodné si nadefinovat konstanty pro správné nastavení registrů dle dokumentace procesoru. [6] Konstanty zapíšeme do hlavičkového souboru, jak je uvedeno v kódu 6.3

Z mikrokontroléru jsou do 51pinoého konektoru vyvedeny dvě UART linky. V tuto chvíli by mělo být rozhraní UART0 nastaveno tak, aby vyhovovlao specifikace modemu. Pokud je pořeba pracovat i s rozhraním UART1, můžeme jeho parametry nastavit separátně, jak napovídá zdrojový kód 6.4. Demonstrační kód představuje ilustrační nastavení druhého portu. Používáme rychlost 57000 bitů za sekundu, sudou paritu, jeden stop bit a přenášíme najednou 8 bitů.

```
if (PLATFORM_BAUDRATE == 19200UL)
  m_byte_time = 200; // 1 TMicor ~= 2.12 us, one byte = 417us
     ~= 200
else if (PLATFORM_BAUDRATE == 57600UL)
  m_byte_time = 68;  // 1 TMicor ~= 2.12 us, one byte = 138us
     ~= 65
else if (PLATFORM_BAUDRATE == 115200UL)
  m_byte_time = 34;  // 1 TMicor ~= 2.12 us, one byte = 69us
     ~= 34
```

Zdrojový kód 6.2: Úprava ./tos/chips/atm128/Atm128UartP.nc

```
enum {
  ATM128_UART_DATA_SIZE_5_BITS = 0,
  ATM128_UART_DATA_SIZE_6_BITS = 1,
  ATM128_UART_DATA_SIZE_7_BITS = 2,
  ATM128_UART_DATA_SIZE_8_BITS = 3,
};

enum {
  ATM128_UART_1x_STOP_BIT = 0,
  ATM128_UART_2x_STOP_BIT = 1,
};

enum {
  ATM128_UART_MODE_ASYNC = 0,
  ATM128_UART_MODE_SYNC  = 1,
};

enum {
  ATM128_UART_PARITY_NONE = 0,
  ATM128_UART_PARITY_EVEN = 2,
  ATM128_UART_PARITY_ODD  = 3,
};

enum {
  ATM128_UART_RISING_EDGE  = 0,
  ATM128_UART_FALLING_EDGE = 1,
};
```

Zdrojový kód 6.3: Možná úprava ./tos/chips/atm128/Atm128Uart.h

Pro komunikaci přes sériové rozhraní UART se používá obecný ovladač `PlatformSerialC`, který je definovaný pro všechny čipy podporující sériovou komunikaci. Například při práci s kontrolérem ATmega1281 se modul napojí na ovladač `Atm128Uart0C`. Tato komponenta nabízí rozhraní `StdControl` pro zapínání a vypínáni linky, `UartStream` pro odeslání toku dat a `UartByte` pro odesílání jednotlivých bitů.

```
command error_t Uart1Init.init() {
...
mode.bits = (struct Atm128_UCSRC_t) {
  ucsz:ATM128_UART_DATA_SIZE_8_BITS,  //!< Velikost znaku
  usbs:ATM128_UART_1x_STOP_BIT,       //!< Stop bity
  upm: ATM128_UART_PARITY_EVEN        //!< Parita
};
...
ubrr1 = call Atm128Calibrate.baudrateRegister(57000L);
```

Zdrojový kód 6.4: Možná úprava ./tos/chips/atm128/HplAtm128UartP.nc

Pro naše potřeby je nevýhodné používat přímo ovladač `PlatformSerialC`, protože podporuje jen jednu sériovou linku a kvůli rozhraní `UartByte`, které my stejně nepoužíváme, obsadí hardwarový časovač. Proto jsme si, z důvodu optimalizace našeho ovladače, napsali vlastní rozhraní `Atm128UartLite0C` a `Atm128UartLite1C` koncipované přesně pro naší aplikaci a platformu Iris. Jelikož se jedná o komponenty specifické pro procesor ATmega128 a ATmega1281, jsou uložené v adresáři `./tos/chips/atm128/`. Jedná se o odlehčené moduly, které nám nabízí pouze rozhraní `StdControl` a `UartStream`.

6.2.2 GSM rozhraní

Práce s GSM modulem byla rozdělena na pět hlavních kategorií tak, jak již bylo dříve zmiňováno kapitole 5.2.1. Finální modul se jmenuje `GsmModuleC` a poskytuje všechna tato rozhraní vyšším vrstvám. Závislosti modulů jsou znázorněny na obrázku 6.4.

Jak je z hierarchické struktury na obrázku patrné, komponenta `GsmModuleC` se skládá z většího počtu subkomponent.

Při postupu od shora dolů po `GsmModuleC` následuje modul `GsmTeltonikaP`, který zprostředkovává obsluhu všech nabízených rozhraní pro vyšší vrstvy. Hlavním úkolem komponenty je mimo jiné rozpoznávat a sestavovat AT příkazy modemu. Tento modul není univerzální a jak název napovídá, je specifický právě pro modemy Teltonika.

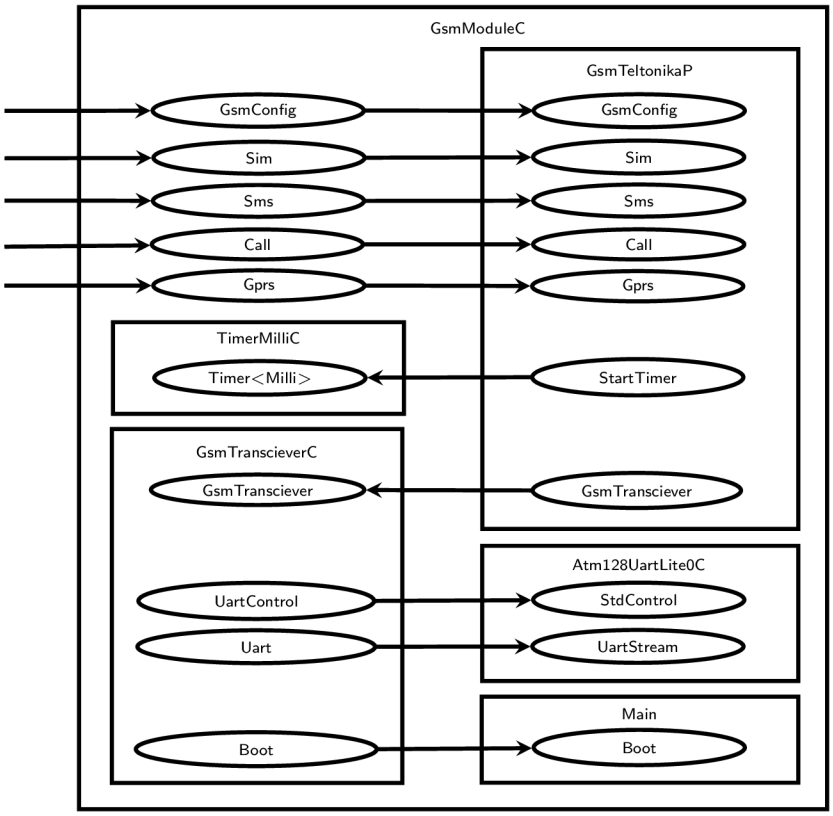
Pro svou správnou funkci vyžaduje `GsmTeltonikaP` časovač `StartTimer`, který používá při opakovaném vysílaní nebo když je potřeba opozdit zprávy pro modem. Modul s linkou UART nekomunikuje přímo, příjmem a odesílání dat pro něj zprostředkovává modul `GsmTranscieverC`.

Komponenta `GsmTranscieverC` implementuje dříve zmiňované FIFO pro ukládání příchozích AT zpráv. Modul využívá ovladač `Atm128UartLite0C`, který pracuje s UART rozhraním na nejnižší úrovni a je celý napsaný v asynchronním kódu. `GsmTranscieverC` v našem ovladači vytváří most mezi asynchronními a synchronními komponentami.

Komponenta `Main` nabízí rozhraní `Boot`, které informuje `GsmTranscieverC` o tom, že došlo k nastartování zařízení a že si má modul správně inicializovat ovladač UART.

6.2.3 Zajímavosti modemu Teltonika

Při implementaci komunikačního softwaru mezi modemem a senzorem jsme narazili na několik zajímavých problémů, které bychom rádi zmínili. Protože jsme pracovali pouze s jedním modemem, nemůžeme určit, jestli se jedná o záležitosti typické pouze pro modem Teltonika, nebo zda-li se jedná o problémy, které je potřeba řešit i u modemů jiných výrobců.



Obrázek 6.4: Schéma ovladače

V první řadě se jedná o to, že modem nemá žádnou vyrovnávací frontu (na rozdíl od našeho ovladače) a proto je schopný přijímat příkazy z linky UART pouze pokud nepracuje na jiném úkolu. Na to je třeba při psaní ovladače brát ohled a po odeslání jakéhokoliv příkazu je vždy nutno čekat na potvrzení od modemu, před tím, než je možno posílat další data.

Bohužel i po tom, co modem odešle příkaz potvrzující ukončení operace, je ještě chvíli „zamrzlý“. Proto v několika specifických případech bylo nutné pro správné provedení operací vkládat mezi odesílané příkazy několika milisekundové čekací rutiny. Jedná se především o příkazy spojené s GPRS, pro bližší informace odkazujeme na zdrojový kód v příloze.

Další zajímavou vlastností modemu je chyba při rychlém odmítnutí hovoru. Když modem zaznamená příchozí hovor, posílá v několikasekundových intervalech zprávu `RING`. Pokud mu jako odpověď přijde `ATH`, tak hovor odmítne, pokud mu dojde `ATA`, tak hovor příjme.

Takový příkaz mu však můžeme zaslat až po druhém signálu `RING`. Pokud modem obdrží příkaz `ATH`/`ATA` dřív, hovor sice odmítne/přijme, ale nedá o tom zprávu GSM síti. Což se projeví tak, že nám přestane přes linku přicházet oznámení `RING`, ale telefonující osoba se o ničem nedozví a stále slyší signalizační vyzvánění.

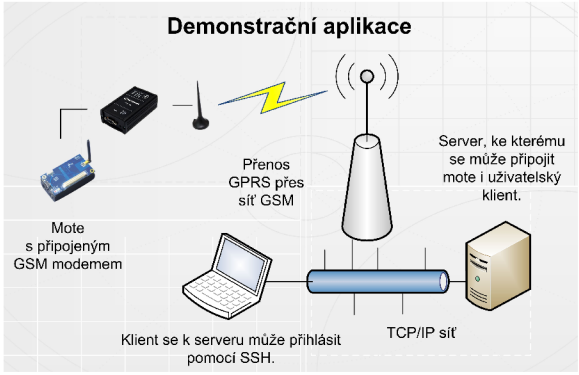
Další brzdící a opakovací rutiny bylo potřeba vkládat při inicializaci GPRS. Modem potřebuje, aby mezi odblokováním SIM karty a aktivací GPRS uběhla doba několika vteřin. Ovladač se proto pokouší inicializovat GPRS pravidelně v tří sekundovém intervalu, dokud od modemu nedostane úspěšnou odpověď.

Kapitola 7

Demonstrační aplikace

Demonstrační aplikace byla vytvořena tak, aby ve zjednodušené míře splňovala všechny dříve definované požadavky na budoucí používání rozhraní. Slouží proto pro ukázku funkčnosti a potenciálu vytvořeného rozhraní. Ze zdrojových kódů `GsmAppC.nc` a `GsmC.nc` je také patrné, jak správně používat ovladač modemu. Proto by tato aplikace měla sloužit jako výchozí bod pro další navazující práce.

Ukázková aplikace pro prezentaci komunikace přes GPRS sestává ze třech oddělených účastníků. Prvním z nich je server, který běží na počítači s veřejnou IP adresou. Druhým účastníkem je uživatelský počítač, jež se připojí k serveru pomocí SSH, Telnetu nebo jiné aplikace pro vzdálený přístup. Uživatel po připojení na serveru spustí aplikaci, která čeká na připojení třetího účastníka, tím je mote Iris s modemem Teltonika. Uživatel je následně pomocí aplikace schopen komunikovat se vzdálenou platformou.



Obrázek 7.1: Schéma demonstrační aplikace

7.1 Server

Může se jednat o libovolný linuxový/unixový server, který má veřejnou IP adresu. Na tomto serveru by měl běžet SSH, Telnet nebo jiný program pro vzdálenou správu. Také zde musí být připravená zkompilovaná serverová aplikace a klient musí mít právo na její spuštění. Tato aplikace je součástí přílohy. Jedná se o jednoduchý server implementovaný v C a obsahuje vlastní `Makefile`, proto pro zkompilování stačí prostý příkaz `make`.

Při vytváření reálných komerčních nebo výzkumných aplikací se samozřejmě počítá s tím, že by server běžel trvale a zvládal by komunikovat paralelně s větším počtem ne-

závislých senzorových sítí. Pro demonstraci funkčnosti projektu však plně postačuje tato jednoduchá aplikace.

7.2 Uživatel

Uživatel se připojuje k serveru pomocí SSH/Telnet a spouští demonstrační aplikaci. Tento postup je zde zahrnut, protože na server je kladena podmínka, že musí mít veřejnou IP adresu, aby se k němu mohla připojit i Iris přes GPRS. S klientem se zde tedy počítá obecně s jakoukoliv platformou, na které je schopen běžet SSH nebo Telnet klient (každý osobní počítač, notebook nebo i smartphone).

Při implementaci reálných aplikací bychom na tomto místě samozřejmě doporučovali vytvořit přístup přes multiplatformní klientskou aplikaci nebo webové rozhraní, které je přijatelnější pro většinu běžných uživatelů.

Přihlášený uživatel spustí serverovou aplikaci. Ta čeká, než se k ní připojí klient – tedy mote Iris s modemem. Po úspěšném připojení klienta aplikace pracuje pouze jako převodní most a přeposílá zprávy od uživatele klientovi a zpět.

Uživatel může na klientské platformě Iris ovládat tři LED diody pomocí těchto příkazů:

- `<R/G/Y><1/0>` — Písmeno určuje barvu ledky (red, green, yellow) na Iris. Číslo `1` značí rozsvítit, zatímco `0` zhasnout. Tím dostaneme jednoduché příkazy jako `R0`, `Y1` a podobně.
- `?` — Klient odešle serveru stav, ve kterém se právě nacházejí všechny jeho ledky.

7.3 Klient

Klientský program je kompletně napsaný v jazyce nesC, proto je pro jeho správnou kompilaci potřeba mít nainstalované prostředí TinyOS. [9] Dále je potřeba upravit nainstalované prostředí TinyOS podle návodu, který je uvedený v kapitole 6.2.1.

Jedná se o aplikaci, která používá vytvořený ovladač modemu Teltonika, připojuje se k vzdálenému serveru pomocí GPRS a zapíná nebo vypíná na Iris LED diody. Po nahrání programu a zapnutí mote se Iris pokouší spojit s modemem. Pokud se mu podaří navázat spojení, zkusí zadat PIN (pokud je vyžadován). Po úspěšném odblokování SIM, zapne aplikace GPRS a pokusí se připojit ke vzdálenému serveru. Jakmile dojde k úspěšnému navázání kontaktu, aplikace čeká na příkazy a podle nich ovládá LED diody na Iris.

Pokud je potřeba zaslat SIM kartě jiný PIN, je možné ho nastavit změnou konstanty ve volání funkce `call Sim.pinEnter`. Pro změnu IP adresy nebo portu vzdáleného serveru musíme správně nastavit parametry funkce `call Socket.connect`. Všechny tyto úpravy se provádí v souboru `GsmC.nc`.

Kapitola 8

Závěr

Podařilo se úspěšně spojit bezdrátový senzor s GSM modemem a implementovat pro tento modem ovladač. Vytvořený ovladač umí odesílat a přijímat zprávy SMS a pracovat s telefonními hovory. Hlavním cílem projektu však bylo umožnit platformě přístup na internet prostřednictvím technologie GPRS. Ten se povedlo taktéž úspěšně splnit. Pro ukázku komunikace přes GPRS byla vytvořena přiložená demonstrační aplikace.

Při implementaci aplikace bylo dbáno na znovupožitelnost kódu, takže můžeme k platformě Iris s minimálními úpravami kódu připojit i modemy jiných výrobců. Stejně tak můžeme bez větších obtíží upravit implementaci ovladače a rozšířit aplikaci i na jiné platformy bezdrátových sítí.

Vzdálené spojení platformy Iris s uživatelským počítačem přes GPRS rozšiřuje možnosti využití senzorových sítí a značně zjednodušuje jejich nasazení do reálných terénních aplikací, jako například autonomní monitorování rozsáhlých ploch, detekce požárů v lesech, automatické zavlažování při nedostatečné vlhkosti pole a podobně.

Tato práce otevírá spoustu možností pro další projekty v oblastech bezdrátových senzorových sítí a jejich internetových aplikací. Jako další navazující projekty můžeme uvést implementaci DNS klienta nebo i jednoduchého webového serveru pro platformu Iris, který by zaručoval kontinuální sledování oblasti přístupné z libovolného místa na světě s minimálními nároky na klientské rozhraní.

Literatura

[1] ZigBee Technology: Wireless Control that Simply Works [online].
URL:
http://www.zigbee.org/imwp/idms/popups/pop_download.asp?contentID=5162,
Říjen 2003 [cit. 2010-05-15].

[2] SP3222E/3232E True +3.0V to +5.5V RS-232 Transceivers [online].
URL: http:
//www.exar.com/Common/Content/Document.ashx?id=619&LanguageId=1033, 2005
[cit. 2010-05-10].

[3] HW server představuje - Sériová linka RS-232 [online].
URL: http://hw.cz/rs-232, 2005 [cit. 2010-05-15].

[4] 3.0V to 5.5V, Low-Power, up to 1Mbps, True RS-232 Transceivers Using Four 0.1μF
External Capacitors [online].
URL: http://datasheets.maxim-ic.com/en/ds/MAX3222-MAX3241.pdf, 2007 [cit.
2010-05-15].

[5] AT Commands Manual [online].
URL: http://www.cse.iitd.ernet.in/~cs5050224/GPRS/Teltonika%20AT%
20commands%20EN%202007%2004%2027.pdf, 2007 [cit. 2010-05-15].

[6] ATmega640/V ATmega1280/V ATmega1281/V ATmega2560/V ATmega2561/V
[online].
URL: http://www.atmel.com/dyn/resources/prod_documents/doc2549.PDF, 2007
[cit. 2010-05-15].

[7] What's so good about mesh networks? [online].
URL: http://www.daintree.net/downloads/whitepapers/mesh-networking.pdf,
2007 [cit. 2010-05-15].

[8] GSM [online].
URL: http://en.wikipedia.org/wiki/GSM, 2010 [cit. 2010-04-20].

[9] TinyOS [online].
URL: http://www.tinyos.net/, 2010 [cit. 2010-05-11].

[10] Datasheet IRIS [online].
URL: http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/IRIS_
Datasheet.pdf, [cit. 2010-05-15].

[11] Datasheet MICAz [online].
URL: http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/MICAz_Datasheet.pdf, [cit. 2010-05-15].

[12] Levis, P.: TinyOS Programming [online].
URL: http://www.tinyos.net/tinyos-2.x/doc/pdf/tinyos-programming.pdf, 2006 [cit. 2010-05-15].

[13] [online], L. L.: PPP Authentication Protocols.
URL: http://tools.ietf.org/html/rfc1334, 2002 [cit. 2010-05-12].

[14] Spáčil, P.: Mobilní agenti v bezdrátových senzorových sítích. 2009.

[15] Straka, J.: PDU formát SMS zpráv.
URL: http://radio.feld.cvut.cz/personal/mikulak/MK/MK06_semestralky/PDUformatSMSzprav_StrakaJ.pdf, 2006 [cit. 2010-05-15].

Příloha A

Obsah CD

- `./klient/` Složka obsahuje zdrojové kódy ukázkové aplikace a ovladačů napsané v jazyce nesC. Pro úspěšnou kompilaci je potřeba mít správně nainstalované prostředí TinyOS.
- `./server/` Zdrojové kódy napsané v jazyce C. Jedná se o ukázkovou serverovou aplikaci pro běh na linuxové/unixové stanici.
- `./text/` Adresář obshahuje elektronickou verzi tohoto dokumentu včetně zdrojových kódů programu LaTeX, šablon a `Makefile`.
- `./README` Soubor shrnuje obsah CD včetně jednoduchého návodu, jak správně pracovat s přiloženými zdrojovými kódy v TinyOS prostředí.