České vysoké učení technické v Praze
Fakulta elektrotechnická

Bakalářská práce

Implementace sběru dat z mikropočítačové aplikace užitím GPRS

Jan Skalický

Vedoucí práce: Ing. Pavel Kubalík

Studijní program: Elektrotechnika a informatika, strukturovaný, bakalářský

Obor: Výpočetní technika

srpen 2007 (aktualizace 2008-11-18)

Poděkování

Děkuji Ing. Pavlu Kubalíkovi za vedení práce. Dále bych rád poděkoval firmě Czechlabs s. r. o., zejména jejímu technickému řediteli Ing. Pavlu Růžičkovi za poskytnutí materiálního zázemí a Ing. Janu Krejsovi za čas, který mi věnoval při konzultacích. Dík patří rovněž všem, kteří moji práci připomínkovali a všem vývojářům svobodného softwaru, jejichž nástroje mi napomáhaly v celém průběhu práce.

Prohlášení

Prohlašuji, že jsem svou práci vypracoval samostatně a použil jsem pouze podklady uvedené v přiloženém seznamu zdrojů.

Nemám závažný důvod proti užití tohoto školního díla ve smyslu §60 Zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon).

V Praze dne 21.srpna 2007

....

Abstract

The goal of this thesis is to prove the capabilities of GPRS technology utilization in realizing data capture from a telemetric application. The device, for which the data capture is implemented within this work, is a GPS collar for wildlife animals, controlled by microcomupter with AVR architecture. The work presents a process, in which the GPRS intergration request is being turned into working results, usable in the design of an existing device.

Abstrakt

Úkolem této práce je prověřit možnosti využití technologie GPRS při sběru dat z telemetrické aplikace. Zařízení, pro které se v rámci práce sběr dat implementuje, je GPS obojek pro divoká zvířata, řízený mikropočítačem architektury AVR. Práce prezentuje postup, při kterém se od požadavku integrace GPRS dochází ke konkrétním výsledkům, použitelným v designu existujícího zařízení.

Obsah

Seznam tabulek

Seznam obrázků

Slovník použitých zkratek

BGA	Ball Grid Array
CPU	Central Processor Unit
CS	Coding Scheme
CSD	Circuit-Switched Data
EDGE	Enhanced Data Rates for GSM Evolution
FIFO	First In, First Out
FTP	File Transfer Protocol
GPRS	General Packet Radio Service
GPS	Global Positioning System
GSM	Global System for Mobile Communications
HSCSD	High-Speed Circuit-Switched Data
HTTP	Hypertext Transfer Protocol
ICE	In-circuit Emulator
ICMP	Internet Control Message Protocol
IDE	Integrated Development Environment
IP	Internet Protocol
JTAG	Joint Test Action Group
LED	Light-emitting Diode
M2M	Machine to Machine
MCU	Microcontroller Unit
MMS	Multimedia Messaging Service
NFS	Network File System (protocol)
NDA	Non-disclosure Agreement
PDP	Packet Data Protocol
PPP	Point-to-Point Protocol
PSK	Phase-shift Keying
RF	Radio Frequency
RFC	Request for Comment
SIM	Subscriber Identity Module
RoHS	Restriction of Hazardous Substances
RTT	Round-trip Delay Time
SMS	Short Messaging Service
TCP	Transmission Control Protocol
UART	Universal Asynchronous Receiver/Transmitter
USB	Universal Serial Bus

1. Úvod

GPS Collar je existující telemetrická aplikace vyvíjená českou firmou Czechlabs s. r. o., určená k monitorování divokých zvířat ve volné přírodě Evropy a Severní Ameriky (www.telemetrysolutions.com). Cílem této práce je vybavit zmíněné zařízení GPRS subsystémem, který bude realizovat sběr naměřených dat prostřednictvím internetu.

Práce tedy má rešeršně-implementační charakter. V první části je třeba provést průzkum trhu na poli GPRS modemů a stanovit na základě jakých parametrů, vzhledem k nasazení aplikace, se uskuteční výběr konkrétního typu k použití. Několik nejlepších kandidátů mezi moduly bude třeba získat (finančně zajistí firma) a nejdůležitější parametry změřit v simulovaných i reálných podmínkách (v terénu), a to buď na vývojových kitech nebo na vlastních testovacích deskách. Na základě toho se obhajitelným způsobem učiní finální výběr hardwaru k implementaci. Součástí analytické části práce bude rovněž studie možných koncepcí architektury přenosu dat tak, aby výsledný model byl vhodný pro typické nasazení produktu.

Implementační část práce bude spočívat v interfacování vybraného modulu do zařízení, a to na HW úrovni (vstupem budou schémata ve formátu PDF a výstupem doporučení jejich modifikace) a zejména pak na SW/FW úrovni, kde výstupem bude odladěné API pro ovládání přenosu dat do internetu, vytvořené pro centrální procesor zařízení – Atmel AVR ATmega2560 (pro účely programování a ladění FW bude k dispozici programátor/debugger Atmel JTAG-ICE mkII a AVR Studio s překladačem jazyka C). Návrh API musí být vypracován s ohledem na analyzovanou možnost přenosu dat i směrem ze sítě do obojku a správnou interpretaci takových dat (dávkové programování, aktualizace firmware – vlastní interpretaci zajistí existující kód). Vzhledem k bateriovému napájení přístroje bude třeba přizpůsobit API pro možnost optimalizace spotřeby energie při přenosu dat.

Práce je rozdělna do kapitol v závislosti na postupu jejího plnění. Hlavní z nich jsou rešerše, analýza, implementace, testování a zhodnocení výsledků. Pro snadnou orientaci v textu jsou hlavní části členěny na subkapitoly tak, aby text jednotlivých pojmenovaných celků pokud možno nepřesáhl jednu stránku a byla tak zajištěna kontinuita logiky dělení. Odkazy v textu na použité zdroje mají formát [x], kde x je číslo ze seznamu zdrojů uvedených v závěru práce.

2. Rešerše

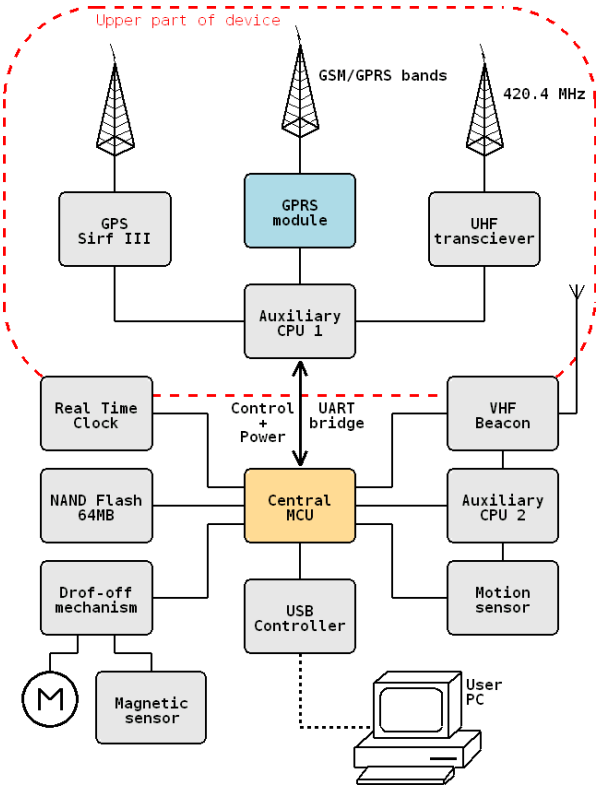
2.1 Seznámení s technologií

Začátkem práce je moje seznámení s cílovou problematikou, kterou budu zpracovávat. Protože požadavkem je integrace GPRS technologie, o níž jsem před začátkem práce mnoho nevěděl, do neznámého existujícího zařízení, bylo potřeba provést bližší seznámení s obojím.

2.1.1 GPS Collar

Obojek (obr. 2.1) je hardware skládající se ze dvou částí elektricky propojených vodiči v jeho obvodu. V horní poloze se nachází deska spojů pro VF obvody a jejich pomocný procesor, který je řídí. V dolní poloze je deska s hlavním mikrokontrolérem (dále jen MCU), externí paměti, vstupně-výstupní elektronika a mechanické části (mechanismus pro odepínání pásu apod.). Pro ilustraci jsem nakreslil blokové schéma zařízení, viz obr. 2.2.

Obrázek 2.1: Finální produkt



Obrázek 2.2: Blokové schéma obojku

Blok realizující GPRS by byl umístěn v horní poloze s anténou vedenou obvodem obojku a datový kanál mezi ním a centrálním MCU by zajišťovala sériová linka mezi centrálním procesorem a pomocným procesorem horní desky. Obvody, které jsou relevantní v otázce integrace GPRS ukazuje tabulka 2.1.

Blok	Hlavní obvod	Popis
Central MCU	Atmel ATmega2560	8-bit AVR with 256kB
Auxilary CPU 1	Atmel ATmega162	8-bit AVR with 16kB
Auxilary CPU 2	Microchip PIC16LF88	low-consumption MCU
USB Controller	FTDI FT245BQ	USB <=> Parallel FIFO
GPRS	? – výstup rešerše	GPRS modem

Tabulka 2.1: Hlavní obvody vybraných bloků obojku

2.1.2 Technologie GPRS a její užití

Z volně dostupných zdrojů [10], [11], [12] jsem načerpal informace o technologii GPRS a shrnul klíčové poznatky:

GPRS (General Packet Radio Service) je mobilní datová služba přístupná pro uživatele GSM mobilních telefonů označována jako síť generace „2.5G“ (technologie mezi druhou (2G) a třetí (3G) generací mobilních telefonů). GPRS je paketově přepínané a jeho specifikace zahrnuje podporu protokolů IP, PPP, OSPIH a X.25 (v praxi podporují operátoři pouze IP a někdy také PPP). Běžně se GPRS používá pro přístup k internetu, WAP a přenos MMS.

EDGE (Enhanced Data Rates for GSM Evolution) je rozšíření GSM, někdy označováno jako síť generace „2.75G“, obsahující EGPRS (Enhanced GPRS) a ECSD (Enhanced Circuit Switched Data, ve skutečnosti rozšíření HSCSD). Hlavní vylepšení spočívá v použití modulace 8-PSK (3 informační bity na 1 rádiový symbol oproti 1 ve starší modulaci GMSK). EGPRS nabízí vyšší rychlosti než GPRS a je s ním zpětně kompatibilní. Pokrytí signálem EGPRS je v současné době však výhradou měst a jiných aglomerací a tudíž obecně nepoužitelné pro danou aplikaci, u níž předpokládám primární nasazení v řídce obydlených lokalitách.

GSM/GPRS zařízení se dělí na 3 třídy podle schopnosti využívání GPRS a hlasových služeb:

- třída A – umožňuje simultánní využívání GPRS i hlasových služeb
- třída B – hovor nebo data (ne současně), provozy je možné držet a přepínat
- třída C – umožňuje pouze jeden druh provozu bez možnosti držení a přepínání

GPRS používá čtyři kódová schémata CS-1 až CS-4, ze kterých se vybírá v závislosti na odstupu signálu od rušení tak, aby byl zajištěn nejlepší přenos dat. Jejich rychlosti (v pořadí od 1 do 4) jsou: 8.0, 12.0, 14.4 a 20.0 kbit/s.

Další rozdělení GPRS terminálů do tříd je podle toho, kolik timeslotů (GPRS využívá časové dělení kanálu) umí použít pro upstream, downstream a kolik z toho současně. Třídy tohoto dělení se nazývají „Multislot class“ (může být rozdílná pro HSCSD, GPRS a EGPRS). Nejběžnější třídou dnešních modemů je třída 8 (4RX + 1TX, max. 5) nebo 10 (4RX + 2TX, max. 5). Daná konfigurace je zvolena podle převládajícího toku dat a mění se dle aktuální situace. Nejvyšší přenosové rychlosti pro terminál pracující ve třídě 10 při použití CS-4 jsou tedy 80 (down) a 40 (up) kbit/s. Tato teoretická hranice nezahrnuje protokolární režii a předpokládá ideální RF signál a nezatíženou buňku mobilní sítě, protože GPRS provoz má obecně nízkou prioritu (při zatížení buňky GSM hovory se kvalita služeb poskytovaných v rámci GPRS dynamicky snižuje). Podotýkám, že ačkoliv GPRS je síť logicky oddělená od GSM, praktické realizace spojují tyto sítě do společných přístupových bodů a tím je ovlivněna kvalita služeb a přístupnost GPRS (pokrytí signálem je podmnožinou pokrytí GSM; odhad interesované osoby hovoří o ~95%). V praxi je GPRS síť závislá na GSM minimálně kvůli autentizaci terminálu.

2.2 Průzkum trhu

2.2.1 Podobná řešení

Pro získání přehledu o potenciální konkurenci jsem na webu hledal podobně zaměřené produkty, které využívají GSM/GPRS síť. Existuje nevelké (jednotky) množství firem zabývajících se vývojem v oblasi „GPS Wildlife Tracking“ a některé z nich nabízejí i produkty, resp. jejich varianty s podporou GSM/GPRS (např. www.televilt.se, www.environmental-studies.de, www.tomatrack.com). V ostatních případech dochází k přenosu dat prostřednictvím dedikovaných vysílačů

v pásmech VHF/UHF, satelitního systému Argos (profesionální řešení) nebo u starších produktů i manuálním sběrem. Překvapilo mě, že pouze u jednoho lokalizačního zařízení (navíc neurčeného ke sledování zvířat) byla uvedena podpora přenosu dat pomocí GPRS. Ostatní produkty využívaly pro odchozí data SMS v síti GSM. Při konzultaci s vývojáři GPS Collar mi bylo řečeno, že očekávaný průměr objemu generovaných dat je 10MB za rok. Na základě této informace jsem odhadem porovnal parametry přenosu takového objemu pomocí SMS s datovým provozem po GPRS, viz tabulku 2.2.

Porovnávaný ukazatel	**SMS**	**GPRS data**
Jednotka dat	160 bytů	např. 1 kB
Rychlost přenosu jednotky	~ 5 sec.	pro 1 kB/s ~ 1 sec.
Počet jednotek na 10MB	~ 60 000	~ 10 000
Doba provozu	~ 100 hod.	~ 3 hod.
Odběr proudu	~ 200 mA	~ 500 mA (pesimisticky)
Celková spotřeba	~ 20 Ah	~ 1.5 Ah

Tabulka 2.2: Porovnání parametrů přenosu SMS vs. GPRS

Vzhledem k objemu dat, který se chystáme přenášet, a faktu, že SMS jsou režijně náročnější už v okamžiku, kdy jejich počtem je dvojciferné číslo (rovnost energetické režie pro výše stanovené parametry nastává při cca 1kB dat), SMS jednoznačně zamítám a považuji je pouze jako budoucí možné řešení přenášení krátkých operativních dat, které ovšem zůstanou v množině všech dat synchronizovatelných přes GPRS nebo pro oznamování problémů souvisejících s GPRS (např. nedostatečné pokrytí). Proti SMS hovoří i ekonomické důvody, protože při konvenčním zpoplatňování SMS a GPRS vycházejí data v SMS měrně cca 100x dražší. Obvyklejší nasazení SMS si vysvětluji historickým vývojem složitosti implementace GPRS v embedded zařízeních (starší modemy neměly stack protokolu TCP/IP a tudíž byla nutná podpora všech vrstev od PPP po aplikační na úrovni řadiče) a tím, že jejich současní uživatelé zřejmě požadují podstatně menší, neúplné a nebo méně časté bloky dat. To se ovšem liší od mého zadání, které požaduje možnost synchronizace všech sebraných dat, a z hlediska systémovosti řešení preferuji realizaci tohoto požadavku právě jednou, vhodnější cestou.

2.2.2 Studie použitelnosti

V průběhu rešeršní části projektu jsem dostal za úkol vytvořit „Feasibility Study“ (studii použitelnosti) pro účely uspokojení zákazníka, na zákadě jehož zadání je prováděn vývoj produktu GPS Collar. Jedná se o text na cca 500 slov, jehož obsah je z velké části redundantní s informacemi z kapitol 2.2 a 2.3. Proto ji nemá cenu zde uvádět a její existenci pouze zmiňuji.

2.2.3 Nabídka výrobců GPRS modemů

V prvním kroku této části jsem hledal výrobce telekomunikačních zařízení, v jejichž portfoliu se nacházejí modemy pro GSM/GPRS, a to jak ve formě modulů, tak jako řešení typu „on-chip“. Vzhledem k tomu, že tento segment trhu je hojně obsazený a já jsem chtěl provést jeho zevrubný průzkum, protože na základě výběru modemu se bude odvíjet celý zbytek projektu, nastudoval jsem nabídku výrobců, které uvádím v tabulce 2.3.

Výrobce	Webová stránka
Advanced Wireless Planet	www.gsm-modem.de
Agere Systems	www.agere.com
AirLink Communications	www.airlink.com
Alcatel	www.alcatel.com
Aplicom	www.aplicom.com
Audiotel	www.audiotel.it
BenQ Mobile	www.benqmobile.com
CalAmp	www.calamp.com
CSI Wireless	www.csi-wireless.com
Comverge	www.comverge.com
DataRemote	www.dataremote.com
EDMI Ltd.	www.edmi.com.au
Enermet	www.enermet.com
eLutions	www.elutions.com
Enfora	www.enfora.com
Fujitsu	www.fujitsu.com

Výrobce	Webová stránka
Hitachi	www.hitachi.com
Kyocera Wireless Corporation (KWC)	www.kyocera-wireless.com
Motorola	www.motorola.com
Nokia	www.nokia.com
OMRON Corporation	www.omron.com
Panasonic	www.panasonic.com
QUALCOMM	www.qualcomm.com
Research In Motion (RIM)	www.rim.net
Round Solutions	www.roundsolutions.com
Sagem	www.sagem.com
Siemens	www.siemens.com
Sierra Wireless	www.sierrawireless.com
SIM Technology	www.sim.com
Sony Ericsson	www.sonyericsson.com
Telenetics	www.telenetics.com
Telit	www.telit.co.it
Thales Telematics	www.thalestelematics.com
Tyco	www.tycoelectronics.com
Wavecom	www.wavecom.com
Wireless Maingate	www.wirelessmaingate.com

Tabulka 2.3: Výrobci telekomunikačních zařízení

Tučně jsou zvýrazněni výrobci, jejichž nabídka je relevantní a jejichž produkty jsem dále porovnával z hlediska jejich parametrů. Zbylí výrobci nenabízeli pro mě principiálně použitelné produkty. Při průzkumu nabídek jsem učinil několik obecných poznatků:

- všechny dnešní GPRS modemy podporují kódovací schémata CS1 – CS4
- běžně jsou dostupné modemy s multislot třídou 8, 10, vzácněji 12 (4+4, Siemens)
- běžně jsou dostupné modemy s integrovaným TCP/IP stackem (např. Telit – „Easy GPRS“)
- „krabicových“ terminálu je na trhu podstatně víc než embedded modulů
- starší moduly nejsou o moc levnější, jsou o dost rozměrnější, mají nízkou multislot třídu a nemají IP stack (Sagem)

- moduly některých firem podporují EDGE, jsou však dražší a rozměrnější (Sony-Ericsson, Wavecom, Aplicom)
- existují moduly s teplotními čidly nebo GPS (např. Siemens XT55) (od myšlenky jejich použití místo designových jsem ale upustil)
- existují i značně robustní moduly s podporou Javy, HSCSD, HTTP atd. (tyto funkcionality by ale zde asi zůstaly nevyužity)
- „automotivní“ moduly mají značný rozsah pracovních teplot, běžně od -40°C (většinou jsou však rozměrnější; Siemens)
- Wavecom integruje do svých modulů funkcionalitu „OpenAT“ (jde o možnost programování modulu v API na úrovni AT příkazů)
- GPRS/GSM „on-chip“ baseband procesory jsou obvykle pouzdřené v high-pin-count BGA a je nutno k nim použít externí RF transciever (Agere, Broadcom, Freescale, Infeon, Intel)
- podrobnou dokumentaci produktů je někdy obtížné získat (Siemens, Wavecom, Simcom – nutnost podepsat NDA)

2.3 Výběr modemu

2.3.1 Klíčové parametry

Na základě zadání, specifikace celého produktu (v příloze) a konzultací s jeho vývojáři jsem stanovil klíčové parametry při výběru vhodného modemu. Jsou jimi (s klesající prioritou):

- rozměry a hmotnost (modem bude osazen v horní poloze obojku pro divoká zvířata)
- spotřeba energie v aktivním stavu (kvůli bateriovému napájení a velkých příkonů v porovnání se zbytkem HW)
- rozsah pracovních teplot, dle specifikace nejlépe už od -40°C (obojek může být nasazen v arktických oblastech)
- GPRS multislot class, zejm. počet TX slotů ovlivňující rychlost upstreamu (upload bude tvořit většinu přenášených dat)
- podporovaná RF pásma – aplikace má být použitelná v Evropě i Americe (nejlépe Quadband – GSM/EGSM/DCS/PCS)
- další aplikační přednosti

Dále jsem určil omezující podmínky na přítomnost TCP/IP stacku, protože bez něj by bylo nutné naprogramovat do MCU tuto protokolovou vrstvu, vč. PPP protokolu a jeho podprotokolů (LCP, PAP, NCP), který paketizuje data vyšších vrstev pro zachování zpětné kompatibility s pozemními modemy. Podpora tohoto stacku není u dnešních modemů problémem a je jimi podporován zejména pro zjednodušení vývoje M2M aplikací založených na nekomplexních embedded systémech. Původní představa firmy preferovala řešení „on-chip“, které je ale náročnější o design RF části (vč. VF layoutu desky spojů), napájení a naopak většinou poskytuje integrované řadiče pro HW, který by se v našem designu nevyskytoval (displaye, klávesnice, multimédia). V tomto ohledu se toto řešení hodí spíš pro výrobce mobilních telefonů a v designu obojku by se z něho za cenu vetšího úsilí stejně stal HW blok podobný modulu. Dále jsou tyto obvody vyráběny výhradně v pouzdrech BGA a ty se nehodí do aplikací s velkým teplotním rozsahem. Na základě těchto důvodů jsem obhájil svoje doporučení pro volbu modemu ve formě zapouzdřeného modulu. Vzhledem k nasazení obojku v přírodě je rovněž žádoucí, aby celý produkt, tudíž i modul, splňoval normu RoHS, omezující užívání nebezpečných látek (zejm. olova, rtuti, kadmia) při výrobě a montáži součástek.

2.3.2 Porovnání produktů

Nalezené modemy jsem zpracoval ve formě tabulky (příloha 9.2) porovnávající jejich parametry, kterou jsem vyrobil pro účely přehlednosti při výběru.

2.3.3 Výběr vhodných modulů k bližšímu porovnání

Z tabulky parametrů, přihlédnutí k jejich prioritám a ostatním poznatkům vybírám nejvhodněji se jevící moduly:

- Enfora Enabler-IIG
- Siemens MC55
- Simcom Sim200 nebo Sim300(D)
- Sony-Ericsson GS64
- Telit GC864
- Wavecom Q2687

2.3.4 Pořízení vhodných modulů, vývojových desek

Z 6 vybraných modulů jsem prostřednictvím českých/slovenských distributorů jejich výrobců sehnal 4. Sony-Ericsson GS64 nebyl v té době ještě uveden na trh a vzhledem k tomu, že divizi tohoto vývoje právě přebírala firma Wavecom, nezdálo se pravděpodobné, že by v dohledné době uveden byl. Dostupnost modulu MC55 jsem nezjistil, protože firma Siemens neodpověděla na žádný z dotazů (psaných mnou jménem firmy Czechlabs) a jejich personál nebylo možné kontaktovat přímo. Od výrobce Wavecom se mi prostřednictvím distributora Spezial Electronic (www.spezial.cz) podařilo sehnat modul **Q2686**, lišící se od Q2687 pouze verzí OS (nová verze ještě nebyla uvolněna). Modul GC864 jsem obdržel od firmy Microdis (www.microdis.cz) a to ve verzi **GC864-PY**, která navíc obsahuje interpret jazyka Python. Modul Sim200 byl v době pořizování testovacích vzorků výběhovým typem a firma S.O.S electronic (www.sos.sk) dodala novější **Sim300**. Modul Enfora **Enabler-IIG** dodala firma Tencom (http://tencom.cz). Ceny (maloobchodní za 1 kus) 3 modulů se pohybovaly mezi 1 600 – 1 700 Kč, pouze modul od Wavecom byl o cca 1 000 Kč dražší než ostatní.

Aby bylo možné získané moduly oživit, blíže se s nimi seznámit a provádět na nich experimenty a měření, bylo nutné k nim pořídit vývojové desky s konektory pro jejich připojení. Ve většině případů se takovéto „development kity“ nacházejí už v nabídce jejich výrobců. Někdy je bylo možné zapůjčit i od distributora. Některé z těchto kitů byly pro moje účely až moc obsáhlé a tomu odpovídala i jejich cena. Proto jsem se rozhodl, že pořídím jenom 2 levnější vývojové desky k modulům Enfora (konkr. SDK0107MG) a Simcom (konkr. SIM300 EVB) a pro zbylé 2 moduly sám navrhnu jednoduché testovací desky, které budou obsahovat pouze ta rozhraní, která jsou pro mě zajímavá.

Návrh vlastních testovacích desek byl proveden v systému OrCAD 10.3 (schéma i layout v příloze) a čerpal jsem při něm zejména z HW datasheetů k modulům a použitým součástkám. Desky, v třídě přesnosti 6, poté vyrobila firma Pragoboard s. r. o. (na základě GERBER výstupů z OrCADu) a osazeny byly mnou. Tyto desky obsahují zejm. sériový port pro připojení k PC, držák SIM karty, header pro měření spotřeby, tlačítka pro zapínání/reset, LED pro indikaci stavu modulu, ev. konektory pro připojení telefonního sluchátka a debugovacích signálů. Konektor pro připojení externí antény se nacházel ve všech případech na těle modulu a byl typu MMCX. K dispozici jsem měl GSM antény v provedení dipólu i čtvrtpólu, se šroubovacím konektorem typu SMA a jeho redukci na MMCX.

2.3.5 Bližší porovnání vybraných modulů

U 4 modulů, které postoupily do užšího výběru, jsem nastudoval podrobnou dokumentaci, vyzkoušel jejich příkazy a dále jsem je porovnával na základě původních parametrů (pokud se lišily) a na základě dalších parametrů, z nichž některé jsem naměřil při jejich provozu. Celkově jsou to:

- velikost a druhy montáže
- nominální citlivost (katalogová) v závislosti na RF pásmu
- rozsah extrémních pracovních teplot (katalogový)
- rozsah napájecího napětí (katalogový)
- minimální napájecí napětí, při kterém se modul zapne
- pokles napájecího napětí, při kterém modem přestane odpovídat na příkazy
- pokles napájecího napětí, při kterém se přeruší probíhající datový přenos
- autobauding
- čas od zapnutí modulu po první úspěšně odeslaná a přijatá data z internetu
- spotřeba modulu při přenosu dat ve směru upstreamu po GPRS v závislosti na síle signálu a využitých pásmech
- spotřeba modulu při přenosu dat ve směru downstreamu po GPRS v závislosti na síle signálu a využitých pásmech
- cena (maloobchodní za 1 kus)

První 4 parametry jsou katalogové. Hmotnost všech modulů je zhruba stejná, okolo 8g. Preferovaný druh montáže je „Board to board“ konektor, ev. SMT, ne však BGA (viz výše). Uvažování parametrů souvisejících s napájecím napětím vznikl z původního záměru napájet GPRS modul přímo z hlavního primárního článku o napětí 3.65V a faktu, že při vysílání vznikají v periodě obsazování timeslotů (GPRS má časové dělení pásma) proudové špičky až 2A a i při dobré filtraci se projeví poklesem napětí v napájecím bodě. Novější koncepce napájení, která vychází z vlastností použitých článků i za extrémních teplot, však počítá se spínaným zdrojem v roli step-up (boost) DC-DC měniče pro napájení modulu. S měničem už parametry týkající se napájení nejsou tak důležité a v návrhu interfacu na HW úrovni se budou týkat zejm. výběru blokovacích kondenzátorů.

Podpora autobaudingu, ev. v jakém rozsahu, souvisí se skutečností, že AVR generuje baudovou rychlost UARTu dělením základní hodinové frekvence (v obojku 8MHz) a různé rychlosti jsou pak generovány s různou přeností hodin. Tudíž nelze zaručit, že zrovna rychlost požadovaná modemem bude generována s přesností

přijatelnou, a proto ideální by bylo zvolit nejpřesněji generovanou rychlost, kterou bude modul schopen přijímat, než se mu nastaví jako fixní (a která zároveň nebude příliš nízká, aby zpomalovala vyšší vrstvu – datový kanál). Proto čím větší rozsah autobaudingu, tím potenciálně lépe.

Stěžejním parametrem jsou naměřené spotřeby (konkr. náboj) při datových přenosech. A to jak ve směru z terminálu do internetu, který bude v aplikaci převládat, tak ve směru opačném, který bude použit např. pro programování obojku nebo stažení nové verze firmware. Cena modulu není v designu obojku rozhodujícím faktorem, ale protože se výrazně liší pouze u jednoho soutěžícího, lze k ní přihlédnout v případě blízkých výsledků ostatních parametrů.

2.3.5.1 Měření na vybraných modulech

Energetické parametry – spotřeby při přenosu dat jsem měřil integračním coulombmetrem vlastní výroby se stupnicí v C (Ampersekunda) a přesností 1% při kalibračním proudu 1A. Přístroj vykazoval progresivní chybu cca +1% na 1A pro vyšší proudy a o něco více pro nižší. Experimentálně jsem ověřil, že změří proudové pulzy s frekvencí 2kHz, což odpovídá průběhu odběru při GSM vysílání (1 timeslot trvá 0.577ms). Konstatoval jsem tedy, že pro změření odběrů modulů za účelem jejich porovnání je použitelný.

Testovací data byly náhodně vygenerované binární bloky o velkosti 2MB pro download a 1MB pro upload, tedy dostatečně velké, abych dlouhými časy přenosů minimalizoval chybu měření vzniklou kolísáním zatížení sítě a ostatní spotřební režií. Pro jistotu jsem ještě vlastní měření prováděl v čase mezi půlnocí a ránem, kdy předpokládám minimální zatížení sítě.

Závislost spotřeby na síle signálu spočívala ve 2 měřeních – při běžné kvalitní síle signálu (město s dobrým pokrytím) a při značně zhoršeném signálu v terénu (les, pahorkatina, žádná BTS v blízkosti). Všechna měření byla prováděna v síti T-Mobile a v případě očividné systematické chyby korigována opakovaným měřením, k čemuž došlo např. vlivem větší změny kvality signálu kvůli nestálosti počasí nebo v případě nekontinuality přenosu v čase. Měřeno bylo nejen v závislosti na síle signálu, ale také na tom, zda-li modem pracoval v režimu „Dualband“ nebo využíval jen spodní pásmo EGSM (900MHz), které má v otevřené přírodě (a tedy v typických podmínkách nasazení obojku) lepší pokrytí.

Na aplikační vrstvě jsem k datovým přenosům využíval 2 typy serverů – FTP server s výhradním přístupem (2 odměry) a HTTP server ve formě webového „speedtestu“ (konkr. http://schema.chello.cz/speed/). U jednotlivých odměrů jsem poznamenával i průměrnou rychlost přenosu, která šla jinak ovlivňovat pouze nepřímo (opakováním měření s rychlostí výrazně se lišící od průměru), protože se dá očekávat, že závislost spotřeby na rychlosti bude sublineární a lepšího porovnání se docílí i přihlédnutím k tomuto. Všechny moduly jsem při měření napájel napětím 4.2V. Výsledky jsou uvedeny v tabulkách 2.4 a 2.5 .

Modul, pásma, protokol			CSQ[1]	Směr uploadu		Směr downloadu	
				Rychlost [kB/s]	Náboj [C]	Rychlost [kB/s]	Náboj [C]
Simcom	Dualband	HTTP	30	3.92	73	9.17	30
		FTP (1)	30	2.95	76	6.17	43
		FTP (2)	31	3.06	84	5.74	47
	GSM9	HTTP	31	4.24	78	8.54	35
		FTP (1)	31	2.99	85	6.60	42
		FTP (2)	31	3.31	83	6.70	41
Telit	Dualband	HTTP	29	1.85	72	6.65	35
		FTP (1)	29	1.19	79	4.96	42
		FTP (2)	29	1.28	87	4.59	45
	GSM9	HTTP	30	2.11	67	6.19	33
		FTP (1)	30	1.47	78	3.72	53
		FTP (2)	30	1.43	80	5.66	39
Wavecom	Dualband	HTTP	31	2.46	66	9.82	29
		FTP (1)	31	1.61	70	7.26	34
		FTP (2)	30	1.56	80	5.95	40
	GSM9	HTTP	30	1.90	58	8.47	30
		FTP (1)	30	1.65	60	5.46	39
		FTP (2)	30	1.58	72	4.99	49
Enfora	Dualband	HTTP	29	1.17	138 (!)	6.37	42
		FTP (1)	30	0.85 (?)	158 (!)	3.19	74
		FTP (2)	30	1.02	144 (!)	4.17	57

Tabulka 2.4: Spotřeby při přenosech – kvalitní signál (okolo -55dBmW)

Z výsledků měření při kvalitním signálu je vidět, že situace, kdy modem operuje ve 2 pásmech, není znatelně energeticky výhodnější, než když se hlásí pouze na EGSM pásmu. Zdá se, že operátor, s jehož SIM kartou testuji (T-Mobile), preferuje GPRS data v dolním frekvenčním pásmu a v horním pásmu hlasové hovory. V opačném případě by se totiž hodnoty v Dualband režimu výrazně lišily, protože v horních pásmech (DCS, PCS – 1800 resp. 1900MHz) se vysílá v průměru skoro polovičními výkony. Takovéto nastavení může uplatňovat jakýkoliv operátor, a proto tedy není závislost naměřených hodnot na využitých pásmech příliš směrodatná. Další měření jsem prováděl vždy už jen v Dualband režimu (v závislosti na nastavení operátora může být Dualband vyžadován už při registraci do sítě).

Dále jsem změřil, že modul Enfora vykazuje podstatně větší spotřebu energie než zbylé moduly, a to obzvlášť na upstreamu, kde se mně ani po několikerém přeměřování nepodařilo přiblížit se konkurenčním rychlostem přenosu. Tímto dále uvažuji už jen moduly Telit, Simcom a Wavecom.

Modul, pásma, protokol			CSQ[1]	Směr uploadu		Směr downloadu	
				Rychlost [kB/s]	Náboj [C]	Rychlost [kB/s]	Náboj [C]
Simcom	Dualband	HTTP	14	2.73	133	8.24	50
		FTP (1)	13	1.65	151	5.97	69
		FTP (2)	16	3.17	111	5.64	65
Telit	Dualband	HTTP	14	2.44	117	6.05	56
		FTP (1)	14	1.87	146	3.63	79
		FTP (2)	13	1.43	174	4.18	73
Wavecom	Dualband	HTTP	16	2.32	127	4.98	75
		FTP (1)	13	1.22	188	4.45	80
		FTP (2)	14	1.68	147	4.27	83

Tabulka 2.5: Spotřeby při přenosech – zhoršený signál (okolo -85dBmW)

Všechna napětí jsem měřil běžným digitálním multimetrem v rozsahu do 20V s 2 desetinnými místy, což odpovídá přesnosti ±10mV, ale mně stačily výsledky zaokrouhlené v rastru 50mV. Jako zdroj napětí jsem použil jednokanálový laboratorní zdroj Manson NP-9615 (30V/5A) a granularitu regulace jsem zvolil stejnou jako rastr zaokrouhlování. Výsledky měření jsou v tabulce 2.6.

1 Okamžitá úroveň kvality signálu, přepočet na dB vzorcem $113-2\times CSQ$ [dbmW]

Modul		Simcom	Telit	Wavecom
Rozsah napájecího napětí [V] (katalogový)	**Minimální**	3.4	3.4	3.2
	Nominální	4.0	3.8	3.6
	Maximální	4.5	4.2	4.8
Minimální napětí pro zapnutí [V]		3.35	3.4	3.25
Pokles napětí, při kterém modem přestane odpovídat na příkazy [V]		3.3	3.4	3.2
Pokles napětí, při kterém se přeruší probíhající datový přenos [V]		3.35	3.45	3.15

Tabulka 2.6: Parametry související s napájecím napětím

Autobauding jsem zkoušel experimentálně reakcí na prázdný "AT" příkaz hned po zapnutí modemu, a to pro všechny rychlosti, jejichž podporu výrobce uváděl (pokud vůbec) a pro některé jejich další celočíselné podíly. Výsledky jsou v tab. 2.7.

Rychlost [bps]	300	600	1200	2400	4800	9600	19200	38400	57600	115200
Simcom			X	X	X	X	X	X	X	X
Telit		X[2]	X	X	X	X	X	X	X	X
Wavecom			X	X	X	X	X	X	X	

Tabulka 2.7: Rychlosti podporované autobaudingem

Parametr výše nazvaný „Čas od zapnutí modulu po první úspěšně odeslaná a přijatá data z internetu“ jsem měřil od aktivní hrany zapínacího signálu po první úspěšnou odezvu z nezatíženého, dobře dostupného serveru v internetu, přičemž připojení k internetu obstarával PPP stack v PC a vytáčení servisního kódu pro připojení přes GPRS se provádělo opakovaně od zapnutí modemu po jeho první úspěšnou odezvu. V tomto čase je tak zahrnuto zapnutí modulu, nabootování jeho OS, vyhledání sítě a registrace do ní, aktivaci GPRS kontextu (definován byl předem), přechod do datového režimu, autokonfiguraci TCP/IP a RTT ICMP paketu do vnější sítě, který indikoval funkční připojení (jeho fluktuance pro různé moduly byla vzhledem k celkovému času zanedbatelná). Toto měření jsem opakoval několikrát a pro všechny moduly byly výsledky takřka totožné v rozmezí 20–25 sec. Připojení ze stavu, kdy byl modem již zaregistrován v GPRS síti, činilo 7–12 sec. Je vidět, že zatížení sítě se zde

2 'X' značí podporovanou rychlost

projevuje větší mírou než časová režie modulů, jejíž rozdíl je v porovnání s vnějšími vlivy minimální, a proto nemá cenu tento parametr dále uvažovat.

2.3.5.2 Ostatní parametry

Neměřené parametry modulů, které jsou spolu s měřenými rozhodující při finálním výběru modulu, sumarizuji v tabulce 2.8.

Modul		Simcom	Telit	Wavecom	Enfora
Velikost a druhy montáže	**Š x V x H [mm]**	40 x 33 x 2.85	37 x 30 x 2.8	40 x 32.2 x 4	46 x 30 x 3.1
	Velikost relativně	středně velký, nízký	nejmenší, nízký	malý, vysoký	větší obdélník
	Montáž	QFN, B2B (60/0.5mm)[3]	BGA, B2B (80/0.5mm)	B2B (100/0.5mm)	B2B (60/0.5mm)
Nominální citlivost [dBmW 900, 1800 MHz]		-106, -104	-107, -106	-104, -102	-106, -106
Rozsah extrémních teplot [°C] (katalogový)		-25 až +70	-30 až +80	-40 až +85	-20 až +60[4]
Cena (maloobch. za kus)		1 700 Kč	1600 Kč	2 600 Kč	1 650 Kč

Tabulka 2.8: Ostatní parametry – velikost, citlivost, rozsah teplot, cena

2.3.6 Finální výběr nejvhodnějšího modulu

Všechny blíže porovnávané moduly mají GPRS class 10 (2 TX timesloty), jsou čtyřpásmové (Quadband), s hmotností okolo 8g. Jejich autobauding zvládá všechny standardní rychlosti sériové linky modemu nad 1200bps, kromě 115200bps u modulu Wavecom (lze nastavit fixně).

3 montáž Board-to-board konektorem s 60 piny a roztečí mezi nimi 0.5mm
4 definována pouze teplota za neextrémních podmínek, s neomezenou funkčností

Modul **Simcom** je druhý největší, varianta v pouzdru pro SMT je o 7mm menší. V testech vykazoval spíše vyšší spotřebu při nejvyšší průměrné rychlosti přenosů. Při zhoršeném signálu měl výsledky relativně lepší, zejm. ve směru downloadu. Spodní hranice pracovních teplot udaná výrobcem je nižší (absolutně) než u konkurence, rovněž katalogová citlivost přijímače je druhá nejhorší. Vyžaduje nejvyšší napájecí napětí, jehož hodnota vylučuje přímé napájení z článku 3.65V. Cenově jde o přijatelný kus.

Modul **Telit** je ze všech nejmenší a zároveň velmi tenký. Generoval nepatrně vyšší odběry energie při silném, avšak o cca 2dBmW slabším, signálu než konkurence, čemuž odpovídá i nižší průměrná rychlost přenosů. Při zhoršeném signálu si vede lépe a pokud srovnáme odběry při podobných rychlostech, má nejmenši spotřebu tento modul. Jeho výrobce udává druhý nejvyšší rozsah teplot a nejvyšší citlivost přijímače. Napájecí a vypínací napětí jsou vysoká a modul je poměrně náchylný k jeho správnému blokování. Proto rovněž zde nedoporučuji přímé napájení z článku. Dodatečnou výhodou tohoto modulu je nejnižší cena a jednoduše a flexibilně navržený TCP/IP stack s podporou protokolů SMTP a FTP.

Modul **Wavecom** je druhý nejmenší, poněkud vysoký. Jeho odběry při silném signálu jsou nejlepší a rovněž průběhy rychlostí uploadů měl nejstabilnější. Ne tak dobrá je situace při zhoršeném signálu, a to v obou směrech toku dat, méně pro upload. Mezi jeho výhody patří nejvyšší udávaný rozsah pracovních teplot, nízké napájení s velkou odolností proti podpětí a dostatek vývojové dokumentace poskytované výrobcem. Nevýhodou je nejmenší citlivost, zejména v horních pásmech a znatelně nejvyšší cena, která o 50% převyšuje ostatní moduly.

Modul **Enfora** je větších rozměrů, hlavně šířkou svého obdélníkového půdorysu. Spotřeby značně převyšovaly konkurenční hodnoty. Rozsah extrémních teplot výrobce neuvádí. Běžné provozní teploty jsou srovnatelné s ostatními. Citlivost a cena jsou průměrné.

Pro design obojku nakonec vybírám modul **Telit GC864**. Má nejméně nevýhod oproti ostatním a nejlepší nebo velmi dobré výsledky ve většině klíčových parametrů. Kromě odolnosti proti podpětí není v ničem nejhorší, ale vzhledem k tomu, že poslední model power managementu obojku počítá s vlastním step-up měničem pro napájení GPRS modemu, není tato nevýhoda nikterak zásádní.

3. Analýza

Analýza problému zadání má 2 nezávislé části. Jedna část se týká softwarových vrstev, jejichž úkolem je přenos dat po internetu na centrální úložiště, ev. opačným směrem (servisní účely, programování). Tato část by měla obsahovat také analýzu podpory pro řízení spotřeby ve smyslu šetření energií. Vrstva mezi MCU a GPRS je pak předmětem implementační kapitoly práce. Druhou částí je hardwarová analýza, jejíž výstupem je sumarizace aspektů integrace GPRS subsystému do stávajícího hardwaru obojku.

3.1 SW analýza

Softwarová analýzá spočívá v enumeraci možných koncepcí přenosu dat, nasbíraných obojkem, na centrální úložiště, kde budou k dispozici k dalšímu zpracování. Je třeba jednotlivé možnosti najít, porovnat je a vybrat nejvhodnější z nich, zejm. s ohledem na flexibilitu a budoucí rozšiřování. Tento požadavek stanovuji na základě faktu, že produkt – obojek je nový a ještě nemusí být zcela zřejmé, jak se bude v budoucnu vyvíjet či modifikovat. Nejprve je třeba uvědomit si charakter sbíraných dat.

3.1.1 Charakter sbíraných dat

Na základě konzultace s vývojáři obojku jsem zjistil, že jeden odměr veličin (GPS souřadnice, teplota aj.) obojkem je binární záznam o velikosti 32B, v debug módu 78B. Perioda odměrů v průměrném nasazení je odhadována na desítky minut, maximální možná četnost odměrů je však 1 minuta. To dává maximálně:

- $78 \times 60 = 4680 \frac{B}{hod} \approx \frac{4.5\text{kB}}{hod}$
- $78 \times 60 \times 24 = 112230 \frac{B}{den} \approx \frac{110\text{kB}}{den}$
- $78 \times 60 \times 24 \times 7 = 786240 \frac{B}{týden} \approx \frac{768\text{kB}}{týden}$

Synchronizace dat každou hodinu je vzhledem k době připojování GPRS a rychlostí přenosu režijně nevýhodná, nehledě na to, že vypočítané objemy jsou horní

hranicí. Granularita po 1 týdnu je už příliš hrubá a mohla by obsahovat velké objemy dat, což by vzhledem k času jejich přenosu a za zhoršených či nestálých podmínek fyzické vrstvy mohlo ohrožovat jejich spolehlivost. Optimální perioda pro synchronizaci odměrů tedy bude 1 den. To je pro účely aplikace dostatečně často a objemy dat jsou v přijatelném rozsahu, který v průměrném případě překryje energetickou a časovou režii vlastního připojování. Tato volba ovšem není závazná a API realizující připojování k síti a přenos dat ji nebude zapouzdřovat, pouze optimalizovat na úrovni návrhu. Může však korespondovat s organizací dat na úložišti.

3.1.2 Enumerace možných koncepcí přenosu dat

Přenos dat na aplikační vrstvě se týká klientské i serverové části. Koncepce přenosu stanovuje pojetí obou a definuje logiku přenosové architekury. Po protokolové stránce lze použít existující řešení nebo navrhnout vlastní. Volba existujícího řešení zjednodušuje implementaci serverové části, která vystupuje v roli úložiště, ev. i zpracovatele, dat. Návrh a implementace vlastního protokolu dává prostor pro podporu všech požadovaných funkcionalit, ale v praxi vyžaduje i tvorbu vlastní serverové části aplikace. Vlastnosti koncepcí se dají nejlépe odvodit právě z volby protokolu aplikační vrstvy. Uvažuji několik možností, které jsou shrnuty v tabulce 3.1.

Protokol	vlastní	FTP	HTTP[5]	e-mail	NFS
Náročnost (klient)	střední	nízká s podporou modemu, jinak střední	nízká s podporou, modemu, jinak střední	nízká s podporou modemu, jinak střední	vysoká
Náročnost (server)	střední	nízká (konfigurace)	střední (konfigurace, formuláře)	žádná	nízká (konfigurace)
Max. objem dat	jakýkoliv	velký (omezení serveru)	střední (omezení serveru, skriptu)	nízká (omezení poštovních uzlů)	velký (omezení serveru)
Závislosti	serverová aplikace	FTP server (free)	HTTP server (ev. s metodou PUT)	schránka, SMTP přistup	server sdílení
Bezpečnost	jakákoliv	nízká (ev. dobrá se SSL)	nízka (ev. dobrá se SSL)	velmi nízká	střední
Organizace dat	jakákoliv	sběr = soubor(y) + složky	sběr = soubor(y), ev. jakákoliv	sběr = e-mail (špatně zpracovatené)	n sběrů = n souborů

Tabulka 3.1: Porovnání možných protokolů aplikační vrstvy

5 Realizace upload skriptem nebo metodou PUT

3.1.3 Výběr koncepce přenosu dat

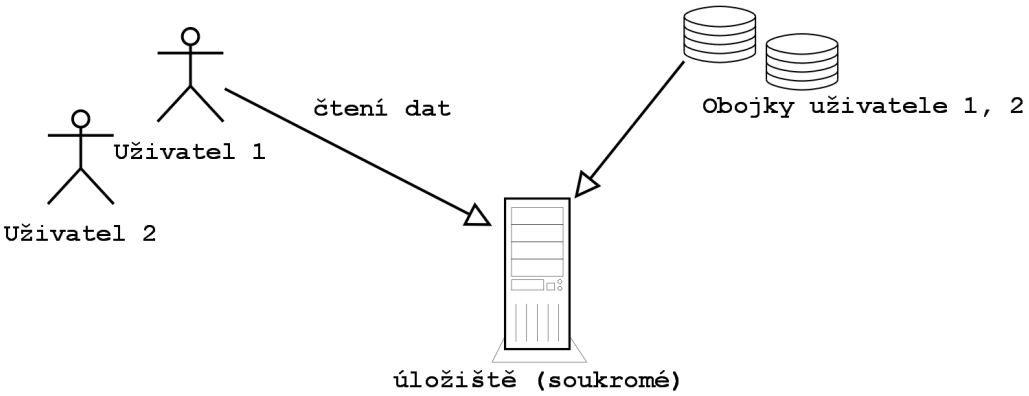
Z hlediska protokolu považuji za optimální použití FTP nebo HTTP. Klientská část nebude náročná, servery jsou běžně dostupné, buď v podobě free software nebo jako profesionální produkty. Při každé synchronizaci by obojek vytvořil soubor(y) s nasbíranými daty, která budou dále univerzálně zpracovatelná buď softwarem realizujícím centrální úložiště nebo samostatným programem. Omezení na velikost souboru je implementačně závislá, ale řádově převyšuje objemy dat přenášených obojkem. Sbíraná data nemají požadavek na úroveň bezpečnosti, takže není třeba implementovat šifrovací vrstvu.

Nenapadá mě žádná funkcionalita, kvůli které by bylo nutné navrhnovat protokol vlastní. Upload na HTTP server má podobné vlastnosti jako použití FTP, pouze vzhledem k tomu, že HTTP je primárně určeno k přenosu dat ve směru od serveru ke klientovi, bylo by nutné upload zajistit buď skriptem (např. PHP) nebo pomocí metody PUT, která je rozšířením protolu HTTP a server i klient by ji museli podporovat. Posílání dat e-mailem je velice „neohrabané“ řešení a data ze zpráv by se stejně nejspíš kopírovala do souborů za účelem rozumného zpracování. Použití NFS – souborového sdílení je naopak pro tuto úlohu příliš komplexní a její klientská implementace by mohla být značně složitá, nehledě k tomu, že různé operační systémy mají protokoly pro sdílení různé.

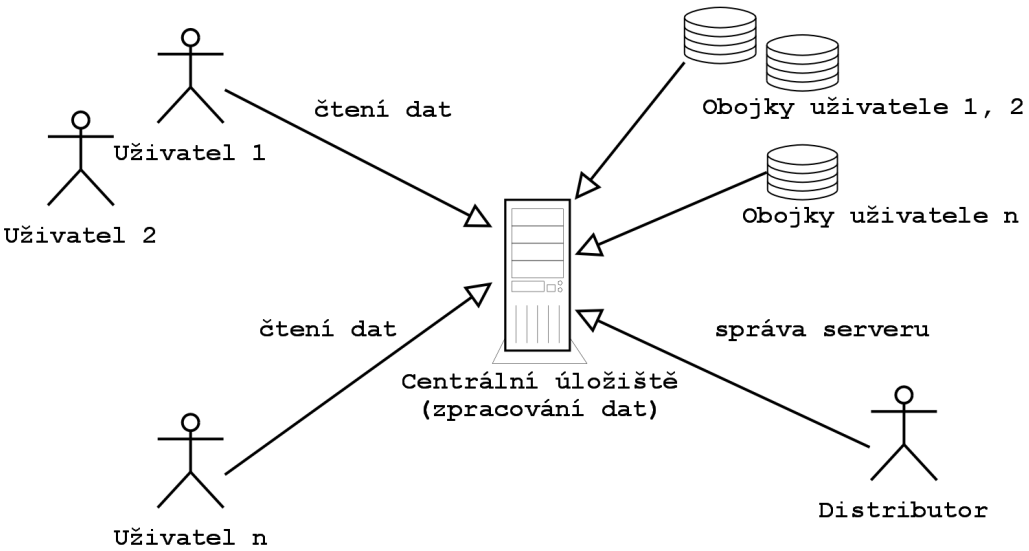
Výhodou FTP bude snazší zprovoznění serveru. Nevýhodou pak nutnost použití 2 TCP socketů, jak je uvedeno v popisu protokolu [9]. Integrované TCP stacky GPRS modemů totiž většinou nemají podporu tzv. multisocketingu, tzn. umějí používat pouze jeden TCP socket současně. Tento poznatek má však pro vybraný modul Telit GC864 minimální dopad, neboť tento modem obsahuje pro FTP protokol interní podporu. Jedná se vlastně o zapouzdření základních FTP příkazů, jejichž vnitřní syntax můžeme zjistit z příslušného RFC a vnější je v dokumentaci k modulu. Důvodem této funkcionality modulu je dle mého názoru výše zmíněná komplikace vycházející z absence multisocketingu. Většina jednoduchých, běžně používaných síťových protokolů aplikační vrstvy totiž vystačí s jedním socketem a bez této nadstavby by nebylo možné s TCP stackem modemu FTP protokol realizovat. S přihlédnutím k faktům, které zmiňuji v tomto odstavci a k tomu, že se jedná o problém bližší typickému nasazení FTP, volím tento jako implementační protokol pro přenos sbíraných dat.

3.1.3.1 Architektura přenosu dat

Architektura přenosu dat umožňuje uživateli spravovat svůj vlastní server (ilustruje obrázek 3.1) nebo využívat virtuální server spravovaný distributorem obojku (ilustruje obrázek 3.2). Zadavatel obojku žádá tuto možnost z marketingových důvodů. Obojku, uploadujícímu data na FTP server, je přirozeně jedno, kdo je jeho provozovatelem. Bude nutné pouze nastavit jméno serveru a přihlašovací údaje. V případě placené správy serveru bude zřejmě praktické i zpracování a webová prezentace dat týmž subjektem. V opačném případě se o vizualizaci dat může postarat aplikace sloužící k ovládání obojku, s tím rozdílem, že data nenačte ze zařízení připojenému k počítači, ale z uploadovaných souborů.



Obrázek 3.1: Architektura přenosu pro úložiště spravované majitelem obojku



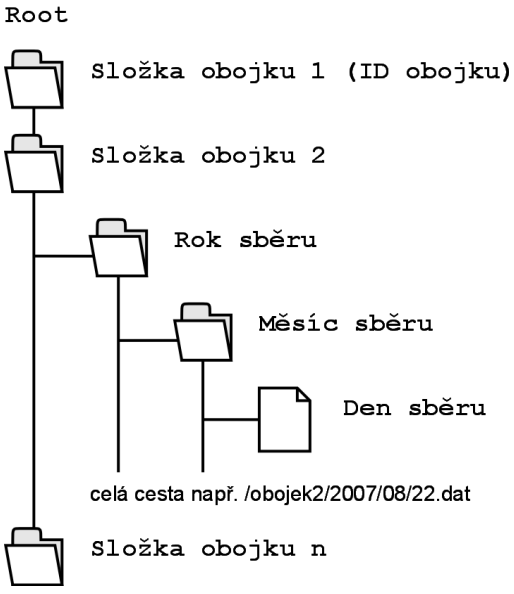
Obrázek 3.2: Architektura přenosu pro úložiště spravované 3. stranou

3.1.3.2 Požadavky na úložiště

Úložištěm dat bude standardní FTP server. Jeho zprovoznění bude spočívat pouze v konfiguraci uživatelů, jejich práv a založení adresářového stromu pro data. Každý obojek spravovaný serverem může mít vlastní login nebo může skupina obojků mít login stejný, v závislosti na potřebách statistiky. Požadavek na diskovou kapacitu je vzhledem k objemům dat vygenerovaných obojkem za jeho životnost (desítky MB) a velikostem dnešních disků velice skromný, pokud se nebude jednat o server spravující řádově desetitisíce obojků, což se s přihlédnutím k očekávanému objemu produkce stane stěží. To samé se týká rychlosti připojení k internetu, protože rychlost uploadu dat přes GPRS bude těžko přesahovat v průměru 2kB/s na obojek a navíc nemusí všechny jednotky provádět synchronizaci současně.

3.1.4 Organizace dat

Server může spravovat teoreticky libovolný počet obojků s životnostmi v řádu let. Dle výše zmíněné úvahy by se mohla synchronizace dat provádět jednou za den. To dává představu o organizaci dat na serveru ilustrovanou obrázkem 3.3.



Obrázek 3.3: Adresářový strom pro data na serveru

Takto vypadající adresářový strom má tu vlastnost, že v žádné složce, unikátní pro konkrétní obojek, se nenachází více než 31 položek. To dává při cca 64B na záznam o souboru max. cca 2kB na data o výpisu adresáře. Takovýto výpis je objem dat, který MCU bude moci uložit a dále analyzovat (např. za účelem automatické selekce položek pro synchronizaci, viz níže). V případě příliš lineární struktury – obsáhlých adresářů, by MCU nebyl schopen uložit celý výpis adresáře, který z FTP serveru obdrží. Tuto strukturu považuji rovněž za poměrně přehlednou.

Mohlo by se stát, že některý den nebudou žádné odměry dat. Potom by nebylo možné zjišťovat kontinuitu dat pouze z přítomnosti souborů. Toho by však mohl s výhodou používat postup automatické selekce položek pro synchronizaci, iniciovaný obojkem, který by procházel adresáře ve stromu dat a doplňoval chybějící soubory, které by měl k dispozici. Možným řešením by bylo v případě absence denních dat vytvářet na serveru prázdný soubor. Efektivně by měl takový soubor obsahovat sekvenci označující tento fakt, aby nemohlo dojít k záměně se situací, kdy datové připojení havaruje těsně po založení nového souboru a ten byl tak záhy interpretován jako správná prázdná data.

3.1.5 Návrh API pro integraci do aplikace

Výstupem implementace bude zdrojový kód modulu(ů) exportujících funkce pro ovládání modemu. Moje představa je taková, že by API mělo obsahovat univerzální funkce pro komunikaci s modemem prostřednictvím AT příkazů a pro účely aplikace by mělo mít podporu některých typických postupů užitím standardních i proprietárních příkazů. Užití proprietárních příkazů se bude týkat zejm. ovládání modemového TCP/IP stacku, protože tato rozšíření zatím nebyla nikým standardizována a např. všechny 4 výše zmíněné modemy realizovaly tuto funkcionalitu jinak. Samozřejmostí je dodržení všech implementačních aspektů uvedených ve standadech, jako např. definované timeouty pro různé kategorie operací apod. Předpokládám rovněž napsání vlastního driveru pro UART mikrokontroléru.

Funkce rozhraní lze rozdělit do následujících skupin:

- funkce pro přímý přístup k modemu (na úrovni AT příkazů)
- funkce pro ovládání GPRS
- funkce pro ovládání FTP
- podpůrné funkce GSM (kvalita signálu apod.)
- podpůrné funkce modemu (selekce pásem apod.)

Proprietární příkazy používají pouze funkce pro ovládání FTP a podpůrné funkce modemu. Zbylé funkce jsou použitelné s jakýmkoliv modemem podporujícím AT příkazovou sadu (standard Hayes rozšířený o ETSI GSM). Protože obojek musí být zejm. spolehlivý, budu klást důraz na ohlašování všech rozlišitelných chybových stavů prostřednictvím návratových hodnot funkcí (např. modem neodpoví ve stanoveném čase, odpoví nesprávně, odpoví správně, ale negativně atd.). Funkce pro ovádání FTP musí být dostatečně flexibilní, aby pokryly požadavky vyšších vrstev. Jde např.

o omezené možnosti přípravy dat pro upload, kdy vzhledem k velikosti paměti RAM MCU a k velikosti zpracovávaných dat (čtených z FLASH) bude nutné dodávat API funkci data po blocích na vícero volání. Časově náročné uploady dat pak bude možné pozastavovat po definované době za účelem obsloužení jiných úloh a následném pokračování v probíhajícím přenosu.

Přesná podoba funkcí prošla několika pracovními verzemi a byla dolaďována v průběhu implementace, kde uvádím podrobnosti posledního modelu.

3.1.6 Šetření energií na SW úrovni

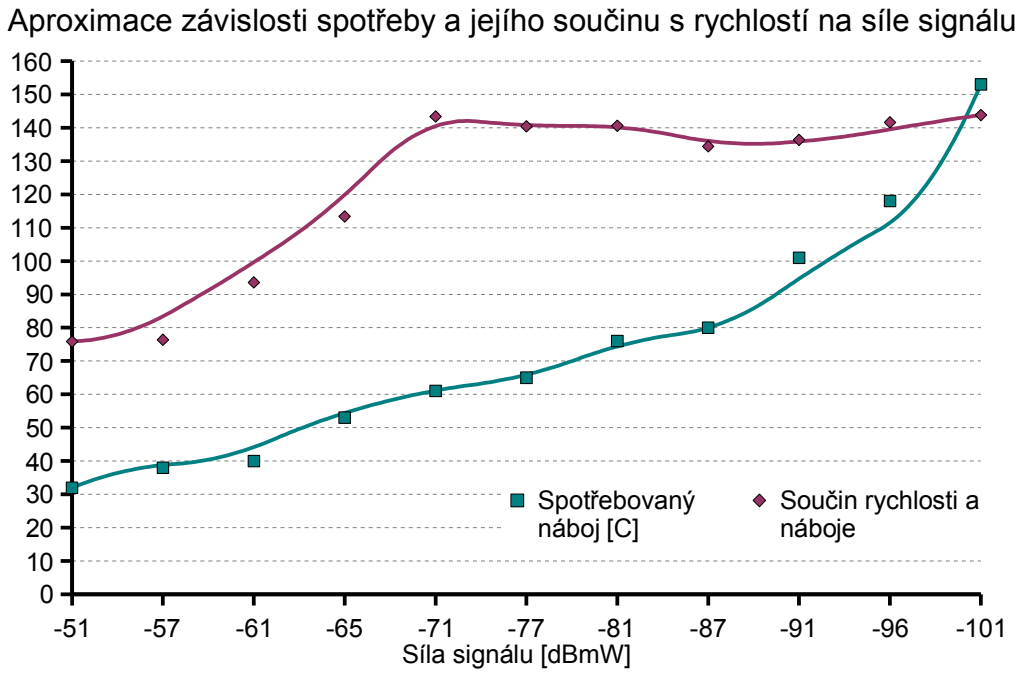
S ohledem na bateriové napájení zařízení je třeba, aby se spotřeba energie při přenosu dat dala řídit na úrovni API pro ovládání modemu. Nemůže jít o přímé řízení, jelikož vlastní výkon vysílání v GSM/GPRS je řízen sítí, ale jeho úroveň je závislá např. na síle signálu, kterou je možno softwarově zjišťovat. Půjde tedy o nepřímé řízení, kdy rozhodování o tom, zda-li se data přenesou, bude závislé na síle signálu. Principiálně lepší řešení zde není možné. Tento mechanismus lze však obohatit např. O odhad spotřeby v závislosti na objemu dat a síle signálu, a sice interpolací této závislosti pro vybraný modul. Vzhledem k objemu dat lze považovat růst spotřeby za lineární (režie je zanedbatelná vzhledem k datům > 10kB). Závislost spotřeby uploadu na síle signálu jsem zjistil jemněji odstupňovaným, takřka ekvidistantním, měřením při různých úrovních signálu, jak ukazuje tabulka 3.2.

Síla signálu [dBmW]	Rychlost uploadu [kB/s]	Spotřebovaný náboj [C]
-51	2.37	32
-57	2.01	38
-61	2.34	40
-65	2.14	53
-71	2.35	61
-77	2.16	65
-81	1.85	76
-87	1.68	80
-91	1.35	101
-96	1.20	118
-101	0.94	153

Tabulka 3.2: Spotřeba vybraného modulu vůči síle signálu pro odhad závislosti

Napájecí napětí experimentu bylo 4.2V (blokované velkým kondenzátorem), vzorek uploadovaných dat měl velikost 500 000B. Snažil jsem se o odstupňování síly signálu od nejlepšího po cca 5dBmW krocích. Ke zhoršování signálu jsem použil útlumové členy se závitem na SMA konektor, které jsem vkládal mezi anténu a modul. Měl jsem k dispozici hodnoty -3, -6, -10, -20 dBmW a kombinoval jsem je k dosažení požadovaného signálu. Doladění o ±2dbmW jsem prováděl změnou polohy nebo polarizace antény. S úrovní signálu pod -101dBmW se mi nepodařilo provést úspěšný přenos z důvodu častého timeoutu (české sítě zahazují pakety při úrovních pod -102 nebo -104dBmW, někdy i více – podle nastavení).

V naměřených datech jsou dobře patrné přechody mezi použitým kódovacím schématem, zejm. mezi CS-4 a CS-3, ke kterému dojde při síle signálu okolo -80dBmW. Takovéto skoky na závislosti mohou právě posloužit jako hranice pro rozhodování, zda-li má k přenosu dojít či nikoliv. Pro lepší ilustraci jsem závislosti spotřeby a jejího součinu s rychlostí, který je úměrný vysílanému výkonu, zpracoval graficky na obr. 3.4.



Obrázek 3.4: Graf aproximace spotřeby

Na grafu vidíme, že odebraný náboj má dle předpokladu monotónní průběh, který je do cca -85dBmW poměrně lineární. Pod touto úrovní signálu však narůstá rychleji a přenosy mohou mít až dvojnásobnou spotřebu. Při rozhodování o provedení přenosu bych jako hranici pro sílu signálu doporučoval -85dBmW, což je úroveň, které lze i při horším pokrytí dosáhnout procházkou po okolí (chování zvířete s obojkem). Pochopitelně se může ovšem stát, že úroveň signálu bude trvale příliš slabá. Pak nezbývá než přistoupit k vysílání i přes tento fakt. API proto poskytne mechanismus kontroly signálu s ohlášením neúspěchu pro příliš slabý signál, avšak po nastaveném počtu neúspěšných pokusů se pokusí data odvysílat i přesto.

Další prostor pro šetření energií, i když ne tak markantní, skýtá rozhodování, zda-li se má vůbec modem zapínat za účelem zjištění síly signálu nebo přímého provedení nějaké operace. Samotné zapnutí a registrace do sítě totiž jistou malou, ale nunulovou fixní energetickou režii generuje. Tento problém se netýká přímo implementace sběru dat, resp. jeho API, a tudíž pouze zmíním postřehy, které mě napadly v kontextu se specifikací obojku.

V hardwaru obojku je mimo jiné obsažen přijímač GPS, hodiny reálného času a náklonové senzory. Výstupy těchto podsystémů by bylo možné použít při odhadu pravděpodobnosti smysluplného zapnutí modemu. Např. při špatném signálu od GPS,

který je, dle vyjádření vývojářů obojku, zjišťován mnohem častěji, než mnou zamýšlené synchronizace dat přes GPRS, je zvíře s největší pravděpodobností v doupěti nebo v hustém porostu, kde bude zřejmě zhoršený nebo úplně nedostatečný signál i pro GPRS. Obráceně bude platit tato souvislost ještě lépe – při dobrém signálu od GPS se zvíře jistě nachází v otevřeném prostranství, kde v případě pokrytí sítí GPRS, bude signál dobrý. Navíc by šlo pomocí GPS zaznamenávat souřadnice míst s dobrým GPRS signálem a při průchodu v blízkosti těchto míst zkoušet realizovat GPRS operace. Takto by postupně vznikala mapa GPRS signálu na území, kde se zvíře pohybuje. Hodiny reálného času by šly využít pro sledování denních cyklů zvířete. V období spánku (řádově hodiny bez pohybu) lze pak opakovaně předpokládat podobné signálové podmínky, nejspíš (ale ne nutně) zhoršené, protože zvíře bude v úkrytu. K detekci podobných situací by mohly sloužit i náklonové senzory, delší dobu neměnící svoje hodnoty. Výstupy zmíněných rozhraní jsou energeticky nenáročné, nebo řádově méně (GPS) než přímé zjišťování situace v GPRS, a proto by jejich využití mohlo mít smysl. To by však vyžadovalo sofistikovanější řešení vstupů těchto rozhraní ve stávajícím firmwaru a jeho větší úpravy než při implementaci vlastního sběru dat po GPRS.

3.2 HW analýza

Hardwarová analýza se týká zejm. napájení GPRS modulu, realizace datové linky mezi ním a MCU, sumarizace obsazených signálů ve stávajícím designu přístroje a fyzického rozmístění přidaných prvků.

3.2.1 Napájení modemu

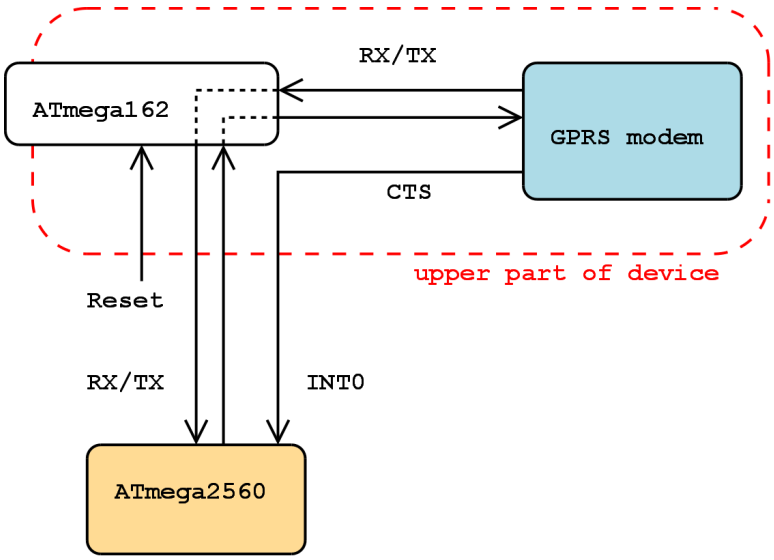
Napájení GPRS subsystému bude obstarávat hlavní primární článek. Zvolená baterie Saft má poměrně stabilní parametry ve velkém teplotním rozsahu, avšak ne dostatečně, a ještě vzhledem k jejímu nominálnímu napětí (3.65V), které je nižší než nominální napájení modulu, jsem doporučil napájení modulu přes spínaný zdroj v roli step-up měniče. To umožní lépe předcházet podpětí na modulu i např. za nízkých teplot, kdy vzrůstá vnitřní odpor článku. Napájecí uzel bude každopádně nutné dobře blokovat. Radiová část modemu má špičkové proudy okolo 2A a na mých testovacích deskách jsem osadil nedostatečné blokovací kondenzátory, což se projevilo nutností posazení napájecích napětí do horní části rozsahu napájení modulů. Zejména vybraný modul Telit GC864 je na nedostatečně blokované napájení dosti citlivý, a proto se tento fakt nesmí podcenit. Doporučuji min. blokování 100µF

tantalovým kondenzátorem a VF blokování menšími kondenzátory co nejblíže napájecích plošek modulu, které vyhladí hrany při začátku/konci vysílání v timeslotu. I přes kvalitní blokování bude, vzhledem k přítomnosti vlastního zdroje, vhodné nastavit napájení blížící se horní hranici rozsahu – 4.2V.

3.2.2 Linka datového kanálu

Data mezi GPRS modulem a MCU budou přenášena sériovou linkou RS232, což je nejběžnější způsob komunikace s modemy. MCU ovšem nemá „full-featured“ UART, tozn. UART obsahující všechny signály dle standardu V.24, ale pouze signály TX (vysílání) a RX (příjem). Pro mé účely postrádá zejména signály pro řízení toku RTS (Request to send) a CTS (Clear to send). Také napěťové hladiny logických úrovní nejsou dle V.28, ale 2.8V na straně modemu a 3.3V na druhé straně, což vyžaduje 2 jednosměrné translátory úrovní.

Modem nebude hlavním MCU komunikovat přímo, ale prostřednictvím pomocného procesoru na horní desce obojku. Toto ale bude pro implementaci API transpaprentní, neboť dolní MCU může instruovat horní procesor k vytvoření softwarového mostu pro UART a provoz na sériových linkách je logicky propojen tak, jak je možné vidět na obrázku 3.5.



Obrázek 3.5: Softwarový most UARTu pro jeho multiplexi

Řízení toku se týká situace, kdy příjemce nemá místo v bufferu pro uložení dalších znaků a řeší to nastavením příslušného signálu (RTS ze strany DTE, zde procesoru, CTS ze strany DCE, zde modulu). Kromě signálů RX/TX obsahuje propojovací kabel mezi horní a spodní deskou obojku pouze jeden další signál, který je možno využít při práci s GPRS. Ten jsem využil k připojení CTS modemu, protože bez něj by nebylo možné spolehlivě uploadovat data. TCP/IP stack modemu totiž po připojení realizuje transparentní datový kanál a informaci o připravenosti DCE přijímat do TCP kanálu data, není možné zjišťovat jinak, než na hardwarové úrovni (softwarové řízení toku modem pro TCP stack nepodporuje). Argumentace o spolehlivosti kanálu na úrovni TCP je bezpředmětná, neboť zde se jedná o problém na nižší úrovni, který je sice nepřipraveností TCP (nebo jiné vyšší vrstvy) podmíněn, ale informace o něm se do vyšší vrstvy přirozeně nedostane a ven se dostává právě prostřednictvím signálu CTS. Toto řešení je vhledem k tranpsarenci datového kanálu korektní.

Výrobce modemu doporučuje při použití integrovaného TCP stacku implementovat rovněž signál DCD (Data carrier detect), který v tomto případě signalizuje otevřený kanál. Pro tento signál již ale ve stávajícím designu obojku nejsou HW prostředky. Narozdíl od CTS, který není principiálně možno obejít, tento problém jsem vyřešil v implementaci softwarově. Rovněž na RTS nezbývá místo, avšak tento signál není kritický, protože při downloadu dat bude firmware alokovat určitou velikost bufferu v RAM MCU a k jejímu zvětšení by vzhledem k její omezené velikosti stejně nemohlo dojít. Download se bude týkat převážně menších bloků dat (programování obojku, listing FTP adresáře) do cca 2kB. Modul pro ovládání modemu má za úkol vystačit s 3kB RAM, takže mohu alokovat 2kB na buffer pro příjem souborů. V případě bloků dat, která jsou větší než dostupná RAM (upgrade firmware) bude nutné data průběžně zpracovávat nebo ukládat do externí paměti (FLASH), což není při probíhajícím přenosu problém stihnout vzhledem k řádově pomalejší a navíc nastavitelné rychlosti sériové linky. Ideální by samozřejmě bylo implementovat obousměrnou signalizaci, resp. řízení toku. To však stávající design obojku neumožňuje a mnou navržené zapojení je minimální k dosažení spolehlivých přenosů a detekce chyb.

3.2.3 Obsazené signály

Veškeré HW signály které GPRS systém využivá jsou:

- napájení = zem a napájecí uzel ze step-up měniče
- zapínání = 1 logický signál z pomocného CPU
- sériová linka = RX a TX na UARTu pomocného CPU
- CTS = 1 logický signál (INT0) z hlavního MCU
- 4 signály (VCC, RST, IO, CLK) pro komunikaci se SIM kartou

Všechny logické signály mají úrovně 2.8V. Zapínací signál má pull-up rezistor uvnitř modulu, takže mu stačí výstup typu open-drain s aktivní úrovní v nule. Komunikace se SIM kartou je v režii modulu a z hlediska jeho vnějšího rozhraní existují tyto 4 signály pouze, pokud by měla být SIM karta umístěna fyzicky jinde než modul, což v designu obojku nebude. Není k tomu důvod a stejně tomu brání absence volných signálů v propojení dolní a horní desky, nehledě k tomu, že pás obojku může mít u větších zvířat pro tyto signály příliš dlouhé vodiče.

3.2.4 Rozmístění přidaných prvků

GPRS modem v podobě modulu, SIM karta a jejich okolní pasivní součástky budou umístěné v horní poloze obojku, osazené na jeho horní desce spojů. Napájecí měnič modulu bude umístěn v blízkosti modulu, tzn. rovněž na horní desce. Anténa pro GSM bude umístěna v pásu obojku. Signály pro datový kanál a napájení měniče povedou stávající vodiče v ohebné DPS v pásu obojku. Ostatní obsazené signály vedou mezi modulem a pomocným procesorem na horní desce.

4. Implementace

Zdrojový kód firmwaru obojku je psán v jazyce C, tudíž byl požadavek na použití tohoto jazyka i v mojí implementaci. Jazyk C má dobrou podporu pro použitý procesor AVR ATmega. Existuje několik kompilátorů tohoto jazyka pro tuto architekturu, např. CodeVision AVR, gcc-avr, IAR Embedded Workbench, ICC AVR, SCC AVR. Výrobce procesoru, firma Atmel, poskytuje zdarma vývojové prostředí AVR Studio, které obsahuje kompilátor assembleru, programátor, simulátor a podporu pro debugging kompatibilní s některými komerčními prográmatory/debuggery a ICE.

4.1 Vývojový toolchain

Firma Czechlabs mně na implementaci zapůjčila programátor Atmel JTAG-ICE mkII. Je připojitelný do COM portu nebo USB a spolupracuje s AVR Studiem. Obojek má vyvedený konektor pro ISP se signály pro JTAG i SPI. JTAG-ICE mkII umí používat obě rozhraní, pro debugging je nutné použít JTAG. Vzhledem k chybě ve schématu obojku – posunutí signálu pro SPI na tomto konektoru jsem používal pouze rozhraní JTAG, které je univerzálnější.

Pro kompilaci jsem zvolil open-source projekt WinAVR (winavr.sourceforge.net), což je balík programů ze sady GNU Binutils pro architekturu AVR včetně GNU C kompilátoru, knihovny libc – runtime knihovny s funkcemi ANSI C a podpůrných hlaviček s definicemi procesorů AVR. Obsahuje ještě simulátor AVR simluavr, GNU debugger gdb, backend avarice pro použití hardwarového debuggeru a některé podpůrné knihovny a další nástroje běžné v linuxovém prostředí (zejm. make), které jsou zde převzaty z projektu MSYS (www.mingw.org). Novější verze AVR Studia má podporu tohoto balíku a při zakládání projektu nabízí možnost spolupráce s jeho kompilátorem. AVR Studio pak slouží jako editor zdrojového kódu a na základě informací o souborech projektu a jeho nastavení připravuje Makefile (soubor s informacemi o postupu kompilace a linkování) pro program make. Já jsem pro editaci souborů se zdrojovými kódy používal linuxový editor vim a pomocí AVR Studia jsem prováděl pouze ladění.

Kompilátor avr-gcc provádí kompilaci a linkování modulů. Generuje binární soubor ve formátu ELF s debugovacími informacemi typu DWARF2. Oba formáty podporuje debugger AVR Studia ve verzi 4.10 a výše. Jednotlivé typy procesorů

v architektuře AVR se liší velikostmi pamětí, hardwarovou výbavou atd. Typ cílového procesoru se posílá kompilátoru v argumentu jako „target specific option“ a dále je využit např. hlavičkovými soubory pro definice názvů a umístění registrů. Ve verzi 4.1.2 již kompilátor podporuje poměrně nový procesor ATmega2560, použitý v designu obojku. Soubory s binárními daty (resp. v textovém formátu .hex) jednotlivých sekcí strojového kódu pro programování pamětí mikrokontroléru (paměť programu FLASH, EEPROM) vyrábí nástroj objcopy z výstupu kompilátoru.

4.2 Integrace do stávajícího návrhu

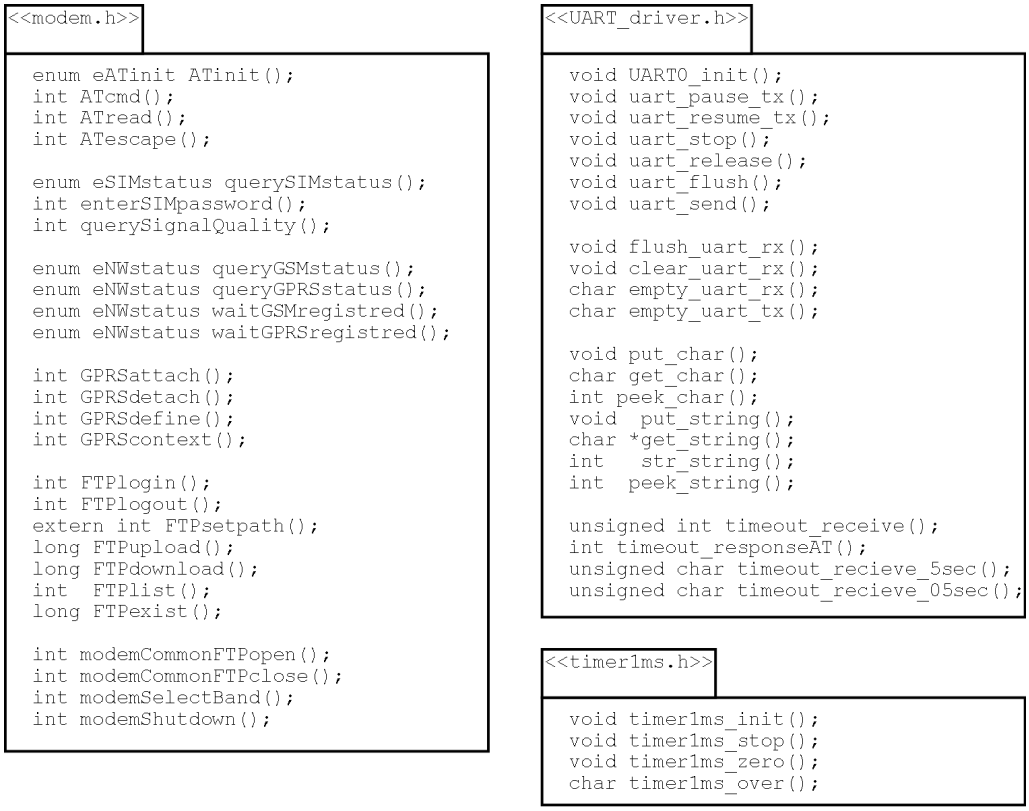
Původní firmware obojku, který jsem dostal k dispozici, sestával ze 17 modulů. Pro mne zajímavé byly 3 moduly a několik hlaviček. Šlo o hlavní modul, modul ovládající napájení částí obojku, pomocí kterého jsem zapínal napájení pro translátory úrovní na sériové lince a ovládal „debugovací“ LED na spodní desce obojku, a modul k ovládání UARTu. Původní driver UARTu jsem však měl potřebu rozšiřovat a nakonec jsem ho přepsal celý. Je ale zpětně kompatibilní s původním. Několik dalších změn, které byly vhodné nebo jsem byl nucen udělat do původního firmwaru, jsem zapisoval do changelogu za účelem jeho budoucí konzultace s vývojáři obojku.

4.3 Zdrojové moduly a exporty

Mnou napsaná část firmwaru jsou 3 moduly:

- modem.c – modul s API funkcemi pro ovládání modemu
- UART_driver.c – ovladač sériového portu MCU
- timer1ms.c – malý modul k realizaci časování (pro měření timeoutů apod.)

Jejich exporty uvádím na obrázku 4.1.



Obrázek 4.1: Exporty implementovaného API

Přesné prototypy všech funkcí zde neuvádím. Jsou uvedeny spolu s komentáři k jejich použití a návratovým hodnotám v hlavičkových souborech zdrojových modulů. Zdrojové kódy jsou komentované na místech, kde jsem to považoval za vhodné. Všechny komentáře, názvy proměnných a jiné identifikátory jsou v jednotném jazyce – angličtině. Text zdrojů má jednotnou úpravu řádkování a odsazování, pořadí maker, deklarací a definicí. Kompilováno bylo se zapnutým varováním na všechno (-Wall), se striktními prototypy a s optimalizacemi úrovně 2 (-O2). Překlad neobsahuje žádná varování.

4.4 Implementační detaily

Přesto, že veškeré implementační detaily lze vyčíst ze zdrojových kódů, v této subkapitole popíšu některé z nich, které považuji za vhodné zmínit i v dokumentaci k práci.

4.4.1 Vyrovnávací paměti driveru pro UART

Obě vyrovnávací paměti – vysílací i přijímací jsou realizovány kruhovými buffery s velikostí nastavitelnou makrem. Vlastní vysílání i přijímání je spouštěno přerušením. Přijímací buffer umí nahlížet („peekovat“) na své relativní pozice odpředu nebo odzadu vzhledem k aktuálnímu stavu zaplnění. Toho je využíváno např. při hledání hodnot čtených parametrů v modemovém výstupu nebo při čekání na přijetí ukončovací sekvence odpovědi na modemový příkaz. Realizace timeoutů pro příkazy modemu používá funkci driveru, která čeká na podmínku formátu posledních dat v přijímacím bufferu nebo dokud neuběhne stanovený čas. Při přetečení přijímacího bufferu se nastavuje globální příznak.

4.4.2 Funkce FTPupload()

Funkce FTPupload() odesílá data do souboru na přihlášeném FTP serveru. Její přesná deklarace je:

```
long FTPupload( char const *file, char const *data, long
datalen, unsigned long time, char keep );
```

Argument file je název uploadovaného souboru v aktuální cestě otevřeného FTP spojení, data a datalen je ukazatel na data v paměti a jejich délka. Další 2 parametry se týkají možnosti použití opakovaného volání a zavedl jsem je z důvodu mnou předpokládaného typického použití této funkce. Doba uploadu většího objemu dat může totiž být nezanedbatelná vzhledem k obsluze ostatních částí obojku, a proto je nutné zavést jakousi primitivní formu kooperativního multitaskingu. Toho se týká argument s názvem time, což je doba maximálního blokování kódu unitř této funkce. V případě jeho použití (pokud není nula) se funkce vrátí nejvýš po uplynutí stanoveného času (v milisekundách). Funkce vrací délku dat, která nestihla poslat, tzn. nulu v případě úspěchu, kladné číslo pro pozastavení kvůli vypršení času. V případě výskytu chyby je vráceno záporné číslo dle definic maker v hlavičkovém souboru. Při časovém pozastavení funkce je očekáváno následné volání nad stejnými daty a funkce pokračuje v jejich odesílání od přerušeného místa. Tato informace je uložena staticky a funkce tedy není reentrantní, což ovšem není na závadu vzhledem k tomu, že je možné obsluhovat max. 1 FTP připojení už na úrovni TCP stacku modemu.

Argmuent keep se týká situace, kdy MCU hodlá odesílat objem dat, který je příliš velký pro jeho celé uložení do RAM. Nenulová hodnota tohoto argumentu způsobí, že funkce neukončuje datové spojení při odeslání celé délky dat (soubor zůstane otevřen). Další volání této funkce pokračuje v uploadování otevřeného souboru novým blokem dat. Tento mechanismus umožňuje MCU kopírovat z FLASH bloky dat o libovolné velikosti a nahrát je do jednoho souboru.

Oba zmíněné přístupy je možné kombinovat a uploadovat soubor např. po 2kB blocích s obsluhou zbytku HW po 0.4 sec, což je doba, za kterou se 2kB blok dat odeslat určitě nestihne. Při přerušení datového kanálu na různých úrovních má funkce výstup dle tabulky 4.1 (odsimulováno).

Chybující vrstva	Typická příčina	Význam návratové hodnoty funkce
Fyzická	přerušení vodičů UARTu	timeout kvůli CTS, event. nemožnost přepnout modem do příkazového módu
Fyzická/linková	výpadek GPRS signálu, pád BTS	timeout kvůli CTS
Síťová/transportní	rozpad TCP pádem síťového uzlu	nepotvrzení přenosu serverem
Aplikační	rozpad TCP pádem FTP serveru	nepotvrzení přenosu serverem
Aplikační	neočekávané odpojení klienta	rozpad datového kanálu při přenosu

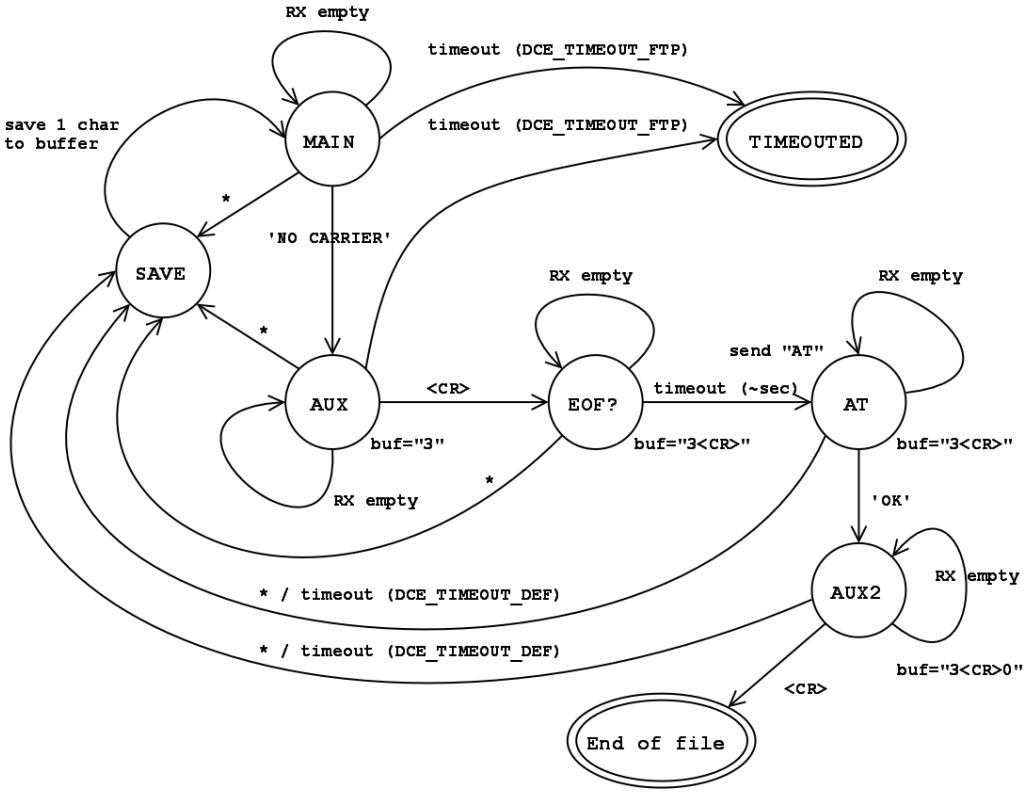
Tabulka 4.1: Návratové hodnoty funkce FTPupload() při přerušení kanálu

4.4.3 Funkce FTPdownload()

je funkce, která přijímá data ze souboru na FTP serveru do bufferu v paměti MCU. Její přesná deklarace je:

```
long FTPdownload( char const *file, char *buffer, int buflen,
char keep );
```

3 argumenty jsou společné s argumenty funkce FTPupload(). Argument keep zde funguje podobně, pro dělení dat do bloků. Pokud je použit, funkce nečeká na konec souboru a vrací řízení okamžitě při naplnění bufferu. Dalším voláním lze přijmout další blok a opakovat tento postup až do konce souboru. Zajímavost této funkce spočívá v řešení detekce konce souboru. Komunikace s modemem je realizována jen pomocí 2 drátové sériové linky (+ při downloadu nepoužité CTS) a v průběhu práce s TCP stackem modulu je celá rezervována pro jeho datový kanál. Konec souboru při uploadu je řešen standardní modemovou escape sekvencí, která má speciální časování. Sekvencí označující konec souboru při downloadu je však odezva na vykonaný příkaz, která přichází ze strany MCU po stejném kanálu, po kterém do té doby přicházela data souboru. Samozřejmě je nutné počítat s tím, že znaková sekvence této odezvy se může vyskytovat v přenášených datech. Navíc je nutné počítat s možným vytimeoutováním celého přenosu. Pro korektní řešení tohoto problému jsem sestavil následující stavový automat naznačený na obrázku 4.2.



Obrázek 4.2: Stavový automat downloadu

Algoritmus pro detekci konce downloadu tedy za platný konec prohlásí pouze sekvenci odezvy modemu, za kterou následuje časová prodleva, a po které se modem sám přepnul do příkazového módu, což zjišťuje jeho odpovědí na testovací příkaz

(v případě, že modem v příkazovém módu není, je testovací příkaz ignorován a nezpůsobí poškození dat). V případě nesplnění všech uvedených podmínek je sekvence interpretována jako souborová data a uložena do bufferu. Mám otestováno, že pro funkci není problém přijmout jakákoliv data (tzn. i soubor plný ukončovacích sekvencí). Z každého stavu automatu vede cesta do stavu TIMEOUTED, po které automat přechází v případě dlouhodobé (makrem nastavitelné) nečinnosti, takže je zaručena maximální doba blokování zbytku kódu voláním této funkce.

4.4.4 Funkce modemCommonFTPopen()

Postup volání, který od zapnutí modemu předchází typickému užití API pro upload nebo download souborů, jsem zapouzdřil do této procedury:

```
int modemCommonFTPopen( int minSQ, int attemps_ignoring_SQ );
```

Má za úkol inicializovat modem, nastavit preferované RF pásmo, zadat PIN pro SIM kartu (pokud je vyžadován), vyčkat na registraci do sítě, zjistit úroveň signálu, definovat PDP kontext pro GPRS, aktivovat ho, přihlásit se na FTP a nastavit domovskou cestu konkrétního obojku jako aktuální. Při neúspěchu některého stupně inicializace je vrácen příslušný chybový kód.

Pokud je použit argument minSQ (je záporný), je požadováno, aby síla signálu byla lepší než tato hodnota (v dBmW). Jestliže tato podmínka není splněna, funkce nepokračuje. Tomuto chování lze předejít užitím druhého argumentu, který vystupuje v roli počtu minulých pokusů o připojení, které selhaly v důsledku špatného signálu, po jehož naplnění dojde k připojení nezávisle na signálu.

Nastavení, která tato funkce používá k inicializaci (PIN k SIM, APN v GPRS, login na FTP apod.) jsou v hodnotách „konfiguračních parametrů“. Tyto parametry jsou připraveny k začlenění mezi ostatní parametry zařízení, uložené v EEPROM, se kterými lze manipulovat pomocí PC aplikace k ovládání obojku. Deinicializaci stavu, do kterého bylo přejito tímto voláním, obstarává funkce modemCommonFTPclose().

4.4.5 Časová robustnost kódu

Všechna místa ve zdrojovém kódu, která vyžadují přesné časování (výpočet předděličky pro hodiny UARTu, timeouty pro příkazy modemu), jsou napsána univerzálně vzhledem k použité frekvenci procesoru. Výpočet souvisejících konstant se děje při překladu, takže nezatěžuje běh programu dodatečnou aritmetikou.

Hodinová frekvence cíle se nastavuje argumentem kompilátoru a projekt v AVR Studiu toto nastavení obsahuje. Všechny použité příkazy modemu mají nastavené timeouty dle specifikace AT příkazové sady. V případě proprietárních příkazů, u kterých výrobce timeout nedefinoval (zejm. ty, které se týkají TCP stacku) jsem zvolil hodnoty, které považuji za použitelně nízké a v praxi s nimi nebyly problémy ani při krátkodobých výpadcích sítě apod. Tyto hodnoty jsou definovány v komentovaných makrech.

4.5 Ladění

Pro ladění implementovaných funkcí jsem vyrobil propojovací kablík mezi konektorem pro horní desku obojku a testovací deskou GPRS modulu. Obsahuje obsazené signály sériové linky a je možné na něm přepínat jejich křížení tak, aby šel propojit jak obojek s modemem, tak obojek s PC (k tomu jsou využity stejné konvertory úrovní na RS232, které za normálních okolností figurují mezi modemem a PC). V emulátoru terminálu jsem pak mohl simulovat obě strany a sledovat jejich odezvy. Za jistých okolností se dal i odposlouchavat živý provoz.

Vlastní ladění jsem prováděl v debuggeru AVR Studia. V případě, kdy jsem potřeboval výstup z obojku (nejčastěji návratovou hodnotu funkce) při „ostrém“ běhu, využil jsem přítomnosti LED na desce obojku. Počet krátkých/dlouhých bliknutí LED korespondoval s výstupním kladným/záporným číslem. Tato velice minimalistická metoda výstupu dat mi vždy stačila a ze strany vývojářů obojku byl dobrý nápad tuto debugovací LED osadit. Ve většině případů jsem měl však procesor v debug režimu přes JTAG. Při odlaďování API pro FTP operaci jsem používal zdarma dostupný server TYPSoft FTP Server, který běžel na mém domácím PC s konektivitou 8Mbit/2Mbit.

Vyskytl se problém, který se projevoval tím, že při uploadu většího souboru (> cca 100kB) byl jeho obsah od jistého okamžiku poškozen. Software by mohl jen těžko generovat chybu takovéhoto charakteru a při analýze poškození jsem si všiml souvislosti mezi původním znakem a znakem, který dorazil na FTP server. Např. při kontinuálním uploadu znaku 'A' začal po jisté době přicházet znak 'P' nebo znak s kódem 0x05. Dále jsem si všiml toho, že k proměně znaků dojde vždy po pauze ve vysílání způsobené aktivací signálu CTS ze strany modemu. Chybné znaky měnily svoji hodnotu i v závislosti na nastavení počtu stopbitů. Souvislost mezi zmíněnými znaky spočívá v kódování rámce pro RS232. Znak 'A' má hodnotu 0x41 = 01000001, vysílaný rámec formátu 8N1 (8 datových bitů, žádná parita, 1 stopbit) vypadá "0 10000010 1". Znak 'P' má hodnotu 0x50 = 01010000, vysílaný rámec stejého formátu vypadá "0 00001010 1". Při rotaci rámce pro 'A' o 2 bity později dostáváme rámec pro 'P'. Při rotaci o 6 bitů pak pro znak 0x05 (přehození nibblů). V obou případech se jedná o validní rámec zakončený stopbitem v podobě jedničky. Dokonce i poměr statistických pravděpodobnosti změny na 0x50 a na 0x05 odpovídal poměrům časů před možnou synchronizací na další sestupnou hranu (úměrné počtu nul před přechodem z '1' jako stopbit do '0' jako starbit). Jednalo se tedy o problém synchronizace na správný začátek rámců přijímaných modemem. Při připojení obojku k PC se však problém neprojevil po libovolně dlouhé době vysílání. Nejprve jsem myslel, že problém tkví v nepřesné frekvenci hodin pro UART na straně MCU. S krystalem 8MHz, který byl použit v designu obojku nelze dobře generovat žádnou ze standardních modemových rychlostí. Já jsem použil nejrychlejší možnou, která měla ještě únosnou chybu, konkr. 38400bps s chybou +0.2% (tj. ~ 38462bps). Zmíněný problém se ovšem projevil i při jiných rychlostech s chybou 0.2% a s vyšší chybou (pak až v řádu %) nefungovala sériová linka vůbec. Zakoupil jsem tedy krystal 7.3728MHz, jehož hodnota je celočíselným násobkem standardních modemových rychlostí a vyměnil původní za tento. Zmíněný problém však přetrvával. Nakonec pomohlo až to, když jsem propojil vysílací signál UARTu drátkem od padu procesoru přímo na konektor UARTu testovací desky s modemem. Obešel jsem tak translátor úrovní z procesorových 2.8V na 3.3V, což je napětí, kterým se komunikuje s horní deskou obojku (UART je vyveden na konektoru, kterým se horní deska připojuje). Vypadá to tedy, jako kdyby po delší době (po pauze ve vysílání kvůli CTS) neměnné úrovně na vstupu onoho translátoru a následné hrany nedošlo k jejímu správnému projevení na výstupu. Přijímač na straně modemu se pak špatně synchronizuje na některou následující. Přijímač v PC však dokáže průběh zotavit a data interpretuje správně. O tomto chování a svých poznatcích jsem následně informoval vývojáře obojku.

Ještě jsem objevil chybu v FTP stacku použitého modulu. Projevuje se po uploadu souborů větších než cca 1MB a to tím, že si firmware modemu pro aktuální přihlášení špatně uchovává informaci o poslední serverové zprávě na řídícím kanálu FTP protokolu. V mém API se to projeví tím, že po uploadu velkého souboru funkce občas vrátí místo úspěchu konstantu, která označuje přenesení souboru, avšak nepotvrzené serverem. Tento problém nemohu ze své pozice vyřešit. Naštěstí se netýká typického použití této funkce aplikací, při kterém se počítá s řádově menšími velikostmi souborů. Pokud by firma Czechlabs projevila zájem o jeho vyřešení, konzultoval bych ho s vývojáři modemu. Také mi nefungovaly některé, pro implementaci nekritické, proprietární příkazy modemu, které výrobce v dokumentaci uvádí. Konkrétně se jednalo o příkazy #FTPTO, #MSCLASS, #CGCLASS a #AUTOBND.

4.6 Statistika implemetace

Implementace obsahuje celkem 57 funkcí, rozpis pro jednotlivé moduly ukazuje tabulka 4.2.

Modul	Funkcí celkem	Exportovaných	Statických	Obsluhy přerušení
modem.c	29	26	3	0
UART_driver.c	24	22	0	2
timer1ms.c	4	4	0	0

Tabulka 4.2: Statistika počtu funkcí modulů

Modul modem.c navíc obsahuje funkci main(), která byla použita při testování. Vzhledem k tomu, že z původního firmwaru obojku jsem do svého projektu použil minimální možnou množinu modulů (konkr. 1 modul a 4 hlavičky) tak, abych mohl svoji část kódu ladit a testovat na živém HW, bylo nutné hlavní funkci někde definovat. K výrobě modulů linkovatelných s hlavním modulem zdrojových kódů obojku ji stačí oddefinovat.

4.6.1 Obsazení prostředků MCU

V implementaci je obsazen UART (resp. USART) s indexem 0 a čítač/časovač 3. Obsazení pamětí je následující:

- Program: 14422 B (5.5% zaplnění, sekce: .text + .data + .bootloader)
- Data: 1086 B (13.3% zaplnění, sekce: .data + .bss + .noinit)

Zhruba polovinu obsazení datové části (tzn. v RAM při run-timu) zabírají buffery ovladače UARTu (konkr. v sekci .noinit). Vzhledem k nárokům původního firmwaru jsem byl s vývojáři obojku dohodnut, že zdrojové kódy pro GPRS subsystém by neměly zabrat víc než cca 2kB v RAM. Tento požadavek byl tedy splněn.

5. Testování

Vzhledem k tomu, že dílčí testování funkcí bylo prováděno v průběhu jejich odlaďování, jsem po skončení implementace prováděl zejm. testování různých přístupů při uploadování a downloadování dat. Zkoušel jsem přenášet data paralelně s obsluhou jiného HW (blikání LED). Dále jsem přenášel soubory po malých blocích na velký počet opakovaných volání a testoval tak budoucí typické použití postupu pro upload dat. Rovněž jsem zkoušel i kombinaci obou přístupů a některé další situace, včetně simulace havarovaného kanálu.

5.1 Podmínky testování

Testovací zapojení obsahovalo dolní desku obojku propojenou datovým kablíkem, (s konektorem pro přípojení horní desky) s testovací deskou GPRS modulu. Obojek byl napájen ze svého primárního článku, modem z laboratorního zdroje. Připojená anténa byla prutová pro GSM a testováno bylo při relativně dobrém signálu. Jako server aplikačního protokolu byl použit výše zmíněný TYPSoft FTP Server a testovacími daty byly většinou regulární vzorky textu.

5.2 Výsledky testování

Všechny testy proběhly s úspěšnými návratovými hodnoty funkcí (kromě simulací havarovaného kanálu, jejichž výstupem byly hodnoty dle tabulky 4.1) a při porovnání přenesených dat s původními nebyl nalezen rozdíl. Při testech uploadu bylo dosahováno průměrných rychlostí okolo 2kB/s. Download vykazoval zhruba 3x větší průměrné rychlosti. Některé z testovacích sekvencí volání jsem ponechal v poslední verzi zdrojových kódů (ve funkci main).

6. Zhodnocení

V této práci se podařilo nastudovat možnosti užití technologie GPRS v telemetrickém obojku pro divoká zvířata. Na základě požadavků aplikace byl vybrán konkrétní typ modemu, který je pro zařízení vhodný. V rámci analýzy jsem navrhl možné postupy pro sběr dat. Vzhledem k prioritě energetických aspektů jsem provedl měření spotřeby v závislosti na parametrech prostředí s přihlédnutím k typickému nasazení ve volné přírodě. Tyto výsledky, spolu s dalšími nabytými poznatky, mohou být respektovány při použití implementovaného softwarového rozhraní pro ovládání modemu. Toto rozhraní je sada funkcí pro realizaci datových přenosů k synchronizaci dat sbíraných obojkem s centrálním úložištěm. Rozhraní obsahuje i další podpůrné funkce pro práci s modemem a je připraveno pro některé, v budoucnu využitelné operace, např. upgrade firmwaru. Implementace byla odladěna v existujícím mikropočítačovém systému s procesorem Atmel AVR. Její statistika je v kapitole 4.6.

6.1 Dílčí problémy a jejich řešení

V rešeršní části se zásadní problémy, až na nemožnost kontaktovat firmu Siemens ohledně jejích produktů, nevyskytly. V analytické části se vystkytl problém s realizací řízení toku na sériové lince kvůli nedostatku volných signálů ve stávajícím designu obojku. To bylo vyřešeno jednosměrným hardwarovým řízením toku signálem CTS a softwarovým získáním informace o stavu signálu DCD, které bylo provedeno v rámci implementace. Implementace probíhala vesměs hladce, jediný závažný problém spočíval v poškození uploadovaných dat větších souborů. Tento problém souvisel s obvodem pro translaci napěťových úrovní osazeným na dolní desce obojku. Podrobněji ho rozebírám v kapitole 4.5.

6.2 Zkušenosti s použitými nástroji

Kompilátor GNU C pracoval celou dobu bezvadně, což je ostatně jedna z jeho známých vlastností. Už horší zkušenost mám s AVR Studiem, které po delší době práce zabíralo velké množství paměti a občas celé IDE spadlo se ztrátou neuložených dat. Atmel označuje na svém webu za stabilní verzi s major číslem 3. Ta však neobsahuje spolupráci a podporu pro debugovací informace z avr-gcc a definice poměrně nového procesoru ATmega2560. Proto jsem upřednostnil novější, avšak jak

se ukázalo, méně stabilnější verzi 4. S debuggerem JTAG-ICE mkII jsem měl však značné potíže. Občas prostřednictvím AVR Studia hlásil nemožnost identifikovat cíl, ačkoliv od jeho minulého fungování se na připojení všeho HW nic nezměnilo. Řešení pak bylo otázkou restartů programátoru, AVR Studia či celého OS a následného připojování částí debugovacího prostředí v definovaném pořadí. Vzhledem k tomu, že AVR Studio obsahuje chybu, kdy neuvolňuje zásuvku debuggeru, nevylučuji, že i zmíněné potíže mohou souviset spíše s ovládacím SW než s programátorem samotným. Naopak textový editor vim, který používám k editaci zdrojových kódů, je velmi spolehlivý SW a i při výpadku systému umí zotavit poslední změny v souborech ze svých odkládacích dat.

7. Závěr

Výsledkem této bakalářské práce je funkční softwarové rozhraní pro sběr dat z mikropočítačové aplikace užitím GPRS. V rámci práce byly obstarány vhodné GPRS modemy, mezi nimiž byl vybrán ten, který nejlépe splňoval požadavky aplikace. Při výběru modemu byly vyrobeny desky spojů, na kterých probíhalo jejich testování. Pro vybraný modem byla provedena softwarová a hardwarová analýza jeho integrace do stávajícího designu zařízení. Na jejich základě pak bylo implementováno rozhraní pro jeho ovládání a úkony souvisejícími se zadáním práce.

Práce značným způsobem obohatila mé znalosti v oblasti GPRS. Dále jsem získal nové zkušenosti s mikrokontroléry AVR, tvorbou softwaru pro tuto architekturu v jazyce C a jejich debugováním. Myslím si, že práce určitě má potenciál budoucího rozšiřování.

8. Zdroje

[1] Vít Záhlava: *OrCAD 10.* Grada Publishing, a. s., 2004.
[2] Vladimír Váňa: *Mikrokontroléry Atmel AVR.* , BEN technická literatura, 2003.
[3] Schéma zapojení a osazovací plán obojku.
[4] Webové stránky a technická podpora výrobců z tab. 2.3.
[5] Vývojová dokumentace k vybraným GSM modemům[6].
[6] Dokumentace k vývojovým deskám modemů Simcom, Enfora[6].
[7] Datasheety k součástkám použitým v HW designu[6].
[8] Manuálové stránky balíku WinAVR, překladače avr-gcc a knihovny avr-libc[6].
[9] RFC 959 (FTP).
[10] Wikipedia, the free encyclopedia.
http://wikipedia.org
[11] GSM World – the world wide web site of the GSM Association.
http://gsmworld.com
[12] GSMweb.cz.
http://gsmweb.cz
[13] Pokyny pro psaní závěrečných prací na katedře počítačů FEL ČVUT.
https://info336.felk.cvut.cz

6 V příloze na CD

9. Přílohy

9.1 Obsah přiloženého CD

- `/bp.pdf` — Vysázený text práce – tento dokument
- `/doc-gsm-brief/` — Stručná dokumentace k modemům
- `/doc-gsm-full/` — Podrobná vývojová dokumentace k modemům
- `/doc-other/` — Datasheety k součástkám, manuál avr-libc
- `/implement/` — Implementace, projektové soubory
  - `backups/` — Zálohy verzí zdrojových kódů
  - `fw/` — Informace k původnímu firmwaru
    - `changelog.skalda` — Změny provedené v původním firmwaru
  - `synchro_corrupts/` — Projevy chyby popsané v kapitole 4.5
- `/txt/` — Zdroje pro sazbu textu práce
  - `appendix/` — Přílohy
  - `bp-zadani.rtf` — Původní znění zadání práce (před schválením)
- `/web-denik/` — „Deníkový web“ začátku práce

9.2 Tabulka porovnávající parametry všech modemů

num.	Type	Producer	Technical features	Power U/I	Operation temperature	Dimensions [mm x mm]	Weight [g]	Pb free ?	Note
1	**GPRS Module 4-Band**	**Advanced Wireless Planet** gsm-modem.de	Band:Quad GPRS: class 10 (class B) AT set: yes	3.4 – 4.2V, nominal: 3.8 Idle mode: < 3.5mA, speech mode: 250mA (average)	-20 to +70	43.9 x 43.9 x 6	20		Easy GPRS (embedded TCP/IP stack) SIM reader inside
2	**GM862-GPRS** **GM862-PCS**	**Round Solutions** roundsolutions.com	Band: Dual/Tri GPRS: class 8/10 AT set: yes	3.4 – 4.2V, nominal: 3.8 Idle mode: < 3.5mA, speech mode: 250mA (average)	-20 to +70	43.9 x 43.9 x 6	20		Easy GPRS (embedded TCP/IP stack) SIM reader inside
3	**Trizium (Gx863)**	**Round Solutions / Telit**	Band: Tri, Quad GPRS: class 10 AT set: yes	3.4 - 4.2V, nominal: 3.8 Idle mode: <3.5 mA, speech mode: 250 mA, GPRS: 350mA	-25 to +75	41.4 x 31.4 x 3.6	9	Verze Quad	Easy GPRS (embedded TCP/IP stack)
15	**GC864**	**Telit** www.telit.co.it	Band: Quad GPRS: class 10 AT set: yes	3.4 – 4.2V Idle mode: <3.5mA, Voice mode: 270mA, GPRS: 500mA max.	-30 to +80	37 x 30 x 2.8	7	RoHS compl.	TCP/IP stack FTP, SMTP Python interpreter (opt)
16	**GE864**	**Telit**	Band: Quad GPRS: class 10 AT set: yes	3.4 – 4.2V Idle mode: <3.5mA, Voice mode: 270mA, GPRS: 500mA max.	-30 to +80	30 x 30 x 2.8	7	RoHS compl.	TCP/IP stack FTP, SMTP Python interpreter (opt) BGA
4	**12, 12i GSM Module**	**Aplicom** www.aplicom.com	Band: Dual (12i = USA) GPRS: class 10 AT set: yes, **EDGE cl. 6**	3.6 – 4.0V Idle: 3mA, Standby: 12-29 GPRS: 310 – 690mA Peak: 2 A	-25 to +55	45 x 36 x 9	15	RoHS compl.	TCP/IP stack HSCSD, HTTP, Java
5	**TC63**	**Siemens** www.siemens.com	Band: Quad GPRS: class 12 (class B) AT set: yes	3.2 – 4.5V, nominal: 3.8 Sleep: 3mA GPRS: 600mA	-30 to +65 (+75 auto-off)	45 x 34 x 3.5	< 10	YES	TCP/IP stack FTP, SMTP Python interpreter (opt)
6	**TC65**	**Siemens**	Band: Quad GPRS: class 12 (class B) AT set: yes	3.2 – 4.5V, nominal: 3.8V Sleep: 3mA GPRS: 600mA	-30 to +65 (+75 auto-off)	45 x 34 x 3.5	< 10	YES	TCP/IP stack HTTP, FTP, POP3 PBCCH, Java I2C, SPI, USB
7	**MC75**	**Siemens**	Band: Quad GPRS: class 12 (class B) AT set: yes, **EDGE cl. 10**	3.2 – 4.5V, nominal: 3.8V current: ? (probably similar to TC6x)	-30 to +65 (+75 auto-off)	45 x 34 x 3.5	< 10	YES	TCP/IP stack HTTP, FTP, POP3 PBCCH, Java
8	**XT55 (900MHz)** **XT56 (850MHz)**	**Siemens**	Band: Tri (w 850 or 900) GPRS: class 10 (class B) AT set: yes	3.3 – 4.8V, nominal: 3.3V Power: 200mW typical, 60mW trickle	-20 to +55	53 x 35 x 5.1	11		I2C, SPI, USB
17	**TC35i**	**Siemens**	Band: Dual GPRS: class 10 (class B) AT set: yes	3.3 – 4.8V, nominal: 3.3V Idle: 3.5mA Voice: 300mA	-20 to +70	54.5 x 36 x 3.6	9	RoHS compl.	TCP/IP stack: no!
18	**MC39i**	**Siemens**	Band: Dual AT set: yes	3.3 – 4.8V Idle: 3mA Voice: 300mA GPRS: 590mA	-20 to +70	54.5 x 36 x 3.6	9	RoHS compl.	TCP/IP stack
19	**MC55/56**	**Siemens**	Band: Tri (w/o 850 or 900) GPRS: class 10 AT set: yes	3.3 – 4.8V Idle: 3.0mA Voice: 260mA GPRS: 450mA	-20 to +70	35 x 32.5 x 2.95	5.5	RoHS compl.	TCP/IP stack HTTP,FTP,POP3, SMTP
20	**AC75**	**Siemens**	Band: Quad **GPRS: class 12** EDGE: class 10 AT set: yes	3.3 – 4.5V Voice: 335mA Iidle: 4.6mA GPRS: 710mA	-30 to +85	33.9 x 55 x 3.15	8.5	RoHS compl.	TCP/IP stack HTTP,FTP,POP3, SMTP Java USB, I2C, GPIO
9	**G20**	**Motorola** www.motorola.com	Band: Quad GPRS: class 8 (class B) AT set: yes SMS: yes	3.0 – 4.2V Idle: <2.5mA TX power: 0.6W/1W/2W (850/1800/900)	-30 to +85	24.4 x 45.2 x 6.0	11.9		TCP/IP stack USB
10	**GM47 (900MHz)** **GM48 (850MHz)**	**Sony Ericsson**	Band: Dual (GSM/EGSM) GPRS: class 8 (class B) AT set: yes SMS: yes	3.4 – 4V Idle: <5mA GPRS: 350mA	-30 to +75	50 x 33 x 7.2	18.5		TCP/IP stack UDP 3x RS232
11	**GR64**	**Sony Ericsson**	Band: Quad GPRS: class 10 (class B) AT set: yes SMS: yes	3.2 – 4.5V Idle: ? GPRS: ? (probably simlar to GM4x)	-30 to +75	50 x 33 x 3.2	9	YES	TCP/IP stack, UDP, PPP FTP 2x UART, SPI, USB
12	**GS64**	**Sony Ericsson** sonyericsson.com	Band: Quad GPRS: class 10 (class B) AT set: yes SMS: yes	3.2 – 4.5V Idle: ? GPRS: ? (probably similar to GM4x)	-30 to +75	37 x 30 x 3.2	5	YES	TCP/IP stack, UDP, PPP FTP, SMTP 2x UART, SPI, USB pads for Antenna
21	**EE54**	**Sony Ericsson**	Band: Quad GPRS: class 10 (class B) AT set: yes SMS: yes	5.5 – 20V Idle: 5mA Voice: 250mA EDGE: 350mA	-20 to +65	69 x 31 x 7.35	11.5	RoHS compl.	TCP/IP stack USB 2.0

num.	Type	Producer	Technical features	Power U/I	Operation temperature	Dimensions [mm x mm]	Weight [g]	Pb free ?	Note
13	Q2406 (900MHz) Q2426 (850MHz)	Wavecom www.wavecom.com	Band: Dual (+DCS/+PCS) GPRS: class 10 AT set: yes SMS: yes	3.6V Idle: 3.5mA Voice: 260mA GPRS: ?	-20 to +55	58 x 32 x 3.9	< 12		TCP/IP stack, UDP FTP, POP3, SMTP 2x RS232, SPI, I2C, IrDA
14	Q2686	Wavecom	Band: Quad GPRS: class 10 (class B) AT set: yes (open AT) SMS: yes	3.6V Idle: 3.5mA Voice: 260mA GPRS: ?, Peak: 1.7A	-40 to +85 (class C)	40 x 32.2 x 4	< 9	RoHS compl.	„Internet Package“ 2x UART, USB GPIO, ADC
22	Q2687	Wavecom	Q2686+EDGE		-35 to +85				
23	GX820	Calamp www.calamp.com	Band: Dual (USA only) GPRS: class 10 AT set: yes SMS: yes						
24	Enabler IIG	Enfora www.enfora.com	Band: Quad GPRS: class 10 AT set: yes	3.3V - 4.5V Idle: <5mA Voice: 175 - 230mA GPRS: 244(1TX) / 350(2TX)	-20 to +60	46.1 x 30.2 x 3.1	?	?	
25	Enabler IIE	Enfora	Enabler IIG+EDGE						
26	SIM100C	Sim Technology www.simcom.com	Band: Tri GPRS: class 10 AT set: yes	3.4 – 4.5V Low power consumption	-20 to +55	72 x 42 x 6.2	18	RoHS compl. (sos.sk)	DIP connector (car terminal)
27	SIM100S	Sim Technology	Band: Tri GPRS: class 10 AT set: yes	3.4 – 4.5V Low power consumption	-20 to +55	53 x 33 x 3.0	11	RoHS compl. (sos.sk)	Compact version of 100C
28	SIM200	Sim Technology	Band: Quad GPRS: class 10 AT set: yes	3.4 – 4.5V Low power consumption	-20 to +55	39.5 x 32.5 x 3	8	RoHS compl. (sos.sk)	TCP/IP stack
29	SIM300C	Sim Technology	Band: Tri (Quad SIM340C) GPRS: class 10 AT set: yes	3.4 – 4.5V Low power consumption	-20 to +55	50 x 33 x 6.2	12	RoHS compl. (sos.sk)	DIP connector
30	SIM300D	Sim Technology	Band: Tri (Quad SIM340D) GPRS: class 10 AT set: yes	3.4 – 4.5V Low power consumption	-20 to +55	33 x 33 x 3	7	RoHS compl. (sos.sk)	TCP/IP stack
31	SIM300	Sim Technology	Band: Tri (Quad SIM340) GPRS: class 10 AT set: yes	3.4 – 4.5V Low power consumption	-20 to +55	40 x 33 x 2.85	8	RoHS compl. (sos.sk)	TCP/IP stack
32	SIM508	Sim Technology	Band: Tri or Quad GPRS: class 10 AT set: yes	3.2 – 4.5V ?	-20 to +55	55 x 34 x 3	10	RoHS compl. (sos.sk)	GPS (10 channel)
33	SIM600	Sim Technology	Band: Quad GPRS: class 10 EDGE AT set: yes	3.4 – 4.5V Low power consumption	-25 to +70	54 x 33 x 3.0	8	RoHS compl. (sos.sk)	SIM toolkit 2*UART, USB IrDA, LCD, BT, audio

Výběhové modely (2005)

Výhodný parametr (nadprůměrný)

Nevýhodný parametr (podprůměrný)

Vhodně se jevící produkt

9.3 Schémata testovacích desek

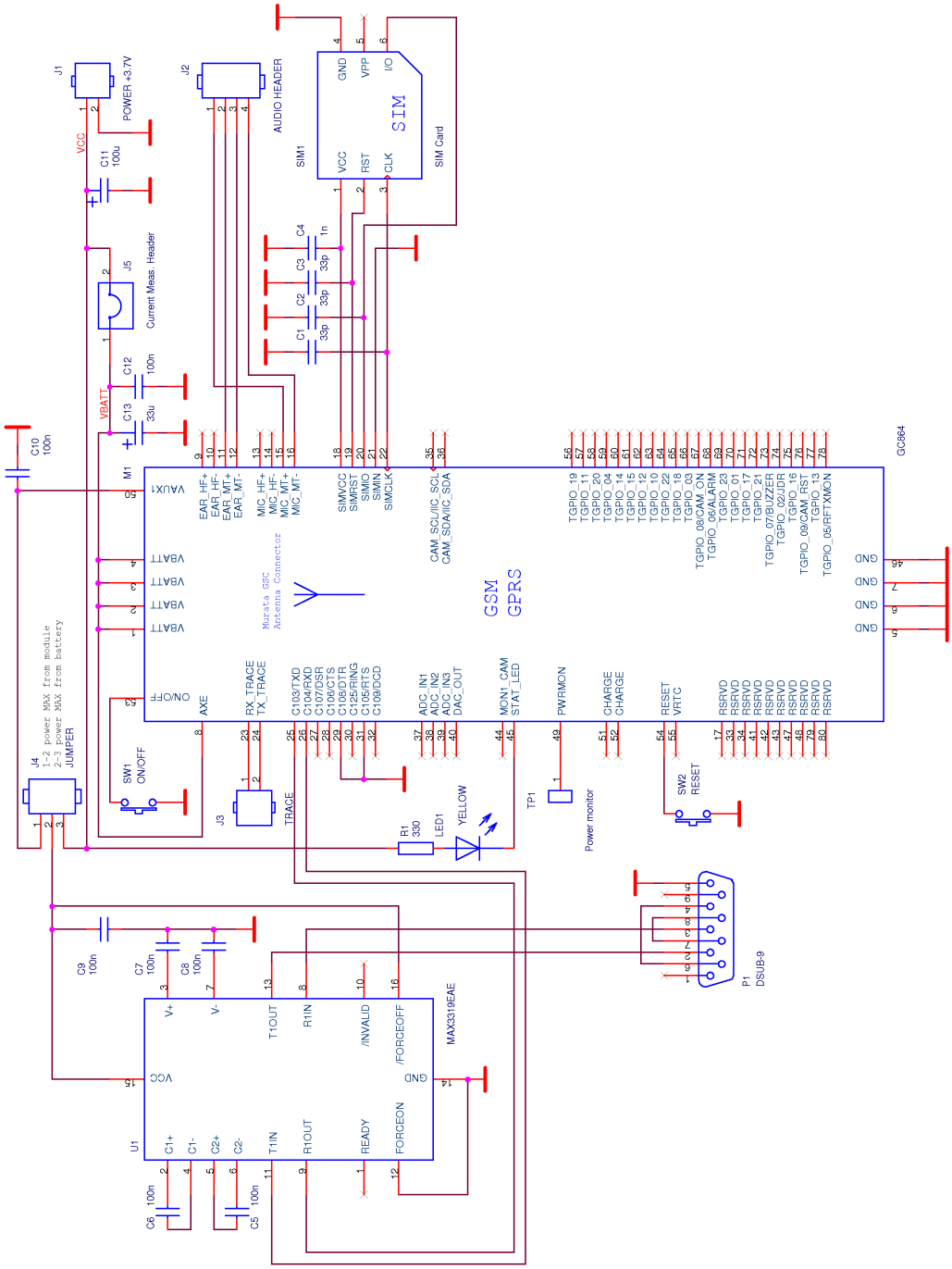


Schéma testovací desky pro modul Telit GC864

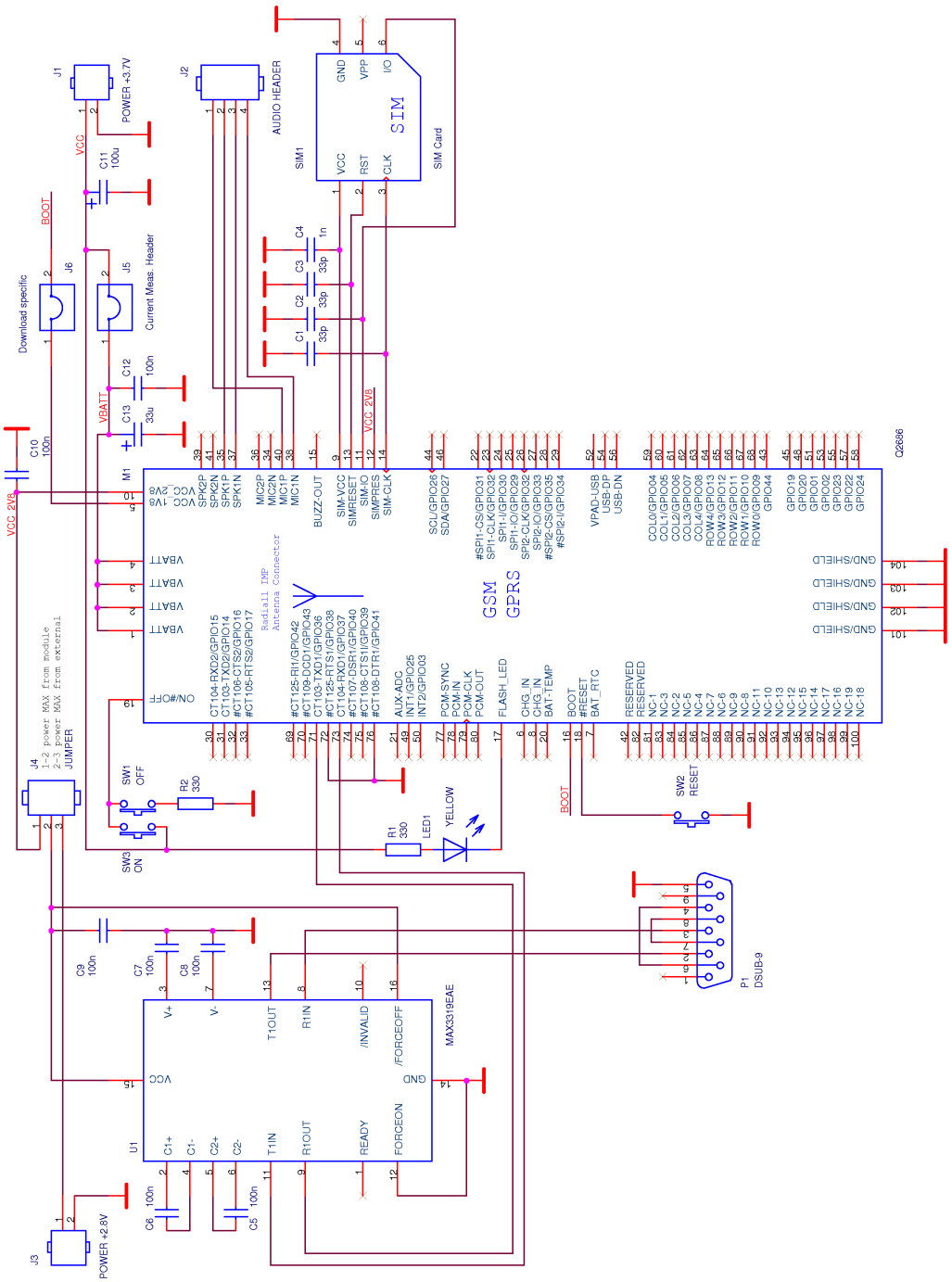
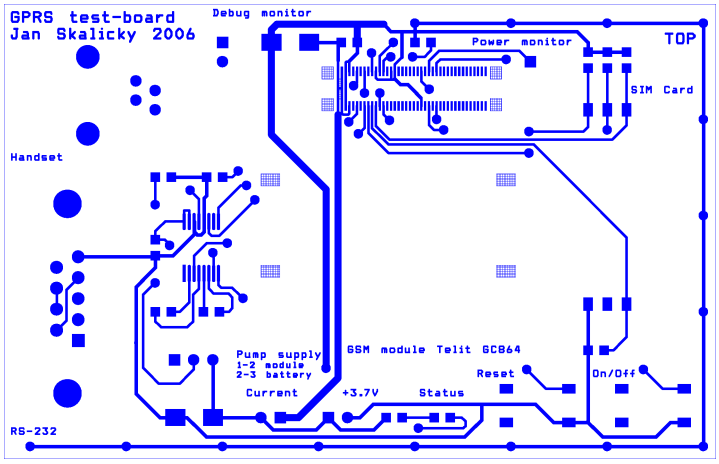
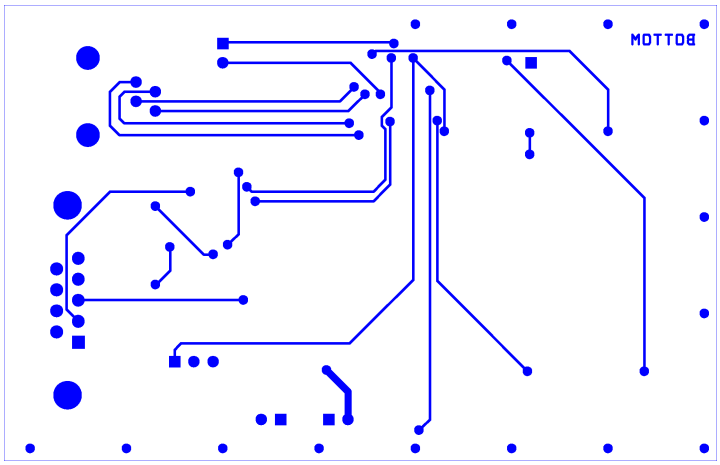


Schéma testovací desky pro modul Wavecom Q2686

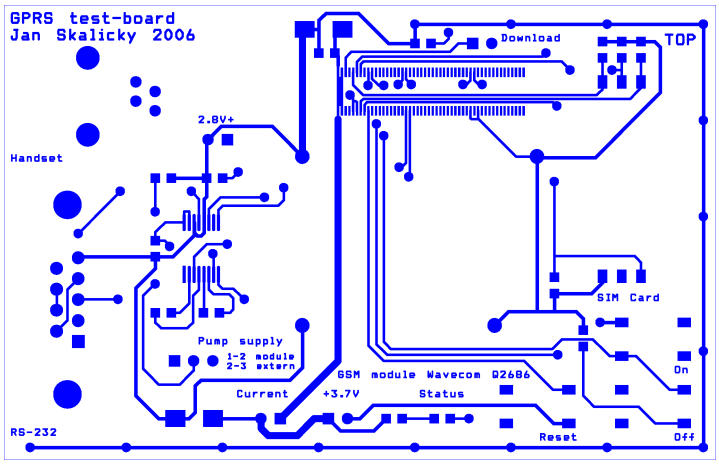
9.4 Výkresy spojů a osazováky testovacích desek



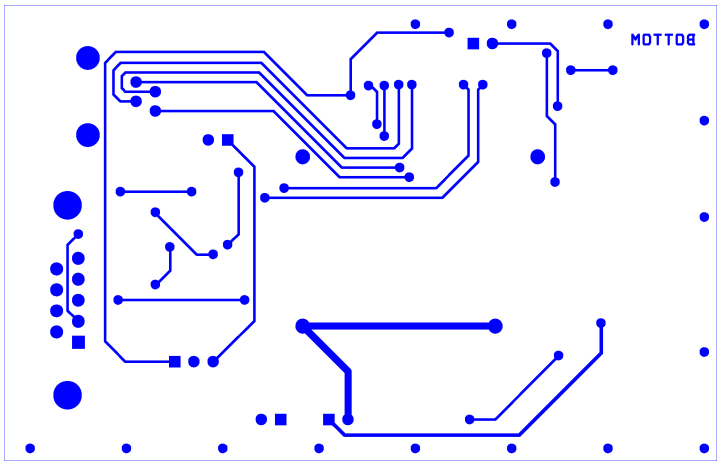
Vrchní strana spojů testovací desky pro modul Telit GC864



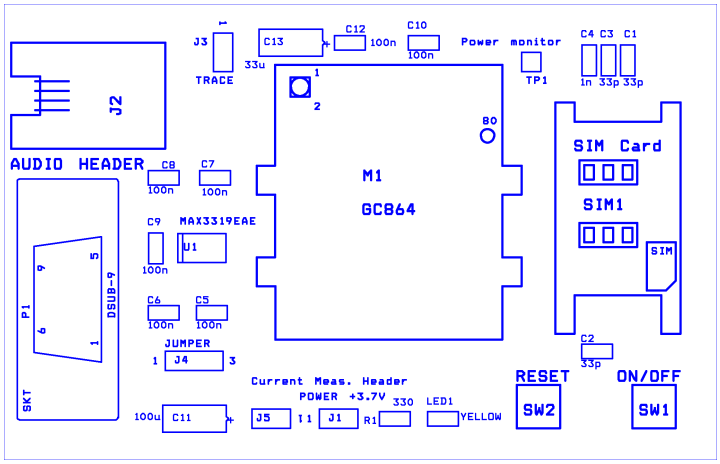
Spodní strana spojů testovací desky pro modul Telit GC864



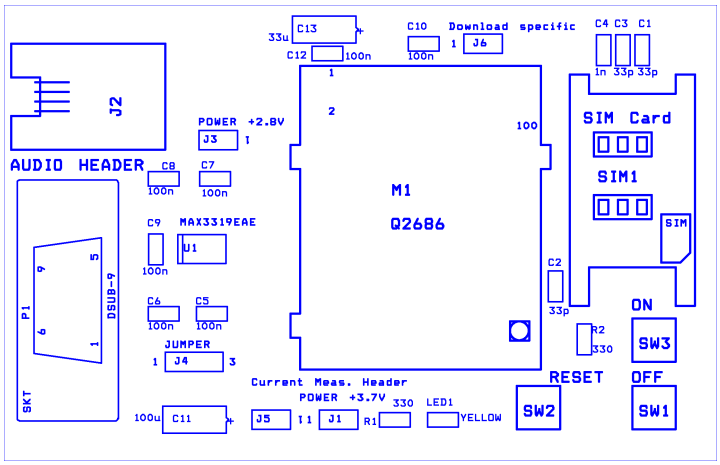
Vrchní strana spojů testovací desky pro modul Wavecom Q2686



Spodní strana spojů testovací desky pro modul Wavecom Q2686

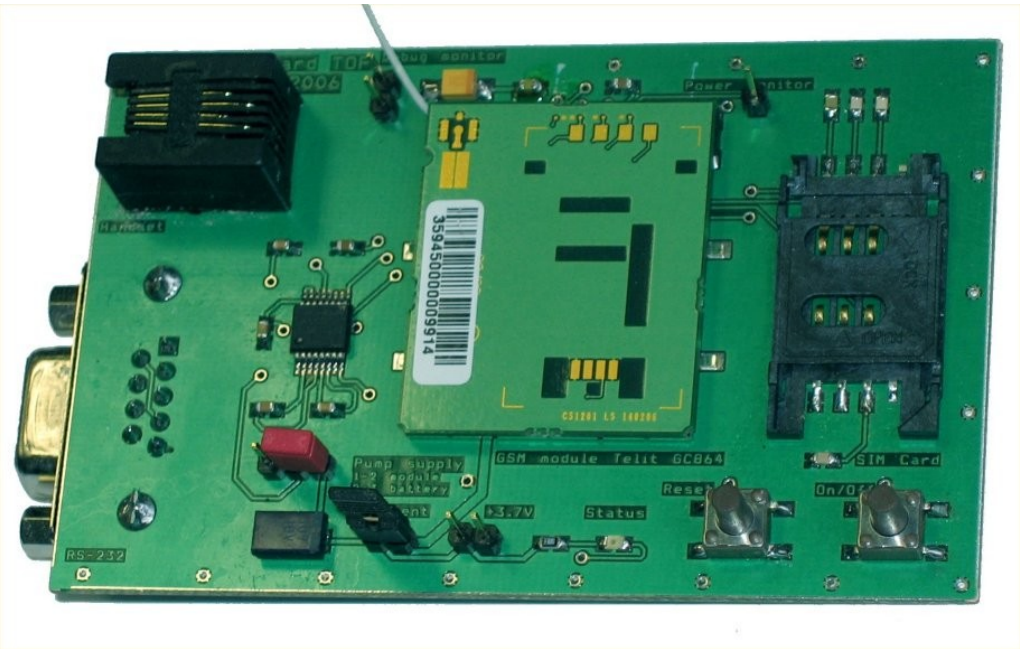


Osazovací plán testovací desky pro modul Telit GC864

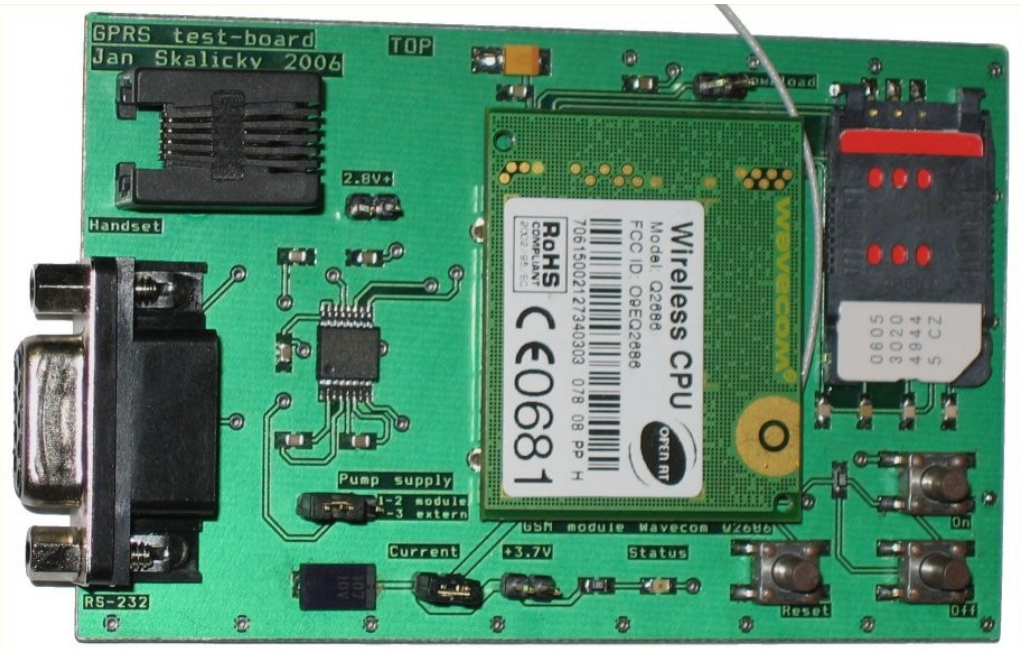


Osazovací plán testovací desky pro modul Wavecom Q2686

9.5 Fotografie osazených testovacích desek



Osazená testovací deska pro modul Telit GC864



Osazená testovací deska pro modul Wavecom Q2686

9.6 Originální specifikace GPS Collar