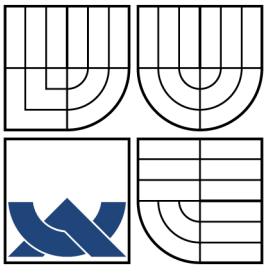
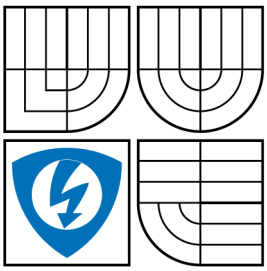


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

IMPLEMENTACE IP TELEFONU NA VÝVOJOVÉM KITU ATSTK1000

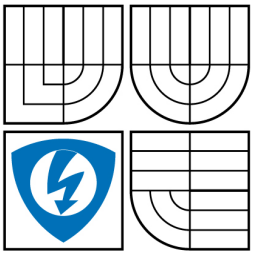
IP PHONE IMPLEMENTATION ON DEVELOPMENT KIT ATSTK1000

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE — KAREL BÖHM
AUTHOR

VEDOUCÍ PRÁCE — Ing. PETR SYSEL, Ph.D.
SUPERVISOR

BRNO 2009



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Bakalářská práce

bakalářský studijní obor
Teleinformatika

Student:	Karel Böhm	***ID:***	70262
Ročník:	3	***Akademický rok:***	2008/2009

NÁZEV TÉMATU:

Implementace IP telefonu na vývojovém kitu ATSTK1000

POKYNY PRO VYPRACOVÁNÍ:

Seznamte se s vývojovým kitem ATSTK1000 a s instalovaným operačním systémem Linux. Seznamte se s dodávaným vývojovým prostředím a s postupy při křížovém překladu aplikací. Seznamte se s volně šířitelnými softwarovými IP telefony, s jejich vlastnosti i požadavky. Zdrojové kódy jednoho zvoleného softwarového IP telefon upravte, přeložte jej a potřebné knihovny a nainstalujte na vývojový kit ATSTK1000. S použitím zvukového modulu otestujte jeho funkci.

DOPORUČENÁ LITERATURA:

[1] AVR32 32-bit Microcontroler [online]. Atmel. 2007. [cite 11.10.2007]. Dostupné na URL http://www.atmel.com/dyn/resources/prod_documents/doc32003.pdf
[2] AT32STK1000 Schematic [online]. Atmel, 2006. [cite 11.10.2007]. Dostupné na URL http://www.atmel.com/dyn/products/tools_card.asp?tool_id=3918
[3] Matthew, N. a kol. Linux: Programujeme profesionálně. První vydání. Brno, Computer Press: 2001. ISBN 80-7226-532-6

Termín zadání:	9.2.2009	***Termín odevzdání:***	2.6.2009

Vedoucí práce: Ing. Petr Sysel, Ph.D.

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

UPOZORNĚNÍ:

Anotace:

Tato práce se zabývá implementací IP telefonu na vývojovém kitu Atmel ATSTK1000. Úvodem je seznámení se s vlastnostmi, technickým a programovým vybavením vývojového kitu sloužícího k vývoji embedded zařízení. Další část seznamuje s volně poskytovanými softwarovými IP telefony, s jejich vlastnostmi i požadavky. Závěr je věnován cross-kompilaci vybraného IP telefonu a jeho zprovoznění na vývojovém kitu ATSTK1000.

Klíčová slova:

avr32-linux-gcc, Buildroot 2.3.0, Cross-kompilace, GCC, GNU, OpenSSH, Speak Freely, Toolchain 2.1.6, VoIP

Abstract:

This thesis deals with the implementation of IP phone on development kit Atmel ATSTK1000. At the beginning, there is an introduction with properties, hardware and software equipment of the development kit which is used to development of embedded devices. The next part introduces with open sources IP phones, with their characteristics and requirements. The conclusion is devoted to cross-compilation of the selected IP phone and its operation on the development kit ATSTK1000.

Keywords:

avr32-linux-gcc, Buildroot 2.3.0, Cross-compilation, GCC, GNU, OpenSSH, Speak Freely, Toolchain 2.1.6, VoIP

Citace práce

BÖHM, K. *Implementace IP telefonu na vývojovém kitu ATSTK1000*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2009. 42 s. Vedoucí bakalářské práce Ing. Petr Sysel, Ph.D.

Prohlášení

Prohlašuji, že svou bakalářskou práci na téma „Implementace IP telefonu na vývojový kit ATSTK1000“ jsem vypracoval samostatně pod vedením vedoucího bakalářské práce s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a jsou uvedeny v seznamu literatury na konci práce.

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

V Brně dne 1. června 2009 .....

podpis autora

Poděkování

Děkuji své manželce Mgr. Lucii Böhmové za podporu a pomoc při vypracovávání práce, dále děkuji vedoucímu bakalářské práce Ing. Petru Syslovi, Ph.D. a Ing. Michalovi Hejtmánkovi za velmi užitečnou metodickou pomoc a cenné rady při zpracování práce.

V Brně dne 1. června 2009 .....

podpis autora

OBSAH

SEZNAM OBRÁZKŮ

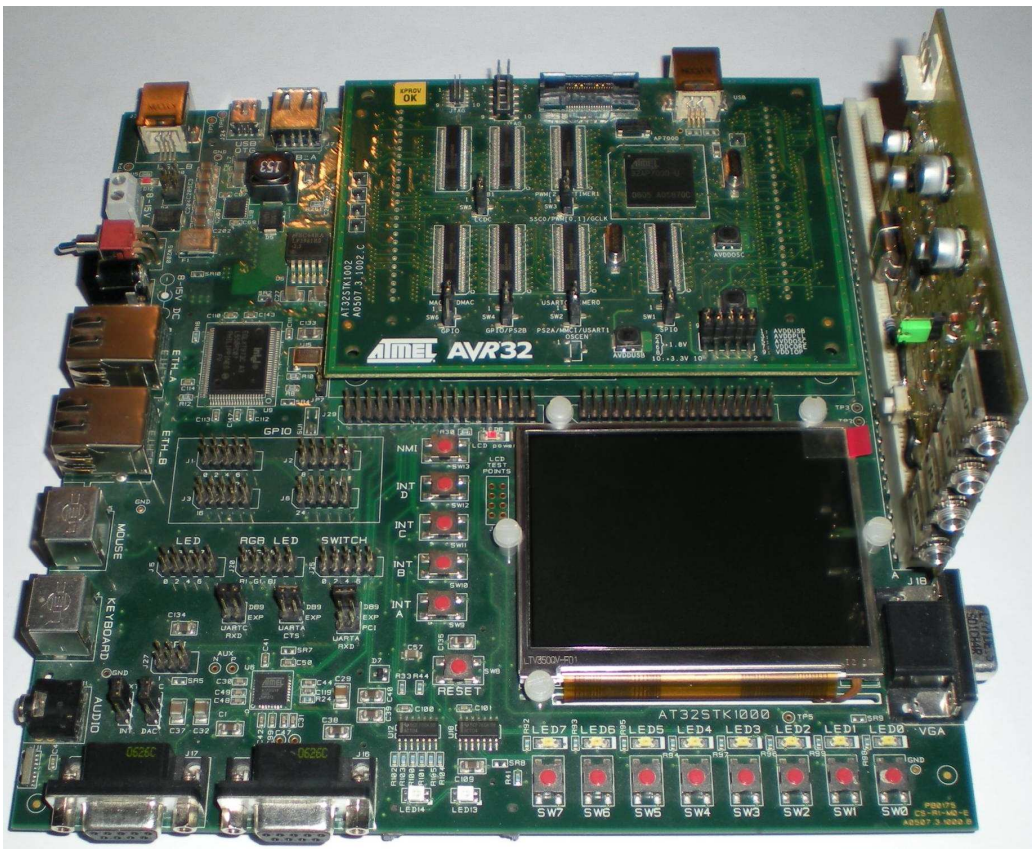
ÚVOD

První část této bakalářské práce se zabývá seznámením s vývojovým kitem ATSTK 1000 od firmy Atmel, stručně popíši jeho hardwarové vybavení a zaměřím se na zprovoznění připojení k síťi ethernet pomocí portu RJ45 a zprovoznění zvukového přídavného modulu. V další části stručně popíši protokoly a open source software využívané pro VoIP komunikaci. Cílem této bakalářské práce je seznámení s vývojovým kitem, s jeho možnostmi, s konfigurací a se softwarovými telefony – z nich vybrat takový, který by bylo možno pomocí cross-kompilace na platformě PC upravit, přenést ho pomocí SSH na na kit ATSTK1000 a následně jej zprovoznit.

1 VÝVOJOVÝ KIT STK1000

Tento vývojový kit byl zkonstruován a určen k návrhu systémů, využívajících jeho snadné rozšiřitelnosti pomocí konektorů tak, aby konečné vyvíjené zařízení mohlo obsluhovat periferie určené dle požadavků zadání vývojářů.

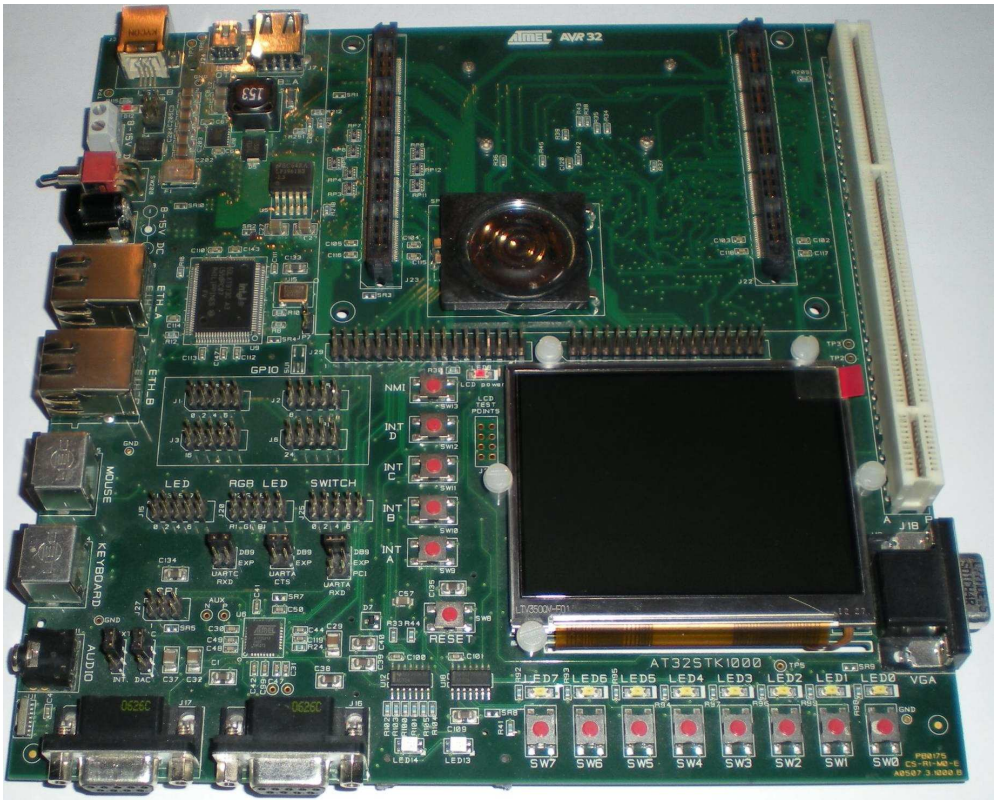
Mimo součástí a periferií, umožňujících široké využití vývojového kitu, základní deska STK1000 navíc obsahuje dceřinou desku STK1002 s paticí BGA256 určenou pro mikrořadič AP7000. Tyto desky jsou k sobě navzájem spojeny dvěma konektory. Jelikož základní deska obsahuje pouze jeden zvukový výstup, je k vývojovému kitu přidána další deska (tzv. zvukový modul), která obsahuje nejen zvukový výstup, ale také zvukový vstup. Aby bylo možno do vývojového kitu implementovat služby přenosu VoIP, je zvukový modul nepostradatelný, protože od služby VoIP se vyžaduje obousměrná komunikace. V dalších částech této bakalářské práce jednotlivé desky popíši. Dokumentace k vývojovému kitu STK1000 je volně dostupná na stránce výrobce [1]. Podrobné informace jsou popsány v uživatelské příručce [2].



Obr. 1.1: Vývojový kit STK1000 včetně zvukového modulu

1.1 Základní deska STK1000

Samostatná základní deska tvoří hlavní část vývojového kitu STK1000, pomocí konektorů a portů na ni připojujeme další potřebná zařízení. Mimo jiné tato deska vlastní rozšiřující konektor podobný slotu PCI (v pravé části obr. 1.2), není však s tímto slotem kompatibilní. Účelem tohoto konektoru je snadná rozšiřitelnost systému libovolnou deskou (kartou) schopnou pracovat se signály zpřístupněnými zmíněným konektorem. Schematické zapojení základní desky STK1000 lze nalézt v příručce [3].



Obr. 1.2: Samostatná základní deska STK1000

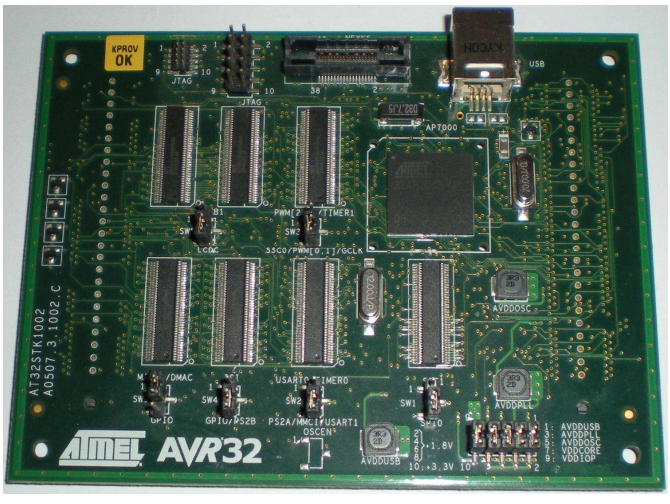
Seznam součástí a periferií osazených na základní desce STK1000:

- paměti 8MB SDRAM a 8MB Flash,
- slot pro paměťovou kartu SD/MMC,
- slot pro paměťovou kartu Compact Flash (CF),
- LCD o rozlišeni QVGA (320x240),
- konektor výstupu VGA pro externí monitor,

- dva konektory PS/2 pro připojení klávesnice a myši,
- dvojitý vysílač/přijímač fyzické vrstvy sítě Fast Ethernet 10/100 Mbps a dvě zásuvky RJ45,
- konektory rozhraní USB a UART (standard RS-232),
- vysilač/přijímač infračerveného přenosu dle specifikace IrDA,
- tlačítka ke generovaní signálů RESET a přerušeni procesoru,
- tlačítka a diody LED k univerzálnímu uživatelskému použiti,
- dvouřadé konektory na plošném spoji (headers) umožňující přistup k určitým signálům,
- propojky s možnostmi nastavení některých periferii,
- vnější stereofonní číslicově-analogovy převodník (DAC) se zesilovačem. Pomocí propojky lze volit výstup do zásuvky stereo Jack 3,5mm mezi vnějším a vnitřním DAC,
- stabilizátory s filtracemi potřebných napájecích napětí pro celý systém.

1.2 Základní deska STK1002

Deska STK1002 je osazená pouzdrem BGA256 pro mikrořadič AP7000. AP7000 je aplikační procesor s architekturou jednočipového systému (System-on-chip), kde jsou vedle procesoru AVR32AP obsažený další podpůrné obvody, řadiče a periferie, umožňující vyvíjet zařízení s co nejmenším množstvím hardwaru. Schematické zapojení základní desky STK1002 se nachází v příručce [4].



Obr. 1.3: Základní deska STK1002

Seznam součástí a periferií osazených na dceřiné základní desce STK1002:

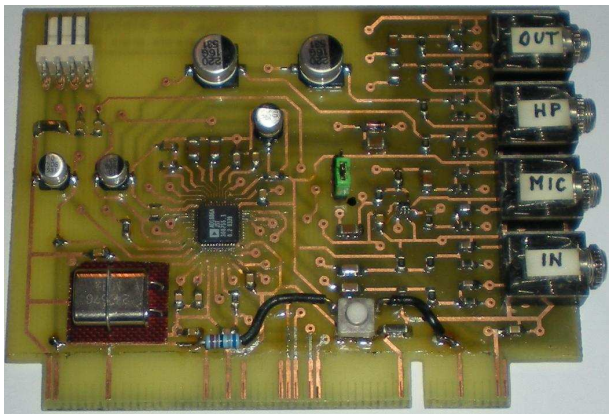
- krystalové rezonátory ke generování taktovacích kmitočtů pro mikrořadič, jeho sběrnice a rozhraní,
- propojky k připojeni rozhraní na periferie a k měření spotřeby proudu různých napájených částí mikrořadiče,
- tvarovací obvody signálu RESET a další pomocné obvody mikrořadiče,
- ladící rozhraní JTAG,
- konektor Nexus 1 rozšiřující možnosti JTAG,
- konektor rozhraní USB.

Seznam obvodů, o které AP7000 ve vnitřní struktuře doplňuje AVR32 AP:

- vysokorychlostní vnitřní sběrnice a mosty propojující jednotlivé vnitřní bloky,
- vnější sběrnice rozhraní paměti SDRAM a Flash,
- rozhraní pro paměťovou kartu SD/MMC,
- generátory taktovacích kmitočtů,
- časovače a čítače,
- řadiče vnějších přerušeni,
- řadiče přímého přístupu do paměti (DMA),
- řadič zobrazovače LCD,
- koprocesor obrazových bodů pro rychlé vektorové výpočty,
- rozhraní snímače obrazu,
- rozhraní PS/2,
- dvoudrátové rozhraní (TWI) podporující standard Philips I2C,
- sériová rozhraní USB, USART a SPI,
- řadič synchronní sériové komunikace (SSC) konfigurovatelný pro různé typy rozhraní, například standard Philips I2S,
- řízeni přístupu k mediu (MAC) sítě Ethernet,
- řadič AC‘97 specifikace Intel pro audio kodeky,
- stereofonní číslicově-analogovy převodník,
- ladící rozhraní JTAG,
- řadič pulsní šířkové modulace (PWM).

1.3 Přídavný zvukový modul

Základ zvukového modulu tvoří integrovaný obvod AD 1886A výrobce Analog Devices, INC. (ADI). Tento obvod obsahuje celou funkcionalitu zvukového kodeku tak, jak předepisuje specifikace AC'97. K němu je připojeno několik vnějších součástek, vstupy/výstupy a celý zvukový modul je připojitelný přes rozhraní AC-link s vývojovým kitem STK1000. Podrobnější informace o tomto modulu jsou popsány v bakalářské práci pana Bc. J. Priškina, jejímž cílem bylo zvukový modul sestrojit a zprovoznit [5].



Obr. 1.4: Zvukový modul

Kodek AD 1886A disponuje těmito vlastnostmi:

- 16-bitový stereofonní plně duplexní kodek,
- architektura ADC a DAC s vícebitovými převodníky,
- dynamický rozsah větší než 90 dB,
- měnitelný vzorkovací kmitočet (režim VRA) od 7040 Hz do 48 kHz, s přesnosti kroku 1 Hz,
- čtyři stereofonní a dva monofonní vstupy,
- monofonní vstup z mikrofonu s možnosti zařadit vnitřní předzesilovač se ziskem 20 dB,
- dva stereofonní a jeden monofonní vystup,
- 20-bitový vystup S/PDIF o symbolové rychlosti 32 kHz, 44,1 kHz, nebo 48 kHz,
- oddělené napájení pro číslicové obvody 3,3V a analogové 5V,
- oddělený číslicový a analogovy společný vodič (zem).

1.4 Zprovoznění vývojového kitu STK1000

Tato kapitola popisuje připojení na vývojový kit STK1000. Nikde jsem nenalezl manuál, který by to popisoval. Vývojový kit ATSTK 1000 lze zprovoznit díky jeho širokému hardwarovému vybavení několika způsoby. Můžeme ho zprovoznit jako samostatně plně funkční zařízení pouze pomocí klávesnice (připojené do konektoru PS/2) a monitoru (připojeného do konektoru VGA). Nebo ho můžeme konfigurovat pomocí počítače tak, že jej k PC přes terminál připojíme sériovým rozhraním SR-232, USB nebo přístupem ze sítě TCP/IP. Informace jsem čerpal v průvodci [6]. Pokud bych tedy chtěl připojit klávesnici a myš přímo ke kitu, musel bych nejprve nastavit správný obnovovací kmitočet. Ten je nastaven pro malý displej na kitu a proto monitor, díky špatné synchronizaci nic nezobrazuje. Dále je třeba dbát na správné napájení kitu. Kit je stabilní při použití napájecího napětí v rozmezí 8 – 15 V DC. Pokud není dodrženo předepsané rozmezí napájecího napětí a použijeme nižší hodnoty, kit je velmi nestabilní a dochází u něj k samovolnému restartování.

Ke zprovoznění a konfiguraci jsem využil možnosti připojení přes sériové rozhraní RS-232 tak, že jsem vývojový kit připojil k počítači s nainstalovaným Microsoft Windows Vista Business. Ke komunikaci jsem použil jednoduchý a spolehlivý terminálový emulátor Tera Term Pro určený pro MS-Windows. V operačním systému Linux Kubuntu 8.10 tento terminálový program také funguje, je však o poznání pomalejší. Pokud na vývojovém kitu necháme spustit operační systém Linux BSP z přiložené paměťové karty SD, zobrazí se nám na displeji modře podsvícené logo AVR 32, jak je patrné z obr. 1.5. Poté můžeme přejít k softwarovému připojení pomocí počítače. Po spuštění programu Tera Term Pro jsem v úvodní nabídce programu zvolil sériové připojení a pak v menu Setup/Serial port je nutné nastavit přenosovou rychlost na hodnotu 115 200 b/s. Pokud se zvolí jiné hodnoty, nebude připojení správně pracovat. Všechny ostatní hodnoty lze ponechat v základním nastavení. V dalším kroku je nutné v terminálu zadat přihlášení a heslo, které zůstaly beze změny od výrobce a jsou:

přihlášení (login) – root

heslo (password) – roota

Po zadání hesla jsme přes terminál připojeni k vývojovému kitu a můžeme ho konfigurovat. Pro zprovoznění zvukového modulu je nutné stisknout tlačítko *reset*, které je umístěno přímo na zvukovém modulu. Do příkazové řádky programu Tera Term Pro musíme zadat příkaz *modprobe snd-atmel-ac97*, který nám zvukový modul zavede. K obnovení předchozích uložených nastavení pro inicializaci a registraci zvukového modulu v podsystému ALSA musíme zadat příkaz *alsactl restore*. Další možností je spustit skript (*alsa.sh*), který jsem vytvořil a uložil do kořenového adresáře paměťové karty. Pokud tedy zadáme *./alsa.sh* budou oba zavaděče spuštěny automaticky. Nyní bychom měli mít vývojový kit ATSTK 1000 zprovozněn včetně přídavného zvukového modulu.

Obr. 1.5: Logo AVR32 – systém je nastartovaný

1.5 Odzkoušení zvukového modulu na kitu STK1000

Dalším krokem ke zprovoznění vybraných programů na vývojovém kitu bylo připojení pomocí terminálového programu TerraTerm Pro v OS Linux k vývojovému kitu a zprovoznění zvukového modulu dle návodu v podkapitole 1.4. Otestoval jsem mikrofonní vstup *[MIC_IN]* a zvukový výstup *[HP_OUT]*, na dané výstupy jsem připojil sluchátka s mikrofonem.

Poté jsem deaktivoval funkci Mute a nastavil hlasitost do sluchátek:
amixer set Headphone 85% unmute.

Pro nahrávání z mikrofonu musí být přepnut záznam zvuku na mikrofonní vstup: *amixer set Mic cap.*

Abych slyšel zvukový signál z mikrofonu v reálném čase přímo přes směšovací obvody kodeku ve zvolené hlasitosti v procentech, zadal jsem:
amixer set Mic 80% unmute cap.

Úroveň signálu vstupující do analogově číslicových převodníků jsem nastavil příkazem: *amixer set Capture 50%.*

Vyzkoušel jsem zaznamenat zvuk z mikrofonu pomocí:
arecort test2.wav.

Ztlumil jsem odposlech mikrofonního vstupu, abych slyšel přehrávaný soubor:
amixer set Mic 80% mute.

Deaktivoval jsem funkci Mute:
amixer set PCM 70% unmute.

Soubor jsem si poslechl pomocí:
aplay test2.wav.

Úspěšným přehráním zvukového souboru *test2.wav* jsem ověřil funkčnost zvukového modulu, na kterém je tedy pravděpodobně možné zprovoznit IP telefon.

1.6 Zprovoznění ethernetu na kitu STK1000

Je několik způsobů, jak nahrávat data na vývojový kit STK1000, jež budeme potřebovat. Lze použít čtečku paměťových karet a data nahrát přímo na SD kartu. SD karta je zformátovaná na formát ext2, v Linuxu Kubuntu 8.10 jsem dokázal z paměťové karty pouze číst. Data se mi nepodařilo na SD kartu zapsat. Další možností, jak přenést

data, je využit port USB. Tuto možnost jsem nevyzkoušel a použil jsem poslední možnost – přenášet data pomocí ethernetu (port RJ45).

Ke zprovoznění ethernetu na vývojovém kitu STK1000 jsem postupoval dle návodu v dokumentaci [7]. Návod je popsán chybně, cesta ke konfiguračnímu souboru není */etc/init.d/network* nýbrž je */etc/network/interfaces*. Obsah tohoto souboru zobrazíme příkazem *cat interfaces*, zobrazí se nám tedy:

 # Configure Loopback
auto lo
iface lo inet loopback
 # Configure Ethernet 0
 # eth0
iface eth0 inet dhcp
 # Configure Ethernet 1, not enabled by default
#auto eth1
iface eth1 inet dhcp

Pro zprovoznění sítě je potřeba změnit pátý řádek, kde je ethernet zakomentován *(#eth0)*, tak aby konfigurace a načtení sítě neprobíhalo po každém spuštění. Dle literatury [8] jsem zjistil, že stačí daný řádek odkomentovat a zadat *auto eth0*, který nám zprovozní port ETH_A. Nyní po zadání příkazu *ifconfig* se zobrazí potřebné informace, signalizující zprovoznění ethernetu:

 # ifconfig
eth0 Link encap:Ethernet HWaddr 00:04:25:19:03:1A
inet addr:192.168.1.105 Bcast:192.168.1.255 Mask:255.255.255.0
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:30 errors:1 dropped:0 overruns:0 frame:0
TX packets:33 errors:1 dropped:0 overruns:0 carrier:1
collisions:0 txqueuelen:1000
RX bytes:3954 (3.8 KiB) TX bytes:3628 (3.5 KiB)
Interrupt:25 Base address:0x1800

2 PROTOKOLY VYUŽÍVANÉ PRO VoIP KOMUNIKACI

V této části bakalářské práce popíši vybrané protokoly pro VoIP komunikaci (Voice over Internet Protocol - je technologie, která umožňuje přenos digitalizovaného hlasu v těle paketů rodiny protokolů UDP/TCP/IP prostřednictvím počítačové sítě nebo jiného média, prostupného pro protokol IP. Využívá se pro telefonování prostřednictvím Internetu, intranetu nebo jakéhokoliv jiného datového spojení.). Popíši jen ty protokoly, které využívají softwarové VoIP telefony (popsané ve 3. kapitole), které chci vyzkoušet při komunikaci na vývojovém kitu STK1000.

V internetové telefonii (VoIP) se hlas přenáší v datových paketech. Nejprve je hlas převeden do digitální podoby, tuto činnost provádí tzv. kodek. Kodek zvuk rozdělí na krátké úseky, které přetransformuje do digitální podoby. Tyto data (pakety) jsou po síti přenášena na místo určení a následně pomocí kodeku stejného typu přetransformována zpět na zvuk. K přenosu těchto datových paketů je potřeba určit způsob, jakým bude zajištěno přenášení, detekce nebo opravy chyb apod. Tento způsob je definován v rámci protokolu. Protokolů, zabývající se přenosem dat(včetně multimediálních) existuje celá řada např. H.323, SIP, RTP, RTCP, SRTP, ZRTP, STUN, IAX, HFA, SCCP, MGCP, MiNET a další.

Tyto protokoly dále dělíme na dvě velké skupiny, na protokoly otevřené a protokoly uzavřené. Otevřené protokoly jsou takové, ke kterým jsou dostupné zdrojové kódy. Naopak u uzavřených protokolů zdrojové kódy k dispozici nejsou (např. Skype). V této práci se budu věnovat pouze protokolům otevřeným. Vybral jsem následující protokoly: H.323, SIP a RTP.

2.1 Protokol H.323

H.323 je doporučení ITU Telecommunication Standardization Sector (ITU – T), které definuje protokoly pro přenos zvuku a videa v jakékoliv paketové síti. Tento protokol patří do skupiny nejstarších protokolů, vznikal společně s linkami ISDN. Bývá implementován v několika internetových real-timeových aplikacích, např. NetMeeting

nebo Ekiga, která využívá implementaci OpenH323. Tento protokol lze použít v jednoduchých segmentech sítí LAN nebo i ve značně složitějších soustavách vzájemně propojených sítí. Protokol podporuje i skupinový způsob přenosu, není vázán na žádnou systémovou platformu. V dnešní době tento protokol už nenachází velké uplatnění, protože není příliš ideální. Jeho hlavní problematika nastává při použití překladu síťových adres (tzv. NAT), které se používá při řešení problémů díky nedostatečnému počtu IP adres.

2.2 Protokol SIP

SIP (Session Initiation Protocol) protokol pro inicializaci relací je určený pro přenos dat ve VoIP telefonii. Tento protokol byl standardizován IETF. První verzi tohoto protokolu popisoval dokument RFC 2543, současnou druhou verzi popisuje RFC 3261. Podobně jako u H.323 při spojení dvou uživatelů není potřeba server (je decentralizovaný). Tento protokol pro zajištění VoIP spojení pracuje v součinnosti s dalšími protokoly. Principielně k přenosu dat používá jiné protokoly, sám data nepřenáší. Přenos hovoru přenáší pomocí protokolu RTP. Naopak detaily o vlastnostech zahajovaného přenosu popisuje protokol SDP, který je přenášen v těle SIP paketu. Protokol SIP byl vyvinut proto, aby protokol H.323 zjednodušil. A proto vychází z osvědčeného protokolu HTTP a je mu velmi podobný. Podobně pracuje s položkami, které jsou velmi podobné položkám SMTP, využívané k posílání e-mailů. Nevýhodou tohoto protokolu je, že jej nelze použít pro uživatele s neveřejnou IP adresou. Tento problém však lze řešit pomocí serveru (VoIP proxy).

2.3 Protokol RTP

RTP (Real-time Transport Protocol) definuje standardní paketový formát pro přenos zvuku a videa. Byl vyvinut korporací audio-video Transport Working Group IETF. Původně byl vyvíjen jako protokol pro výběrové vysílání. Protokol našel využití při kontinuálním přenosu audiovizuálního materiálu (streaming media) jako video telefonní konference a v systémech s poloduplexním přenosem (push to talk). Díky tomu se protokol RTP stal nejčastěji používaným protokolem pro přenos dat Voice over IP technologii. Pakety protokolu RTP jsou přenášeny pomocí UDP protokolu.

3 SOFTWAROVÉ VoIP TELEFONY PRO LINUX

3.1 Přehled dostupných softwarových VoIP telefonů

Ke zprovoznění VoIP komunikace na vývojovém kitu STK1000 jsem byl postaven před problém jaký program použít pro přenos hlasu. V dnešní době je na internetu dostupné velké množství aplikací s otevřeným zdrojovým kódem (open-source) určených k tomuto účelu. Jsou to např. Ekiga, GnomeMeeting, KPhoneSI, Twinkle, OpenH323, GPhone, Speak Freely, Voicechat, Nautilus, GnoPhone. Na svém domácím 64 bitovém PC jsem si nainstaloval výše zmíněný 32 bitový operační systém Linux Kubuntu, verzi 8.10. Myšlenkou bylo vybraný komunikační software nejprve zkompilovat a otestovat na stolním PC (popsáno ve 4. kapitole) a teprve po získání zkušeností se pokusit o přenos programu na vývojový kit STK1000. Nyní stručně popíši několik VoIP telefonů, které ke své komunikaci využívají výše zmíněné protokoly.

3.1.1 Ekiga

První vybraný softwarový VoIP telefon Ekiga jsem vybral, protože jej pod operačním systémem Linux používám a znám jeho vlastnosti. Tento program je nástupcem programu GnomeMeeting. Podporuje protokoly SIP a H.323. Ekiga umožňuje video hovory pomocí protokolu H.261 nebo H.245 k textovému chatu. Ekiga podporuje LDAP a bezproblémově pracuje jak v prostředí GNOME tak i v KDE. Na tomto programu je vidět dlouhá historie jeho vývoje. Dokumentace k projektu je na velmi vysoké úrovni.

3.1.2 GPhone

GPhone využívá protokol RTP, velikostí je nejmenší ze všech tří vybraných programů. GPhone umí navázat spojení, přepínání režimu naslouchání/mluvení, obsahuje funkci mute pro umlčení mikrofonu. Pomocí parametrů zadaných z příkazové řádky lze zvolit port, na kterém naslouchá, a také port, na kterém vysílá. Tento program je ze všech tří kandidátů hardwarově nejméně náročný. GPhone funguje pouze v připojení mezi Linux – Linux (není multiplatformní).

3.1.3 Speak Freely

Speak Freely ke komunikaci využívá RTP protokol, podporuje však také vlastní speakfreely protokol. Zahrnuje podporu šifrování (PGP, DES, IDEA), zároveň používá kompresi (GSM, LPC, ADPCM). Obsahuje všechny standardní i nadstandardní komunikační funkce. Speak Freely by měl fungovat v příkazovém řádku a neměl by být tudíž vázán na grafické prostředí. Dále umožňuje mezi platformní komunikaci. Díky své nízké náročnosti se pro mne stal společně s výše popsaným GPhone nejsilnějším adeptem pro zprovoznění na vývojovém kitu.

3.2 Instalace Speak Freely na platformě PC

Pro použití VoIP komunikace na vývojovém kitu STK1000 se jeví jako nejvhodnější použití programu Speak Freely, protože tento software pracuje bez nutné návaznosti na grafické knihovny. Speak Freely jsem se rozhodl zprovoznit nejdříve mezi PC s operačním systémem Linux Kubuntu 8.10 a PC s operačním systémem Microsoft Windows Vista Bussines (tzv. meziplatformní komunikace).

3.2.1 Instalace pod OS Windows

Pro nainstalování programu Speak Freely jsem stáhl instalační soubor z internetových stránek výrobce [9]. Po nainstalování a puštění tohoto programu je dobré kliknout v menu na *Help/About Speak Freely,* kde se vypíše IP adresa počítače. Po spuštění programu, není zapotřebí žádného nastavování, program je automaticky v režimu naslouchání sítě. Více o programu Speak Freely pod OS Windows je k dispozici na internetových stránkách výrobce [10]. Rozhodl jsem se nainstalovaný program ve Windows nechat naslouchat a volat ho z OS Linux.

3.2.2 Instalace pod OS Linux

Software pro Linux jsem stáhl od výrobce z internetových stránek [11]. Po stažení jsem zadal do příkazové řádky příkaz *tar xf speak-freely_7.6a.orig.tar*, abych Speak Freely dekomprimoval. Po zadání příkazu *ls* se mi zobrazily dekomprimované soubory a adresáře. Po získání zdrojových kódů vhodných pro kompilaci jsem zadal příkaz *make.* Lze použít i *make install*. Po jeho zadání se mi ovšem vypsalo:

domacnost@domacnost:~/Dokumenty/speak_freely-7.6a$ make
gcc -Wall -O3 -DHEXDUMP -Iadpcm -Iaes -Icelp -Ilpc -Igsm/inc -Ilpc10 -Imd5 -Ides -Iidea -Ilibdes -Iblowfish -DInternet_Port=2074 -DAUDIO_BLOCKING -DLINUX -DHALF_DUPLEX -DM_LITTLE_ENDIAN -DNEEDED_LINEAR -DLINUX_DSP_SMALL_BUFFER -DHAVE_DEV_RANDOM -c -o speaker.o speaker.c
In file included from speaker.c:11:
speakfree.h:32:20: error: curses.h: No such file or directory
speaker.c: In function 'makeSessionKey':
speaker.c:514: warning: ignoring return value of 'getcwd', declared with attribute warn_unused_result
speaker.c:522: warning: ignoring return value of 'getdomainname', declared withattribute warn_unused_result
speaker.c: In function 'playbuffer':
speaker.c:1475: warning: ignoring return value of 'fwrite', declared with attribute warn_unused_result
speaker.c: In function 'main':
speaker.c:2194: warning: implicit declaration of function 'initscr'
speaker.c:2195: warning: implicit declaration of function 'move'
speaker.c:2196: warning: implicit declaration of function 'printw'
speaker.c:2203: warning: implicit declaration of function 'noecho'
speaker.c:2204: warning: implicit declaration of function 'getstr'
speaker.c:2206: warning: implicit declaration of function 'echo'
speaker.c:2207: warning: implicit declaration of function 'endwin'
speaker.c:2263: warning: pointer targets in passing argument 3 of 'getsockname'differ in signedness
speaker.c:2401: warning: pointer targets in passing argument 6 of 'recvfrom' differ in signedness
speaker.c:2820: warning: operation on 'ident' may be undefined

speaker.c:2127: warning: ignoring return value of ‘fwrite’, declared with attribute warn_unused_result

speaker.c:2544: warning: ignoring return value of ‘system’, declared with attribute warn_unused_result

speaker.c:3082: warning: ignoring return value of ‘system’, declared with attribute warn_unused_result

speaker.c:3212: warning: ignoring return value of ‘fwrite’, declared with attribute warn_unused_result

speaker.c:3389: warning: ignoring return value of ‘fwrite’, declared with attribute warn_unused_result

*make: *** [speaker.o] Error 1*

domacnost@domacnost:~/Dokumenty/speak_freely-7.6a$

V souboru *speak freely.h* byla využívaná funkce push to talk, která byla provázána s knihovnou *courses.h.* Nejprve jsem se pokusil funkci zakomentovat, to ale vedlo k dalším chybám. Nakonec jsem zjistil, že je potřeba nainstalovat balík *ncurses-dev.* Po úspěšném nainstalování tohoto balíku příkazem *sudo apt-get install ncurses-dev* jsem zadal příkaz pro kompilaci *make* a kompilace programu Speak Freely úspěšně proběhla.

Po zkompilování a zadání příkazu *ls* jsem dostal výpis souborů včetně potřebných zkompilovaných, které jsou spustitelné.

Popis funkcí binárních souborů:

- *sfspeaker* -naslouchá příchozím spojením
- *sfmike* -navazuje komunikaci a odesílá zvuk
- *sflaunch* -spouští sfmike a sfspeaker tak, že jsou použitelné i na half duplex
- *sfecho* -echo server
- *sflwld* -phonebook server
- *sflw* -phonebook klient
- *sfreflect* -reflektor pro multikonference - server pro konferenční hovory
- *xspekfreely* -spuštění pomocí x-window do grafického režimu

3.2.3 Navázání meziplatformní komunikace

Na PC s OS Windows jsem postupoval dle kapitoly 4.1. V OS Linux jsem pak v příkazové řádce spustil program *sfmike* s parametrem tvořeným IP adresou volaného PC. V mém případě *sfmike 192.168.1.100*. Navázání komunikace proběhlo okamžitě. V příkazové řádce se napsal komentář *pause*, který nám říká, že program je spuštěn a momentálně naslouchá druhé straně – volaného účastníka. Po zmáčknutí mezerníku se *pause* přepíše na *speak* a uživatel může začít hovořit do mikrofonu. Druhá strana ho slyší. Pokud uživatelé na obou stranách nechají nastaven režim speak, komunikace probíhá plně duplexně a není potřeba program spouštět pomocí *sflaunch*.

4 PŘÍPRAVA A INSTALACE POTŘEBNÉHO SOFTWARU PRO CROSS-KOMPILACE

Ke zprovoznění VoIP komunikace na vývojovém kitu je zapotřebí nejdříve vybraný komunikační software upravit a nastavit tak, aby byl spustitelný na kitu STK1000. Nachystat a pomocí cross-kompilace lze software upravit několika způsoby. Je možno využít nástroj Toolchain 2.1.6 (viz kapitola 5.1), AVR32 studio 2.1.2 (viz kapitola 5.2) a nebo Buildroot 2.3.0 (viz kapitola 5.3). Všechny tyto nástroje lze nainstalovat do OS Linuxu a nebo, u jiných operačních systémů, lze využít tzv. virtuální stroj např. vmware.

4.1 Toolchain 2.1.6

Toolchain 2.1.6 je vývojové prostředí složené ze série command-line služeb pro ladění a kompilování aplikací. Je spustitelné pod Linuxem i MS Windowsem (bez virtuálního stroje). Prostředí Toolchain 2.1.6 postavené na nástrojích GNU se skládá z následujících částí:

- *avr32-binutils* 2.18.atmel.1.0.1
- *avr32-gcc with Newlib* 4.2.2-atmel.1.1.4
- *avr32-newlib* 1.16.0.atmel.1.0.0
- *avr32-gdb* 6.7.1.atmel.1.0.3
- *avr32program* 3.1.4
- *avr32gdbproxy* 3.1.5
- *avr32trace* 2.1.0
- *avr32headers* 2.0.6
- *avr32parts* 2.0.3
- *avrfwupgrade* 1.1.2
- *libavrtools* 3.1.6
- *libavr32ocd* 3.1.3
- *libavr32sim* 0.2.4
- *libelfdwarfparser* 2.1.3

Samotné prostředí tooolchain lze nainstalovat na operační systémy Windows 2000, Windows XP, Windows Vista a na Linux distribuce: Fedora 8, Fedora 9, Ubuntu Linux 7.10 (Gutsy), Ubuntu Linux 8.04 (Hardy), openSUSE Linux 10.3, openSUSE Linux 11.0. Pro Linux může být Toolchain 2.1.6 nainstalován jako RPM nebo pomocí balíků.

Toolchain 2.1.6 nelze nainstalovat na operační systémy Windows 95, 98, NT ani ME. Tento software je k dispozici na AVR32 CD dodávaný ke kitu a nebo ho lze stáhnout z internetových stránek výrobce [12] pod položkou "Tools & Software".

Vzhledem k tomu, že AVR32 Studio 2.1.2 v sobě může integrovat Toolchain 2.1.6 se všemi jeho částmi, rozhodl jsem se samotný Toolchain 2.1.6 nepoužívat a přistupovat k němu právě skrze toto studio. Podrobnější popis GNU Toolchain 2.1.6 je popsán v dokumentaci [13].

4.2 AVR32 Studio 2.1.2 pod OS Windows

AVR32 Studio 2.1.2 je integrované vývojové prostředí (IDE) pro vývoj aplikací kompatibilní s architekturou AVR32. Toto studio nabízí kompletní sadu funkcí pro projektový vývoj a ladění, C/C++ editor se syntaxí zvýrazňování, debugger podporující spuštění kontroly a je postaveno na Eclipse, které umožňuje snadnou integraci s pluginy třetí strany. AVR32 Studio 2.1.2 podporuje všechny AVR32 32 bitové procesory a základní desky jako jsou ATSTK1000 a NGW100. AVR32 Studio 2.1.2 nabízí dva druhy budování systému, ve dvou režimech provozu (*Internal build* a *managet make*). Více podrobných informací o těchto režimech a dalších možnostech AVR32 Studiaje popsáno v příručce [14]. Aby bylo možné spustit AVR32 Studio 2.1.2 je nutné mít nainstalovaný následující software:

- Windows 2000 nebo Windows XP,
- Fedora Core 4, 5 nebo 6, Ubuntu Linux 6.06 (Dapper) nebo SUSE Linux 10.2,
- Sun Java 2 platformu verze 1.5 nebo novější,
- Internet Explorer, Mozilla nebo Firefox,
- AVR32 GNU Toolchain.

AVR32 Studio 2.1.2 nepodporuje Windows 98, Windows NT a Windows ME a nebylo testováno na Windows Vista, či 64-bitových operačních systémech. Aby bylo možné ladit linuxové aplikace na cílový systém je nutné nainstalovat i GNU Debugger a Buildroot 2.3.0 (viz kapitola 5.3). AVR32 Studio 2.1.2 i ostatní software je k dispozici na AVR32 CD dodávaný ke kitu a nebo ho lze stáhnout z internetových stránek výrobce [15] pod položkou "Tools & Software".

Po nainstalování výše zmíněného softwaru, jsem nainstaloval AVR32 Studio 2.1.2 na PC s OS Windows XP SP2. Studio však po spuštění spadlo, aniž by napsalo jakoukoliv chybovou hlášku. Zkusil jsem Studio nainstalovat na jiný PC a zde mi pro změnu nešla spustit instalace, což bylo doprovázeno chybovou hlášku, viz obr. 5.1.

Obr. 4.1: Chybová hláška při pokusu o instalaci AVR32 studia

Na internetu jsem zjistil, že se jedná o chybu Windows InstallShieldu, která by měla být vyřešena stáhnutím a uložením instalačního souboru do počítače. Opětovný pokus o instalaci by měl být úspěšný. Problém však nadále přetrval. Další instalace AVR32 Studia na notebook s OS Windows XP dopadla stejně. Jako poslední pokus jsem zkusil nainstalovat studio pod OS Windows Vista (32 bit), kde se konečně podařilo studio bez dalších komplikací spustit.

Pro ozkoušení cross-kompilace jsem se rozhodl nejprve zprovoznit jednoduchý projekt Hello World! Založil jsem nový projekt, jako cílový MCU jsem dle návodu [16] zvolil typ projektu *AVR32 Standalone Executable.* V okně *Project explorer* jsem vytvořil nový zdrojový soubor typu C++.

Do prostoru pro zdrojový kód jsem vložil následující kód:

```
#include <stdio.h>

int main(int argc, char** argv)
{
   printf("Hello World!\n");
   return 0;
}
```

Pro použití příkazu printf je potřeba nastavit hlavičkou *stdio.h* na knihovnu vstupně výstupních rutin. Můžeme to provést staticky či dynamicky. Zvolil jsem statické odkazování, protože dynamické zahrnuje celou knihovnu a poté ji celou nahrává na cíl. Program Speak Freely obsahuje velké množství hlaviček, takže bych mohl výsledný software zbytečně obohatit o velké množství knihoven. U statického odkazování bude linker pracovat pouze s potřebnými částmi knihoven. Vybral jsem projekt v panelu nástrojů a otevřel jeho vlastnosti. Ve vlastnostech jsem vybral položku *C/C++ Build*. V záložce *Tool Settings* jsem vybral *AVR32/GNU C++ Linker*. Konečně se objevila okna knihoven a popis jejich cest k nim, kde jsem zvolil potřebnou statickou knihovnu libstdc++ a její cestu k ní na pevném disku. V záložce *Miscellaneous* v menu *AVR32/GNU C++ Linker* jsem nastavil do rámečku *Other Options* volbu *–static*. Dále jsem projekt zkompiloval, výsledný soubor *hello.elf* je ve formátu, který lze spustit na kitu STK1000. Dle kapitoly 6.1 jsem soubor, pomocí SSH serveru, zkopíroval na SD kartu vývojového kitu a na kit se pomocí příkazu *ssh root@192.168.1.100* připojil. Po zadání *hesla* roota jsem změnil příkazem *chmod 777 hello.elf* práva tohoto souboru a příkazem *./hello.elf* jej spustil. Po spuštění se ovšem vypsala chybová hláška *Segmentation fault*. Segmentation fault se často objevuje, pokud překladač a běžící jádro nejsou kompatibilní. Může to být problém s nastavením překladače, či chybným nastavením použitých knihoven. Pokusil jsem se chybu odstranit, také jsem studio přeinstaloval a nainstaloval nejnovější verze uvedených programů, zkusil jsem také s AVR32 studiem použít Toolchain jiné verze než 2.1.6 a od různých výrobců. Vše však bezúspěšně. Po konzultaci s profesionálním programátorem firmy AVG panem Ing. Michalem Hejtmánkem jsem se rozhodl použít ke cross-kompilaci Buildroot 2.3.0 popsaný v následující kapitole a práci s AVR32 studiem jsem ukončil.

4.3 Buildroot 2.3.0

Buildroot 2.3.0 je sada skriptů, které jsou schopny postavit kořenový systém souborů pro požadovaný cíl. Toolchain kompiluje od začátku bez toho, aby se spoléhal na to, co je nainstalované na hostitelském zařízení. Buildroot skripty jsou založeny na kombinaci makefile a kconfigu, které se běžně využívají v mnoha projektech. V Kconfigu jsou uloženy konfigurace rozhraní tak, aby je uživatel mohl co nejsnadněji nastavit. Makefile pak načítá hodnoty uloženy v kconfigu a konfiguruje soubor pravidel, kterými je software kompilován.

Buildroot 2.3.0 nejprve začne kompilací Toolchainu. Uživatel může použít implementovaný v Buildrootu 2.3.0 nebo může využít externí Toolchain. V dalším kroku Buildroot 2.3.0 načte informace z jádra Linuxu a ze softwaru. Nakonec v sobě spojí všechny aplikace s potřebnými knihovnami a jádrem Linuxu. Výsledkem je obraz souborového systému, připravený pro uživatele, aby ho použil na vývojový kit.

Buildroot 2.3.0 je z velké míry závislý na Linuxu. Ke spuštění jeho instalace je velmi doporučované využít samotného Linuxu, nebo při použití jiných operačních systémů, Linuxu spuštěném ve virtuálním stroji. Buildroot 2.3.0 je podporován většinou hostitelských architektur a vyžaduje poměrně hodně volného místa na pevném disku, před započetím prací minimálně 5 GB. Pokud chceme kompilovat nějaký software, je také potřeba předinstalovat soubor hostitelských nástrojů. U většiny Linuxových ditribucí je to umožněno formou instalací potřebných balíčků.

Seznam požadavků (nástrojů):

- *C compiler (GCC)*
- *C++ compiler (pro Qtopia® (G++))*
- *GNU make*
- *sed*
- *flex*
- *bison*
- *patch*
- *gettext*

- *libtool*
- *texinfo*
- *autoconf*
- *automake*
- *ncurses library*
- *zlib library*
- *libacl library*
- *lzo2 library*

Po stažení Buildrootu 2.3.0 z internetových stránek výrobce [17] je potřeba zadat: *make atstk1002_defconfig.* Buildroot 2.3.0 nahraje potřebné informace a uloží je do souboru config. Dalším příkazem je *make source,* pomocí něhož jsou staženy zdrojové soubory. Pomocí dalšího příkazu *make*, doděláme samotnou instalaci Buildrootu 2.3.0. Buildroot 2.3.0 začne stahovat softwarové balíky z internetu, extrahuje je na lokální souborový systém a bude nainstalován pro AVR32. Po nahrání Buildrootu 2.3.0, bude vytvořen kořenový adresář s binárními kódy.

Buildroot 2.3.0 bude mít následující strukturu (v závislosti na konfiguraci systému):

- *Config.in*
- *.defconfig*
- *docs/*
- *Makefile*
- *package/*
- *project/*
- *target/*
- *TODO*
- *toolchain/*

Po úspěšné instalaci bude v základním adresáři Buildrootu dalších následujících 6 adresářů, včetně níže popsaných souborů, které jsem uznal za důležité je jednotlivě popsat, z důvodu dalšího využití. Níže popsané i další důležité soubory jsou popsané v dokumentu [18].

- *binaries/*
- *build_avr32/*
- *dl/*
- *include/*
- *project_build_avr32/*
- *toolchain_build_avr32/*

Adresář *binares/* obsahuje úspěšně kompilovaný software. Adresář *build_avr32/* obsahuje knihovny a aplikace potřebné ke kompilaci, obsahuje také *staging_dir/* obsahující Toolchain. Adresář *dl/* obsahuje všechny stažené zdrojové tar archivy. V adresáři *docs/* lze nalést dokumentaci pro Buildroot2.3.0. Adresář *include/* obsahuje data automaticky generované kconfigem. Adresář *package/* obsahuje všechny soubory makefile, popisující jak mají být zkompilovány jednotlivé knihovny a aplikace. V adresáři *project_build_avr32/* je konkrétní cilové jádro Linuxu a nekomprimovaný kořenový souborový systém. Adresář *toolchain/* obsahuje makefile soubory pro Toolchain a v adresáři *toolchain_build_avr32/* obsahuje extrahované zdroje tar. Soubor *config* obsahuje konfiguraci systému. Soubor *config.in* obsahuje nejvyšší úroveň konfigurací. Soubor *defconfig* obsahuje výchozí obecnou konfiguraci pro Buildroot 2.3.0 (zejména pro architekturu x86). Soubor makefile popisuje pravidla, jak sestavit kořenový adresář systému a zahrnuje všechny ostatní makefile soubory rozprostřené v podadresářích. Informace o tom, jak s Buildrootem 2.3.0 pracovat je uvedeno v dokumentaci [19] a [20]. Samotnou cross-kompilaci programu Speak Freely jsem popsal v následující kapitole 5.1.

4.4 SSH

Data, která připravím cross-kompilací na platformě PC je potřeba nějakým způsobem na kit STK1000 přepravit. Přenos dat jsem se rozhodl realizovat pomocí SSH. SSH (Secure Shell) je protokol, který umožňuje vzdáleně přistoupit na server a vykonávat na něm činnosti tak, jako by u něj uživatel přímo seděl. To vše bezpečně, přes šifrované spojení, znemožňující odposlechnutí hesla a další následnou komunikaci. Nejrozšířenější je SSH bezpochyby na Linuxu, i když existují SSH servery i pro jiné operační systémy. Službu SSH také nemusíme použít jen pro práci v příkazové řádce

vzdáleného počítače, ale můžeme přes ni i kopírovat, mazat a přenášet soubory podobně jako přes FTP, jen bezpečněji. Jedna z možností způsobu přenosu souborů přes SSH se nazývá SFTP, Secure FTP, což se nesmí zaměňovat s FTPS, šifrovanou formou klasického FTP protokolu pracující jinak než SFTP.

K použití SSH potřebujeme dva počítače, hosta a klienta, mezi kterými probíhá komunikace. Můžeme se přes SSH samozřejmě připojit z jednoho počítače na ten samý, ale v tom případě SSH nebude představovat žádnou výhodu. Na hostitelském počítači musí být nainstalován SSH server a na klientském počítači klient. Mezi těmito dvěma programy pak probíhá komunikace pomocí SSH v tzv. SSH tunnelu. SSH tunnel může být provozován ve více verzích protokolu SSH, mezi nejznámější patří SSH 1, SSH 1.3, SSH 1.5 a SSH 2. Verze se od sebe mírně liší, každá opravuje některé vady předchozí verze. Protokol vždy používá některý z šifrovacích algoritmů, mezi které patří 3DES, AES nebo Blowfish. Vše, co je posílané mezi hostem a klientem, je na straně odesílajícího počítače zašifrováno a na straně přijímajícího počítače rozšifrováno (platí to samozřejmě pro oba směry). Z bezpečnostních důvodů se po určité době neaktivity jednoho z počítačů spojení přeruší, čemuž se dá zabránit posíláním null packetů, které nenesou žádnou informaci, pouze uchovávají spojení. Komunikaci zabezpečenou pomocí SSH není možné za použití klasických metod přečíst.

OpenSSH postavený na šifrování OpenSSL, které lze šířit za podmínek licence GPL. OpenSSH bylo portováno pro mnoho systémů (např. Linux, Solaris, FreeBSD, NetBSD, AIX, IRIX, HP-UX), ale jeho "domovským systémem" je OpenBSD, jehož tvůrčí tým jsou také tvůrci OpenSSH. Celý balíček OpenSSH se skládá z následujících klientských programů: *ssh, scp, sftp,* serverového programu (daemona) *sshd* a přídavných programů *ssh-add, ssh-agent, ssh-keysign, ssh-keyscan, ssh-keygen sftp-server*. Více o OpenSSH je popsáno na internetových stránkách výrobce [21], z kterých jsem čerpal.

Pro kopírování dat na kit STK1000 jsem využíval Open SSH server. Na PC s OS Linux Kubuntu 8.10, odkud jsem data kopíroval, bylo potřeba příkazem *sudo apt-get install openssh-server* SSH server nainstalovat. Po úspěšné instalaci jsem se mohl na STK1000 připojit pomocí příkazu *ssh root@192.168.1.100*. Pro odpojení od SSH serveru jsem použil příkaz *exit*.

5 IMPLEMENTACE SOFTWAROVÝCH TELEFONŮ NA VÝVOJOVÉM KITU STK1000

Tuto závěrečnou část bakalářské práce jsem rozdělil do dvou částí: kapitola 5.1 volně navazuje na kapitolu 4.3 Buildroot 2.3.0, pomocí kterého jsem cross-kompilaci provedl. Kapitola 5.2 navazuje na kapitolu 4.4 SSH, pomocí kterého jsem kompilovaný software Speak Freely v konečné podobě na kit STK1000 přenesl a spustil.

5.1 Speak Freely – cross-kompilace pomocí Buildrootu 2.3.0

Kapitola 4.3 popisuje teoretické nainstalování Buildrootu 2.3.0. Praktická instalace ovšem proběhne odlišně. Je potřeba počítat s chybami, které bohužel vývojáři firmy Atmel neodstranily. Je také dobré pokaždé stahovat co nejnovější software, který obsahuje méně chyb, než předchozí verze.

Po stažení Buildrootu 2.3.0 z internetových stránek výrobce [17] jsem zadal: *make atstk1002_defconfig.* Vytvořil se systém včetně binárních kódů. V *menuconfig* je potřeba vypnout v *packages* stahované balíky *docwiki* a *docstart*. Tyto balíky neexistují, pokud nebudou vypnuty, nelze pokračovat. Pro odstranění dalších chybových hlášek je potřeba stáhnout patch z internetových stránek firmy Atmel [22] a ten aplikovat na *package/ncurses/ncurses.mk* umístěného v adresáři Buildrootu 2.3.0 (*patch-p0 < patchfile.patch http://www.atmel.no/buildroot/source/buildroot-avr32-v2.3.0-ncurses-backport-ncurses-package-from-BR-20090416.patch*). Příkazem *make* jsem docílil přeložení Toolchainu.

Dalším důležitým krokem bylo vyexportovat cestu k wrapperům (toolchain), zadal jsem: *export PATH = ˝$PATCH:/build avr32/staging/dir/bin/avr32-linux-gcc˝* (cesta může být odlišná). Poté je potřeba stáhnout Speak Freely ve zdrojovém kódu [11], viz kapitola 3.2. Zde jsem musel v makefile upravit řádek č. 54. Zde bylo uvedeno, že k překladu bude využito *gcc. Gcc* je potřeba změnit na námi vykompilované *avr32-linux-gcc.* Po zadání *make* pro cross-kompilaci programu Speak Freely se vyskytne

poslední chyba. V *makefile* se chybně spustí *perl script* místo *avr32-linux-gcc* použil jsem *host gcc*. Čili v makefile *-$(CC) + gcc* jsem nahradil *$(CC)* za *gcc*, bylo to na řádku č. 378. Po opětovném zadáni *make* již došlo k úplné bezchybné cross-kompilaci programu Speak Freely a výsledkem byl binární kód použitelný pro kit STK1000.

5.2 Speak Freely – zprovoznění na STK1000

Pro kopírování dat jsem využil službu SSH dle kapitoly 4.4. Pro kopírování jednoho souboru např. *pokus.elf,* na kit STK1000 do kořenového adresáře jsem zvolil příkaz *scp pokus.elf root@192.168.1.100:~/*

Pro zkopírování adresáře *speak_freely* i s obsahem (který je tvořen binárním kódem po cross-kompilaci) na vývojový kit STK1000 do adresáře *speak_freely* jsem použil příkaz *scp * root@192.168.1.100:/speak_freely.* Přenos souborů po síti ethernet proběhl bez komplikací. Na kit STK1000 jsem se připojil pomocí SSH příkazem *ssh root@192.168.1.100.* Zde jsem pomocí skriptu *alsa.sh* inicializoval zvukový modul (viz kapitola 1.4) a otevřel adresář *speak_freely.* Dle kapitoly 3.2.3 jsem zadal příkaz *sfmike 192.168.1.100* a provedl spojení s PC, na kterém již naslouchal nainstalovaný a spuštěný Speak Freely pod OS Windows Vista. Navázání spojení proběhlo okamžitě bez jakýchkoliv problémů! Po zmáčknutí mezerníku jsem promluvil do mikrofonu a na druhé straně se zvuk ozval s malou prodlevou. Komunikaci jsem ozkoušel i naopak. Přenos zvuku proběhl úspěšně.

6 ZÁVĚR

K úspěšnému cross-kompilování softwaru na vývojový kit STK1000 si uživatel musí uvědomit několik důležitých věcí. Vývoji potřebných aplikací pro AVR32 se věnuje úzké spektrum lidí, čili je nutné počítat s chybami v dokumentaci i ve vývojovém softwaru od výrobce, k ladění těchto chyb je potřeba věnovat dostatek času. Klíč k úspěchu tvoří Buildroot 2.3.0 složen z knihoven, Toolchainu a aplikací, které zvolí uživatel, aby byly zahrnuty v systému. Další významnou komponentou je hlavní část Toolchainu wraper avr32-linux-gcc, který se postará o správné linkování knihoven potřebného typu.

V navazující práci je dle tohoto návodu možno kompilovat široké spektrum softwaru. Užitečné se jeví zprovoznění LCD a portů PS/2 k připojení polohovacích zařízení. Lze zprovoznit i port USB k připojení dalších zařízení např. webkamery.

SEZNAM LITERATURY

[1] Atmel AVR32 32-bit MCU, Stránka produktu ATSTK1000. <http://www.atmel.com/dyn/products/tools_card.asp?tool_id=3918>

[2] Atmel AVR32 Linux BSP User Guide, 3MB, verze 2.0.0, březen 2007. <www.atmel.com-dyn-resources-prod_documents-avr32_linux_user_guide_2.0.0.tar.gz>

[3] Atmel ATSTK1000 Schematics, 18 stran, září 2008. <http://www.atmel.com/dyn/resources/prod_documents/ATSTK1000_schematics.pdf >

[4] Atmel ATSTK1002 Schematics, 7 stran, září 2008. <http://www.atmel.com/dyn/resources/prod_documents/ATSTK1002_schematics.pdf >

[5] PRIŠKIN, J. *Zvukový modul pro vývojový kit*. Bakalářská práce. Brno: VUT FEKT, 2008.

[6] Atmel STK1000 Quick Start Guide, 2 strany, květen 2006. < http://www.atmel.com/dyn/resources/prod_documents/doc8029.pdf>

[7] Documentation: AVR32 Linux Development/Bootup and system configurations, leden 2008. <http://www.avrfreaks.net/wiki/index.php/Documentation:AVR32_Linux_Development/Bootup_and_system_configurations>

[8] KRČMÁŘ, P. *Linux-postavte si počítačovou síť.* Praha: Grada, 2008. 184 s. ISBN: 978-80-247-1290-1.

[9] Speak Freely 7.2 for Windows, instalační soubor 802kB, <http://www.speakfreely.org/download/speakfi72.exe>

[10] Speak Freely 7.2, stránka produktu Speak Freely, <http://www.speakfreely.org/>

[11] Speak Freely 7.6 for Linux, instalační soubor 769kB, <http://sourceforge.net/project/downloading.php?groupname=speak-freely&filename=speak-freely_7.6a.orig.tar.gz&use_mirror=switch>

[12] Atmel AVR32 32-bit MCU, stránky produktu GNU Toolchain <http://www.atmel.com/products/AVR32/>

[13] GNU Toolchain/Release Notes 2.1.6, 224 kB, 9 stran, vision 1.4, <http://www.atmel.com/dyn/resources/prod_documents/Release-Notes-AVR32-GNU-Toolchain-2.1.6.pdf>

[14] AVR32015:AVR32 Studio getting started, 24 stran, duben 2008, <http://www.atmel.com/dyn/resources/prod_documents/doc32086.pdf>

[15] AVR32 Studio for Windows, instalační soubor, 265MB, vision 2.1.2, May 2009, < http://www.atmel.com/dyn/products/tools_card.asp?tool_id=4116>

[16] Dokumentation: NGW/HelloWorld, august 2007, <http://www.avrfreaks.net/wiki/index.php/Documentation:NGW/CPPHelloWorld>

[17] Buildroot for AVR32 AP7, instalační soubor, 7MB, verze 2.3.0, March 2009, <http://www.atmel.com/dyn/resources/prod_documents/buildroot-avr32-v2.3.0.tar.tar>

[18] AVR32003: AVR32 AP7 linux Buildroot, 8stran, revize E, November 2008, <http://www.atmel.com/dyn/resources/prod_documents/doc32082.pdf>

[19] AVR32004: AVR32 AP7 How to addsoftware package to Buildroot, 9 stran, revize B, November 2008, <http://www.atmel.com/dyn/resources/prod_documents/doc32085.pdf>

[20] AVR32005: AVR32 AP7 How to add a cystom board to Buildroot, 6 stran revize B, November 2008, <http://www.atmel.com/dyn/resources/prod_documents/doc32062.pdf>

[21] Open SSH Server, Stránky produktu SSH <http://www.openssh.com/>

[22] patch firmy Atmel, 17 April 2009 <http://www.atmel.no/buildroot/source/buildroot-avr32-v2.3.0-ncurses-backport-ncurses-package-from-BR-20090416.patch>

SEZNAM SYMBOLŮ A ZKRATEK

ADC	Analog to Digital Converter – analogově číslicový převodník
AES	Advanced Encryption Standard – pokročilá šifrovací norma/algorytmus
ALSA	Advanced Linux Sound Architecture – pokročilý zvukový podsystém operačního systému Linux
API	Application Programming Interface – rozhraní k programování aplikací
AVR32	32-bit RISC microprocessor with DSP instructions – 32 bitový RSC mikroprocesor s DSP instrukcemi od firmy Atmel
BLOWFISH	Symetrický blokový šifrovací algoritmus
BSP	Board Support Package – podpůrný balíček k hardwaru
C	C with Classes – programovací jazyk C s třídami
CPU	Central Processing Unit – centrální procesorová jednotka, tzv. procesor
C++	Objektově orientovaný programovací jazyk
DAC	Digital to Analog Converter – číslicově analogový převodník
DES	Data Encryption Standard – Datová šifrovací norma/algoritmus
DSP	Digital Signal Processor – digitální signální procesor
ETHERNET	Typ lokální sítě
ETH_A	Zásuvka typu RJ-45 pro připojení sítě ETHERNET
EXT2	Second EXtended filesystem –souborový systém používán jádrem Linux
FTP	File Transfer Protocol – protokol určený pro přenos souborů
GCC	GNU Compiler Collection – sbírka překladačů programovacích jazyků projektu GNU

GNU	GNU’s Not Unix – projekt svobodného softwaru inspirovaný operačním systémem UNIX
GPL	General Public License – Všeobecná veřejná licence
H.323	Doporučení , definující protokoly pro přenos zvuku a videa v jakékoliv paketové síti.
IDE	Integrated Development Environment – integrované vývojové prostředí
I²C	Inter-IC – dvouvodičová sběrnice standardu Philips Semiconductors
I²S	Inter-IC Sound – třívodičová sběrnice Philips Semiconductors navržená pro přenos zvukových dat
LCD	Liquid Crystal Display – Displej z tekutých krystalů
MCU	Multipoint Control Unit – vícebodová řídící jednotka
MMC	Multi Media Card – typ paměťové karty Flash
NGW100	základní deska kombinující AVR32 DSP CPU s komunikačními rozhraními
OPENBSD	OPEN Berkeley Software Distribution – otevřená distribuce softwaru z Kalifornské univerzity v Berkeley
OPENSSH	OPEN Secure Shell – protokol pro zabezpečenou komunikaci
OPENSSL	OPEN Secure Sockets Layer – otevřená vrstva bezpečných soketů
OS	Operation Systém – operační systém
OSS	Open Sound Systém – zvukový podsystém operačních systémů UNIX/Linux
PC	Personal Computer – osobní počítač
PCI	Peripheral Component Interconnect – standard sběrnice pro připojení periferních zařízení
PCM	Pulse Code Modulation – pulsně kódová modulace

PS/2	konektor pro připojení myši nebo klávesnice
RISC	Reduced Instruction Set Computer – systém s omezenou instrukční sadou
RPM	RPM Package Manager – podpůrný balíčkový systém
RS-232	Typ sériového portu/linky
RTP	Real-time Transport Protocol - definuje standardní paketový formát pro přenos zvuku a videa
SD	Secure Digital – typ paměťové karty Flash
SDRAM	Synchronous Dynamic Random Access Memory – synchronní paměť s náhodným přístupem
SFTP	Secure FTP – zabezpečený FTP
SIMD	Single Instruction Multiple Data – paralelní zpracování více toků dat jednou instrukcí
SIP	Session Initiation Protocol – protokol pro iniciaci relací
S/PDIF	Sony/Philips Digital Interconnect Format – sériové rozhraní pro přenos zvukových dat vyvinuté firmami Sony a Philips
SPI	Serial Peripheral Interface – sériové rozhraní pro periferie
SSC	Synchronous Seriál Controller – řadič synchronní sériové komunikace
SSH	Secure SHell – zabezpečený protokol umožňující vzdáleně přistupovat na server, kopírovat data atd.
TCP/IP	Sada protokolů pro komunikaci v počítačové síti
TDM	Time Division Multiplexing – časový multiplex
TWI	Two Write Interface – dvouvodičové rozhraní
UDP	User Datagram Protocol – protokol sloužící k odesílání dat
USB	Universal Seriál Bus – univerzální sériová sběrnice

VGA	Video Graphics Array – počítačový standard pro zobrazovací techniku
VMWARE	virtualizační nástroj, umožňující virtuálně spustit další operační systémy na jednom PC
VoIP	Voice over Internet Protocol – technologie, umožňující přenos digitalizovaného zvuku v těle paketů
VRA	Variable Rate Audio – režim kodeku umožňující změnit vzorkovací kmitočet
3DES	Triple DES – blokový šifrovací algoritmus založený na DES