Vestavěný Linux pro inteligentní dům

Tomáš Novotný, Aleš Marvan, Josef Hájek, Martin Drahanský
{inovottom,imarvan,ihajek,drahan}@fit.vutbr.cz

Abstrakt

S rozvojem výpočetní techniky, sítí a snižováním ceny integrovaných obvodů je dnes možné vytvořit velkou a levnou síť vestavěných počítačů. Tyto možnosti společně s požadavky na komfort v moderních obydlích vedou k rozvoji inteligentních domů. Příspěvek se nejprve zabývá popisem inteligentního domu a toho, jaké jsou vůbec důvody „inteligenci“ doplňovat. V dalších částech jsou postupně probrány možnosti a využití Linuxu ve vestavěných systémech, protože implementace řídicí jednotky inteligentního domu je vestavěným systémem. Poslední část je věnována propojení těchto dvou témat. Jde o popis návrhu a tvorby centrální jednotky, která je postavena na Linuxu.

1 Inteligentní domy

Úvodní část se věnuje popisu inteligentního domu, jeho součástem, řízení a komunikaci mezi jednotkami.

1.1 Inteligentní dům

Moderní inteligentní domy jsou v současné době velmi populární a jejich uživatelé si je pořizují nejen pro zvýšení komfortu vlastního bydlení. Infrastruktura elektroniky řízených a akčních prvků rozmístěných po budově umožňuje monitorovat a ovládat takřka cokoliv, co se v něm a jeho bezprostřední blízkosti děje. Dnešní systémy domácí automatizace však nejsou již jednostranně zaměřené, ale lze si vybírat i podle zaměření, například:

- Úspora energie – takto zaměřený systém se specializuje na ušetření všech druhů energií a využívá se především v tzv. nízkoenergetických domech, kde je potřeba toky energií usměrňovat.
- Bezpečí – dům může hlídat sám sebe. Struktura vhodně rozmístěných bezpečnostních čidel detekuje podezřelé aktivity přímo v něm či jeho okolí. Ať už se jedná o požáry, zatopení sklepních prostor či úniku plynu. Tyto systémy také dokáží zastoupit EZS (elektronický zabezpečovací systém) a mohou přímo komunikovat s policií skrze pulty centrální ochrany.
- Komfort – bezesporu jeden z nejrozšířenějších důvodů, proč si uživatelé do svého domu „inteligenci“ pořizují. Umožňuje pak např. automaticky stahovat žaluzie na základě osvětlení, automaticky otevírat garážová vrata při příjezdu auta, regulovat teplotu v místnosti. Patří zde však i jednoduché úkony jako třeba automatické spuštění zavlažování nebo rozsvícení světla vypínačem. Použití těchto prvků v domě se běžně označuje termínem systémová integrace. Díky automatizaci je užívání takového domu výrazně jednodušší. Jedním tlačítkem lze při odchodu zhasnout světla, vypnout televizi, zatáhnout žaluzie, regulovat vytápění, aktivovat EZS a odpojit rizikovější spotřebiče jako sporák, rychlovarná konvice apod.

1.2 Struktura řízení inteligentního domu

Modelů pro řízení inteligentní budovy existuje více. Vzhledem k povaze této práce, kdy je využíván Linux v centrální jednotce, se však zaměříme na scénář řídicí jednotky (Master) a řízených podjednotek (Slave).

1.2.1 Centrální jednotka

Řídicí jednotka je jakýmsi mozkem celého systému. Je to arbitr, který komunikuje s ostatními jednotkami. Musí být tedy propojena se všemi ostatními moduly v systému. Uživatel většinou pomocí nějakého programu, aplikace na PC nebo chytrého telefonu definuje program, podle kterého se bude řízení domu chovat. Informace ze senzorů jsou pak zasílány do centrální jednotky, která je vyhodnotí a provede příslušné výstupní akce. Definuje tak vztahy mezi vstupními a výstupními jednotkami. Možnosti takového systému jsou takřka neomezené a záleží jen na fantazii a potřebě uživatele.

Dále je potřeba definovat terminologii vstupních a výstupních jednotek, užívanou v souvislosti s inteligentními domy.

- Aktor – výstup, který provádí daný úkol na základě příkazu z centrální jednotky.
- Senzor – vstup, na jehož základě řídicí jednotka může definovat nějakou akci.

Řídicí jednotka je v podstatě programovatelný logický automat, jehož programovací rozhraní je uživateli prezentováno ve formě akce – reakce. Toto rozhraní je často definováno jako aplikace, ve které si uživatel definuje skelet svého domu včetně jednotlivých místností, do kterých pak umístí senzory a aktory. Tím se stává programování centrální jednotky velice jednoduché a pohodlné.

1.2.2 Ostatní jednotky

V zásadě se dají podružné jednotky rozdělit na senzory a aktory podle toho, zda jsou určeny pro monitorování nebo provedení nějaké akce. Pro představu jsou zde uvedeny příklady několika jednotek a k čemu slouží:

- Spínač (senzor) – jeden z nejzákladnějších vstupů. Ve spojení s centrální jednotkou jej lze použít ke spínání takřka čehokoliv.
- Relé (aktor) – jeden z nejzákladnějších výstupů. Umožňuje zapnout/vypnout připojený spotřebič.
- Hladinový snímač (senzor) – monitoruje výšku hladiny kapaliny. Využití v bazénech, sklepech (při zatopení) apod.
- Termostat (senzor) – slouží k nastavení teploty. Pozor, nejedná se však o aktor. O samotnou regulaci se stará řídicí jednotka.
- Spínací hodiny (aktor) – zapne/vypne připojený spotřebič v nastaveném čase. Je možné je také realizovat ve spojení relé a řídicí jednotky, která jej spíná ve zvolených intervalech.

1.3 Komunikace mezi jednotkami

Senzory a aktory musí být nějakým způsobem propojeny s řídicí jednotkou pomocí vhodné sběrnice. Napájení je však kromě výjimek přivedeno samostatnými vodiči. Dále jsou uvedeny nejpoužívanější standardy pro komunikaci v inteligentním domě.

LonWorks LON, Local Operating Network [2] – sběrnice vyvinutá v letech 1989 – 1992 firmou Echelon Corporation ve spolupráci s firmami Toshiba a Siemens. Jako fyzická vrstva může sloužit prakticky cokoliv – kroucená dvoulinka, síťové rozvody, RF, optika, RS-485 atd. Síť může být rozdělena na domény. Využívá p2p architektury (komunikace uzel – uzel). Základem je tzv. inteligentní node. Ten je založen na speciálních mikroprocesorech Neuron chip, na nichž běží LongTalk protokol, který řídí zprávy a směruje samotné zpráv. Tento protokol tvoří firmware každého Neuron chipu. V jeho EEPROM je také uložena 48 bitová adresa jako jedinečný identifikátor.

BACnet Building Automation and Controls Network [3] – je od roku 2003 evropským standardem v rámci CEN s označením ISO 16 484-5. Skládá se ze specifické struktury BACnet objektů, služeb, přenosového protokolu (síťová vrstva) a fyzická vrstva. Jako fyzická vrstva se běžně používá RS-485 nebo Ethernet. Je založena na objektové reprezentaci a organizaci fyzických vstupů a výstupů. Objekt může reprezentovat samostatný fyzický uzel, nebo i skupinu uzlů tvořících pohromadě

nějakou funkci. Každý BACnet objekt je definován sadou vlastností, které popisují jeho chování a řídí provoz.

KNX Konnex bus [4] – sdružuje tři existující sběrnice EIB (European Installation Bus), BatiBus a EHS (European Home System). Umožňuje tak komunikaci mezi přístroji více výrobců. Umožňuje komunikovat rychlostí až 32 kbit/s v závislosti na použitém přenosovém médiu a datové pakety mohou mít velikost 14 B nebo 248 B. Rozsah (délka) sítě může být až 1000 m (ovšem mezi jednotkami 700 m). Umožňuje navíc napájet jednotky přímo po sběrnici, není tedy zapotřebí zvláštních napájecích vodičů. Plně definuje síťovou transportní a aplikační vrstvu. Jakou fyzickou vrstvu lze použít prakticky cokoliv – kroucenou dvojlinku, síťové vedení, rádiový přenos, infračervený přenos, média založená na IP (Ethernet, WiFi, BlueTooth, FireWire).

CIB Common Instalation Bus [5] – tato sběrnice je založena na technologiích firmy Teco, a.s. Umožňuje komunikovat rychlostí 19,2 kb/s s odezvou do 150ms při plném zatížení. Současně s daty se přenáší také napájení pro připojené jednotky. Tím pádem se minimalizuje počet vodičů nutných pro napájení. Jediné, co je nutné dodržet je správná polarita 24 V napájecího okruhu a vyvarovat se kruhu v topologii. Na jednu větev může být připojeno až 32 jednotek, je-li potřeba více, lze síť rozšiřovat pomocí dalších master modulů.

2 Linux a embedded systémy

Tato část popisuje možnosti pro výběr Linuxové platformy a dále nastiňuje vlastnosti a vztah Linuxu k vestavěným systémům.

2.1 Vymezení pojmu Linux

Pojem Linux označuje jen jádro samotného operačního systému. Proto se někdy používá označení GNU/Linux, které vyjadřuje kompletní operační systém, kde jsou zahrnuty i části z projektu GNU. Jde především o knihovnu jazyka C (glibc), překladač (gcc) a další. Samotný tvůrce Linuxu však jednou k tomuto tématu napsal:

> *Umm, this discussion has gone on quite long enough, thank you very much. It doesn't really matter what people call Linux, as long as credit is given where credit is due (on both sides). Personally, I'll very much continue to call it "Linux", but there have already been distributions that call it "Linux Pro(tm)" etc, which I don't find all that surprising.*
>
> – Linus Torvalds, příspěvek ve skupině comp.os.linux.misc

Z tohoto důvodu budeme dále používat nejen pro jádro, ale i pro celý systém prosté označení Linux.

2.2 Výběr platformy

Výběr vhodné platformy je důležitou částí při tvorbě jakéhokoli řídicího systému. Po výběru architektury a rodiny procesoru musíme vybrat, do jaké míry budeme vlastní hardware tvořit. Možností je několik:

- Použít existující řešení
- Využít CoM (Computer on Module) a vytvořit vlastní nosnou desku
- Vytvořit kompletně vlastní řešení

Použití existujícího řešení přináší mnoho výhod. Počínaje hotovým a funkčním návrhem, který je často otestovaný a ověřený komunitou uživatelů, přes možnost ihned objednat funkční desky. Oproti tomu je vývojář limitován životností a velikostí daného produktu. Navíc je potřeba vycházet z toho, jak byl produkt připraven (omezení vstupně výstupních periferií apod.). Někteří výrobci přímo podporují

některé operační systémy, proto může být při tvorbě výsledného systému věnován čas jen přípravě samotné aplikace. Mezi zástupce patří například deska BeagleBoard[1].

Na pomyslném středu stojí využití CoM (někdy také označováno SoM – System on Module). Jde o menší desku, která typicky obsahuje procesor, operační paměť a paměť typu flash pro uložení dat. Při tvorbě systému pak vytváříme jen nosnou desku, do které se CoM vkládá. Výhodou tohoto přístupu je, že tvorba nosné desky je v porovnání s tvorbou celé platformy mnohem jednodušší. Odpadá také jedna z nejnáročnějších částí náchylných na chyby – propojení procesoru s pamětmi. Mezi zástupce patří například moduly firmy Toradex[2].

Návrh vlastního řešení přináší možnost vytvořit zařízení téměř libovolného rozměru spolu s možností využití potřebných periferií, vždy ale podle možností vybraného procesoru a okolních komponent. Životnost produktu je limitována životností jednotlivých komponent a také je snížena cena výsledného řešení. Oproti použití existujícího řešení zde ale přichází zásadní nevýhody v podobě vývoje a testování celého zařízení. Při návrhu vlastního řešení je možné a dokonce vítané využít znalostí z návrhu různých vývojových kitů, které existují vždy pro dané rodiny procesorů. Dokumentace k vývojovým kitům často obsahuje řešení problému z errata dokumentů a mnohdy se jedná opět o otestované zapojení, včetně podpory menší komunity ohledně konfigurace systému.

V našem případě došlo k výběru návrhu vlastního řešení. Ačkoliv toto řešení přináší víc práce, testování, problémů a delšího vývoje, potřeba svého vlastního zařízení o konkrétním rozměru, menší ceně a nezávislosti na výrobci zvítězila.

2.3 Historie

První verze Linuxu byla vydána v létě roku 1991 finským vývojářem Linusem Torvaldsem. Původně šlo o OS vytvořený ve volném čase. V devadesátých letech pomohl rozvoj Internetu a komunit kolem Linuxu k vytvoření prvních distribucí. Během několika let se tento univerzální operační systém vyvinul tak, že je možné jej použít v široké škále zařízení – od vestavěných systémů, přes pracovní stanice až k vysoce výkonným serverům.

2.4 Použití ve vestavěných systémech

Linux má několik vlastností, které jsou vhodné při nasazení ve vestavěném systému [6].

První může být kvalita a spolehlivost kódu. Ač jde o těžko měřitelnou veličinu, je několik faktorů, podle kterých ji můžeme porovnávat. Kód jádra je modulární s přesně definovanou strukturou. Díky tomu je možné snadno přidávat nebo odebírat různé funkce. Při znalosti principů OS je také kód jádra dobře čitelný. Z pohledu stability je jádro prověřeno mimo jiné díky velkému množství nasazení v „kritických“ aplikacích (např. servery).

Další vlastností, která plyne z vývoje s otevřeným zdrojovým kódem, je dostupnost zdrojových kódů a to, že Linux nepatří žádné firmě. Zdrojové kódy jsou vždy dostupné na Internetu – a to bez poplatku. Licence, pod kterou jsou vydávány, je GNU GPLv2. Ta umožňuje svobodné šíření a použití za předpokladu, že všechny změny jsou opět „vráceny“ komunitě. Výhodou je, že se do výsledné ceny produktu licence za software vůbec nemusí promítnout.

Na trhu existuje několik firem, které nabízí podporu pro tento OS. Nabízí konzultace až vývoj ovladačů. Je také možné se při vývoji spolehnout na pomoc komunity, která ale není zaručena a nejde vymáhat.

V souvislosti s komunitou a samotným rozšířením Linuxu najdeme další silnou stránku. Jde o podporu hardware, síťových protokolů a dostupné utility. Někteří výrobci procesorů už přímo podporují Linux a uvolňují vlastní podporu. Vždy se však objeví nějaký hardware, pro který nemusí být ovladač dostupný. Pro vestavěné systémy, které potřebují pracovat s počítačovou sítí, nabízí Linux velkou podporu ze strany síťové vrstvy.

[1] Domovská stránka: http://beagleboard.org/.

[2] Více viz http://www.toradex.com/Products/Colibri-Computer-On-Module.

3 Příprava Linuxového systému

Pro úspěšný provoz Linuxového systému je potřeba provést několik kroků. Na začátek je potřeba zvolit způsob propojení vývojového stroje a cílové platformy. Následně vytvořit vývojové prostředí, ve kterém bude možné překládat programy pro cílovou platformu. Dále je potřeba vybrat části jádra, které budou přeloženy. Následuje část tvorby obsahu souborového systému. V neposlední řadě je potřeba připravit software pro zavedení systému (bootloader).

3.1 Propojení s cílovou platformou

Pro vývoj je možné použít počítač s operačním systémem Linux i Microsoft Windows. V případě Linuxu je však vývoj značně ulehčen. Většina utilit a vývojových nástrojů je určena právě pro tento operační systém. Jedinou oblastí, pro které bývá podpora pro OS Windows větší, jsou různé hardwarové nástroje (debuggery, programovací kabely). Pro většinu však existuje alespoň neoficiální podpora ze strany linuxové komunity.

Při vývoji je potřeba zvolit způsob komunikace mezi cílovou platformou a vývojovým strojem. Existují tři způsoby.

3.1.1 Přímé propojení

V tomto případě je cílová platforma s vývojovým strojem propojena fyzicky pomocí kabelu. Může jít např. o sériovou linku nebo ethernet. Bootloader, jádro i samotný souborový systém jsou přenášeny pomocí tohoto kabelu. Někdy je možné zároveň pomocí tohoto propoje provádět ladění. Výhodou je, že změny provedené v souborovém systému jsou hned dostupné na cílové platformě. K propojení se používají typicky protokoly TFTP (pro přenos jádra) a NFS (pro souborový systém).

3.1.2 Vyměnitelné médium

K přenosu vytvořených obrazů jádra a souborového systému se v tomto případě používá vyměnitelné médium. V případě cíle může jít např. o paměťovou kartu. Ve srovnání s předchozím způsobem je přenos výsledků pomalejší. Na druhou stranu je cílová platforma při běhu nezávislá na vývojovém stroji, což odpovídá výslednému nasazení.

3.1.3 Přímý vývoj

Vývoj lze provádět přímo na cílové platformě. Je možné vytvořit překladač a všechny potřebné nástroje na cílové platformě. Cílová platforma však musí být dostatečně výkonná a vstup/výstup musí umožňovat vývojářům práci. Výhodou je, že není potřeba vytvářet žádné cross-platform (viz dále) nástroje, protože vývoj probíhá přímo na cílové platformě.

3.2 Vývojové prostředí

V případě vývoje na odlišných architekturách (což je typické pro vývoj vestavěných systémů), je potřeba vytvořit tzv. cross-platform nástroje, kde jde především o překladač. Dále je potřeba vybrat a přeložit knihovnu jazyka C.

Existuje několik nástrojů, které tuto práci automatizují a jsou dostupné zdarma. Kromě tvorby překladače se zároveň starají o překlad jádra, programů a vytvoření výsledného systému souborů. Mezi tyto nástroje patří např. Ptxdist[3], Buildroot[4], OpenEmbedded[5], LTIB[6] a ELDK[7]. Jednoznačnou výhodou těchto nástrojů je urychlení vývoje a reprodukovatelnost výsledného systému. Stačí zachovat konfiguraci pro daný nástroj a systém je možné přeložit na jiném stroji. Většina stahuje zdrojové kódy přímo z Internetu.

[3] Domovská stránka: http://www.pengutronix.de/software/ptxdist/index_en.html
[4] Domovská stránka: http://buildroot.uclibc.org/
[5] Domovská stránka: http://www.openembedded.org/index.php/Main_Page
[6] Domovská stránka: http://bitshrine.org/ltib/
[7] Domovská stránka: http://www.denx.de/wiki/ELDK-5

Jako v případě podpory pro Linux i zde existují společnosti, které se zabývají přípravou těchto nástrojů. Někdy jde o vlastní, v jiných případech rozšiřují funkčnost stávajících.

Ve světě vestavěných linuxových systémů je nejznámějším překladačem gcc (GNU C compiler). Umožňuje překládat spustitelné programy pro velké množství architektur (x86, PowerPC, MIPS, ARM, Motorola 68000, ...). Nedílnou součástí výsledného programu v jazyce C je i standardní knihovna. V jejím případě existuje několik implementací. Ty vznikly především proto, že knihovna glibc, používaná na pracovních stanicích a serverech, je příliš velká (řádově jednotky až desítky megabajtů). Místo je ale v případě vestavěných systému často velmi cenný zdroj. Kromě již zmíněné knihovny glibc[8] je možné použít knihovny uClibc[9] a Diet libc[10]. Tyto knihovny se vyznačují především malou velikostí (řádově stovky kilobajtů). Na druhou stranu nepodporují všechny části standardů POSIX. Jde však většinou o části, které nejsou z hlediska vestavěných systémů podstatné. Naproti tomu existuje implementace EGLIBC[11] (Embedded glibc), která je v základním nastavení binárně kompatibilní s knihovnou glibc.

3.3 Překlad výsledného systému

Tato část se zabývá překladem jádra, programy pro systém, tvorbou systému souborů a bootloaderu. Tyto části automatizují nástroje představené v předchozí kapitole. V konfiguraci se jen zvolí, co a jak přeložit. Následně se spustí překlad a za několik desítek minut až hodin je hotov obraz systému. Při změnách nastavení trvá překlad řádově kratší dobu (není potřeba stahovat a kompilovat již vytvořené části). Vytvořený souborový systém odpovídá FHS (Filesystem Hierarchy Standard [1]).

3.3.1 BusyBox

Jedním z programů, který se často objevuje ve vestavěných systémech, je BusyBox[12]. Jde o implementaci většiny Unixových příkazů, která je při statické kompilaci s uClibc velká přibližně 500 kilobajtů. V systému je uložena jako jeden binární program. Všechny utility na ni ukazují symbolickou linkou. Právě podle tohoto odkazu program zjistí, co za utilitu má spustit. Kromě standardních utilit jako `cp`, `ifconfig` a `sleep` může být do překladu zahrnut i jednoduchý webový nebo FTP server.

3.3.2 Bootloader

Nedílnou součástí vestavěného systému s operačním systémem Linux je bootloader. Těch bývá při startu systému v řadě více za sebou. Start systému pak může probíhat následujícím způsobem: Po připojení napájení začne procesor provádět bootloader uložený v ROM procesoru (označovaný např. RBL – ROM Bootloader). Ten má za úkol načíst první blok z nevolatilní paměti (např. flash) do interní paměti RAM a spustit běh. Protože je tento blok malý a ještě není inicializována externí paměť RAM, bývá v něm uložen jednoduchý bootloader (často označovaný jako stage 1). Jeho úkolem je inicializace externí RAM a načtení dalšího, většího bootloaderu (stage 2). Ten už umožňuje zavedení jádra operačního systému.

Stage 2 bootloadery umožňují načtení jádra přes síť, přímo z pamětí flash nebo ze souborového systému na paměti flash. Bootloader také může provádět zápis do flash paměti.

Menší bootloadery (stage 1) jsou často spojeny s konkrétním procesorem. U stage 2 bootloaderů už ale nalezneme několik implementací. Jmenujme např. volně dostupné (pod licencí GPL) zástupce U-Boot[13] a RedBoot[14].

Bootloader U-Boot nabízí možnost vytvoření tzv. SPL (Second Program Loader) bootloaderu, který umožní nahrazení stage 1 i stage 2 bootloaderu. Po nastavení a několika úpravách je možné při standardním překladu vytvořit dva programy: SPL a vlastní U-Boot. Myšlenka je taková, že samotný „velký“ U-Boot, který má velkou funkcionalitu, není při standardním startu potřeba. Proto je možné

[8] Domovská stránka: http://www.gnu.org/s/libc/
[9] Domovská stránka: http://uclibc.org/
[10] Domovská stránka: http://www.fefe.de/dietlibc/
[11] Domovská stránka: http://www.eglibc.org/
[12] Domovská stránka: http://www.busybox.net/
[13] Domovská stránka: http://www.denx.de/wiki/U-Boot
[14] Domovská stránka: http://sourceware.org/redboot/

jej spustit jen v případě potřeby (např. výroba nebo servisní zásah). SPL tedy rovnou načte jádro a tím zastoupí funkci obou bootloaderů. Výsledkem je jednodušší a rychlejší start systému.

4 Paměťová zařízení a souborové systémy

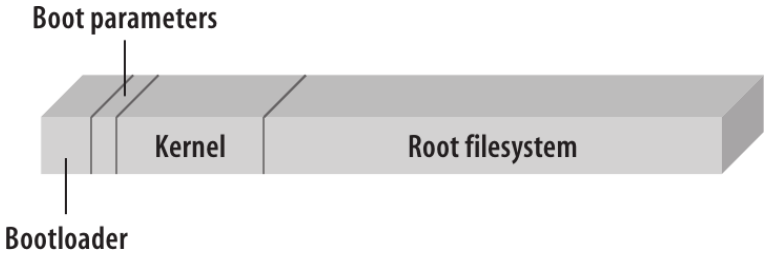
Uložení dat ve vestavěných systémech se většinou liší od způsobu, jakým je provedeno na pracovních stanicích a serverech. Vestavěné systémy používají úložiště na bázi flash pamětí nebo flash disků. Tyto paměti se liší od klasických disků (velikostí, způsobem přístupu, rozdělením na oddíly, spolehlivostí, ...), proto jsou i nástroje na jejich správu odlišné.

4.1 Flash paměti

U tohoto typu pamětí (NAND nebo NOR) je potřeba brát v úvahu jejich životnost. V závislosti na typu se udává počet zápisů na jeden blok v rozmezí desítek až stovek tisíc. „Klasické“ použití tohoto typu paměti a souborového systému by vedlo k rychlému zničení. Proto byly vytvořeny speciální přístupy a souborové systémy právě pro tento typ pamětí.

Na nejnižší vrstvě jde o MTD (Memory Technology Devices). Tento subsystém vytváří záznamy v adresáři `/dev` (kde se vyskytují další záznamy pro přístup k jednotlivým zařízením Linuxu). Ty se pak používají pro fyzický přístup k paměti. U těchto záznamů se objevil problém s jejich zařazením. Klasicky jsou zařízení v Linuxu rozdělena na znaková (`char`) a bloková (`block`). Flash paměť však nespadá ani do jedné kategorie. Přístup po znacích se do paměti typicky neprovádí. Na druhou stranu, bloky flash paměti jsou řádově větší než klasický blok (512 bajtů). Navíc do flash paměti není umožněn přímý zápis. Příslušný blok musí být nejprve smazán, až pak je možné do něj začít zapisovat. Výsledkem je to, že existují jak znakové, tak blokové typy zařízení. Příslušné nadřazené vrstvy pak vybírají vhodnou reprezentaci a adekvátně s ní pracují.

Subsystém MTD se také stará o rozdělení paměti na jednotlivé části (oblast pro bootloader, jádro, souborový systém). Na rozdíl od rozdělení na partice, které známe z pevných disků počítačů, se paměť rozděluje na oblasti dané svým začátkem a koncem. Toto rozdělení se nezapisuje do paměti, ale je uloženo jako struktura v jádru. Definice této struktury se provádí buď přímo ve zdrojovém souboru jádra nebo se předává při startu jádra jako parametr. Typické rozdělení paměti ve vestavěním systému nalezneme na obrázku 1.



Obrázek 1: Rozdělení paměti ve vestavěném systému. Převzato z [6].

Důležitou částí, která se používá u flash pamětí, je FTL (Flash Translation Layer). Jde o mezivrstvu, která se stará o rozkládání zápisů na jednotlivé bloky flash paměti. Jejím úkolem je rozložit zápisy rovnoměrně tak, aby nedocházelo k opotřebení jen jedné části flash paměti. Tuto mezivrstvu obsahují všechny USB paměti. Jde typicky o samostatný čip, který ve vestavěném systému nenajdeme – proto potřebuje speciální přístupy.

Zajímavý přístup pro vestavěné systémy nabízí eMMC. Jde o čip, který v sobě kombinuje paměťovou kartu a řadič. Paměť má rozhraní MMC, tedy zajišťuje vrstvu FTL. Vzhledem k bohatému výběru souborových systémů, které jsou určené přímo pro použití na NAND nebo NOR čipech, je ale toto řešení vhodné spíše do menších vestavěných systémů.

4.2 Souborové systémy

Vzhledem k nízkým životnostem pamětí flash a potřebám vestavěných systémů bylo potřeba vytvořit nové souborové systémy. Ty musí rovnoměrně rozkládat využívání bloků, musí být odolné vůči výpadku napětí a musí být rychlé.

Na jedné straně existuje část souborového systému, která je neměnná po celou dobu života vestavěného systému. Pro tyto části je možné použít souborový systém, který je pouze pro čtení. Jejich výhodou je jednoduchost, rychlost a odolnost proti výpadku. Zástupci z této rodiny jsou např. Cramfs a SquashFS (oba zároveň podporují kompresi dat).

Na druhou stranu jsou případy, kdy potřebujeme neustále ukládat nebo přepisovat data. I nejlepší souborové systémy pro flash paměti by po čase způsobily jejich zničení. Proto se používají souborové systémy v paměti RAM. Ty sice ztrácí obsah při odpojení napájení a mezi restarty systému, na druhou stranu vůbec neničí flash paměť. Používají se typicky pro adresář `/tmp`, jehož velikost bývá malá a není potřeba uchovávat obsah mezi restarty.

Poslední skupinou jsou souborové systémy, které umožňují zápis a zároveň pracují přímo na paměti flash. Jmenujme několik zástupců, např. JFFS2, UBIFS nebo YAFFS. Vzhledem k odlišným přístupům popíšeme princip prvních dvou zmiňovaných.

JFFS2 pracuje na principu logování. Začíná na začátku paměti a každou změnu zapisuje postupně ke konci flash paměti. V případě, že dojde paměť, spustí se tzv. garbage collector. Ten projde místa, kde jsou označeny smazané soubory. V jejich místech smaže celý blok, takže je znovu použitelný. Nevýhodou je pomalejší připojení (mount) souborového systému. Musí se procházet kvůli rekonstrukci souborů.

UBIFS je souborový systém, který je postaven na mezivrstvě UBI, která se stará o mapování fyzických bloků na logické bloky. Má rezervu několika fyzických bloků. V případě, že dojde ke zničení fyzického bloku, je jeho logický blok přesunut na jiný fyzický blok. Tato operace je transparentní pro UBIFS, proto je tento souborový systém ve výsledku jednodušší. Oproti JFFS2 je UBIFS tzv. write-back souborový systém. Změny jsou na disk ukládány z cache, až je to opravdu nutné. Mezi další možnosti UBIFS patří komprese dat.

5 Návrh a implementace centrální jednotky

Ze strany software byl pro centrální jednotku požadavek, aby využívala Linux. Z pohledu hardware vylo vyžadováno připojení do LAN, úložiště pro provozní data a dostupnost několika sériových portů. Díky zkušenostem s procesory od společnosti Texas Instruments byl vybrán procesor AM1707. Další součásti byly záležitostí dodacích podmínek distributorů a zkušeností s daným výrobcem.

Procesor AM1707 je postaven na starším jádře ARM926EJ-S, obsahuje 2 rozhraní pro přístup k paměti (jedno primárně pro nevolatilní paměť, druhé pouze pro volatilní paměť), 3 sériové linky, ethernet, SPI a mnoho další periférií.

5.1 Rozvržení systému

Centrální jednotka je rozdělena do 3 částí (modulů). Všechny moduly jsou propojeny pomocí pinheaderů.

První modul obsahuje napájecí část celého systému a rozhraní na speciální sběrnici elektroinstalace. Tato sběrnice je převedena na sériovou linku a přivedena do druhého modulu.

Druhý modul je „mozkem“ celého systému. Obsahuje již dříve zmíněný procesor AM1707, NAND paměť o velikosti 2 Gb s 16 bitovým přístupem, 128 MB SDRAM paměť se 32 bitovým datovým přístupem, rozhraní pro ethernet, malou flash paměť pro uchovávání konfigurací a programovatelné rozšiřující rozhraní realizované pomocí mikrokontroléru STM32F100. Dále různé podpůrné obvody, zapojení a konektory, které umožňují přístup z dalších modulů.

Třetí a poslední modul vytváří určité rozhraní směrem k uživateli. Obsahuje malý LCD displej, jednoduchou klávesnici, RF rozhraní, konektor pro ethernet, slot pro paměťovou kartu, servisní rozhraní a jednoduchou signalizaci.

Všechny tři moduly jsou do sebe zapojeny jako „sendvič“.

5.2 Chyby v návrhu

Při návrhu se počítalo se zapojením obou sběrnic pro připojení pamětí. Jedna pro NAND flash a druhá sběrnice pro SDRAM. Současně bylo plánováno využít i rozhraní pro SD kartu, které procesor AM1707 nabízí. Vodiče pro přímý přístup na SD/MMC kartu jsou sdíleny s vodiči pro NAND paměť, nicméně na procesoru je výstup pro chip select, který je možné využít pro multiplexor (řídí, jestli se využívá NAND nebo paměťová karta).

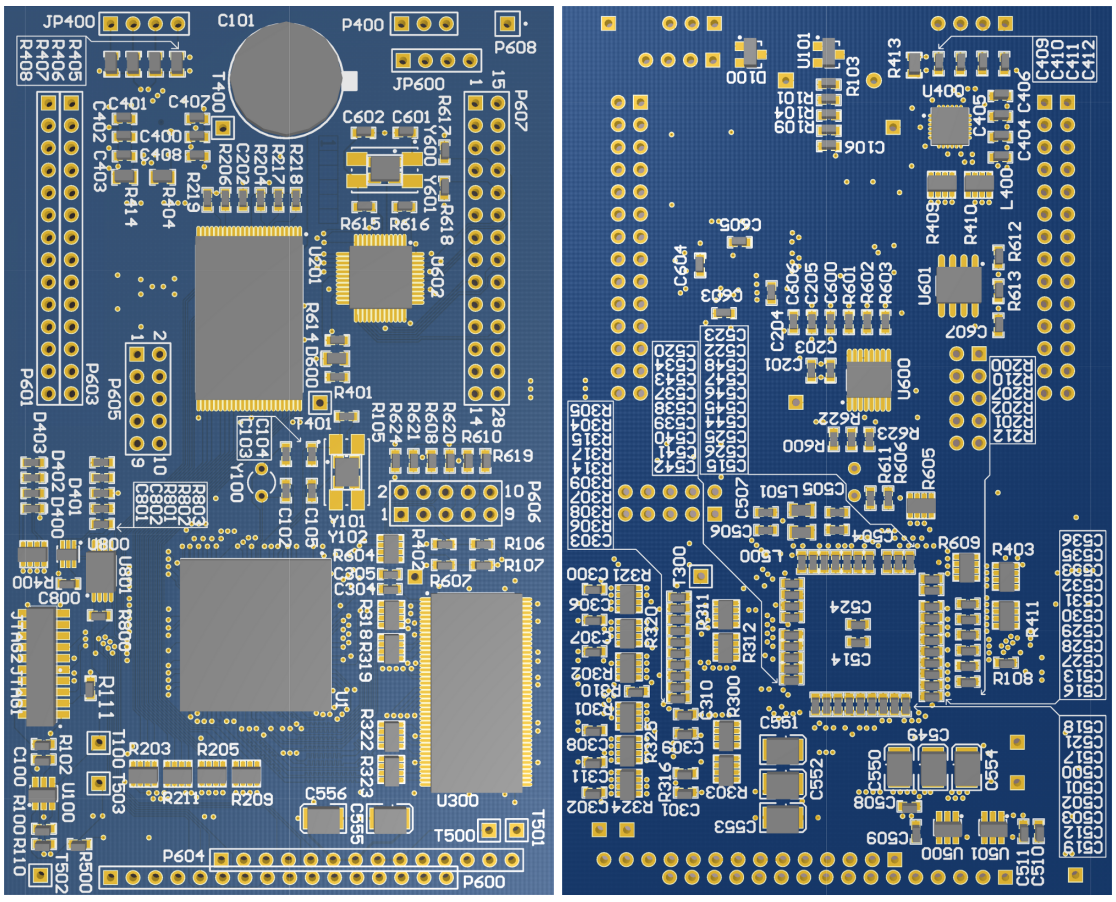
Procesor tedy sice obsahuje obě potřebná rozhraní, neumožňuje ale jejich sdílený běh. Při konfiguraci jádra je nutné si vybrat, které zařízení bude využito a po celý běh je potřeba přistupovat pouze k tomuto typu zařízení. Tuto limitaci je možné obejít, ale zásah do Linuxu by byl velký. Bylo by nutné vždy při zápisu na NAND provést přemapování pinů na modul, který obsluhuje NAND a následně provést zápis. Stejná akce by musela být provedena při zápisu na paměťovou kartu. Zároveň by musely být doplněny zámky, které by hlídaly přístup.

Kvůli těmto obtížím a tomu, že NAND paměť je pro zařízení kritická, rozhodli jsme se využít paměťové rozhraní pouze pro NAND paměť a paměťovou kartu připojit pomocí rozhraní SPI.

5.3 Oživení desky plošného spoje

Zapojení centrální jednotky bylo inspirováno vývojovým kitem k danému procesoru. Texas Instruments, jakožto výrobce čipu, má navíc přímou podporu pro U-Boot a Linux na jeho vývojovém kitu. Přestože byla většina komponent odlišná a využito bylo jen několik periférií, vychází podpora pro námi vytvořenou jednotku z podpory vývojového kitu.

Na následujícím obrázku je vizualizace vytvořené desky.



(a) Horní strana (b) Spodní strana

Obrázek 2: Vizualizace desky

Vytvořená deska má rozměry 101 mm × 62,5 mm. Skládá se z šesti vrstev a je vytvořena ve třídě přesnosti VII.

Při oživování bylo využito pro základní testy rozhraní JTAG, přes které byly nahrávány jednoduché programy, které testovaly funkčnost periférií. Po úspěšném provedení testů přišla další fáze – portace bootloaderu a Linuxu. Ta spočívala především v úpravách nebo doplňování definic specifických pro naši desku.

5.4 Portace software a vývoj

Jako vývojový nástroj pro tvorbu překladače, knihovny jazyka C a dalších komponent výsledného systému byl využit dříve zmiňovaný Buildroot. Jde o jednoduchý nástroj založený na klasických makefilech. Pro vytvoření malého Linuxového systému je plně dostačující. V úvahu připadaly i jiné nástroje, např. OpenEmbedded, ale ten je vhodný u větších a složitějších systémů. Umožňuje totiž mimo jiné i tvorbu balíčků, které usnadňují další správu a úpravy software vestavěného systému. V našem případě však tuto možnost nevyužijeme.

Procesor AM1707 umožňuje natažení spouštěného programu i přes sériovou linku. Jde o jeden z několika boot režimů, které nabízí. Díky tomuto způsobu může být vývoj prováděn i bez rozhraní JTAG. Zároveň jde o způsob, kterým může být nahrán první software do desky. Přes sériovou linku se nahraje malý program, který si z počítače vyžádá bootloader (také přes sériovou linku) a ten nahraje do paměti. Pak spustí právě nahraný bootloader, který se při prvním spuštění postará o natažení jádra a souborového systému. Tento větší objem dat je ale přenášen díky možnostem bootloaderu přes síť – natažení tak velkého objemu dat by přes sériovou linku zabralo několik jednotek až desítek minut.

Díky přítomnosti síťového rozhraní bylo také možné rychle a efektivně ladit Linux. Jádro bylo vždy přeneseno přes síť a okamžitě z bootloaderu spuštěno, nebylo tedy potřeba nic ukládat do flash paměti – tím nedocházelo k jejímu zbytečnému opotřebení.

5.4.1 Podpora AIS

Z pohledu rychlého startu a jednoduchosti bootloaderů je jednou z výhod procesoru AM1707 podpora protokolu AIS (Application Image Script). Tento protokol najdeme i v jiných procesorech firmy Texas Instruments. V ROM procesoru je nahrán jednoduchý bootloader, který umí zpracovat několik základních povelů. Jde především o zápis dat do paměti RAM. Pokud najde ROM bootloader při startu magickou konstantu, začne podle jednoduchých instrukcí provádět program.

Příprava probíhá tak, že je na počítači po přeložení bootloaderu U-Boot programem AISgen vygenerován binární soubor s potřebnými instrukcemi. Kromě samotného bootloaderu je do binárního souboru připojena definice registrů periférií mikroprocesoru. Tento binární soubor je nahrán do NAND paměti.

Při startu procesoru je v paměti NAND nalezena magická konstanta, takže pokračuje vykonávání instrukcí. V první řadě jsou na příslušná místa překopírovány obsahy registrů periférií, čímž dojde k jejich inicializaci. Dále jsou v binárním souboru AIS nalezeny instrukce, které říkají, že další bloky paměti NAND mají být překopírovány do RAM. Zde dojde k překopírování bootloaderu U-Boot. Poslední instrukce v souboru AIS říká, že se má začít vykonávat právě natažený U-Boot v paměti RAM (skočí se na první adresu a začne vykonávání programu). Tato metoda je velmi rychlá a umožňuje nám úplnou eliminaci stage 1 bootloaderu.

6 Závěr

Práce shrnuje základní popis inteligentních domů, dále přechází k obecnému popisu použití Linuxu ve vestavěných systémech a následně popisuje zkušenosti při tvorbě centrální jednotky inteligentního domu. Jde samozřejmě jen o jeden ze způsobů řízení. Inteligentní domy, případně i budovy, se začínají stále více prosazovat, takže se v budoucnu dočkáme dalších přístupu a rozšíření technologií v této oblasti.

Poděkování

Tento výzkum a vývoj byl financován v rámci projektů „Výzkum informačních technologií z hlediska bezpečnosti“ – MSM0021630528 (ČR), „Pokročilé bezpečné, spolehlivé a adaptivní IT“ – FIT-S-11-1 (ČR) a „Centrum excelence IT4Innovations“ – IT4I-CZ 1.05/1.1.00/02.0070 (ČR).

Reference

[1] *Filesystem Hierarchy Standard*. Filesystem Hierarchy Standard Group. 2004. Dostupné z: http://www.pathname.com/fhs/pub/fhs-2.3.pdf

[2] VOJÁČEK, Antonín. Sběrnice LonWorks. [online]. 2005 [cit. 2012-09-14]. Dostupné z: http://automatizace.hw.cz/clanek/2005040501

[3] VOJÁČEK, Antonín. Úvod do BACnetu – Building Automation and Controls Network. [online] 2012 [cit. 2012-09-14]. Dostupné z: http://automatizace.hw.cz/uvod-do-bacnetu-building-automation-and-controls-network

[4] VOJÁČEK, Antonín. Sběrnice KNX pro řízení budov. [online]. 2006 [cit. 2012-09-14]. Dostupné z: http://automatizace.hw.cz/clanek/2006061001

[5] KLABAN, Jaromír. Inels a sběrnice CIB – moderní systém inteligentní elektroinstalace. [online]. 2008 [cit. 2012-09-14]. Dostupné z: http://www.odbornecasopisy.cz/index.php?id_document=38218

[6] YAGHMOUR, Karim, et al. *Building Embedded Linux Systems*. 2. vydání. USA: O‘Reilly Media, Inc., 2008. 442 s. ISBN 978-0-596-52968-0.