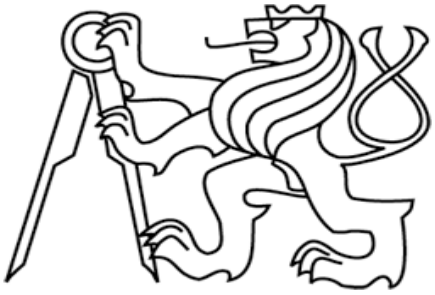
České vysoké učení technické v Praze

Fakulta elektrotechnická



Diplomová práce

Řídicí systém ovládání domu s procesorem ARM

Bc. Ondřej Šafránek

Vedoucí práce: Ing. Pavel Kubalík, Ph.D.

Studijní program: Elektrotechnika a informatika, strukturovaný

Obor: Výpočetní technika

Květen 2011

Poděkování

Děkuji především vedoucímu diplomové práce panu Pavlovi Kubalíkovi za vedení, Marku Hummelovi za podporu a cenné rady v průběhu zpracování této práce

Prohlášení

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

Nemám žádný 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 13. května 2011 ……

Abstract

The thesis describes the design, construction and realization of a main control panel for controlling industrial buildings and other automated units. The core of the main control unit is a microcontroller AT91SAM7X512. The program of the entire device is implemented in EtherNut operating system and allows connection and remote control via the Internet. It deals with the establishment of communication between the card input, output, and keyboard information via the protocol on the parallel and serial RS-485 bus. The work includes a complete hardware implementation of the control unit, including PCB design and verification of the functionality of the system.

Abstrakt

Práce popisuje návrh, konstrukci a vlastní realizaci hlavní řídící desky určené pro řízení průmyslových budov a jiných automatizovaných celků. Jádrem hlavní řídící jednotky je mikrokontrolér AT91SAM7X512. Program celého zařízení je realizován v operačním systému EtherNut a umožňuje připojení a vzdálené řízení prostřednictvím sítě internet. Předmětem práce je vytvoření komunikace mezi kartami vstupů, výstupů a klávesnice prostřednictvím informačního protokolu na paralelní i sériové sběrnici RS-485. Práce obsahuje kompletní hardwarovou realizaci řídící jednotky, včetně návrhu plošného spoje a ověření funkčnosti celého systému.

Obsah

Seznam obrázků

Seznam tabulek

Specifikace

Úkolem celého projektu je navrhnout, zkonstruovat, naprogramovat zařízení určené především pro řízení průmyslových budov, rodinných domů, nebo jiných automatizovaných celků. Systém bude mít velké množství (řádově stovky) datových vstupů a výstupů. Zařízení by mělo komunikovat s okolím prostřednictvím ethernetové přípojky, případně s ostatními vzdálenými periferiemi pomocí sériové sběrnice RS-485.

Výstup celého zařízení by měl, na uživatelsky nastavitelném grafickém výstupu, umožňovat komunikaci s uživatelem. K zařízení by mělo být v budoucnu možné připojovat i další periferie (komunikující prostřednictvím paralelního nebo sériového adresového protokolu) jako jsou například externí klávesnice, display, výkonové prvky (spínající silové obvody), měřiče teploty, záplavová čidla atd. Vlastní kombinačně-sekvenční algoritmus vstupně/výstupní propojovací matice musí být jednoduše uživatelsky nastavitelný. Z důvodu zálohování napájení celého systému bateriemi je při návrhu celého systému kladen důraz na minimalizací odběru energie.

Úkolem práce je předělat systém, přijímající a vysílající stejná data s důrazem na eliminování výše uvedených problémů. Dalším požadavkem na systém je komunikace prostřednictvím webové aplikace, přes kterou bude možno kontrolovat aktuální stav systému včetně jeho ovládání.

Popis existujícího systému

Systém, který vytváříme, vychází z existujícího řešení, které se skládá z těchto prvků:

- **Karta vstupů**
  Úkolem karty je zpracovat údaje přicházejících na 24 digitálních vstupů a převést tyto údaje na paralelní sběrnici obsluhovanou hlavním procesorem. Srdcem karty je osmibitový microkontrolér PIC16F876A. Karta je zkonstruovaná pro připojení na paralelní sběrnici.

- **Karta slaboproudých relé**
  Karta je z hlediska programového vybavení velmi podobná kartě vstupů. Úkolem karty slaboproudých relé je zpracovat data přicházející po paralelní sběrnici a převést tyto data na 15 relátek. Karta (vzhledem k tomu, že se předpokládá její umístění v 19“ subracku) je primárně určena ke spínání slaboproudých rozvodů (jako jsou sirény, elektrické zámky a další). Srdcem celé karty je rovněž osmibitový jednočipový mikrokontrolér PIC 16F876A.

- **Modul silnoproudých relé**
  Obdobně jako karta slaboproudých relé spíná tato karta na základě požadavků hlavní řídící jednotky až 16 silových obvodů (např. světla, čerpadla, elektromotory atd.). Karta může být použita ve větší vzdálenosti, než karta slaboproudých relé, neboť komunikuje s hlavní řídící jednotkou prostřednictvím sériové čtyř-drátové full duplexní sběrnici RS-485. Tranzistory ovládající relé jsou přímo připojeny na brány mikrokontrolér.

- **Hlavní řídící jednotka**
  V současné době hlavní jednotka s microkontrolérem ATmega128. Procesor cyklicky vysílá požadavky o zaslání stavu na všechny periferní karty. Po obeslání všech karet se zahájí výpočet výstupních dat podle napevno stanovené konfigurace. Výsledek je opět odesílán na jednotlivé periferní karty, které data zpracují.
  Jedná se o dočasné řešení, které bylo zkonstruováno na univerzálním procesorovém kitu.

Hlavním nedostatkem stávajícího systému je narůstající prodleva mezi vytvořením a provedením požadavku společně s přidáváním karet a stálé vytěžování periférií, které neprovádí změny. Jako komunikační protokol mezi periferií a hlavním procesorem je osmibytová komunikace, skládající se ze čtyř bytů adresy a čtyř bytů dat. Takový protokol je zcela nevyhovující, neboť není z odečtených dat zřejmé, zda se jedná o adresu nebo data.

Z uživatelského pohledu je jakékoliv provedení změny konfigurace velmi obtížné, neboť konfigurace je přímo zkompilovaná ve zdrojovém kódu mikroprocesoru. Dalším omezením je velmi obtížná integrace nové karty do systému. Takové změny s sebou nesou nutnost úpravy zdrojových kódu mikroprocesoru a tak potenciální výskyt chyb.

Celý projekt vychází z výše uvedeného systému, ze kterého je převzata v podstatě pouze hardwarová část komunikačních vstupních karet, karet slaboproudých relé a výkonových modulů.

V rámci projektu je třeba vytvořit:

- nový komunikační protokol mezi kartami na paralelní sběrnici
- přeprogramovat všechny karty na paralelní sběrnici
- vytvořit komunikační protokol pro sériovou linku
- přeprogramovat všechny powermoduly
- navrhnout software klávesnice s displejem a připojit ji po sériové lince k systému
- navrhnout a zkonstruovat hlavní řídící jednotku celého systému vč. návrhu PCB a osazení
- oživit hlavní řídící jednotku
- vytvořit software pro hlavní řídící jednotku splňující zadání (komunikace po sériové a paralelní sběrnici, komunikaci po internetu, webové stránky systému umožňující ovládání, konfiguraci a nastavení celého systém)

Analýza

3.1. Požadavky na hlavní jednotku

Hlavní jednotka by měla být srdcem celého systému. Musí komunikovat prostřednictvím sériové a paralelní sběrnice se zhotovenými kartami a vyhodnocovat výstupy. Funkce hlavní řídící jednotky musí být uživatelsky přívětivě nastavitelné prostřednictvím webových stránek.

Na hlavní řídící jednotku jsou kladeny následující požadavky:

- Komunikace prostřednictvím RS-485, RS-232 a paralelní sběrnice pro připojení karet
- Podpora ethernetu, CGI skriptů (dynamické generování stránek – viz kapitola 5.1.4.1. HTTP server a CGI skripty) pro vytvoření uživatelského prostředí.
- Z důvodu záložního bateriového napájení je kladen velký důraz na minimální spotřebu celého systému.
- Existence Real-Time operačního systému
  - Mikroprocesor bude obsluhovat přicházející požadavky z periferních karet po sériových a paralelních sběrnicích, požadavky z internetu - nutná podpora vláken.
  - Rychlá inicializace systému – při výpadku systému musí být systém schopen obnovení původního stavu během velmi krátké doby.
- Rychlost.
- Velká externí paměť pro ukládání konfigurace.
- Paměť uchovávající stránky / webovou aplikaci.
- 5 V kompatibilní logika pro komunikaci po paralelní sběrnici.
- Velikost pouzdra, kitu – celá hlavní řídící jednotka musí mít rozměry 149 x 140 mm pro zasunutí do Racku.
- Dostupnost vývojového boardu, prostředí a programátoru.
- Nízká cena.

3.2. Potenciální kandidáti

V přehledu jsou zdůrazněné pouze parametry důležité pro náš systém:

3.2.1. ATmega128, ATmega256

8 bitové procesory firmy Atmel [1], [2].

3.2.1.1. ATmega128

- 8 bitový
- 16 MIPS na 16 MHz
- 4 KB interní SRAM
- 4 KB EEPROM
- 128 KB programové paměti

- Napájení: 4,5 až 5,5 V
- Odběr: do 20 mA při 5 V
- 2x USART

3.2.1.2. ATmega256

- 8 bitový
- 16 MIPS na 16 MHz
- 8 KB interní SRAM
- 4 KB EEPROM
- 256 KB paměť programu
- Napájení: 2,7 až 5,5 V
- Odběr: od 14 mA při 5 V
- 4x USART
- Obtížná dostupnost vývojového kitu

Procesory řady ATmega jsou pro uvedené účely výkonově nedostatečné. V současné době je hlavní procesorová jednotka realizována procesorem ATmega128 a plnění jen základní funkce (přepočítání momentálních stavů na základě přijatých stavů) naráží na výkonnostní problém. Nahrazením procesoru na ATmega256 by velké zlepšení nepřineslo, neboť hlavním rozdílem je zvětšení paměti programu. V podstatě ale chybí podpora ethernetu.

3.2.2. AT91RM9200

- 200 MIPS na 180 MHz
- Technologie Thumb®
- 16 KB datová a 16 KB instrukční cache
- Napájení: 3,0 až 3,6 V
- Odběr: běžně 24,4 mA, v režimu StandBy 520 µA
- 4x USART včetně podpory RS-485
- Ethernet MAC 10/100 Base-T s integrovanou 28 bytovou FIFO, dedikovaný DMA kanál pro příjem a odesílání
- Vhodný operační systém Linux

AT91RM920 [3] je silný procesor splňující základní podmínky pro hlavní procesorovou jednotku. Procesor podporovaný bezplatným operačním systémem Linux. Nevýhodou je celkem vysoká cena.

3.2.3. PIC32

- 32 bitový
- 32 KB až 512 KB paměti Flash, až 128 KB RAM
- Napájení: 2,5 až 3,6 V
- 2x UART s podporou RS-232 a RS-485
- Placené vývojové prostředky
- Až 85 MHz pracovní frekvence

Pro naše účely nevyhovující.

3.2.4. AT91SAM7X128 / AT91SAM7X256 / AT91SAM7X512

- 32 bitový
- Technologie Thumb®
- 55 MHz, 0,9 MIPS
- 128/256/512 KB Flash paměť programu
- 32/64/128 KB SRAM
- ARM 7 jádro
- Odběr: 40 mA při 3,3 V
- 2x paralelní vstupně/výstupní kontrolér
- Ethernet MAC 10/100 Base-T s integrovanou 28 bytovou FIFO, dedikovaný DMA kanál pro příjem a odesílání
- 2x USART s podporou RS-485
- USB 2.0
- Sam-Ba® BootManager
- Vhodný operační systém: Linux, Ethernut, FreeRTOS

Zcela postačující procesory [4] pro určenou aplikaci. Pro procesor jsou podporované bezplatné OpenSource operační systémy např.: Linux, FreeRTOS, Ethernut; bezplatný program Sam-Ba pro nahrávání programu prostřednictvím USB.

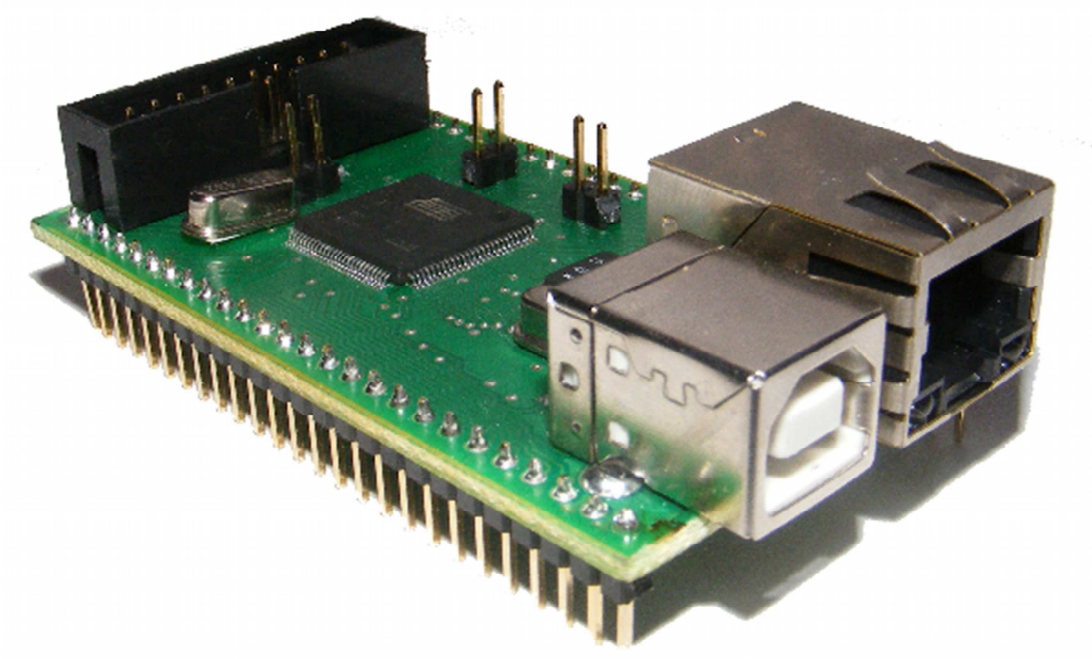
Procesory Atmel s jádrem ARM patří k nejrozšířenějším procesorům pro středně náročné až velmi náročné jednočipové aplikace. Mikrokontrolér AT91SAM7X512 se jeví jako výkonově a cenově nejvýhodnější. Výhodou jsou také snadno dostupné vývojové KITy od několika výrobců v cenově přívětivých relacích.

Rozbor HW a SW

4.1. AT91SAM7X512-KIT

Vývojový kit AT91SAM7X512-KIT (obrázek 4.1) je postaven na mikrokontroléru AT91SAM7X512. Díky tomuto mikrokontroléru může aplikace využít až 128 kB interní paměti RAM a 512 kB interní paměti FLASH. Na kitu je rovněž připravena datová paměť FLASH o kapacitě 4 Mbit pro ukládání dat či webových stránek. Díky jednoduché konstrukci je možné používat vývojový kit jako OEM modul. Samotný kit obsahuje konektor RJ-45 pro připojení k datové síti ethernet, konektor USB-B a konektor JTAG pro ladění přímo na desce. Mikrokontrolér je taktován hodinovým kmitočtem 18,432MHz. Koncepce kitu vychází z AT91SAM7X512. Vývody kitu jsou v rastru 2,54 mm a na každé straně se nachází pouze jedna řada pinů. Díky této vlastnosti je možné vývojový kit používat nejen v aplikacích ale i na nepájivém kontaktním poli [8].

Výhodou mikrokontrolérů na bázi jádra ARM7 je značná podpora OpenSource aplikací jako jsou EtherNut nebo FreeRTOS.



Obrázek 4.1: Vývojový kit AT91SAM7X512-KIT

4.2. Nut/OS

Modulární architektura operačního systému Ethernut (Nut/OS) a jeho síťové vrstvy (Nut/Net) umožňují sestavit operační systém s podporou síťové vrstvy. Lze sestavit systém, který dokáže efektivně řešit požadovaný problém s minimálními požadavky na paměť RAM a FLASH.

4.2.1. Vlastnosti operačního systému EtherNut

- Nut/OS je postaven na kooperativním (nepreemptivním) plánování CPU. Poskytuje tak aplikacím rozhraní pro správu vláken. Jednotlivá vlákna lze spouštět, zastavovat a řídit jejich prioritu.
- Pro práci s časově závislými událostmi a řízení meziprocesních vazeb nabízí Nut/OS rozhraní pro zpracování a generování událostí.
- Většina funkcí aplikačního rozhraní umožňuje definovat požadovaný TimeOut, jako maximální dobu pro dokončení operace. Práce s těmito funkcemi je pohodlnější, ale také šetří paměť a přispívá k větší přehlednosti kódu.
- Často je potřeba obsluhovat určité děje v periodických intervalech - takové požadavky řeší aplikační rozhraní časovačů (timerů). Lze vytvářet timery s jednorázovým, ale i periodickým časováním.
- Práce se všemi vstupy/výstupy v Nut/OS je přístupná přes datové proudy. Tak je možné psát programy, které pracují s proudem dat, jehož zdrojem je port UART, TCP spojení se vzdáleným klientem, nebo třeba jen soubor uložený v paměti procesoru. Přitom jsou použité standardní funkce fprintf, fscanf, fread, fwrite, fgetc, fputc atd.
- Nut/OS umožňuje uložit při překladu do programové paměti soubory. K těmto souborům lze přistupovat jako k datovému proudu. Samozřejmě že jen pro čtení, nikoliv pro zápis. Pro práci s čtením dat takového souboru z FLASH paměti procesoru je důležitá sada standardních funkcí, které jsou upraveny přímo pro tuto činnost. Snižuje se tak množství potřebné paměti RAM, protože se vyčítají data přímo z FLASH paměti.
- Samozřejmostí v Nut/OS je podpora práce s dynamicky přidělovanou pamětí … . [14]

4.3. Yagarto

Volně dostupný překladač zdrojových C souborů do binárního souboru pro procesory ARM.

Yagarto překladač zahrnuje rozhraní JTAG debuggerů jako jsou J-Link GDB Server nebo Open On-Chip debuggeru (používaný v naší práci) [16].

4.4. Sam-Ba

Jednoduchý software pro programování mikroprocesorů řady SAM3, SAM7 a SAM9 od firmy Atmel [5].

Realizace systému

Celý systém, zobrazený na obrázku 5.1, se skládá z těchto základních prvků:

- **Karty vstupů**
  U karet vstupů je zapotřebí změnit komunikační protokol ze čtyřbytového (popsaný v kapitole 2. Popis existujícího systému) na nový, bezpečnější protokol. Dále je potřeba naprogramovat funkci generování přerušení při změně stavů karty.

- **Karta slaboproudých relé**
  U těchto karet je stejně, jako u karet vstupů zapotřebí změnit komunikační protokol.

- **Karta silnoproudých relé**
  Úkolem modulu je převést data na výkonové relé a sepnout silové obvody. Deska je vybavena celkem šestnácti silovými relé. Jádrem desky je mikrořadič PIC 16F876A. Data přicházejí do modulu po sériové čtyř-drátové full duplex sběrnici RS-485 a jsou po převedení do TTL úrovní pomocí obvodu MAX1482. Tranzistory ovládající relé jsou přímo připojeny na bránu mikrořadiče.
  Také u této karty je nutné změnit komunikační protokol z původního, nevyhovujícího protokolu na nový protokol SLIP [15].

- **Klávesnice**
  Tato periferie se chová současně jako vstupní i výstupní periferie. Umožňuje na větší vzdálenost volitelně ovládat až 32 výstupů (akce jsou definovány v hlavní procesorové jednotce). Její konstrukce je koncipována jak na použití uvnitř objektu, tak i na venkovní použití. Je vybavena tepelným snímačem a topením pro zajištění funkčnosti v chladných podmínkách, bzučákem, tamperem pro identifikaci odejmutí klávesnici a dvouřádkovým podsvíceným displejem pro intuitivní ovládání.

- **Hlavní procesorová jednotka**
  Jádrem celého systému je hlavní procesorová jednotka AT91SAM7S512-KIT osazená procesorem ARM 7 AT91SAM7S512 společnosti Atmel. Základním úkolem jednotky je sbírat data z připojených karet a na jejich základě vyhodnocovat výstupy pro ostatní připojené karty.
  Hlavní řídící jednotka je vybavena třemi sériovými rozhraními RS-232 pro debugování programu, sériovou sběrnicí RS-485 pro komunikaci s periferiemi, a paralelní sběrnicí.
  Karta je připojená prostřednictvím ethernetové přípojky do lokální sítě.

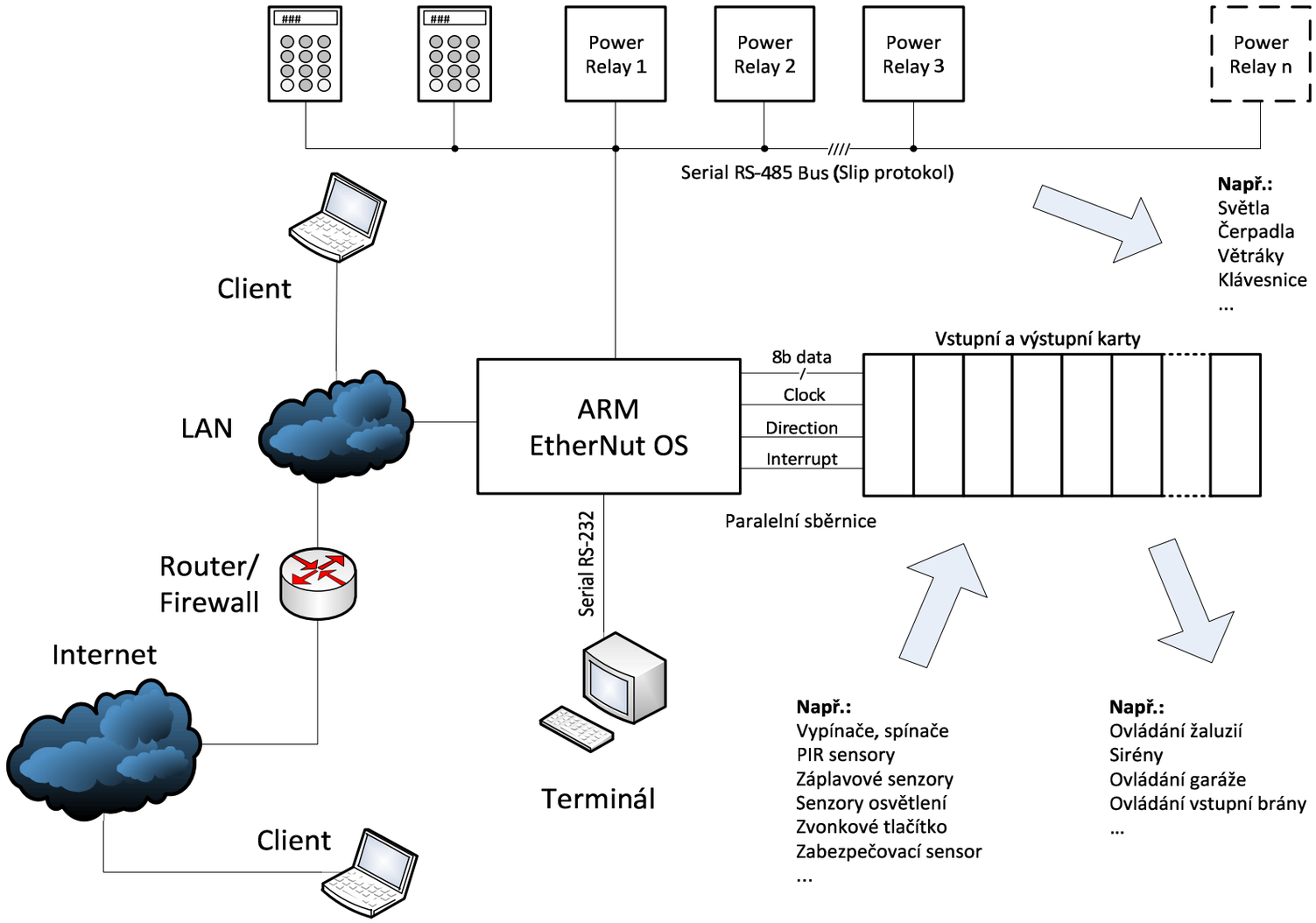
Komunikace mezi paralelními kartami (kartou vstupů a slaboproudých relé) je zajištěna po paralelní osmibitové sběrnici s řídícími signály (CLK, ADR, DIR a IRQ). U těchto karet bylo zapotřebí zcela nahradit původní komunikační protokol za nový, popsaný v kapitole 5.2. Popis komunikace. Modul silnoproudých relé, připojený k sériové lince byl nahrazen nevyhovující protokol novým - protokolem

SLIP [15]. Karta, která se v předchozí verzi nevyskytovala je klávesnice, komunikující prostřednictvím sériové sběrnice RS-485.

Pro ladění systému a základní nastavení je možné připojení hlavní řídící jednotky prostřednictvím sériového rozhraní RS-232 ke konzoli (například PC).

Celý systém je připojený ethernetovím rozhraním do internetové sítě a umožňuje prostřednictvím webových stránek měnit konfiguraci, a ovládat jednotlivé karty.

Popis jednotlivých částí je popsaný v jednotlivých kapitolách.



Obrázek 5.1: Struktura kompletního systému

5.1. Hlavní řídící jednotka

Řídící jednotka je srdcem celého systému. Skládá se z následujících bloků (obrázek 5.2):

- **Konfigurace**
  Konfigurace popisuje chování systému, uchovává vnitřní stavy a nastavení celého systému. Je tvořena dynamickými strukturami obsahující informace o připojených kartách. Nad konfigurací je spuštěno vlákno „Matrix“, vypočítávající výstupní stavy z vnitřních stavů.
  Konfigurace je měnitelná podle požadavků uživatele, přicházejících z ethernetu přes webový server. Pro zachování konfigurace je její každá změna (každá změna v konfiguraci, nikoliv změna ve stavech) ukládána přes SPI sběrnici do Flash paměti a na SD kartu o velikosti 512 KB.
  Uložená konfigurace musí být v takovém formátu, aby ji bylo možné načíst (alespoň částečně) a dále používat i při změně struktur ve firmware.

- **SPI Flash**
  Externí paměť o velikost 512 KB, připojená prostřednictvím SPI k mikrokontroléru. Ve Flash paměti bude uchovávána konfigurace systému.

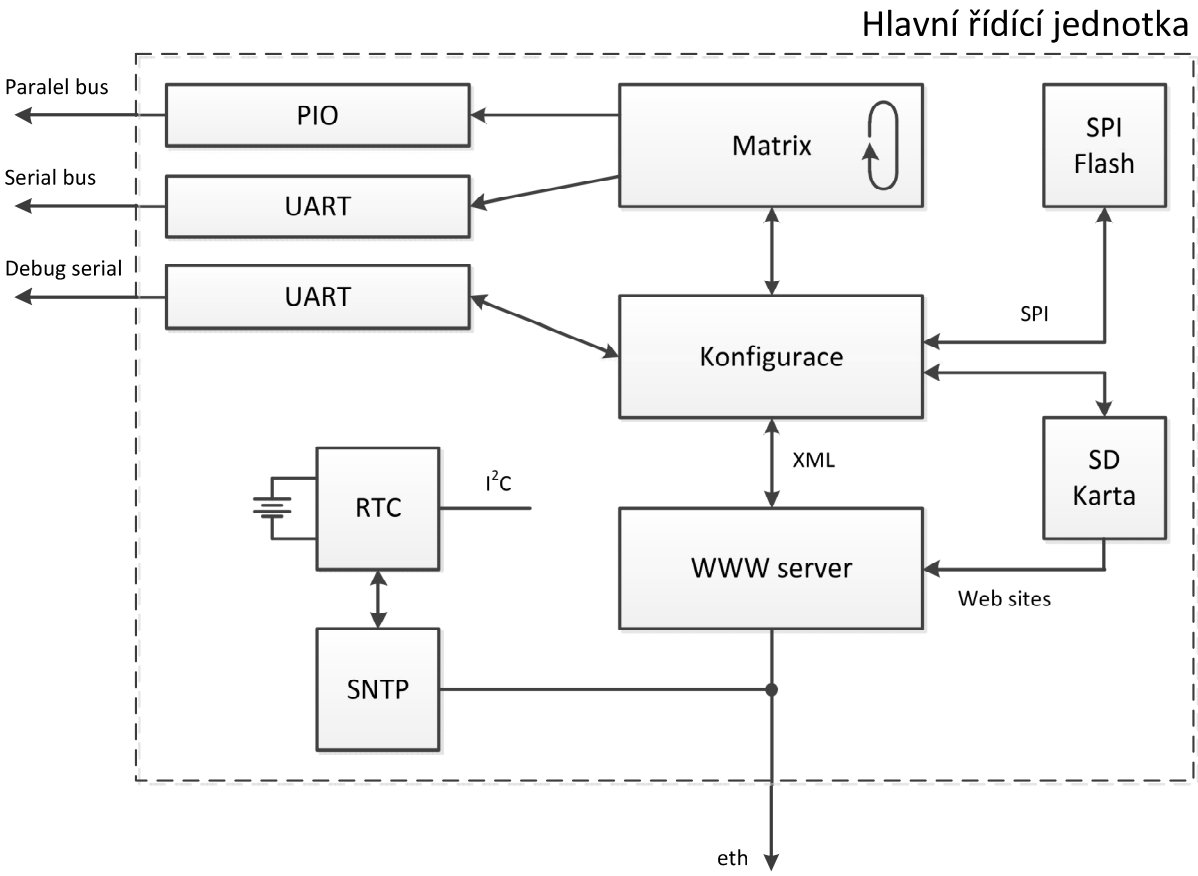
- **SD karta**
  Externí odnímatelná paměť o velikost maximálně 2048 KB, připojená prostřednictvím SPI k mikrokontroléru. Na SD kartu jsou ukládány konfigurace, obrázky a webové stránky. SD karta je naformátovaná na souborový systém FAT16, což umožňuje při vložení do PC snadno měnit webové stránky, jejich konfiguraci nebo zazálohovat uloženou konfiguraci zcela mimo systém.

- **Matrix**
  Zde se vypočítávají výstupy systému na základě aktuální konfigurace a načtených vstupních hodnot. Vypočítané výsledky jsou ukládány zpět do konfigurace a později rozesílány připojeným kartám.

- **WWW server**
  Webový server komunikující s uživatelem přes uživatelský interface s konfigurací. Na základě požadavků uživatelů je možno měnit vnitřní strukturu konfigurace, číst části konfigurace, měnit hodnoty v konfiguraci nebo měnit hodnoty stavů v konfiguraci a ovládat tak objekt z venčí.
  Dále webový server komunikuje s SD kartou, pro načítání komunikačního interface.

- **RTC (hodiny reálného času)**
  Počítačové hodiny, udržující údaje o aktuálním čase. Díky svému externímu zdroji napájení je čas k dispozici i po delším vypnutí desky. Konfigurace času probíhá v daných intervalech operačním systémem, který získá přesný čas pomocí protokolu **SNTP** (Simple Network Time Protocol) z webu. Komunikace je zprostředkována prostřednictvím protokolu I^2C. Obsluha je již implementovaná v systému EtherNut.

» **PIO, UART**
Vstup / výstup ze systému. PIO obsahuje paralelní port, výstup na LED diody, UART přes sériovou linku. Pro přenos dat po sériové lince je použit protokol SLIP [15].



Obrázek 5.2: Blokové schéma hlavní řídící jednotky

5.1.1. Vlákna

Operační systém EtherNut je postaven na kooperativním (nepreemptivním) plánování CPU. Poskytuje tak běh a správu několika vláken. Nepreemptivní plánování znamená, že procesor může přepnout na čekající vlákno (thread) v případě, až momentálně běžící vlákno poskytne procesor jinému vláknu přechodem do stavu spánku, nebo ukončením sama sebe. Do té doby jsou ostatní vlákna blokovány. Pokud se prováděné vlákno zacyklí (a cyklus nebude úmyslný), procesor již nepřepne na jiné vlákno a program přestane fungovat. Z tohoto důvodu je při programování důležité využívat EtherNutové služby pro synchronizaci vláken a vyvarovat se aktivnímu čekání. V aplikaci budou implementované následující samostatná vlákna:

» **idle**
Vlákno vytvořené na pozadí po inicializaci systému. Po inicializaci sytému vytvoří a spustí hlavní (**main**) vlákno. Během dalšího provádění programu je idle vlákno uspané.

» **main**
Hlavní vlákno systému – vlákno provádějící zbylou inicializaci (načtení konfigurace, vytvoření struktur, nakonfigurování periferijních zařízení atd.) a dále již jen vypočítávající výsledky na základě příchozích stavů z periferií (matrix proces, simulující sekvenční logiku). Výsledek přeposílá kartám.

» **emacrx**
Vlákno, které je součástí nejnižší vrstvy síťového driveru. Čeká na příchozí data, vytváří ethernetové rámce. Po vytvoření se volá přerušení, které rámec předá vyšší vrstvě.
Toto vlákno má nejvyšší prioritu z celého systému.

» **tcpsm**
TCPSM – TCP State Machine. Vlákno, starající se o navazování spojení a obsluhu TCP socketu na portu 80 na síťové vrstvě. Tzn. skládá příchozí pakety, řeší timeouty a ignorování TCP paketů.

» **httpd1, …, httpd6**
Šest souběžně uživatelsky obsluhovaných vláken. Vlákna zpracovávají příchozí síťové http požadavky - pracují s konfigurací, podle požadavků konfiguraci přetvářejí a generují XML exporty.

» **updateTime**
Vlákno synchronizující reálný čas s konfigurací. V daných intervalech načte z operačního systému reálný čas a uloží ho do hlavní konfigurace pro následné používání.

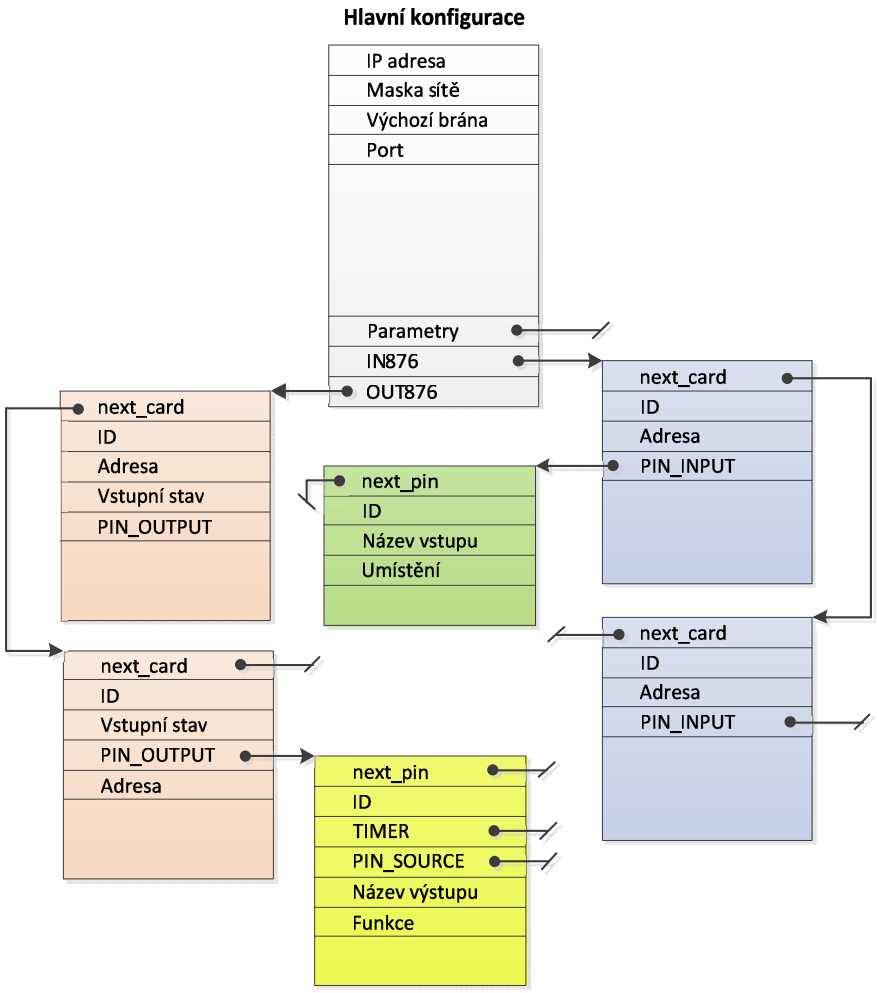
» **debug**
Vlákno spravující sériovou debugovací linku. Čeká na příchozí povely z linky, volá jejich obsluhy.

5.1.2. Konfigurace

Jedním z důležitých požadavků na aplikaci je uživatelské rozhraní, v němž bude možné jednoduše přidávat a odebírat karty, které má řídící jednotka ovládat. Proto musí být nastavení celého systému uloženo v paměti v dynamických strukturách, které budou přidávány a odebírány.

Hlavní struktura obsahuje obecné informace a nastavení systému jako jsou síťová nastavení, SNTP nastavení pro získávání data a času. Dál obsahuje ukazatele na další dynamické struktury periferií vytvářené při běhu systému. Struktury periferií v sobě nesou informace o jednotlivých kartách jako je adresa, název karty, umístění karty v objektu atd. Tyto struktury jsou mezi sebou v každém svém typu zřetězeny. Každá struktura periferie může vlastnit další dynamické podstruktury udržující informace o opakujících se blocích periferie. Jedná se například o struktury jednotlivých vstupů karty, konfigurace výstupních pinů, časovačů atd.

Ukázka takové struktury je zobrazena obrázkem 5.3.



Obrázek 5.3: Schématické znázornění konfigurace

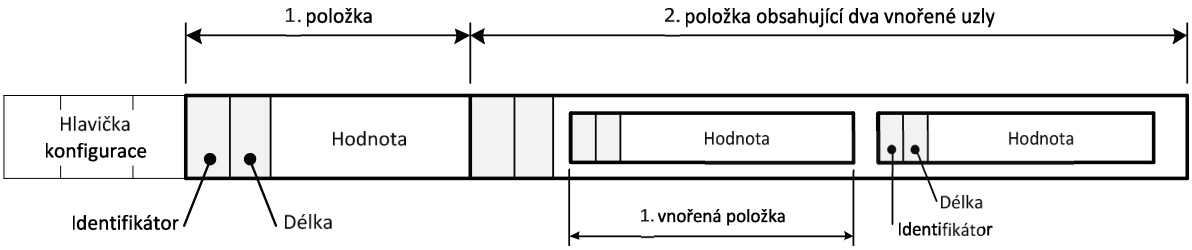
5.1.2.1. Ukládání konfigurace

Konfigurace, se kterou program pracuje, je uložena v pracovní paměti RAM velikosti 128 KB. Při ztrátě napájení nebo resetu jednotky se tato paměť maže a tím se konfigurace nenávratně ztrácí. To je jedním z důvodů nutnosti ukládání konfigurace. Požadavkem na data uložená v externí paměti (Flash, SD karta) je zachování nakonfigurovaných dat i při změně firmware.

Při chodu systému se do konfigurace kopírují vstupní stavy karet a ukládají vypočítané hodnoty. Takové hodnoty jsou platné jen v daný okamžik chodu aplikace, a proto by neměly být uloženy v externí paměti. Při provádění změn konfigurace se z důvodu omezených počtech zápisů pracuje pouze s dynamickými proměnnými. Ukládání se provádí manuálně, po všech provedených změnách. Data jsou uložena prostřednictvím SPI protokolu do paměti Flash o velikosti 512 KB a poté na externí SD kartu, kde mohou být uživatelem zazálohovány. Po spuštění aplikace se primárně konfigurace načítá z paměti Flash. Neobsahuje-li Flash paměť korektní konfiguraci, načtení se opakuje se změnou úložiště na SD kartu.

Uložená konfigurace začíná hlavičkou, která nese počet bytů konfigurace v paměti (bez hlavičky), inverzní hodnotu velikosti konfigurace a kontrolní součet. Při načítání konfigurace lze po načtení samotné hlavičky rozeznat, zda je konfigurace uložená v paměti, či nikoli.

Jednou z možností, jakým způsobem data ukládat je pomocí parseru binárních dat. Každá položka obsahuje hlavičku s identifikátorem položky ve struktuře (v každé struktuře unikátní pro každou položku) a svou velikost. Pomocí informace o velikosti je možné data v souvislé paměti rozsekat na jednotlivé položky a identifikátorem společně s převodní tabulkou rozeznat data a přiřadit je do vytvořených struktur (včetně podstruktur). Ukázka struktury uložených dat v paměti je znázorněna na obrázku 5.4.



Obrázek 5.4: Ukládání dat pomocí parseru

Převodní tabulka (realizovaný pomocí stavového automatu) musí být naprogramovaný samostatně pro ukládání struktur do spojité paměti a pro načítání (parsování) dat.

Toto řešení není složité, umožňuje splňovat požadavek na zachování dat konfigurace při změně firmware a uložené struktury v paměti neobsahují mnoho pomocných dat ke zpětnému vytvoření struktur v paměti. Nevýhodou je ale složitost a rozsáhlost obou automatů - neboť při úpravě / doplnění struktur se musí upravovat oba automaty současně. Oba automaty jsou rozsáhlé a i nepřehledné.

Jinou možností, jak ukládat a načítat data z paměti je vytvořením šablon pro každou strukturu konfigurace.

Každá struktura konfigurace má svou šablonu (zpravidla pojmenovanou stejně jako struktura s prefixem „def_“) skládající se z položek struktury ukládajících se do paměti a dodatečných informací, na základě kterých je možné uložit a načíst jakoukoliv statickou i dynamickou strukturu i po změně firmware. Šablony neobsahují data!

Příklad vypsané šablony pro strukturu klávesnice CARD_KEY877 je znázorněna v tabulce 5.1. Ukázka tvoření takové šablony je zobrazena obrázkem zdrojového kódu 5.5.

Tabulka 5.1:
Ukázka struktury klávesnice

Struktura		Šablona										
		key	var_name	size	offset	type	def_num	def_txt	xml_type	first_id	param	next_itm
CARD_KEY877	***next_keyboard**	-	-	-	-	-	-	-	-	-	-	-
long	**id**	1	id	4	4	TP_ID	0	0	XML_U16	0xFF	0	0
u_char	**address**	2	address	1	8	TP_NUM	0	0	XML_U8	0	1	0
char	**name [32]**	3	name	32	9	TP_TXT	0	Key card	XML_TXT	0	1	0
u_short	**cfg_checkusm**	4	cfg_check	2	48	TP_NUM	0	0	XML_U8	0	1	0
u_char	**cfg_inv_checksum**	5	cfg_inv_check	1	50	TP_NUM	0	0	XML_U8	0	1	0
u_char	**miss**	6	miss	1	62	TP_NUM	0	0	XML_MISS	0	0	0
POSITION	**screen**	7	screen	4	42	TP_NUM	0	0	XML_POS	0	1	0
u_char	**location**	8	location	1	46	TP_NUM	0xFFFF	0	XML_LOC	0	1	0
u_char	**params**	9	params	1	60	TP_NUM	0	0	XML_PAR	0	0	0
u_char	**cfg_invalid_flag**	-	-	-	-	-	-	-	-	-	-	-
KEY877_ITEM	***key877_item**	100	key877_item	60	52	TP_NODE	0	0	0	0	1	0x???
KEY877_PIN	***key877_pin**	101	key877_pin	44	56	TP_NODE	0	0	0	0	1	0x???
Zakončení šablony		0	0	0	0	0	0	0	0	0	0	0

Ukázky jedné struktury (konkrétně struktury klávesnice) je znázorněna v tabulce. Pravá část (sloupce „Struktura“) znázorňuje obsah struktury, pro kterou je šablona vytvořena. Pro zjednodušení procházení struktur je vždy na konec přidaná položka ukončující šablonu nuly.

Samotná šablona se **skládá** z těchto atributů:

- identifikátor položky unikátní vůči dané struktuře (**key**)
- název proměnné (**name**) pro vyhledávání v tabulkách struktur
- velikost položky (**size**), kterou položka zabere v paměti
- pozice položky ve struktuře (**offset**)
- typ položky (**type**) - číslo, text, vnořená struktura nebo identifikátor struktury
- defaultní hodnota, kterou položka získá při vytvoření – defaultní hodnota je určena samostatně pro typ číslo (**def_num**) a textovou hodnotu (**def_txt**)
- typ položky vzhledem XML exportu (**xml_type**)
- binární mapu pro určení skupiny, která je exportována v XML datech (**param**)
- ukazatel na definici dalšího uzlu, na který položka ukazuje (**next_itm**)

Následující příklad ukazuje zapsanou tabulku ve zdrojovém kódu.

PUT_ID, PUT_VALUE_NUM, PUT_VALUE_TXT a PUT_END jsou kvůli zjednodušení jen přepisovací pravidla pro kompilátor, jak nastavit řádek tabulky pro daný typ dat, abychom se nemuseli zabývat atributy položek, který je pro daný typ nepodstatný (např. u číselných hodnot nastavování defaultního textového řetězce).

```
const CONFIG_ITEM def_key877_cards[] = {
        PUT_ID(       CARD_KEY877, 1,   id,               0                                  ),
        PUT_VALUE_NUM(CARD_KEY877, 2,   address,          XML_U8,      0,            EPROM_SAVE),
        PUT_VALUE_TXT(CARD_KEY877, 3,   name,             XML_TXT,     "Key card",   EPROM_SAVE),
        PUT_VALUE_NUM(CARD_KEY877, 4,   cfg_chec,         XML_U8,      0             EPROM_SAVE),
        PUT_VALUE_NUM(CARD_KEY877, 5,   cfg_inv_check,    XML_U8,      0,            EPROM_SAVE),
        PUT_VALUE_NUM(CARD_KEY877, 6,   miss,             XML_MISS,    0,            0         ),
        PUT_VALUE_NUM(CARD_KEY877, 7,   screen,           XML_POS,     0x00,         EPROM_SAVE),
        PUT_VALUE_NUM(CARD_KEY877, 8,   location,         XML_LOC,     0xFFFF,       EPROM_SAVE),
        PUT_VALUE_NUM(CARD_KEY877, 9,   params,           XML_PAR,     0,            0         ),
        PUT_NODE(     CARD_KEY877, 100, key877_item,      KEY877_ITEM,               EPROM_SAVE),
        PUT_NODE(     CARD_KEY877, 101, key877_pin,       KEY877_PIN,                EPROM_SAVE),
        PUT_END
};

#define EPROM_SAVE  (1 << 0)

#define PUT_NODE (type, key, name, p_type, save)
        {key, #name, sizeof(p_type), offsetof(type, name), TP_NODE, 0, 0, 0, 0, save, def_##name}
#define PUT_ID (type, key, name, first_id)
        {key, #name, sizeof((&x_##type)->name), offsetof(type, name), TP_ID, 0, 0, XML_U16, 0xFF, 0}
#define PUT_END {0, 0, 0, 0, 0, 0, 0, 0, 0, 0}
```

Obrázek 5.5: Ukázka zdrojového dódu pro vytvoření šablony struktury

Položky miss a params z ukázky nebudou ukládány do externí paměti, neboť poslední parametr položek (v šabloně pojmenovaný jako „param“) pomocí bitové masky určuje povolení ukládání do paměti. Dalším využitím parametru je redukování XML výstupu, který bude popsaný v kapitole XML export.

Ukládání struktur do paměti probíhá procházením šablon. Šablony se začnou procházet od hlavní konfigurace a zanořují se algoritmem prohledávání do hloubky pomocí ukazatelů na další šablony., Jelikož data nejsou uložena v šablonách, jsou získávány z dynamických struktur pomocí adresování, které je popsáno v načítání dat z paměti. Dada se do paměti ukládají stejně jako v prvním způsobu. Konfigurace začíná hlavičkou, obsahující velikost konfigurace (2B), inverze k velikosti konfigurace (2B) a kontrolního součtu (počítaná bez této hlavičky). Samotné položky struktur obsahují opět hlavičku obsahující index (vždy jedinečný v daném zanoření) a velikost dat. Ta není nutná pro samotné načtení, ale slouží jako jedna z kontrol načtení správných hodnot při změně firmware, nebo načtení starých hodnot ze zálohy.

5.1.2.2. Načítání konfigurace

Načítání dat je již složitější. Konfigurace je uložena v paměti Flash (zde je brána jako primární) a zároveň na SD kartě pro případ, že ve Flash paměti konfigurace nebude, nebo bude poškozená. Absence konfigurace ve Flash paměti se rozpozná kontrolou hlavičky, kde délka konfigurace nebude totožná se svou inverzí uloženou také v hlavičce. Zapisování do Flash paměti probíhá načtením bloku, provedením změny a uložení bloku zpět do Flash. Pře nezměnění prvého bloku Flash paměti (kde konfigurace začíná) se může zdát, že v paměti je platná konfigurace (neboť velikost i její inverzi uložila správně předešlá konfigurace). Proto se následně testuje kontrolní součet – součet všech bytů konfigurace se základem $0x55_h$.

Při načítání konfigurace se prochází současně data načtená z externí paměti a jedna určitá struktura. Při inicializaci je zvolena hlavní (nejnadřazenější) struktura nesoucí globální data. Uložená data vždy začínají touto strukturou. Z dat se přečte nejdříve hlavička, tj. klíč a velikost. Na základě klíče se

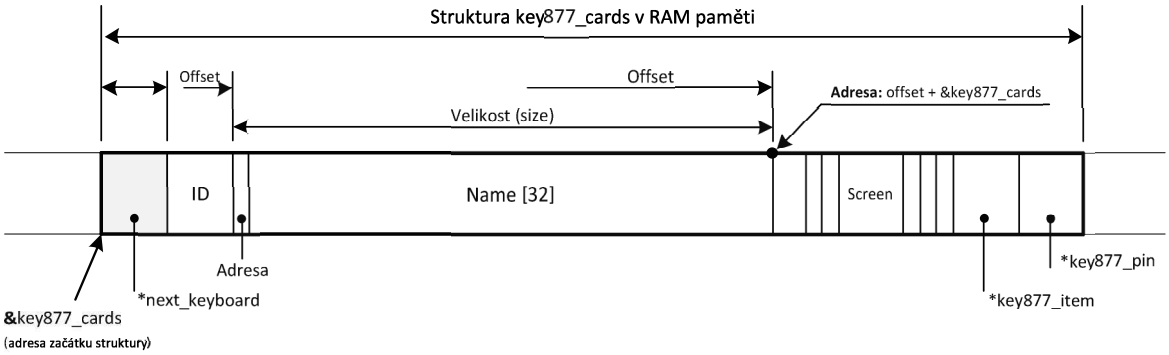
vyhledá v šabloně příslušící položka. Zkontroluje se, zda souhlasí velikost položky v šabloně a v hlavičce. Pokud nesouhlasí, nebo pokud položka v šabloně není nalezena, je ukazatel na data z paměti posunut o délku dat (posunut na další hlavičku) a chybná data jsou ignorována. Takový případ může nastat změnou firmware a konkrétně odstraněním položky ze struktury v novém firmware.

Dynamické struktury jsou uloženy v paměti stejným způsobem, jako hodnoty. Obsahují hlavičku s klíčem (opět unikátní pro konkrétní šablonu, kde je ukazatel na podstrukturu) a délkou, která zahrnuje celou novou strukturu. Data pak obsahují opět hlavičky a položky nové struktury, jak je zobrazeno na obrázku.

Před načítáním tohoto nového uzlu se musí nejdřív uzel vytvořit. Alokuje se paměť o velikosti zjištěné v šabloně nadřazené struktury, a naplní se defaultními hodnotami přidružené struktury. To zaručí, že i kdyby v paměti nebyla žádná platná data, bude struktura korektně naplněna. Po skončení procesu se struktura začne plnit stejnou funkcí, jak bylo popsáno.

5.1.2.3. Načítání hodnot do struktur

Na základě adresy struktury v paměti RAM (získaný alokační funkcí) a offsetu (získaného ze šablony pro konkrétní položku) je určena adresa, kam se položka nakopíruje. U každého typu položky probíhá kopírování odlišně. Pro číselný typ se přetypuje načtená hodnota do příslušného datového typu, a vloží se na danou adresu. U textových položek se kopírují jednotlivé byty. Pokud struktura ve firmware neobsahuje klíč (z důvodu změny firmware), položka je ignorovaná. Obsahuje-li naopak firmware víc položek ve struktuře než je přečteno z uložené konfigurace, položka se v dynamické struktuře nepřepíše a zůstane jí defaultní hodnota, kterou získala po alokaci.



Obrázek 5.6: Struktura ukládající se do paměti RAM

Na obrázku 5.6 je naznačena adresa ukazatele (báze) dynamické struktury „&key877_cards“. Každá dynamická struktura má v prvních čtyřech bytech ukazatele na další dynamickou strukturu svého typu (ukazatel *next_keyboard), svou adresu, název atd. Přečteme-li v datech hlavičku [key, size] = [3, 32] ze šablony zjistíme, že pod klíčem 3 se nachází název (položka „name“) o délce 32 a typu text. Provede se tedy takový příkaz:

```
strncpy(((char *)p_config + d_board[i].offset), ((char*)p_data_tmp), d_board[i].size - 1);
```

Kde: ukazatel p_config je v tomto případě &key877_cards, d_board je šablona pro klávesnici, p_data_tmp je ukazatel na načtenou datovou hodnotu z paměti.

Po prvním načtení konfigurace z paměti (např. po restartu) jsou struktury opět uložené do paměti pro případ, že během vypnutí jednotky došlo k update firmware.

Oproti prvnímu způsobu ukládání dat je u této varianty zapotřebí pouze vytvářet šablony pro nové struktury. Odpadá vytváření převodních automatů.

5.1.2.4. XML export

Pro komunikaci s okolím je zapotřebí zobrazit celou nakonfigurovanou strukturu tak, aby zachovávala veškeré vlastnosti (struktury jsou vnořené, mají předchůdce a následníka, obsahují data a identifikátory). Některá data rozložila do čitelné podoby (např. bitové hodnoty uložené ve struktuře ve stavových registrech rozložila na samostatné parametry).

Při takovém vyexportování není kladen požadavek na objem výstupních dat, ale na zobrazení věrné kopie struktur v paměti.

Pro tyto účely je vhodné použít značkovací jazyk XML (Extensive Markup Language). Jazyk XML vyvinut a standardizován konsorciem W3C nahrazuje textovou databázi a je v současné době velmi používaný pro přenos dat mezi objekty bez ohledu na platfomu. Existuje mnoho způsobů, jak lze s XML dokumentem pracovat – např. provádět nad XML dotazy pomocí XQuery, XPath nebo přiřazovat šablony na zobrazení pomocí XSL.

Velkým nedostatkem XML dokumentů může být větší nároky na velikost XML souboru, neboť veškerá data, včetně dat popisující strukturu jsou zapsaná textem. Obecným řešením nedostatku je komprimace dokumentu nejrůznějšími komprimačními algoritmy, které dokáží XML zredukovat až o 90%. Pro účely exportu konfigurace není tato nevýhoda rozhodující.

Exportování se provádí procházením struktur algoritmem procházení do hloubky. Nejedná-li se o výjimku, jmenují se názvy elementů (TAGů) stejně, jako názvy položek struktur. Při průchodu strukturou se opět procházejí i příslušná šablona, kde jsou nadefinovány názvy položek struktury a jakým způsobem budou data v XML souboru reprezentovány. Typy dat mohou být textové (XML_TXT), číselné bez znaménka podle typu (XML_U8, XML_U16, XML_U32), číselné se znaménkem (XML_S8, XML_S16, XML_S32) nebo s desetinnou čárkou (XML_FLOAT). Dále je nadefinováno velké množství dalších typů jako například XML_IP pro převod IP adresy z číselné hodnoty a její vypsání do tečkové notace.

Předposledním parametrem šablony struktur je osmibitová maska „param“ která filtruje položky z XML exportu. Každá bitová pozice tohoto parametru znamená profil pro export.

```
#define ALL           (0 << 0)
#define EPROM_SAVE    (1 << 0)                    // Zápis konfigurace
#define EXP_PIN_STATE (1 << 1)                    parametr = EPROM_SAVE | EXP_GLOBALS;
#define EXP_GLOBALS   (1 << 2)
```

Obrázek 5.7: Znázornění konfigurace exportovacího profilu

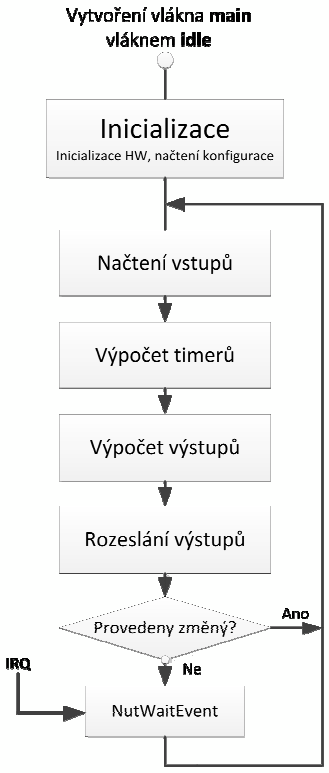
Na obrázku 5.7 je znázorněna konfigurace exportovacího profilu. Při volání exportu do XML se jako jeden z parametrů uvádí exportovací profil (**ALL** pro vyexportování všech položek, **EPROM_SAVE** určující položky, které mají být uloženy do externí paměti, **EXP_PIN_STATE** pro export pouze stavů hodnot, atd.). Parametr lze použít i pro filtraci položek, které se mají ukládat do externí paměti – například aktuální stavy vstupů a výstupů není potřeba do paměti ukládat, neboť jsou platné pouze pro daný okamžik. Touto filtrací prořezáváme data a necháváme takové hodnoty, které z daného exportu potřebujeme získat.

5.1.3. Matrix proces

Matrix se nazývá hlavní nekonečná smyčka vlákna main, kde se vykonává naprogramovaná logika v konfiguraci.

Každý průchod smyčkou je složen z pěti hlavních částí (obrázek 5.8):

1. Načtení vstupních hodnot karet a klávesnice
2. Vypočítání časovačů
3. Vypočítání výstupních hodnot karet
4. Rozeslání výsledků
5. Uspání



Obrázek 5.8: Matrix proces

Úložiště pro matrix je hlavní konfigurace popsaná v předchozí kapitole. Výsledky, zdrojová data a mezivýpočty jsou ukládány právě do hlavní konfigurace.

5.1.3.1. Načtení vstupních hodnot karet a klávesnice

Jako první je nutné načíst všechny vstupní stavy, jako základ pro výpočet výstupních hodnot. Vstupní stavy se získávají z karet vstupů, na které se zasílá požadavek o zaslání jejich vstupních hodnot popsaný v kapitole komunikace. Dalším zdrojem vstupních hodnot jsou klávesnice. U klávesnice není tato akce pouhým načtením vstupních stavů, ale i kontrolou synchronizace klávesnice s hlavní řídící jednotkou.

Zjišťování stavů klávesnice se provádí ve dvou krocích:

1. **Načtení a odeslání stavů pozorovaných hodnot**
   Do klávesnic lze navolit až 32 různých povelů, které budou provádět požadavky na delší vzdálenost. Uživatel, který ovládá klávesnici, musí mít zpětnou vazbu o tom, v jakém stavu se zařízení nachází. Například rozsvěcení a zhasínání světel schodišťovým přepínačem. V klávesnici viditelný stav přepínače jako On nemusí znamenat, že světlo ovládané přepínačem svítí. Z tohoto důvodu je do klávesnice zavedená zpětná vazba, která je zcela volitelná a nezávislá na zařízení, které je ovládáno.
   V tomto kroku jsou jištěny stavy všech pozorovaných objektů a následně odeslány jejich stavy do klávesnice.
2. **Synchronizace / Načtení stavů klávesnice**
   Tento krok může v jednom průchodu vykonávat jednu ze dvou úloh: Načtení stavů vstupů klávesnice nebo proces synchronizace (upload konfigurace z hlavní řídící jednotky do klávesnice).
   a) **Načtení vstupních stavů klávesnice**
      Z hlavní řídící jednotky je odeslaný požadavek konkrétní klávesnici, pro zaslání svých vstupních hodnot.
      Jelikož hlavní řídící jednotka rozhoduje o jakýchkoliv svých výstupů, tak rozhoduje o změnách ovládání výstupů. Klávesnice je takovým rozhodnutím změnit vstupní stavy podřízená a tyto rozhodnutí je povinna respektovat. Této vlastnosti se využívá

například v případě, kdy klávesnice změní stav, ve kterém má nakonfigurovaný typ PULSE o určité délce. Po uplynutí nastavené doby je příslušný signál vrácen zpět do své původní hodnoty. Takové rozhodnutí provádí hlavní řídící jednotka. Klávesnici je pouze zaslána informace o výsledných stavech. Pro snížení přenosů dat jsou stavy z hlavní řídící jednotky přenášeny ve zprávách požadujících o zaslání vstupních hodnot klávesnice.

Odpovědí klávesnice jsou opět nově provedené změny stavů od posledního požadavku. Součástí této zprávy jsou dva byty kontrolního součtu konfigurace klávesnice. Tyty dva byty se porovnávají s kontrolním součtem, vypočítaným v hlavní řídící jednotce po změně konfigurace nebo po spuštění systému (v procesu inicializace).

Nesouhlasí-li tyto dva byty, hlavní řídící jednotka přestane zjišťovat vstupní stavy této klávesnice a začne provádět synchronizaci.

b) Synchronizace

Jak již bylo vysvětleno, synchronizace se provádí v případě, kdy nesouhlasí kontrolní součet konfigurace v klávesnici a hlavní řídící jednotce. To mohlo být způsobeno restartem klávesnice, neboť veškerá konfigurace je v klávesnice z bezpečnostních důvodů uložena v RAM paměti, která se po odejmutí napájení ztrácí, nebo změnou nastavení konfigurace v řídící jednotce (provedené uživatelem pře webové rozhraní).

Konfigurace klávesnice se skládá ze třech částí:

1. Konfigurace přihlašovacích kódů (pinů)
2. Konfigurace textů a jejich oprávnění vůči pinům
3. Konfigurace textů zpětných vazeb

Při nesouhlasu kontrolních součtů se vždy začíná synchronizovat částí s největší prioritou. Vypsané části jsou seřazeny od největší priority. Po dokončení odeslání jedné části konfigurace se opět provede kontrola kontrolních součtů. Takto lze synchronizaci urychlit, byla-li ovlivněna uživatelem přes webové rozhraní (neboť změnou přihlašovacích kódů není nutné synchronizovat i veškeré texty).

Po dobu této synchronizace je nastaven příznak zakazující čekání po průchodu hlavní smyčkou matrix.

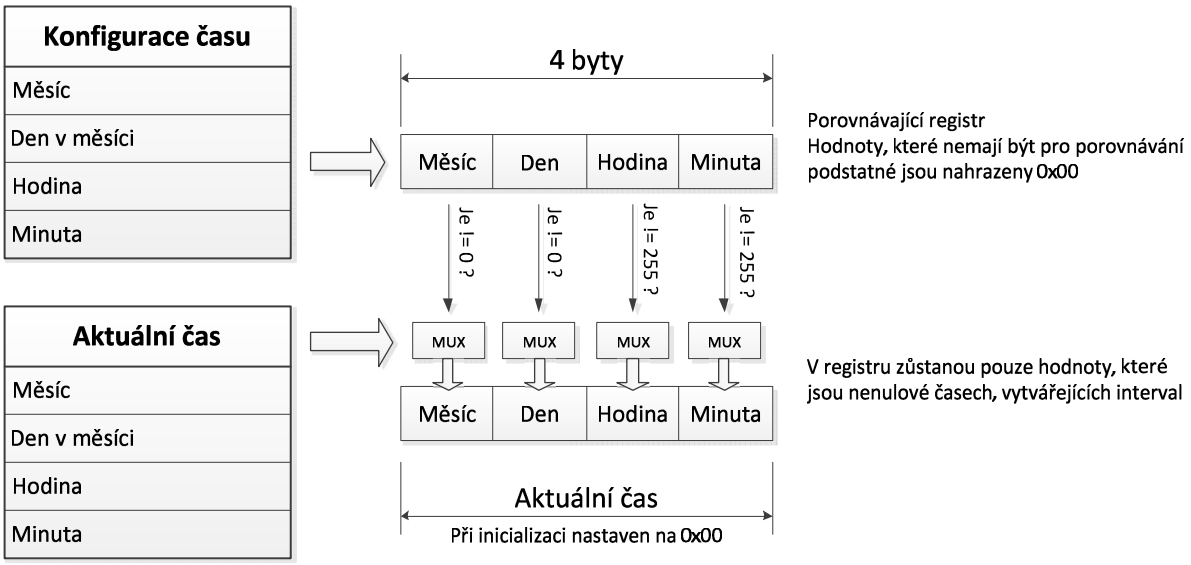
5.1.3.2. Vypočítání časovačů (timerů)

Časovače jsou pro hlavní proces vstupními jednotkami, které nastavují svůj výstup do logických hodnot podle aktuálního času získaného z RTC nebo webu. Do systému jsou přidány pro spínání / rozepínání obvodů v definované časy. Jako příklad jej lze využít společně v kombinaci vstupního zařízení zjišťující míru osvětlení pro rozsvěcení a zhasínání nočního osvětlení. Osvětlení se zapne v definovaný čas, nebo při nedostatku světla. Vypnutí je možné určit vždy, například v půlnoci.

V systému je možné nadefinovat libovolný počet časovačů. Každý časovač obsahuje jeden výstup, nabývající hodnotu log. 0 nebo log. 1.

Časovače lze nakonfigurovat od řádů minut až po měsíce v intervalech od jedné hodiny. Objektem, který nese nakonfigurované časové závislosti je struktura Konfigurace času. Ta v sobě nese měsíc (0 až 12), den v měsíci (0 až 31), hodinu (0 až 12 nebo 255) a minutu (0 až 59 nebo 255). Je-li nastavena hodnota 0 u měsíce nebo dnu v měsíci, je hodnota chápána jako bezvýznamná pro časovač (při

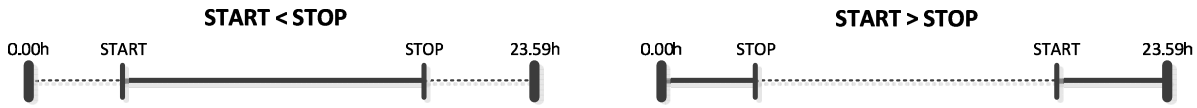
vyhodnocení má být ignorována). Pro hodinu a minutu je taková hodnota 255 (0xFF). Jelikož jsou všechny hodnoty struktury typu unsigned char (mají 1B), lze je převést na 4bytové slovo long, pomocí kterého bude snadněji realizováno porovnávání s aktuálním časem. Převod ze struktury konfigurace do registrů času jsou znázorněny obrázkem 5.9.



Obrázek 5.9: Výpočet časových registrů

Registr aktuálního času je vytvořený z hodnot vrácených RTC nebo webu. Pro účely porovnávání se převede do stejného formátu, který byl vytvořen pro nakonfigurovaný čas – do 4B slova long tak, že registr, se vytvoří registr aktuálního času obsahující samé nuly. Následně se do něho začnou kopírovat hodnoty aktuálního času, nemá-li být aktuální hodnota času ignorována - v tom případě zůstane v registru aktuálního času u hodnoty 0x00. Takto vytvořené registry lze snadno mezi sebou porovnávat.

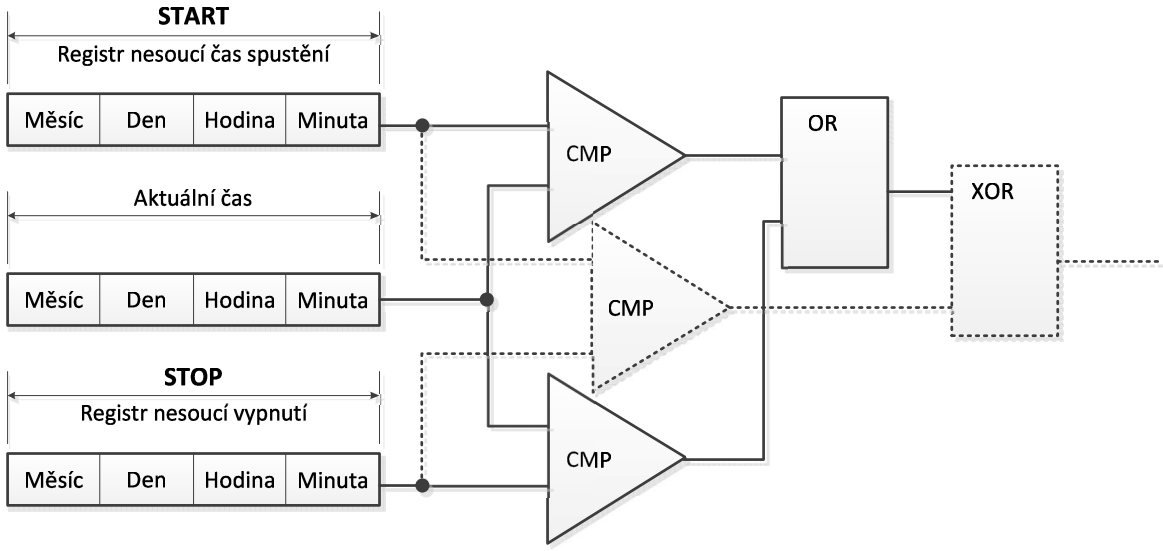
Jsou-li registry vytvořeny, je snadné rozeznat, zda se aktuální čas nachází mezi určenými hranicemi. Je důležité nezapomenout na případy, kdy má být výstup sepnutý přes půlnoc (obrázek 5.10 vpravo). Potom je hodnota spouštěcího registru větší, než hodnota vypínacího registru (jak zobrazuje obrázek 5.10).



Obrázek 5.10: Znázornění nastavení hranic času

Samotný princip porovnávání registrů je blokově znázorněn na obrázku 5.11. Část obrázku nakresleného tečkovaně opravuje případ, když určený interval prochází přes půlnoc.

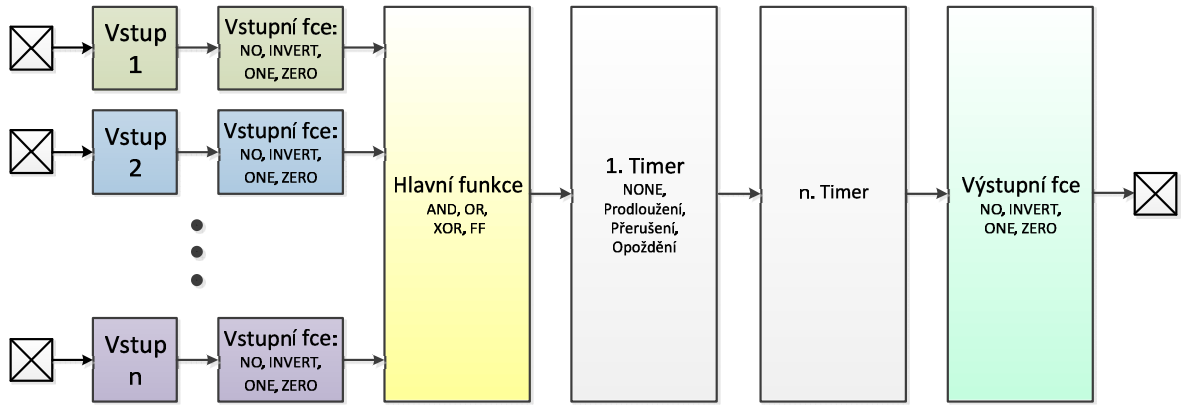
K získání výpočtu jsou tedy zapotřebí dvě časová porovnání, nebo-li 2 porovnání long čísla. Jelikož pracujeme na 32bitovém procesoru, je porovnávání vytvořeno jednou instrukcí.



Obrázek 5.11: Blokové schéma porovnání časů

5.1.3.3. Výpočet výstupních hodnot

Struktury výstupních karet obsahují předpisy, které vytváří logické funkce. Každý výstupní port je připojen k buňce, vytvářející programovatelný logický obvod znázorněný obrázkem 5.12. Vstupem tohoto obvodu může být libovolný počet hodnot reprezentované vstupními kartami, výsledkem z jiné výstupní karty nebo výstupem z virtuální karty. Hodnoty jsou reprezentovány adresami struktur a čísly pinů. Na této adrese v příslušné struktuře je vypočtena momentální logická hodnota.

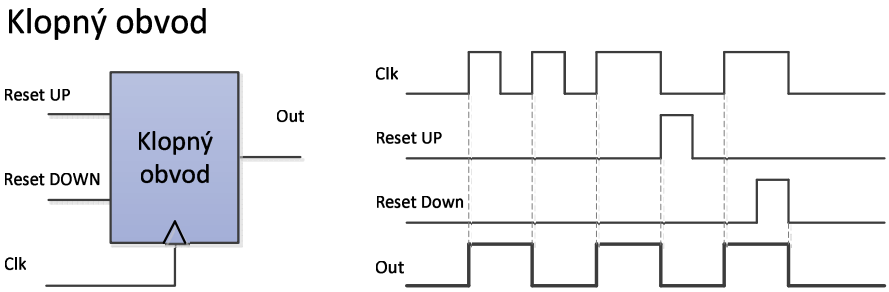


Obrázek 5.12: Blokové schéma zapojení výstupního pinu

Po vstupu do buňky může být hodnota invertovaná, nastavena do logické „1“ resp. „0“ (především se využívá, pokud na vstupu buňky není přiveden jiný člen – logická hodnota je pak volena na základě následujícího logického členu), nebo může být ponechána v nezměněné podobě. Na hodnoty se dále aplikuje jedna ze zvolené základní funkce AND, OR, XOR nebo klopný obvod FF.

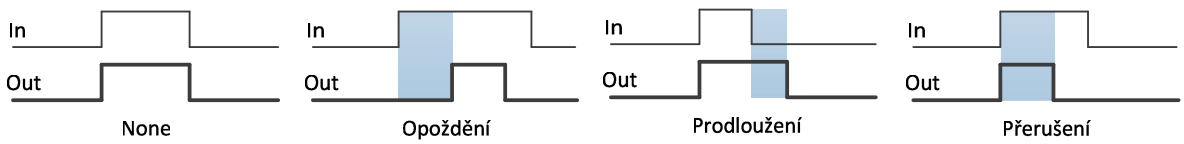
Klopný obvod (znázorněný na obrázku 5.13) je funkčně navržen pro realizaci programování objektů ovládající se dvěma tlačítky jako například žaluzie, jejichž zapojení je zobrazeno obrázkem 5.15. Žaluzie, jako objekt je ovládána dvěma signály – nahoru a dolu. Mimo těchto dvou funkcí tlačítek mají

ještě funkci resetu, kdy musí být obě v log. 1. Při resetu ztrácí žaluzie mezní body, které jsou určeny softwarově. Zapojení žaluzie je zobrazeno na konci této podkapitoly.



Obrázek 5.13: Popis klopného obvodu

Přivedení vypočítaného výsledku lze za pomoci časových bloků pozdržet o definovaný čas, prodloužit jeho hodnotu i po její změně nebo zaručit, že vypočítaná hodnota bude na výstupu přesně definovaný stav a poté se vrátí do původního stavu. Počet časových bloků není omezený a lze tak vytvářet například pulzy. Časové průběhy pospaných možností jsou zobrazeny obrázkem 5.14.

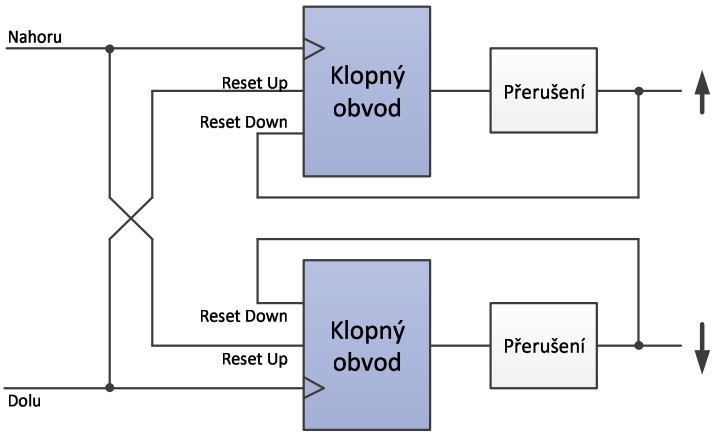


Obrázek 5.14: Časové průběhy konfigurovatelné časovači

Výsledná hodnota je nakonec přizpůsobena potřebě zařízení – tzn. znegovat resp. nastavit na konkrétní logickou hodnotu.

Z takových logických buněk je možné sestrojit nejrůznější sekvenční, i kombinační obvody včetně prodlužování nebo zkracování hran.

Pro vytvoření složitějších logických obvodů je možné výstup logické buňky použít jako vstup do dalšího logického bloku pomocí virtuální karty. Virtuální karta má všechny vlastnosti, které má karta výstupů s rozdílem, že vypočtené hodnoty nejsou odesílány kartám a nemá omezený počet výstupních pinů. Z programového hlediska se jedná o kartu výstupů s nultým indexem.



Obrázek 5.15: Blokové schéma zapojení žaluzií

Při výpočtu výstupních hodnot se pozoruje, zda výpočet změnil původní hodnotu vypočítanou v předchozím průchodu smyčky matrix. Pokud nastala změna, musí se výpočty opakovat. To je zaručeno nastavením příznaku okamžitého opakování smyčky matrix.

5.1.3.4. Rozesílání výstupů

Poslední akcí v matrixu je odeslání výsledných hodnot kartám výstupu (OUT876) a kartám výkonových relé (POWER876) popsané v kapitole 5.2. Popis komunikace.

5.1.3.5. Uspání

Při provádění každé části se zjišťuje, zda byl ovlivněn některý z výstupů. Pokud tomu tak je, výpočet není kompletní, a je zapotřebí okamžitě projít všechny části znovu, dokud nebude provedena žádná změna. Takové případy nastanou použitím virtuální karty, nebo vytvořením složitějších funkcí. Další událostí, která způsobí okamžité projití všech částí matrix procesu, jsou nesesynchronizované klávesnice – klávesnice, které si ještě nenačetly přihlašovací hesla a povely z procesoru.

Po provedení všech pěti částí matrixu bez požadavku na okamžité opakování se vlákno uspí funkcí NutWaitEvent na definovaný čas. Funkce NutWaitEvent je funkce operačního systému Nut/OS používaná pro synchronizaci vláken. Jedním z parametrů je čas, v našem případě maximální čas uspání matrix procesu a druhým parametrem je ukazatel na událost (handler), který způsobí probuzení vlákna, které bylo na funkci NutWaitEvent uspáno. V našem případě je handler svázán na změnu signálu IRQ (přerušení od vstupní karty). Tokovým způsobem je možné okamžitě reagovat na poslaný požadavek.

5.1.4. WWW server

Operační systém EtherNut obsahuje ve svém základu podporu síťové vrstvy. Nabízí podporu IP s jeho servisními protokoly [14]:

- **ICMP** – sloužící pro testování přítomnosti zařízení v síti a podporu chybových zpráv
- **ARP** – překlad IP adres na MAC adresy
- Podpora transportních protokolů:
  - **TCP/IP** – řízený datový proud zajišťující transportní službu se spojením (zabezpečený přenos dat)
  - **UDP/IP** – transportní služba bez spojení (nezabezpečený přenos dat)

- **PPP** – transportní protokol IP s podporou doplňkových služeb pro řízení přenosu

» Servisní protokoly vyšší úrovně:
- **DHCP** – automatická konfigurace IP adresy a masky našeho zařízení z DHCP serveru
- **DNS** – podpora jmenných serverů

» Aplikační protokoly:
- **HTTP** – protokol verze 1.0. Dovoluje velice jednoduše napsat http server, který pak poskytuje na své IP adrese možnost konfigurovat zařízení pomocí webové stránky. S tímto úzce souvisí CGI (Common Gateway Interface) skripty.

Po spuštění systému, ve fázi inicializace ethernetového zařízení se nastavuje síťová konfigurace, jako je IP adresa zařízení, maska sítě, výchozí brána a komunikační port, určené z hlavní konfigurace.

Dalším krokem inicializace je nastavení CGI skriptů a určení zdroje dat, odkud webový server přeposílá statická data (XSL šablony, CSS kaskádové styly, JS JavaScripty, IMG obrázky atd.).

V závěru inicializace se tvoří šest vláken obsluhující webové požadavky. Po vytvoření jsou tyto vlákna sami sebou uspané do doby, než přijde http požadavek. Systém je nastavený tak, aby dokázal v jeden okamžik obsloužit maximálně 6 spojení. Takový počet je dostatečný, neboť webový browser může navázat s jedním serverem maximálně čtyři spojení v jeden okamžik. Pokud by na server přistupovalo více prohlížečů současně, budou stránky v nejhorším případě načítány zpožděně (po vypršení time-outu jednoho požadavku).

5.1.4.1. HTTP server a CGI skripty

Podstatnou vlastností celé implementace http serveru v EtherNutu je možnost vytvořit stromovou strukturu adresářů a souborů složených z HTML stránek.

Jednotlivým adresářům lze přiřazovat hesla, o která je požádán http klient v případě přístupu na požadovanou stránku uloženou v heslem chráněném adresáři.

Mnohem důležitějším faktem je, že CGI skripty lze generovat dynamicky a tak vytvářet stránky s aktualizovanými údaji pro uživatele. [14]

Pro naše účely nás bude zajímat především zasílání požadavků na stránku společně s GET parametry a individuální tvorba XML souborů, ze kterých se v kombinaci s XSL šablonami vytvoří na uživatelském prohlížeči HTML stránka.

5.1.4.1.1. Předávání parametrů serveru

Jedním z nejrozšířenějších způsobů, jak přenést data od uživatele k serveru je pomocí parametrů GET, nebo POST. POST parametry jsou přenášeny v hlavičce požadavku (requestu). Protože jsou data zasílána právě v hlavičce a nejsou na první pohled viditelná, využívá se POST parametry vždy při přihlašování uživatele, kde je nutné uvést přihlašovací heslo.

Druhou metodou jsou GET parametry, které se přenáší v URL adrese požadavku (ta je do parametrů oddělená otazníkem a samotné parametry mezi sebou oddělené symbolem „&“) a jsou tak vždy čitelné. Oproti POST parametrům jsou omezeny normou RFC v počtu znaků, které nesmí přesáhnout. Další odlišností je chování při obnovení stránky. U GET parametrů se po znovunačtení stránky opět odešlou parametry a není-li tato vlastnost softwarově ošetřena, způsobí opětovné provedení akce. POST parametry se sice zachovají stejně, nicméně uživatel je předen informován, že je provedeno znovunačtení stránky s odesláním POST parametrů. Výhodou GET parametrů je snadné generování url odkazů, které mohou nést zároveň GET parametry.

Operační systém EtherNut již dokáže přečíst a rozparsovat GET parametry do podoby, se kterou se lépe pracuje. POST parametry z hlavičky přečíst nedokáže. Nicméně EtherNut je otevřený systém, a lze data z hlavičky vyparsovat. Pro naše účely si vystačíme s GET parametry.

Zmiňovaný problém GET parametrů a znovunačtení stránky vyřešíme odesíláním klíče při každém načtení XML stránky. Server společně s XML daty odešle klientovi klíč (který se s každým požadavkem inkrementuje). Odešle-li klient požadavek na změnu konfigurace, musí být součástí GET parametrů klíč ze stránky, která odeslala požadavek. Neshoduje-li se odeslaný klíč s posledním klíčem na straně řídící jednotky, jsou parametry pro změnu konfigurace ignorovány. Tento postup zabrání opakovanému provádění povelů při několikanásobném stisknutí F5.

5.1.4.1.2. Zpracování příchozích požadavků statické stránky

Přijde-li serveru požadavek o stránku a daná stránka nemá registrovaný ukazatel na CGI funkci, je přeposlána ze zdrojového úložiště (v našem případě z PATH0:/, což je SD karta připojená na SPI sběrnici) včetně vygenerované příslušné hlavičky. Pokud se stránka nenachází ani v úložišti, je odeslána pouze hlavička s chybovým příznakem 404 (Stránka nenalezena).

V našem systému používáme mimo jiné soubory XSL. Operační systém ethernut takový soubor nezná, a vytvoří mu obecnou hlavičku pro přenos textu. Což je chyba. Některé webové prohlížeče se pokusí data rozpoznat, a stránku korektně zobrazit – to ale není zaručené řešení.

Nápravou je upravení jádra operačního systému přidáním zmíněného typu souboru.

V souboru `/nut/pro/httpd.c` :

```
MIMETYPES mimeTypes[] = {
        {".txt",    "text/plain",                   NULL},
        {".html",   "text/html",                    NULL},
        {".shtml",  "text/html",                    NULL},
        {".asp",    "text/html",                    NULL},
        {".htm",    "text/html",                    NULL},
        {".gif",    "image/gif",                    NULL},
        {".jpg",    "image/jpeg",                   NULL},
        {".png",    "image/png",                    NULL},
        {".pdf",    "application/pdf",              NULL},
        {".js",     "application/x-javascript",     NULL},
        {".jar",    "application/x-java-archive",   NULL},
        {".css",    "text/css",                     NULL},
        {".xml",    "text/xml",                     NULL},
        {".xsl",    "text/xsl",                     NULL},
        {NULL,      NULL,                           NULL}
};
```

V tabulce jsou vypsány veškeré typy souborů, pro které je EtherNut schopný vytvořit hlavičku. Pro XSL soubor jsme byli nuceni přidat předposlední (tučně napsaný) řádek definující typ souboru jako „text/xsl“.

5.1.4.1.3. Zpracování příchozích CGI požadavků

CGI skripty nám nejen dovolují zasílat serveru příkazy, ale i generovat stránky. Dynamické generování stránek (jako umožňuje například PHP) je pro takový mikroprocesor časově náročné. Musí se načíst stránka (která může mít několik KB) a provést veškeré náhrady zástupných znaků za procesorem určené proměnné, manipulovat se stránkou (podmínky) nebo generovat stránku (cykly). To vyžaduje napsání poměrně složitého parseru včetně překladače.

Jiným řešením, které se v našem případě zcela nabízí je využít technologie XSL šablon popsané v kapitole 6. Webové rozhraní. Procesor nám na zaslané požadavky neodešle vygenerovanou XHTML stránku, ale vygeneruje XML export popsaný v kapitole 5.1.2. Konfigurace a pomocí statické XSL šablony vygeneruje prohlížeč XHTML stránku. Generování probíhá na straně klienta, a nevytěžuje tak procesor. XSL šablonami navíc získáme velkou flexibilitu ve změnách stránek, možnost vytvářet statické konfigurační XML soubory (které budou lehce editovatelné), spojovat XML soubory, řadit velké množství dat a další operace, které by se obtížně programovala v procesoru.

CGI skripty se tedy využívají pouze pro generování XML exportů na základě zaslaných GET parametrů. GET parametry navíc mění strukturu konfigurace – ať změnou funkcí v administračním rozhraní, nebo změnami stavů karet v uživatelském rozhraní. Po dobu, kdy se vykonávají operace s konfigurací, pracuje se se synchronizacemi vláken, neboť se musí vyloučit přístup main vlákna do matrix processu.

Přihlašování do stránek je řešeno autorizací provádějící webovým prohlížečem. V systému je označen kořenový adresář, který má být přístupný pouze pod správně vyplněným jménem uživatele a heslem. Po přístupu na tyto stránky / podstránky server odešle prohlížeči požadavek 401 – požadavek o http autorizaci (Authorization Required). V další probíhající komunikaci, po úspěšném přihlášení, zasílá prohlížeč zakódované vyplněné přihlašovací jméno a heslo zadané uživatelem v hlavičce http requestů.

```
Authorization: Basic c2F6ZWNza2E6cG9kdWxzaQ==
```

5.1.5. Terminál

Terminál (debugovací linka) je připojen k procesoru prostřednictvím sériové linky RS-232. Je určena především pro ladění firmware. Může však posloužit i pro základní ovládání hlavní řídící jednotky.

Po připojení k sériové lince a odesláním symbolu „?“ se vyvolá nápověda zobrazená na obrázku 5.16.

Ve vrchní části je zobrazena verze firmwaru a síťové nastavení hlavní řídící jednotky. Dále jsou vypsány příkazy sloužící jednak pro základní ovládání jednotky tak pro výpis stavu řídící jednotky.

```
-------------------------------------------
HJ-HS, build 750, Apr 26 2011, 19:12:51
-------------------------------------------
IP Addr:  10.112.2.240
IP Mask:  255.255.255.0
IP Gate:  10.112.2.1
DNS1:     8.8.8.8
DNS2:     0.0.0.0
-------------------------------------------
? ..... Help, tato napoveda
t ..... Print Thread List
1 ..... SPI Read Addr 0x0000, Size 32 bytes
2 ..... MMC Read
3 ..... MMC Dir
4 ..... Print SPI Reg
s ..... Save config to SPI
l ..... Load config from SPI
x ..... Print XML Config
e ..... Erase first block on SPI
5 ..... Test MMC file
6 ..... Test MMC file
h ..... Print Heap
f ..... FlipFlop
r ..... Reset
7 ..... Make Keybard checksum
k ..... Keyboard Transfer
k100 ... Keyboard Transfer
gt ..... Get Time
nt ..... Get Time from SNTP server
-------------------------------------------
>
```

Obrázek 5.16: Terminál

» **t (výpis vláken)** – vypíše vlákna, která v systému existují. Ve výpisu je pro každé vlákno uvedena priorita, stav vlákna (Run – právě běžící, Ready – vlákno připravené pro běh, Sleep –

uspané vlákno), velikost alokované paměti pro každé vlákno a příznak přetečení velikosti alokované paměti (přetečení nikdy nesmí nastat – parametr je důležitý pro ladění aplikace).

- **1 (výpis FLASH paměti)** – vypíše 32 bytů FLASH paměti od adresy 0x0000. Výpisem se zjišťuje uložení konfigurace v paměti.
- **3 (MMC adresářová struktura)** – vypíše adresářovou root strukturu MMC/SD karty.
- **s (uložení konfigurace do FLASH paměti)** – příkaz pro uložení konfigurace do FLASH paměti.
- **l (načtení konfigurace z FLASH paměti)** – ruční vyvolání akce načtení konfigurace z FLASH paměti (neobsahuje-li paměť platnou konfiguraci, bude vytvořená defaultní).
- **x (výpis XML)** – na terminál vypíše XML strukturu aktuální konfigurace.
- **e (vymazání prvého bloku paměti FLASH)** – vymazáním prvního bloku FLASH paměti zneplatníme konfiguraci uloženou ve FLASH paměti.
- **h (velikost volné paměti)** – vypíše volnou RAM paměť.
- **gt (vrať čas)** – vypíše aktuální reálný čas v systému
- **nt (načti čas ze serveru)** – příkaz načte do systému aktuální čas získaný z internetu
- **r (reset)** – ruční reset systému

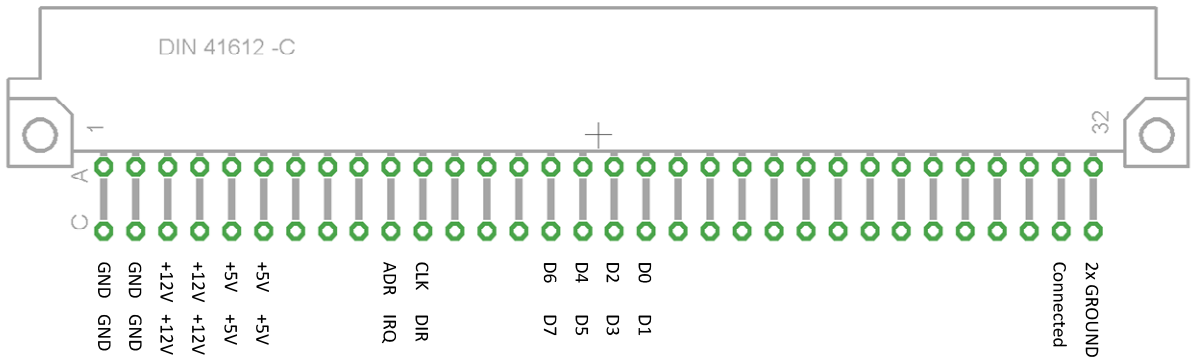
5.2. Popis komunikace

Komunikace mezi periferiemi probíhá po paralelní a sériové sběrnici podle typu zařízení. Komunikaci zahajuje vždy řídící jednotka (jedná se o komunikaci master-slave), která cyklicky po daných intervalech obesílá periferie a zjišťuje stav. Pro oživování stavů karet je cyklické procházení nezbytné alespoň v 500 ms intervalech. Tak vzniká latence mezi vyvoláním požadavku a provedení vypočítané akce. Latenci lze zkrátit pomocí signálu přerušení IRQ (u karet vstupů na paralelní sběrnici), softwarově v řídící jednotce pomocí zachytávání změn na výstupních kartách a zrychlením komunikace pro aktivní periferii (např. pro klávesnici, která je ve stavu konfigurace).

5.2.1. Komunikace po paralelní sběrnici

Paralelní sběrnice je vyvedena jako jediný interface na zadní straně hlavní řídící desky prostřednictvím konektoru DIN41612 (obrázek 5.17). Umístění je nutné pro zasunutí karty do Racku obsahující paralelní sběrnici. Hlavní řídící deska je detailně popsaná v kapitole 8. MainBoard.

Sběrnice přenáší mimo samotných dat i napájení paralelních karet +12V, +5V a zem. Další piny jsou využity pro data - 8bitů a řídící signály: příznak adresy (ADR), řízení určující směr komunikace (DIR), hodinový signál (CLK) a signál přerušení (IRQ). Ostatní vodiče sběrnice jsou nevyužité.



Obrázek 5.17: Schéma zapojeni konektoru DIN41612

Hodinový signál (CLK) je generovaný hlavní řídící jednotkou (Masterem) s periodou T = 130 µs. Data jsou vždy platná s náběžnou hranou hodinového signálu.

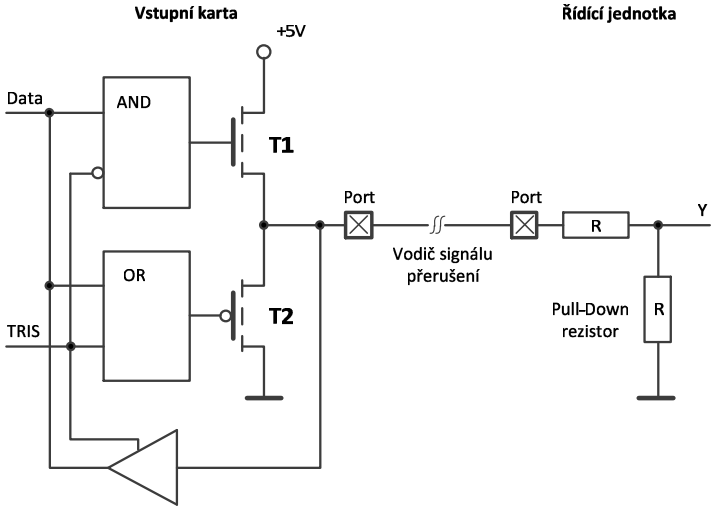
Prostřednictvím paralelní sběrnice probíhá komunikace mezi kartami vstupů (IN876) a kartami výstupů (OUT876). Komunikaci vždy zahajuje řídící deska (Master).

Rozeznávají se dva druhy zpráv – čtení z karty a zápis do karet.

5.2.1.1. Čtení z karty

Signál přerušení (IRQ) na paralelní sběrnici slouží k urychlení vstupních odezev vstupů na výstup. Na straně řídící karty je připojený Pull-Down rezistor. Všechny připojené karty včetně řídící jednotky jsou připojené na pin jako vstupní. V klidovém stavu je na vodiči logická nula. Při změně vstupu na vstupní kartě se brána přepne ze vstupní na výstupní a na vodič vystaví log. 1. Tím řídící jednotka detekuje událost a následně zahájí čtení ze vstupních karet.

Na obrázku 5.18 je naznačen princip zapojení přerušení. V levé části obrázku je výstupní brána mikrokontroléru PIC876, v pravé části schématu je vstup do řídící jednotky přes Pull-Down rezistor. Z tabulky popisující chování obvodu je patrné, že pro nastavení výstupní brány do vysoké impedance (do stavu vstupu) musí být tranzistory T1 a T2 zavřené – TRIS nastavený v log.1. V tento okamžik je na vodiči přerušení logická 0 díky rezistoru Pull-Down.



Tabulka chování

TRIS	Data	T1	T2	Y
0	**0**	Z	O	**0**
0	**1**	O	Z	**1**
1	0	Z	Z	Z
1	1	Z	Z	Z

Z u T1, T2 tranzistor uzavřen
O tranzistor otevřen
Z u Y výstup ve vysoké impendaci (chová se jako vstup)

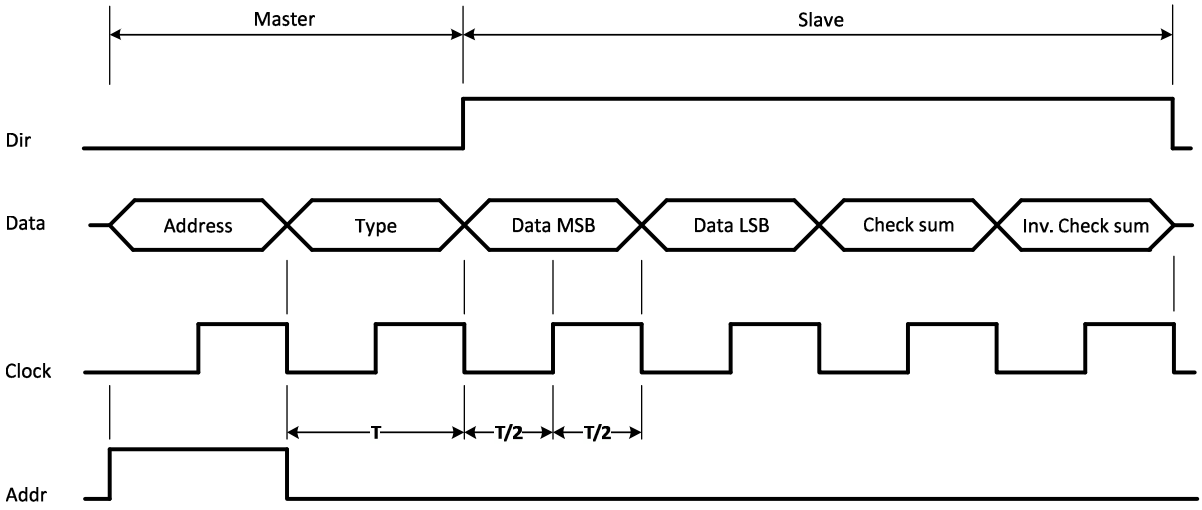
Obrázek 5.18: Schéma funkce přerušení

Čtení karet probíhá cyklicky přes všechny vstupní karty připojené na paralelní sběrnici od nejnižší adresy. Každá karta má v paměti EPROM uloženou jednobytovou adresu v rozsahu 1 až 254 (adresa 0 je určena pro hlavní řídící jednotku, adresa 255 zůstává pro další účely nevyžita) a identifikátor typu karty (paralelní vstupní karta má identifikátor 0x01).

Komunikaci zahajuje nastavením signálu Direction (DIR), čímž se karty nastaví do režimu čtení. Signál adresy (ADR) odlišuje adresový byte od ostatních dat. Po bytu adresy (Address) následuje určení druhu periferie (Type), kterému je zpráva adresovaná. Po odeslání těchto dvou bytů se řídící jednotka přepne do stavu naslouchání a čeká na odpověď zaadresované karty.

Karta odpoví 2 byty dat (Data MSB a Data LSB) obsahujících stavy vstupů. Následuje kontrolní součet paketu celého paketu (Check sum) a nakonec jeho inverze (Inv. Check sum). Přijímací strana (hlavní řídící jednotka) pomocí kontrolního součtu rozezná, zda se jedná o správně přijatý paket či nikoli. Při nesprávném kontrolním součtu, nebo nesouhlasu kontrolního součtu a inverzního kontrolního součtu je zpráva ignorována.

Popsaná komunikace je znázorněna na obrázku 5.19.



Obrázek 5.19: Průběh čtení z paralelní karty

Kontrolní součet zprávy je vypočítaný pomocí funkce XOR jednotlivých bytů zprávy o základu $0x55_h$.

Musí tedy platit vztahy:

```
(CheckSum)^(Address)^(Type)^(Data MSB)^(Data LSB) = 0x55
(CheckSum)^(Inv.CheckSum) = 0xFF
```

5.2.1.2. Zápis do karet

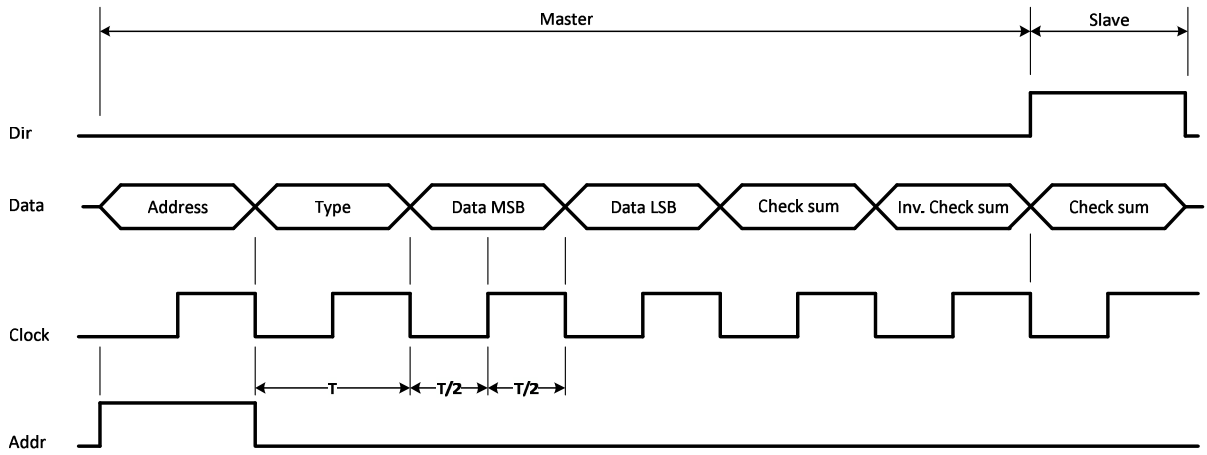
Komunikace s výstupními kartami je obdobná, jak čtení z karet.

Komunikaci zahajuje hlavní řídící jednotka vysláním adresy (Address) a typu zařízení (Type). Dále se odesílají dva byty dat (Data MSB a Data LSB). Nakonec hlavní jednotka zašle kontrolní součet (Check

sum) a inverzní součet (Inv. Check sum) vypočítaný stejným způsobem, jako u karet vstupů. Potom se přepne do stavu poslouchání a čeká na opověď karty signálem (Dir).

Karta přijme zprávu a nezávisle na hlavní jednotce vypočítá kontrolní součet vč. inverzního kontrolního součtu. Souhlasí-li vypočítaný a přijatý kontrolní součet včetně inverzního kontrolního součtu, je zpráva považovaná za platnou. Jako informaci o přijetí platné zprávy karta odpoví odesláním kontrolního součtu (Check sum).

Časový průběh komunikace je zobrazen na obrázku 5.20.



Obrázek 5.20: Průběh zápisu do paralelní karty

Hlavní řídící jednotka přijímající kontrolní součet (Check sum) využívá přijetí součtu jen jako zpětnou vazbu mezi odesláním požadavku a vykonáním operace. U karty vstupů je zpětná vazba realizována přijetím stavů vstupů.

Pomocí těchto informací můžeme vyhodnocovat dostupnost zařízení připojených na sběrnici. Každá periferie má v řídící jednotce registr neplatných pokusů komunikace. Při nezdařeném pokusu o komunikaci mezi hlavní jednotkou a kartou je registr zvětšen o jedničku, dokud nedojde k jeho naplnění. Po úspěšně proběhlé komunikaci s kartou je registr nulován. Překročením hodnoty registru nad určitou hranici můžeme prohlásit zařízení za nedosažitelné.

U paralelních karet je hranice pro nedosažitelnost zařízení defaultně nastavena na 10 pokusů. Informace je pak zobrazena v uživatelském nastavení systému, jako nedostupné karty.

5.2.2. Komunikace po sériové lince

Paralelní sběrnice je vhodná pro komunikaci na vzdálenosti několik centimetrů. Pro ovládání zařízení umístěných ve větších vzdálenostech od řídící jednotky je karta vybavena dvěma rozhraními RS-485. Řídící jednotka je schopná mezi těmito rozhraními přepínat a vyhledávat nakonfigurované karty. Při nalezení karty na druhém rozhraní, než byla původně připojena, se řídící jednotka přizpůsobí a kartu již adresuje z druhého rozhraní.

V současné době se sériová linka využívá pro komunikaci s karty výkonových relé (POWER876) a klávesnic komunikujících pomocí protokolu SLIP (IP protokol po sériové lince) [15].

5.2.2.1. Struktura zprávy

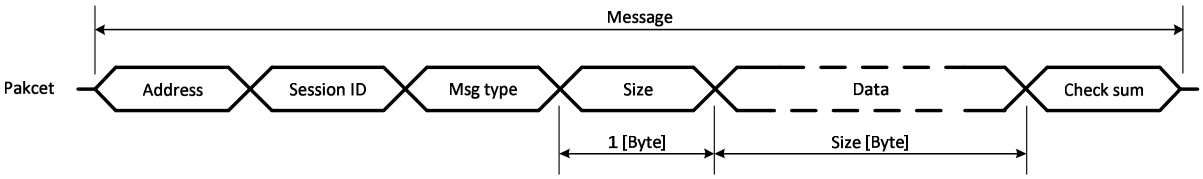
Každé zařízení připojené na sériovou sběrnici naslouchá veškerou komunikaci probíhající na sběrnici.

Při přijetí zprávy zařízení nejdříve testuje, zda je zpráva určena jí a zda souhlasí kontrolní součet, který zpráva obsahuje. Pokud zpráva není adresována dané kartě nebo nesouhlasí kontrolní součet, bude zpráva ignorována. V opačném případě se pokračuje ve vykonávání akcí, určené typem zprávy.

Zprávy posílající se po sběrnici se skládají z těchto položek:

- **adresa zařízení (Address)** – určuje, pro které zařízení je zpráva určena (velikost je 1 byte, kde 0x00 je vyhrazeno pro adresu serveru a 0xFF je rezervováno)
- **identifikátor (Session ID)** – dokáže odhalit prohození zpráv. Při rozesílání zpráv nastaví server tento byte na hodnotu v rozsahu 0 až 255. V odpovědi očekává shodný identifikátor. V případě neshody je paket identifikovaný jako zastaralý.
- **typ zprávy (Msg type)** – definuje typ zprávy. Typy zpráv budou popsány níž.
- **délka zprávy (Size)** – určuje, kolik nese paket datových bytů (maximálně může vést 255 bytů).
- **Data (Data)** – nese data paketu. Význam dat je dán typem dat. Počet přenášených bytů je dán délkou zprávy.
- **kontrolní součet (Check sum)** – pro zjištění, zda paket není poškozen, nebo zda se nejedná o
- naindukované byty vyskytující se bezi komunikacemi na sběrnici.
  Stejně, jako u paralelní komunikace je kontrolní součet vypočítán pomocí funkce XOR jednotlivých bytů zprávy se základem $0x55_h$.

Obecný rámec zprávy je znázorněn na obrázku 5.21.



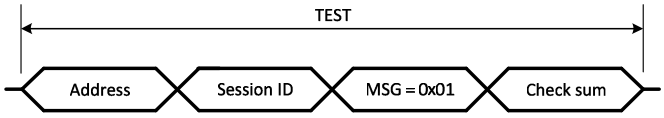
Obrázek 5.21: Obecný formát zprávy sériové linky

5.2.2.2. Typy zpráv

Karty rozlišují několik druhů zpráv lišících se délkou, typem a druhem dat. Každé zařízení se podle příchozí zprávy nastaví do určitého stavu a případně odešle odpověď.

5.2.2.2.1. TEST

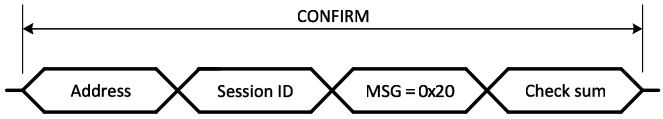
Testovací zpráva (obrázek 5.22) – zpráva tohoto typu testuje, zda je zařízení přítomné a zda odpovídá na požadavky. Periferie, která tuto zprávu obdržela, musí odpovědět zprávou **CONFIRM**. Zprávu využívá každé zařízení na sériové sběrnici. Jestliže hlavní řídící jednotka nepřijme zprávu CONFIRM, inkrementuje registr neúspěšných pokusů. Na základě těchto registrů se vyhodnotí, zda je karta připojená a řádně pracuje, či nikoli.



Obrázek 5.22: Zpráva TEST

5.2.2.2.2. CONFIRM

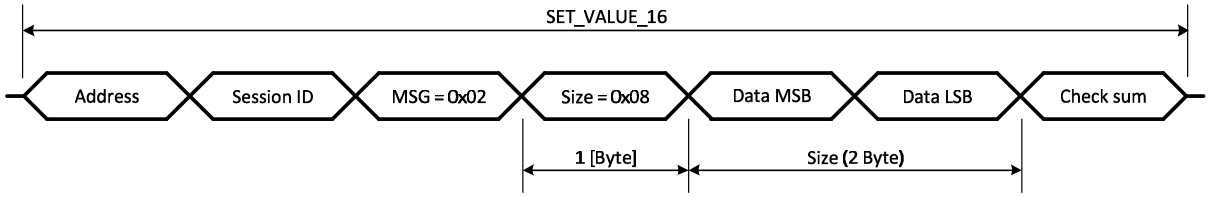
Obecné potvrzení požadavku. Potvrzení se odesílá na hlavní procesorovou jednotku. Zpráva je znázorněna na obrázku 5.23.



Obrázek 5.23: Zpráva CONFIRM

5.2.2.2.3. SET_VALUE_16

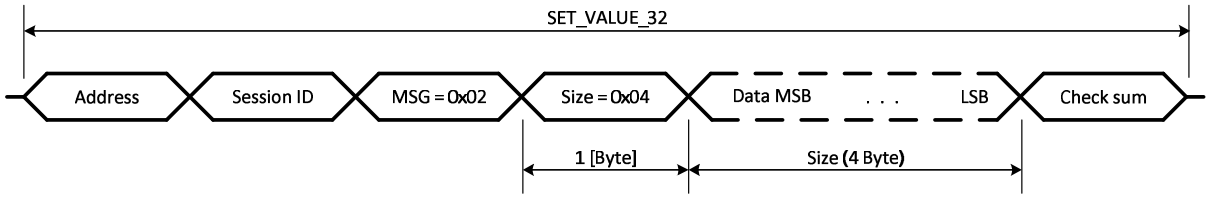
Zpráva, zobrazená obrázkem 5.24, odesílající hlavní řídící jednotka obsahující registr aktuálního stavu výstupu. Využívá karta výkonových relé (POWER876).



Obrázek 5.24: Zpráva SET_VALUE_16

5.2.2.2.4. SET_VALUE_32

Zpráva generována hlavní řídící jednotkou odesílající 32bitovou hodnotu. Využívá např. klávesnice k nastavování pozorovaných stavů. Znázornění zprávy je na obrázku 5.25.

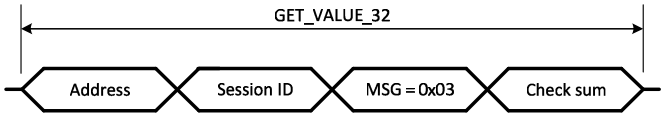


Obrázek 5.25: Zpráva SET_VALUE_32

5.2.2.2.5. GET_VALUE_32

Požadavek hlavní řídící jednotky, znázorněný obrázkem 5.26, pro získání 32bitové hodnoty z cílové karty.

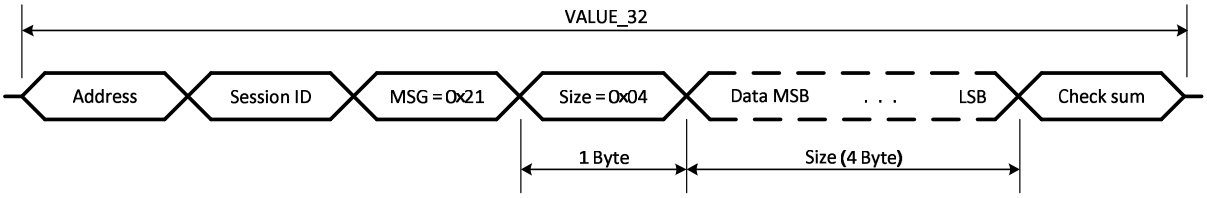
Master očekává vrácení zprávy **VALUE_32**.



Obrázek 5.26: Zpráva GET_VALUE_32

5.2.2.2.6. VALUE_32

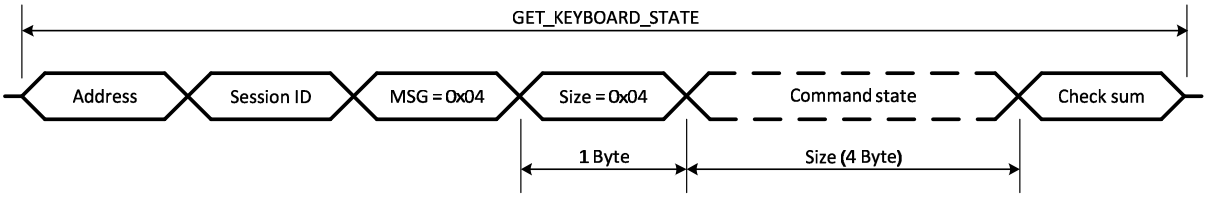
Odpověď karty na požadavek o zaslání 32bitové hodnoty **GET_VALUE_32**. Odpověď je znázorněna na obrázku 5.27.



Obrázek 5.27: Zpráva VALUE_32

5.2.2.2.7. GET_KEYBOARD_STATE

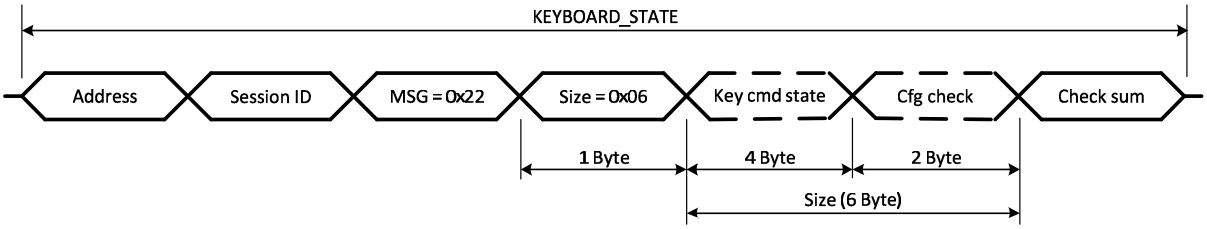
Požadavek pro klávesnici (zobrazen obrázkem 5.28), o zaslání registru akcí, které má řídící karta provést. Zpráva obsahuje aktuálně nastavené hodnoty registru v hlavní procesorové kartě (Command state). Po přijetí si klávesnice tyto data přepíše a odešle hlavní řídící jednotce aktualizovaná data (1 bite = 1 příkaz z klávesnice) a zašle aktuální stav pomocí zprávy **KEYBOARD_STATE**.



Obrázek 5.28: Zpráva GET_KEYBOARD_STATE

5.2.2.2.8. KEYBOARD_STATE

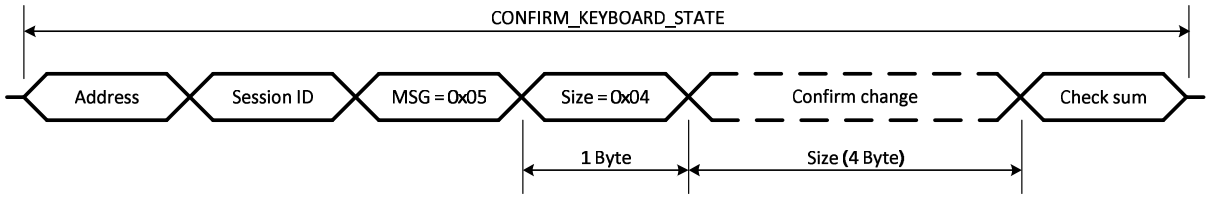
Odpověď klávesnice na zprávu **GET_KEYBOARD_STATE**. Zpráva (obrázek 5.29), obsahuje registr akcí (Key cmd state), které má řídící provést. Dále obsahuje 2 Byty kontrolního součtu nastavení klávesnice, podle kterých vyhodnocuje hlavní jednotka aktualizace klávesnice. Nesouhlasí-li tento kontrolní součet s vypočítaným kontrolním součtem (např. z důvodu restartu klávesnice, nebo změny konfigurace), spustí se proces aktualizace.



Obrázek 5.29: Zpráva KEYBOARD_STATE

5.2.2.2.9. CONFIRM_KEYBOARD_STATE

Vynulování interního stavového registru klávesnice. Nulování požadovaného bitu je provedeno nastavením hodnoty do log. 1 čtyřbytovém slově (Confirm change). Zpráva je znázorněna obrázkem 5.30. Řídící jednotka očekává vrácení zprávy **CONFIRM**.

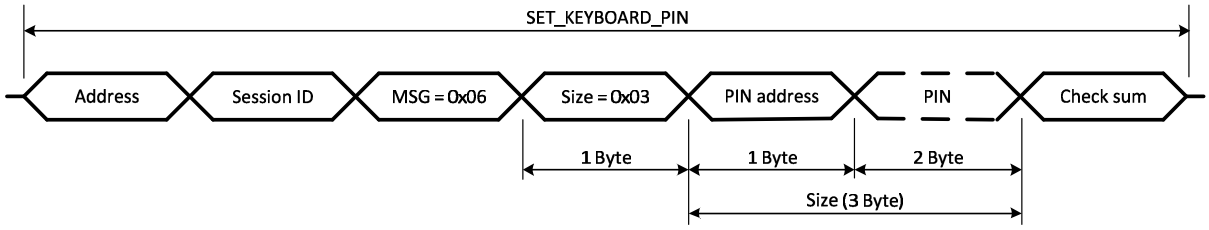


Obrázek 5.30: Zpráva CONFIRM_KEYBOARD_STATE

5.2.2.2.10. SET_KEYBOARD_PIN

Zpráva (zobrazená obrázkem 5.31) obsahující právě jeden přístupový kód klávesnice. Pozice přístupového kódu (0 až 9) je určena bytem „PIN address“. Délka přístupového kódu (PIN) jsou 2 byty.

Jako odpověď je očekávána zpráva **CONFIRM**.

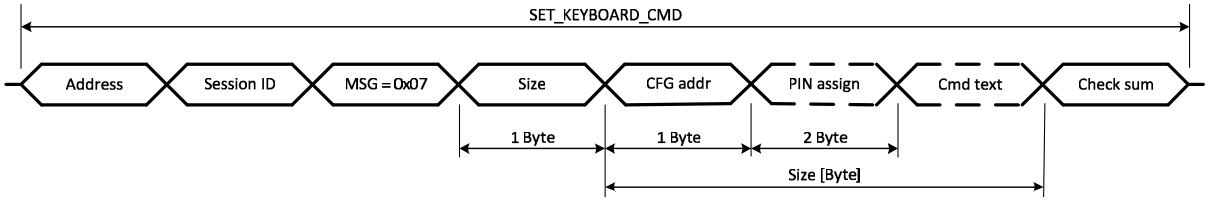


Obrázek 5.31: Zpráva SET_KEYBOARD_PIN

5.2.2.2.11. SET_KEYBOARD_CMD

Zpráva obsahující právě jeden řádek textu akce (Cmd text) o maximálně 16ti znacích do klávesnice včetně oprávnění (PIN assign), který pin může tuto volbu využít. Oprávnění jsou reprezentována bitovou maskou, kde bit odpovídá jednomu PINu. CFG addr určuje pozici textu (0 až 31), která je posílána. Struktura zprávy je znázorněna obrázkem 5.32.

Odpovědí je očekávána zpráva **CONFIRM**.

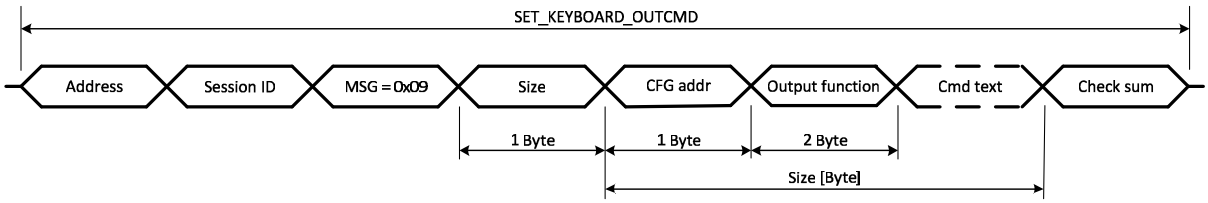


Obrázek 5.32: Zpráva SET_KEYBOARD_CMD

5.2.2.2.12. SET_KEYBOARD_OUTCMD

Zpráva (obrázek 5.33), obsahující texty informující o provedené akci (Cmd text). Tyto texty jsou zobrazeny pod jednotlivými příkazy. Vytvářejí zpětnou vazbu mezi požadavkem a provedením akce. Output function definuje popisek zobrazující se na displeji (Open/Close nebo On/Off) . CFG addr určuje pozici textu (0 až 31), která se vyskytuje ve zprávě.

Odpovědí je očekávána zpráva **CONFIRM**.



Obrázek 5.33: Zpráva SET_KEYBOARD_OUTCMD

Přehled všech typů zpráv sériové linky společně s jejich parametry je shrnutý v tabulce 5.2.

Tabulka 5.2
Přehled typů zpráv sériové komunikace

Typ zprávy	ID	Délka [Byte]	Odesílá		Odpověď
			Master	Slave	
TEST	0x01	**4**	x		CONFIRM
CONFIRM	0x20	**4**		x	-
SET_VALUE_16	0x08	**7**	x		-
SET_VALUE_32	0x02	**9**	x		-
GET_VALUE_32	0x03	**4**	x		VALUE_32
VALUE_32	0x21	**9**		x	-
GET_KEYBOARD_STATE	0x04	**9**	x		KEYBOARD_STATE
KEYBOARD_STATE	0x22	**11**		x	-
CONFIRM_KEYBOARD_STATE	0x05	**9**	x		CONFIRM
SET_KEYBOARD_PIN	0x06	**8**	x		CONFIRM
SET_KEYBOARD_CMD	0x07	**8 až 24**	x		CONFIRM
SET_KEYBOARD_OUTCMD	0x09	**7 až 38**	x		CONFIRM

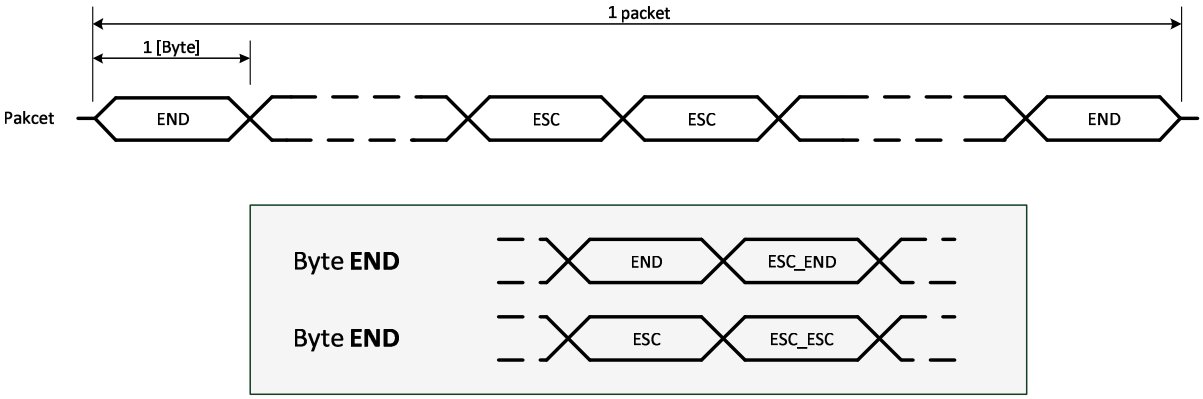
5.2.2.3. SLIP protokol

Pro posílání správ po sériové lince je použit protokol SLIP (IP protokol po sériové lince) popsaný v RFC 1055 [15].

Protokol rozsekává komunikaci na zprávy (pakety) funkčním bytem „0xC0“ (END). Aby nedošlo k rozložení zprávy na dvě (či více), nesmí se již tento znak vyskytnout nikde v těle zprávy. Toho docílíme nahrazením znaku posloupností dvou funkčních bytů, které znak zakódují.

Jak je patrné z obrázku 5.34, pro zakódování bytu 0xC0 (který je shodou okolností funkční byt END) musíme použít sekvenci escape byte „0xDB“ (ESC) a zástupný byte „0xDC“ (ESC_END).

Obdobně pro zakódování funkčního bytu ESC musíme uvést kombinace „0xDB“ (ESC) a zástupný byte „0xDD“ (ESC_ESC).



Obrázek 5.34: SLIP protokol

Na obrázku 5.35 jsou uvedeny zdrojové kódy pro kódování a dekódování zprávy z/do SLIP paketu [15].

```
void sendPacket(unsigned char *p, int len){
  uart1_PutChar(END);
  while(len--){
    switch(*p){
      case END:
        uart1_PutChar(ESC);
        uart1_PutChar(ESC_END);
      break;

      case ESC:
        uart1_PutChar(ESC);
        uart1_PutChar(ESC_ESC);
      break;

      default:
        uart1_PutChar(*p);
    }
    p++;
  }
  uart1_PutChar(END);
}
```

```
u_short recvPacket(u_char *p, int len, u_char c){
  u_short tmp_len;
  static u_short received = 0;
  static char last_state = 0;

  switch(c){
    case END:
      tmp_len = received; received = 0;
      return tmp_len;
    break;
    case ESC:
      last_state = c;
      return 0;
    break;
  }

  if(last_state == (char)ESC) {
    switch(c){
      case ESC_END:    c = END;       break;
      case ESC_ESC:    c = ESC;       break;
    }
  }
  last_state = 0;

  if(received < len) p[received++] = c;

  return 0;
}
```

Obrázek 5.35: Ukázka zdrojového kódu vytvářející a přijímající SLIP paket

Kódování do SLIP [15] paketu lze provést jednoduchým automatem, který nahrazuje určité byty ESC. sekvencemi.

Dekódování je o něco složitější, neboť je zapotřebí zohledňovat předešlí byte. Funkce se volá při každém příchozím byte na sériové sběrnici. Jestliže je přijat funkční byte END, vrací funkce délku, kterou načetla (jinak vrací 0x00), což v programu obstarávající příchod paketu signalizuje přijetí nového paketu.

Zjednodušený příklad obsluhy přijímání paketu je znázorněn v následujícím kódu:

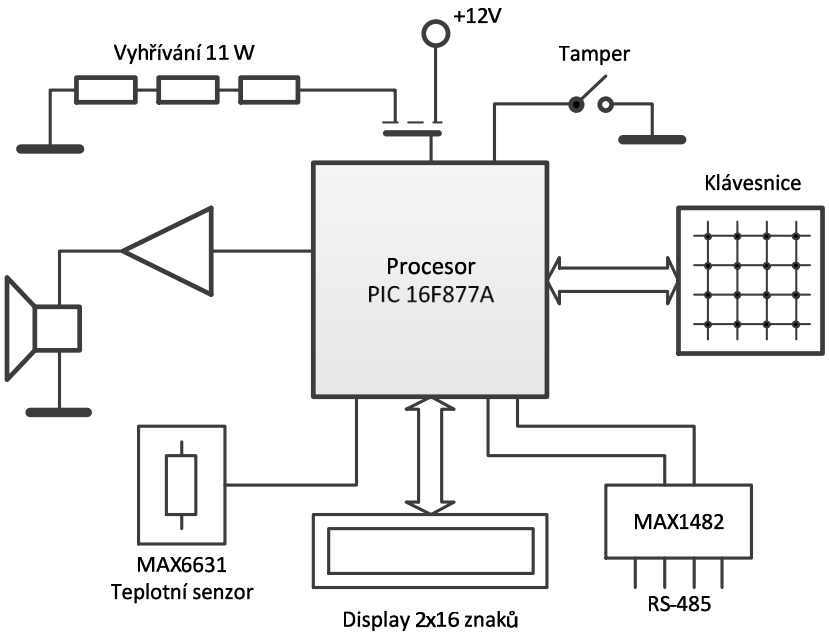
```
while (time_out --){
  if(uart1_RxTest() == 0) {NutMicroDelay(1);  continue; }

  Rx_Size = recvPacket(Buffer, PACKET_BUFFER_SIZE, (unsigned char)uart1_GetChar());
  if (Rx_Size) break;
}
```

5.3. Klávesnice

Pro vzdálenou komunikaci se systémem je navržena vandal odolná klávesnice vybavená dvouřádkovým LCD displejem. Klávesnice komunikuje s řídící deskou pomocí čtyřdrátové full duplex RS-485 sběrnice. Jádrem celé klávesnice je PIC 16F877A od firmy Microchip. K microkontroléru je připojen LCD modul MC1602 (2x16 znaků) a kovová odolná klávesnice s 12-ti znaky. Vzhledem k tomu, že klávesnice je určena do exteriéru, je na desce plošných spojů umístěno teplotní čidlo MAX6631 a celkem 24 rezistorů používaných jako topení (celkem asi 11W). Dále je na desce umístěn reproduktor a 1 W můstkový zesilovač, připravený pro další použití. Celý modul je opatřen magnetickým kontaktem, který indikuje neoprávněné odejmutí klávesnice.

Blokové schéma klávesnice je zobrazena obrázkem 5.36.



Obrázek 5.36: Blokové schéma zapojení klávesnice

5.3.1. Program klávesnice

Program se skládá ze dvou hlavních částí: hlavní smyčky (main), kde se cyklicky testují nastavené příznaky a na jejich základě se provádějí akce, k vykonání a přerušení.

Procesor Microchip PIC16F876A má pouze jeden příznak přerušení, který způsobí odskočení programu do přerušovací rutiny. V ní se testuje, jaké přerušení nastalo. Program pracuje se dvěma druhy přerušení – přerušení od sériové linky (která vykoná vložení přijatého byte do bufferu) a periodické přerušování od časovače. V přerušení od časovače se čte stav klávesnice, teplota, zapíná / vypíná tón klávesnice (pípání není realizované přes čekající smyčky) a v případě, že je klávesnice aktivní odpočítává čas od poslední akce. Vyjmenované úkony vystavují příznaky, které se následně vykonají v hlavní (main) smyčce.

5.3.1.1. Vstup z klávesnice

Klávesy jsou zapojeny maticově, kde vertikálně se vystaví hodnota a horizontálně se hodnota přečte. Tento proces je umístěn v přerušení od časovače a nastavuje proměnou, která nese hodnotu stisknuté klávesy (zákmity jsou odstraněny opětovným testem stisknuté klávesy po 25 opakovacích cyklech od časovače). Ta je jedním vstupem do ovládání automatu zobrazení displeje (dalším vstupem jsou příchozí zprávy) a procesem přihlášení.

5.3.1.2. Přihlašovací piny

Piny jsou z důvodu bezpečnosti ukládány v paměti RAM (při odebrání napájení obsah paměti RAM zmizí). V paměti je vyhrazeno místo pro 10 pinů velikosti dvou bytů a deseti oprávnění k 32 voleb pro určení, jaké povely může přihlášený pin vykonávat.

Speciálním pinem je tzv. servisní pin, který zpřístupňuje menu pro konfiguraci klávesnice (nastavení zapnutí a vypnutí topení a adresy klávesnice).

5.3.1.3. Texty

Klávesnice ukládá 2 x 32 šestnáctiznakových textů – 32 příkazových textů určující volbu, která má být vykonána a 32 textů určujících v jakém stavu je zařízení před/po vykonání příkazu (např. prvý text: „Odemknutí vchodu“, k němu druhý text „Vchod zamknut“). Z důvodu rychlosti (jelikož texty se mění zřídka) jsou texty ukládány do paměti EEPROM, která uchovává informace i po odebrání napájení.

Texty i piny jsou uloženy v hlavní procesorové jednotce a jsou odesílány do klávesnice. Klávesnice si ukládá 2 byty kontrolního součtu všech textů, pinů i jejich oprávnění a na žádost je odesílá hlavní jednotce. Při zjištění, že kontrolní součet není shodný (data v hlavní jednotce se změnila, popř. klávesnice byla restartována a neobsahuje platné piny), začne hlavní jednotka odesílat konfiguraci.

Hlavní řídící jednotka nejdříve odešle přístupové kódy (PINy). Po přijetí desátého pinu si klávesnice přepočítá kontrolní součet a odešle hlavní procesorové jednotce. Jestliže se součet stále neshoduje, pokračuje se se synchronizací příkazových textů a oprávnění pinů. Po přijetí 32. povelu se proces porovnání kontrolních součtů opakuje. Při opětovné neshodě se nakonec sesynchronizují zbylé texty informující o aktuálním stavu zařízení.

5.3.1.4. Akce

Každý pár textů odpovídá jedné akci. Klávesnice tak může odesílat do hlavní řídící jednotky 32 navolených povelů (on x off). Klávesnice si uchovává stavový registr o velikosti 4 byty, kde je bitová maska jednotlivých akcí. Při listování v klávesnici jednotlivými povely se posouvá ukazatel po stavovém registru a při potvrzení vybrané volby bit neguje. Takto modifikovaný registr se po zažádání serveru odešle. Na jeho základě hlavní řídící jednotka vykoná akci (případně naplánuje akci) a klávesnici informuje o přijetí registru a skutečným stavu zařízení.

Komunikační protokol mezi klávesnicí a hlavní řídící jednotkou jsou popsané v kapitole 5.2 Popis komunikace.

Webové rozhraní

6.1. XSL šablony

XSL šablony (eXtensible Stylesheet Language Transformations) jsou definované mezinárodním konsorciem W3C roku 1999. Slouží pro převod (transformaci) dat zapsané v XML do definovaných formátů, jako například do HTML stránky, PDF dokumentu nebo i jiného XML dokumentu.

Proces vytváření výstupního dokumentu (v našem případě XHTML stránky) probíhá na straně klienta pomocí XSL procesoru, který v dnešní době obsahuje každý webový prohlížeč. Generování výstupu na straně klienta je v našem případě užitečné, neboť nemusíme složitým generováním výstupu vytěžovat procesor. Nevýhodou by mohla být nutnost používat na straně klienta „moderní“ internetový prohlížeč, který s XSL šablonami umí pracovat. V dnešní době by tato situace nastala asi jen opravdu ojediněle. Podpora XSL šablon v majoritních prohlížečích je vypsána v tabulce 6.1.

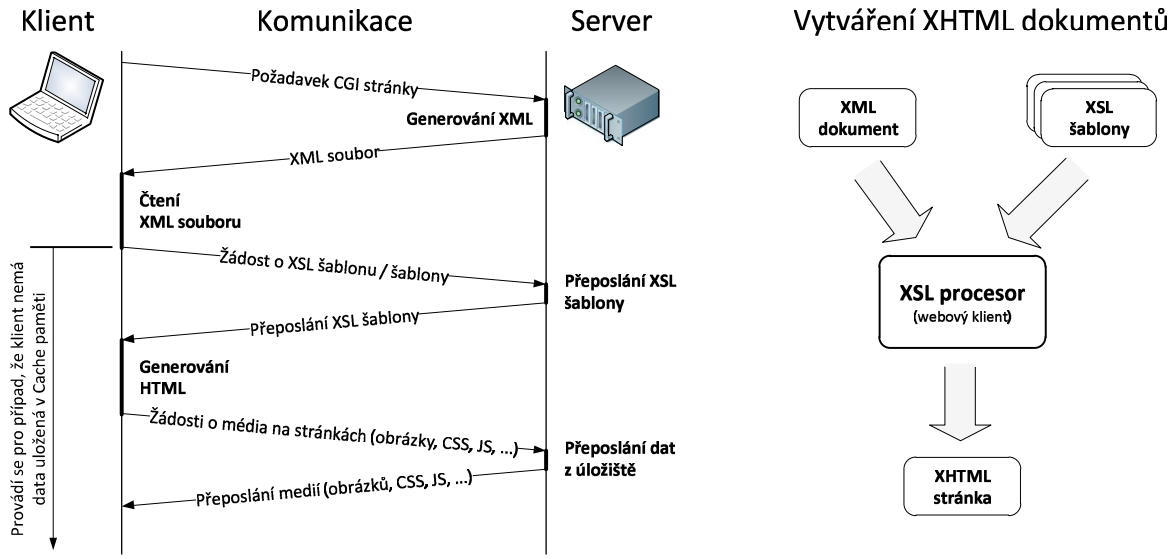
Tabulka 6.1
Tabulka podpory XSL šablon

	Podpora XML, XSL a Xpath od verze	Aktuální verze
Mozilla Firefox	3	4
Internet Explorer	6	9
Google Chrome	1	12
Opera	9	11
Aplle Safary	3	5

Pro vygenerování XHTML stránky musí mít XSL procesor zdrojová data – XML dokument a příslušnou XSL šablonu, jejíž URL adresa je uvede v hlavičce XML dokumentu. Prohlížeč si poprvé šablony stáhne (pro generování jedné XHTML stránky může být zapotřebí stahovat i více XSL souborů) z hlavní řídící jednotky (serveru). Při dalších přístupech má klientský browser XSL šablony uložené ve vyrovnávací paměti a šablony již nepotřebuje stahovat. Tím se komunikace mezi klientem a serverem zredukuje a práce se stránkami zrychlí.

Po načtení XML i XSL dokumentů provede XSL procesor transformaci na XHTML stránku (jak je znázorněno obrázkem 6.1). XHTML stránka může obsahovat odkazy na další zdroje potřebné pro zobrazení stránky, jako jsou JavaScriptové soubory, kaskádové styly CSS, obrázky nebo další XML data. Požadavky na tyto soubory jsou odeslány až v průběhu zpracovávání XHTML souboru prohlížečem. Opět platí, že načítání objektů se provádí vždy při prvém zobrazení stránky. Tyto soubory společně s XSL šablonami (jelikož jsou statické), jsou uloženy v externí paměti na SD kartě. SD kartu je tak možné snadno z hlavní řídící desky vyjímat a pomocí PC data i šablony upravovat. Tak

můžeme provádět rozsáhlé úpravy na zobrazovaných webových stránkách bez nutnosti zásahu do firmware.



Obrázek 6.1: Vytváření XHTML dokumentu

Další výhoda, která je v našem projektu využívána, je spojování XML souborů. Nejen že XSL procesor dokáže reprezentovat XML data v čitelné podobě, ale dokáže navíc XML data spojovat a dotazovat. XML se potom pracuje obdobně jako s databází. Dotazovací jazyk postaven na XQuery společně s XPath. V projektu se toto využívá pro konfiguraci webových stránek. Na SD kartě je umístěný soubor config.xml, který obsahuje linky na obrázky (z SD karty), nadpisy a další položky potřebné k zpřehlednění webových stránek. Soubor je umístěn na SD kartu z důvodu snadného přístupu pro možnou editaci.

6.2. Zrychlení komunikace

Při vytváření webových stránek jsme velmi omezeni výkonem procesoru, na kterém server běží. Pro vytvoření hezkého uživatelského rozhraní je nutné uplatnit nejrůznější techniky, které sníží vytížení procesoru hlavní řídící jednotky. Všechny techniky pro zrychlení komunikace mezi procesorovou jednotkou a webovým klientem jsou používané v SEO optimalizacích.

6.2.1. Používání CSS

CSS odděluje data webové stránky od vzhledu. Při správném použitím CSS jednak klesne celkový objem dat pro vytvoření stránek, a jednak se uplatní cáchování na straně uživatele. Jak bylo popsáno v kapitole 6.1 XSL šablony, tyto přijatá data se ze serveru načtou jednou a poté se již berou z paměti prohlížeče.

6.2.2. Komprese dat

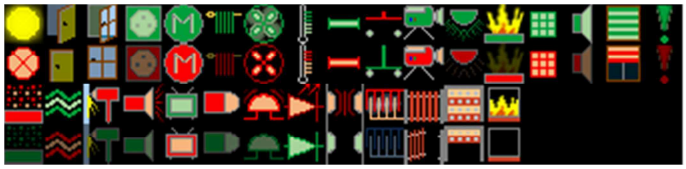
Jedním ze způsobů je komprese zdrojových dat a tím eliminování přenášeného objemu dat. Komprimovat data pomocí gzip, jak se děje na webových serverech zde není realizována, neboť operační systém kompresy neumožňuje, a dopsání funkce pro její složitost není rozhodující. V našem projektu si vystačíme s kompresí XSL, CSS a JS souborů odstraněním bílých znaků (pro program pracující s těmito daty jsou bílé znaky bezvýznamné). Například pro doplněk dataTables klesne velikost souboru o 70%.

6.2.3. Vytváření palet obrázků

Při měření času načítání kompletní webové stránky se jako nejdelší časové úseky projevují navázání spojení a posílání požadavků o data. Tyto požadavky je možné eliminovat spojováním. U CSS stylů a JS souborů lze vytvářet spojené, delší soubory, než několik menších. Doba načtení několika menších je pak větší, než načtení jednoho většího CSS souboru.

U obrázků situace není tak jednoduchá. Při tvorbě webových stránek jsem se snažil používat minimální počet obrázků. Avšak vstupy a výstupy karet jsou v přehledu poschodí reprezentovány šedesáti samostatnými obrázky 20 x 20 px. To je již velké množství.

Pro tyto obrázky jsem zvolil způsob, kdy se obrázky spojí do jednoho po určitých částech - vytvoří se tzv. paleta) a pomocí CSS vybírám z palety jen určitý úsek. Paleta není větší, než velikost samostatně nakreslených ikon a při načítání ze serveru se použije pouze jeden požadavek. Příklad takové palety je na obrázku 6.2.

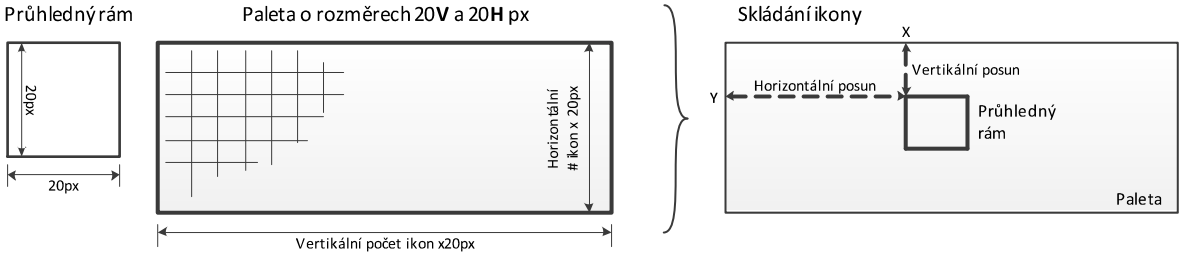


Obrázek 6.2: Paleta objektů

Vytvoření ikony z palety pak probíhá následovně:

- Ikonu nahrazuje rámec (**DIV**) o velikosti jedné ikony (20x20px)
- Jako pozadí rámce se zvolí paleta
- Paletě se podle potřeby nastaví souřadnice posunutí

Postup je znázorněn obrázkem 6.3.



Obrázek 6.3: Princip vytváření ikon

6.3. Další použité technologie

Jak již bylo zmiňováno, webové rozhraní je tvořeno pomocí XML souborů, generující hlavní řídící jednotkou a XSL šablonami uložené na SD kartě jednotky.

Samotné XHTML je pouhá statická webová stránka obsahující hypertextové odkazy. Pro zpříjemnění práce se stránkami se používá JavaScript, který dokáže na stránce (na straně klienta) vytvořit dynamické chování – může manipulovat se strukturou stránek, odesílat požadavky v těle stránky atd. V našem webovém rozhraní je samozřejmě také JavaScript použit.

Pro snadnější práci s JavaScriptem existují nejrůznější frameworky obsahující naprogramované a často používané funkce. Nejrozšířenějším frameworkem je JQuery. Jedná se o open source projekt, do kterého existuje nespočet přídavných balíčků, vytvářející na stránkách doslova miniaplikace. Veškerá dokumentace včetně samotného frameworku jsou k dispozici na webových stránkách http://www.jquery.com.

Jednou takovou miniaplikací je přehledová tabulka v administračním rozhraní webového rozhraní DataTables (http://www.datatables.net). Tato miniaplikace vytvoří pro tabulku, na kterou je aplikovaná, filtry – přes sloupce i fulltextové. Jelikož předpokládáme v systému několik set vstupů a výstupů, je dobré na tabulku, která tato data ponese, aplikovat zmiňovaný doplněk pro urychlení vyhledávání.

Dalším výraznějším doplňkem je draggable (od tvůrců JQuery), který umožňuje posouvání objektů po webových stránkách. Pro funkčnost tohoto doplňku jsou nutné podpůrné knihovny, jako například ui.core, ui.mouse, a ui.widget (dodávané doplňkem draggable).

V poslední řadě drobné doplňky jsou dHtmlToolTip pro vylepšení bublinkové nápovědy stránek a ui.slider (opět od tvůrců JQuery). Ui.slider je použit pro vytvoření posuvníku (použitý pro změnu průhlednosti obrázku), který do HTML verze 4 není ve svém základu implementován pro všechny prohlížeče.

6.4. Konfigurace webových stránek

Konfigurace webových stránek je zcela oddělená (a tak naprosto nezávislá) od firmware řídící jednotky. Je tvořena XML souborem config.xml umístěném na SD kartě. Umístění je zvoleno pro snadnou uživatelskou dostupnost a editaci.

Soubor obsahuje informace nezávislé na provádění obsluhy karet. Obsahuje převážně informaci o objektu, kde je systém použitý a vzhled stránek.

Hlavním root uzlem konfiguračního souboru je „config“ obsahující sekce parameters (obecné parametry), elemetns (informace vstupních / výstupních objektech webových stránek) a location obsahující informace o poschodích.

Parameters

Elementy v této sekci jsou **refresh_time** pro určení časového intervalu obnovy stavů vstupů a výstupů (čas je zapsaný v desítkách milisekund – tzn. refresh_time = 100 znamená 1 sekunda) a **bg_opacity** pro určení výchozí průhlednosti podkladu poschodí v procentech (rozdah 0 až 100).

```
<parameters>
        <refresh_time>100</refresh_time>
        <bg_opacity>70</bg_opacity>
</parameters>
```

Elements

Obsahuje informace o vstupních a výstupních objektech. Každý objekt je uvozen v sekci **type** o identifikátoru id, jedinečného pro každý typ objektu. Každý objekt obsahuje název (**name**), souřadnice (**x** a **y**) palety ikon pro stav v log. 1 (**on**) a pro log. 0 (**off**).

```
<elements>
        <!-- ... -->
        <type id="1">
                <name>Světlo</name>
                <modify>0</modify>
                <on>
                        <x>0</x>
                        <y>0</y>
                </on>
                <off>
                        <x>0</x>
                        <y>20</y>
                </off>
        </type>
<elements>
```

Location

V této sekci jsou určeny jednotlivé místnosti (**room**) a poschodí (**stock**) s unikátním identifikátorem. Místnost obsahuje pouze svůj název, poschodí obsahuje kromě názvu (**name**) i umístění svého obrázku na SD kartě (**url**) a zmenšenou ikonu pro výběr poschodí (**icon**).

```
<location>
        <room id="1">Prádelna</room>
        <!-- ... -->

        <stock id="1">
                <name>Sklep</name>
                <url>../sites/images/stock/sklep.gif</url>
                <icon>../sites/images/stock/sklep_small.gif</icon>
        </stock>
        <!-- ... -->

</location>
```

Uživatelský interface

Webové rozhraní, vytvořené popsanými technologiemi se skládá ze dvou částí – administračního rozhraní pro konfiguraci řízení a uživatelského rozhraní pro zobrazení stavů vstupů / výstupů a jejich ovládání.

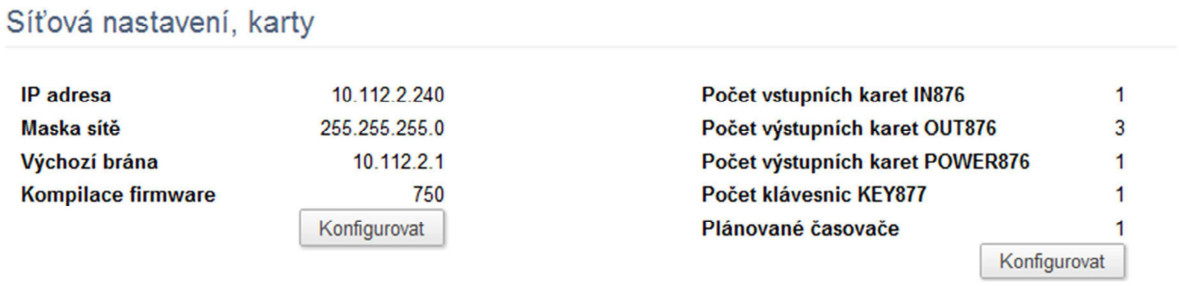
7.1. Administrační část

V této části uživatel nastavuje hlavní řídící jednotku - vytváří konfiguraci a konfiguruje jednotlivé funkce systému.

7.1.1. Hlavní přehled

Při přihlášení do této části se zobrazí stránka s aktuálním stavem celého systému. Ve vrchní části je zobrazena síťová konfigurace (IP adresa, Maska sítě, Výchozí brána) a číslo kompilace firmware. Číslo kompilace je vytvářeno jednoduchým programem, který vytváří hlavičkových soubor ver.h obsahující číslo kompilace. Při spuštění dávky pro kompilaci program inkrementuje číslo, a uloží do souboru ver.h.

V pravém horním rohu (obrázek 7.1) je přehled nakonfigurovaných karet v systému – karet vstupů, výstupů, výkonových relé, klávesnic a časovačů.



Obrázek 7.1: Přehled hlavní konfigurace systému

Jestliže nějaké ze zařízení neodpovídá, protože je buď odpojené, nebo má závadu, zobrazí se v prostřední části přehled těchto zařízení. Obrázek 7.2 zobrazuje příklad nedostupných karet v systému.

Karty neodpovídají na požadavky

Typ karty	Název karty	Adresa
in876_cards	Vstupni karta1	1
out876_cards	Vystupni karta1	1
out876_cards	Vystupni karta2	2
key877_cards	KeyBoard 01	250

Obrázek 7.2: Výpis nedostupných karet

Ve spodní části stránky (obrázek 7.3) je přehled nakonfigurovaných vstupů / výstupů (nikoliv rozpis karet!). Z tohoto rozcestníku je možné přejít přímo na editaci konkrétní karty, konfigurace objektů v poschodí nebo přejít na konfiguraci výstupů. Tabulka je vytvořená tak, aby mohla obsahovat velké množství řádků (řádů stovek) a bylo možné v ní dobře vyhledávat – fulltextově v horní části, nebo pomocí filtrů u každého sloupce. Neodpovídající karty jsou rozlišeny červeným vykřičníkem před typem karty.

Přehled využitých portů

Search all columns:

Typ karty	Název karty	Poschodí	Místnost	Název pinu	Typ pinu
! key877_cards	KeyBoard 01	Sklep	Prádelna	Odemknuti vrat	Klávesnice
! key877_cards	KeyBoard 01	Sklep	Prádelna	Garazova vrata	Klávesnice
! in876_cards	Vstupni karta1	Patro	Záchod	Spínač světla	Vypínač
! out876_cards	Vystupni karta1	Patro	Záchod	Vetrak na toalete	Větrák
power876_cards	Power1	Zahrada	Zahrada	Čerpadlo bazénu	Čerpadlo

Zobrazeno 1 až 5 položek (z 5 položek)

Obrázek 7.3: Přehled využitých portů

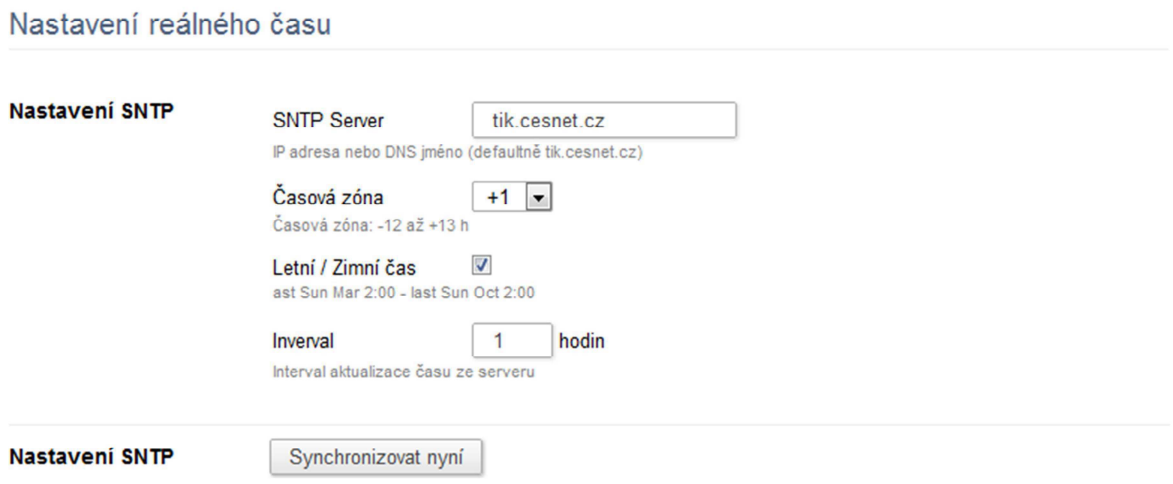
7.1.2. Globální konfigurace

Globální konfigurace je přístupná z hlavního rozcestníku. V konfiguraci lze nastavit síťové připojení, uložení konfigurace do paměti (pokud byla provedena změna konfigurace, je tato volba zvýrazněna červenou barvou) a nastavení reálného času.

Pro nastavení reálného (obrázek 7.4) času lze nastavit SNTP server, odkud je načítán čas (defaultně nastavený na tik.cesnet.cz), časovou zónu (defaultně nastaven hodinový posun), povolení přepínání letního / zimního času a interval v hodinách, ve kterém je čas ze serveru načítán.

Nastavení sítě

Uložení	IP adresa	Maska sítě	Výchozí brána	Port
	10.112.2.240	255.255.255.0	10.112.2.1	80



Obrázek 7.4: Nastavení systému reálného času

Další volbou je ruční spuštění synchronizace ze serveru a ruční nastavení času (obrázek 7.5).

Obrázek 7.5: Ruční nastavení SNTP

7.1.3. Přehled nakonfigurovaných karet

Přehled nakonfigurovaných karet je opět přístupný přes hlavní stránku administračního rozhraní. Obsahuje přehled všech karet, které mají v systému konfiguraci.

V levé části tabulky (obrázek 7.6) jsou rozepsané typy karet, se kterými umí hlavní řídící jednotka pracovat. Výjimky tvoří „karty“ timer_card a virtual_card. Tyto karty jsou tvořeny softwarově v procesorové jednotce a starají se o časové spínání, nebo o rozšíření karet výstupů (jak bylo popsáno v kapitole 5.1.3.3. Výpočet výstupních hodnot). Po rozkliknutí se vypíší do pravé části tabulky. Poté je možné karty odstraňovat nebo přidávat.

Obrázek 7.6: Správa karet

7.1.3.1. Karta vstupů

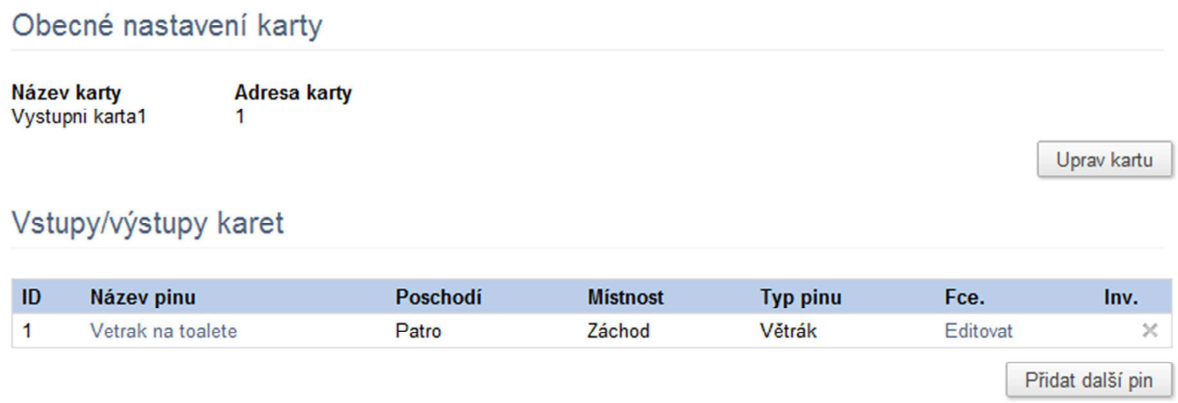
Konfigurace karty vstupů není nikterak složitá. Obsahuje název karty, její adresu a výpis vstupů, kterých může být maximálně 16. U každého vstupu je nastaveno umístění v objektu pro snadnější nalezení v konfiguraci. Umístění se skládá z určení patra (poschodí), místnosti a typu vstupu. Tyto volby jsou nastaveny v konfiguračním XML souboru umístěného na SD kartě. Jeho editací je tak možné snadno přidávat nebo odebírat tyto volby. Poslední možností je invertování vstupu. Tato možnost je zvolena pro zařízení, které mají stav sepnuto v logické 0 (například záplavový detektor).

V případě, že karta neodpovídá, je uživatel informovaný červenou hláškou v horní části konfigurace (taková hláška funguje pro všechny typy karet, a je viditelná v obr. 7.7).

Obrázek 7.7: Přehled karet vstupů

7.1.3.2. Karta výstupů, výkonových relé a virtuální karta

Tyto tři karty jsou z hlediska konfigurace totožné. Konfigurace je velice podobná kartě vstupů. Přehled nastavení těchto karet je zobrazen na obrázku 7.8.

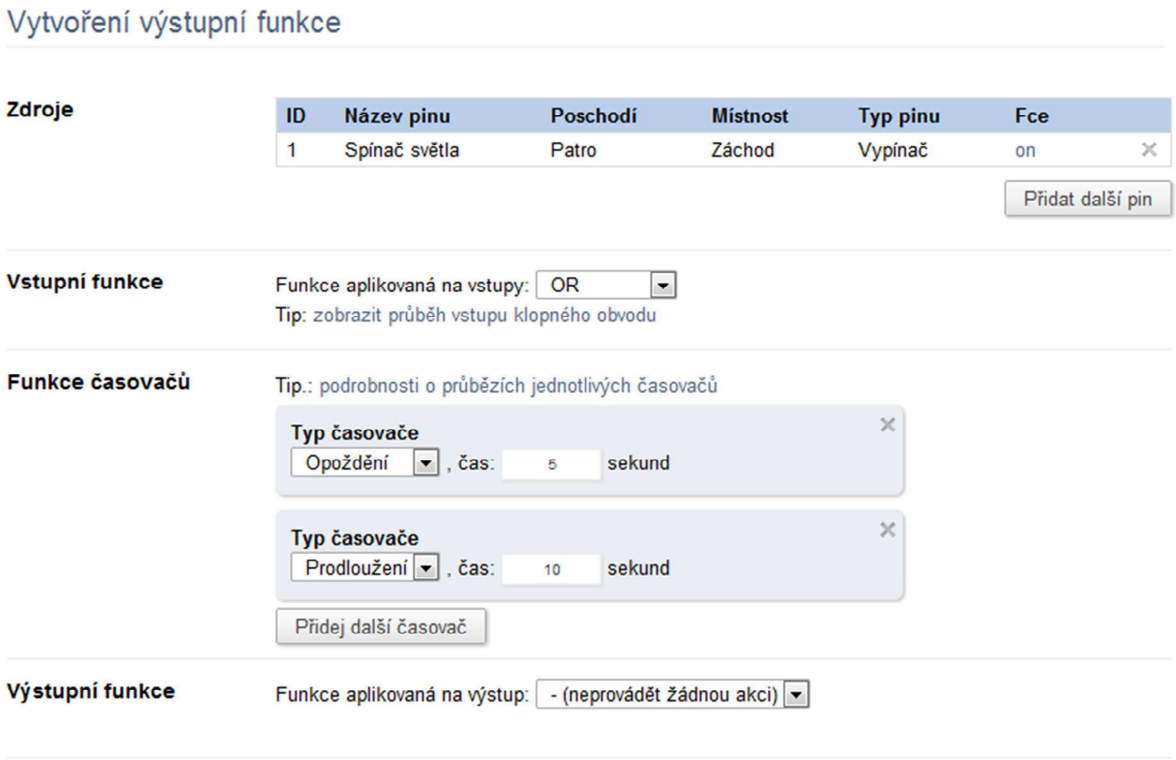


Obrázek 7.8: Karta výstupů, výkonových relé a virtuálních karet

Hlavním rozdílem, od karty vstupů je možnost vytváření výstupní funkce (po kliknutí na tlačítko Editovat ve sloupci „Fce.“ u konkrétního výstupu).

Konfigurace výstupní funkce (popsaná v kapitole 5.1.3.3. Výpočet výstupních hodnot) se skládá z vybraných zdrojů, vstupní funkce (která se aplikuje na vstupy), časovačů (neomezeně mnoho) a výstupní funkce (pro možnost odpojení výstupu a nastavení do jedné z logických hodnot nebo invertování).

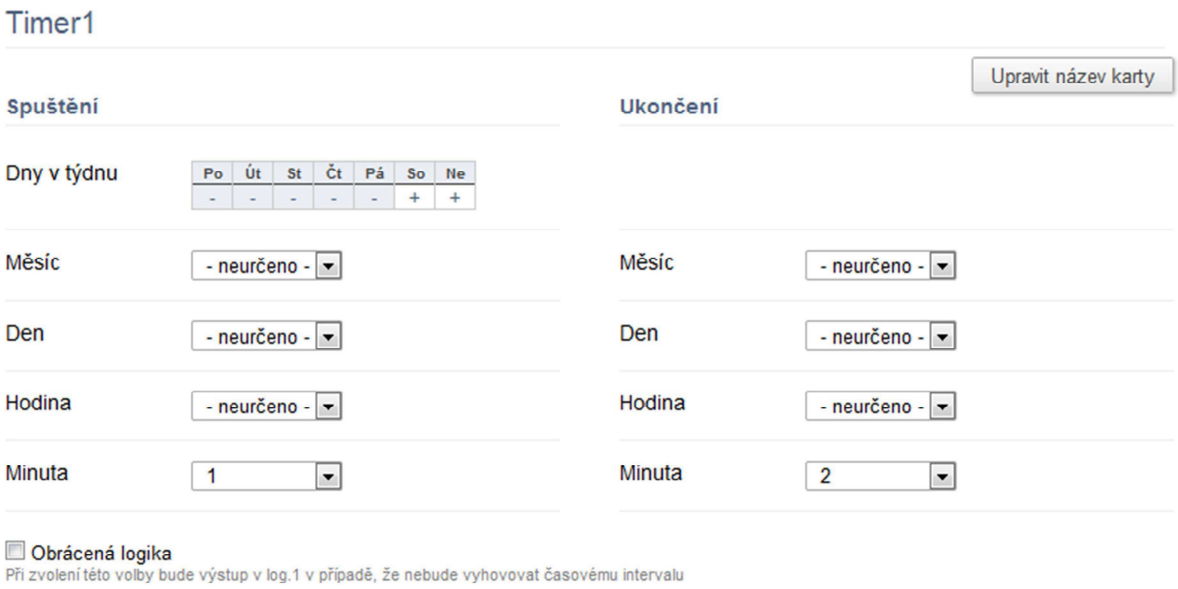
Na obrázku 7.9 je zobrazeno nastavení větráku na toaletě. Větrák je připojený na světlo, který ovládá uživatel. Větrák se spustí v případě, že toaleta bude déle než 1 minutu rozsvícená a vypne se po 3 minutách, co světlo zhasne.



Obrázek 7.9: Vytvoření výstupní funkce výstupních karet

7.1.3.3. Karta časovačů

Tyto karty slouží pro vytvoření volání událostí v určitých časových intervalech. Zde se nastavují dny v týdnu, ve kterých má být karta aktivní a v časech spuštění a vypnutí. Je-li v nějakém času nastaveno „neuvedeno“, nebude tento časový údaj pro rozhodnutí respektovaný. Stejně jako ostatní karty, i zde lze výstup invertovat. Nastavení je zobrazeno na obrázku 7.10.



Obrázek 7.10: Nastavení časovačů

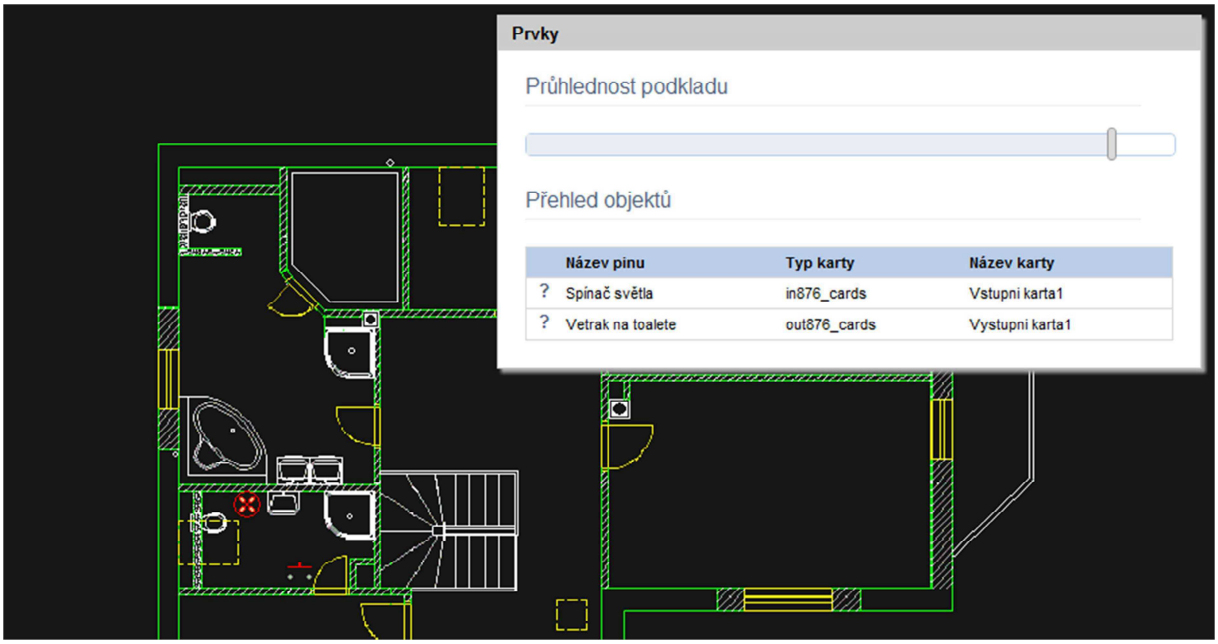
7.1.4. Editace poschodí

Zde si uživatel zobrazí půdorys objektu (resp. jakýkoliv pohled na objekt) uložený jako obrázek na SD kartě (název obrázku spojený s názvem poschodí je opět uložený v konfiguračním souboru config.xml).

Na obrázku jsou rozmístěny prvky, které systém ovládá – světla, motory, větráky atd. a prvky, ze kterých systém čerpá vstupy (detektor úniku plynu, záplavový detektor, dveřní kontakty a mnoho dalších). Vstupy a výstupy jsou reprezentovány technikou popsanou v kapitole 6.2.3. Vytváření palet obrázků.

Na této stránce lze rozmisťovat tyto ikony do míst, kde se fyzicky nacházejí. Každá ikona nese svůj název určený v konfiguraci, který se zobrazí po najetí myši. Přehled všech objektů je zobrazený v pravé části v plovoucím okně (vhodné pro větší plátna na menších displejích – tímto objektem lze posouvat, abychom nepřekrývali část plátna) společně s názvem karty a typem. Dalším užitečným prvek umístěným v plovoucím okně je průhlednost podkladu, která zprůhlední (až vypne) podklad s ikonami. Funkce je užitečná pro nalezení ikon, které se ztrácejí v podkladu.

Ukázka editace objektů pro poschodí je zobrazena na obrázku 7.11.



Obrázek 7.11: Konfigurace objektů v poschodí

7.2. Uživatelská část

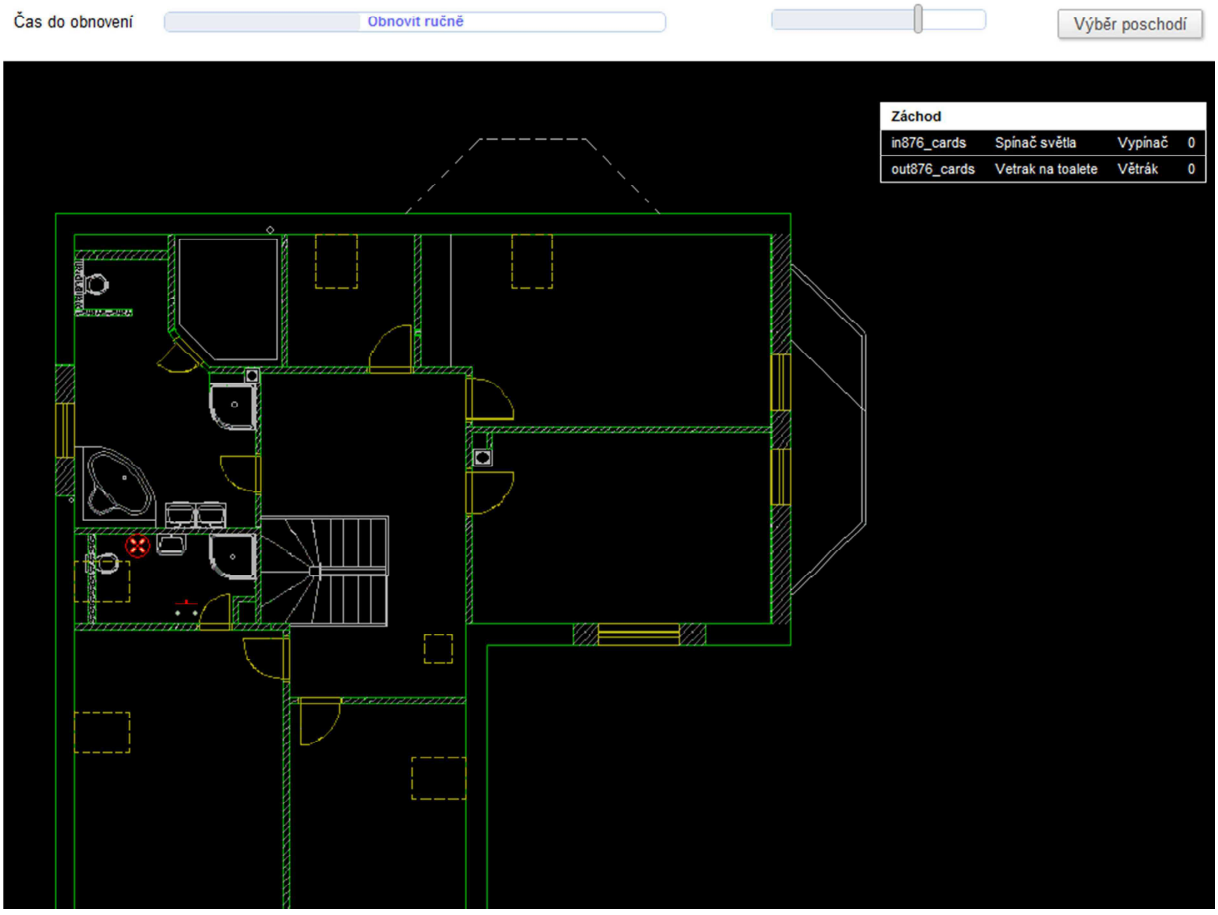
V uživatelské části, přístupné pod jiným přihlašovacím jménem a heslem je přístupné rozhraní, pro ovládání domu na dálku.

Stránka se skládá z částí, které jsou nakonfigurovány v Editaci poschodí. V horní části stránky se nachází odpočet času obnovení. Interval obnovení lze opět nastavit v konfiguračním souboru config.xml. Pro případ rychlejší obnovy lze na průběh kliknout a obnovení se provede okamžitě. V pravé vrchní části je posuvník průhlednosti nastavený na hodnotu určenou konfigurací v config.xml a tlačítko pro výběr jiného poschodí.

Ikonami již nelze pohybovat, ale lze pozorovat jejich stavy a měnit jejich stavy. U vstupních objektů, jako jsou vypínače z tohoto rozhraní měnit jejich vstupní hodnotu. Výsledek je pak viditelný na výstupních objektech.

Ne všechny vstupní objekty lze z webového rozhraní měnit. Například záplavový detektor, u kterého nemusí být logické ovládání na dálku. Takové vstupy, které mají být možné měnit prostřednictvím webového rozhraní, jsou opět nakonfigurované v config.xml.

Ukázka uživatelské části je zobrazena na obrázku 7.12.



Obrázek 7.12: Uživatelská část

MainBoard

Celé zařízení je navrženo jako modulární tak, aby bylo možné koncepci technologického zařízení umístit do 19“ technologického racku. Základním prvkem celého systému je základní deska (MainBoard), opatřená konektory, do které se zasouvají jednotlivé modulární desky, jejichž úkolem je zajišťovat vlastní funkci systému.

Vlastní základní deska slouží pouze ke kontaktu mezi deskami. Do základní desky je nasunuta hlavní řídící jednotka (MainBoard), desky vstupů, desky slaboproudých relé, případně další modulární desky.

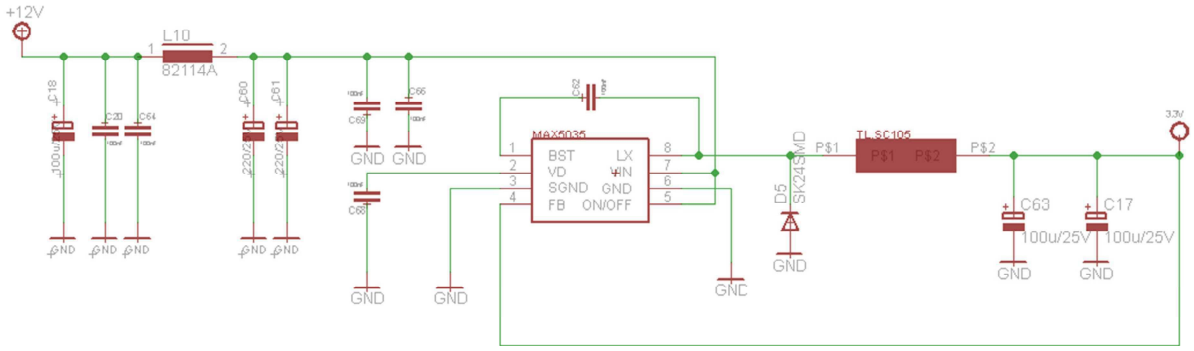
Úkolem hlavní procesorové karty je připojení hlavního procesoru AT91SAM7X512 k ostatním částem zařízení prostřednictvím osmibitové paralelní sběrnice, USARTů, I^2C, SPI a ethernet komunikace.

Karta musí obsahovat převodníky pro tyto komunikační protokoly, zabezpečit napájecí napětí vhodné velikosti a poskytnout dostatečné ochrany na jednotlivých interfacích.

Srdcem celé karty a celého systému je již zmiňovaný vývojový kit AT91SAM7X512-KIT, který je dodáván jako samostatný modul o velikost 140 x 149 mm. Modul obsahuje 52 pinů, a je zasunut do karty prostřednictvím jehličkových konektorů.

8.1.1. Napájecí zdroj 3,3 V

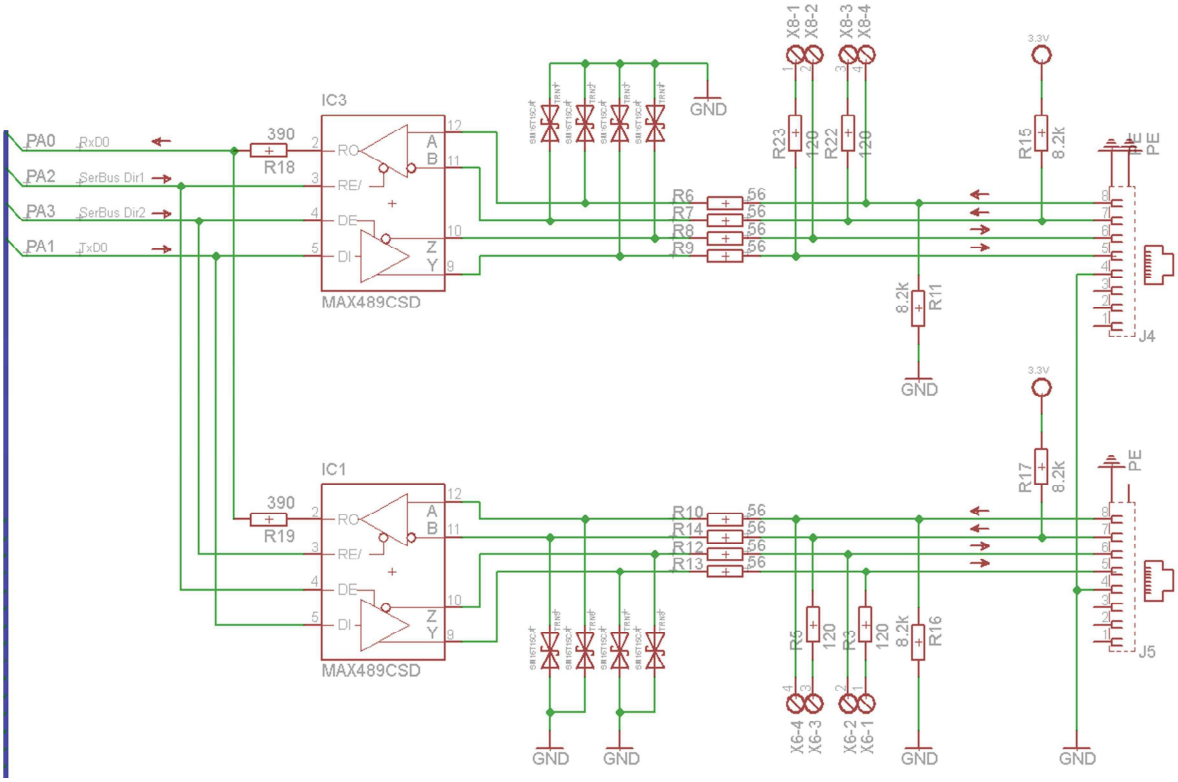
Pro napájení hlavní řídící jednotky je použitý spínaný napájecí zdroj 3,3 V / 1 A od firmy Maxim MAX5035. Vstupním napětím obvodu je hlavní napájecí napětí celého racku +12 V zálohované hlavní baterií. MAX5035 využívá pulzně šířkovou modulaci s pracovní frekvencí 125 KHz. Klidový odběr zdroje je 350 µA. Obvod je použit v pouzdře SO8 [12]. Schéma zapojení je znázorněno obrázkem 8.1.



Obrázek 8.1: Napájecí zdroj 3,3V

8.1.2. Převodníky pro komunikaci na RS-485

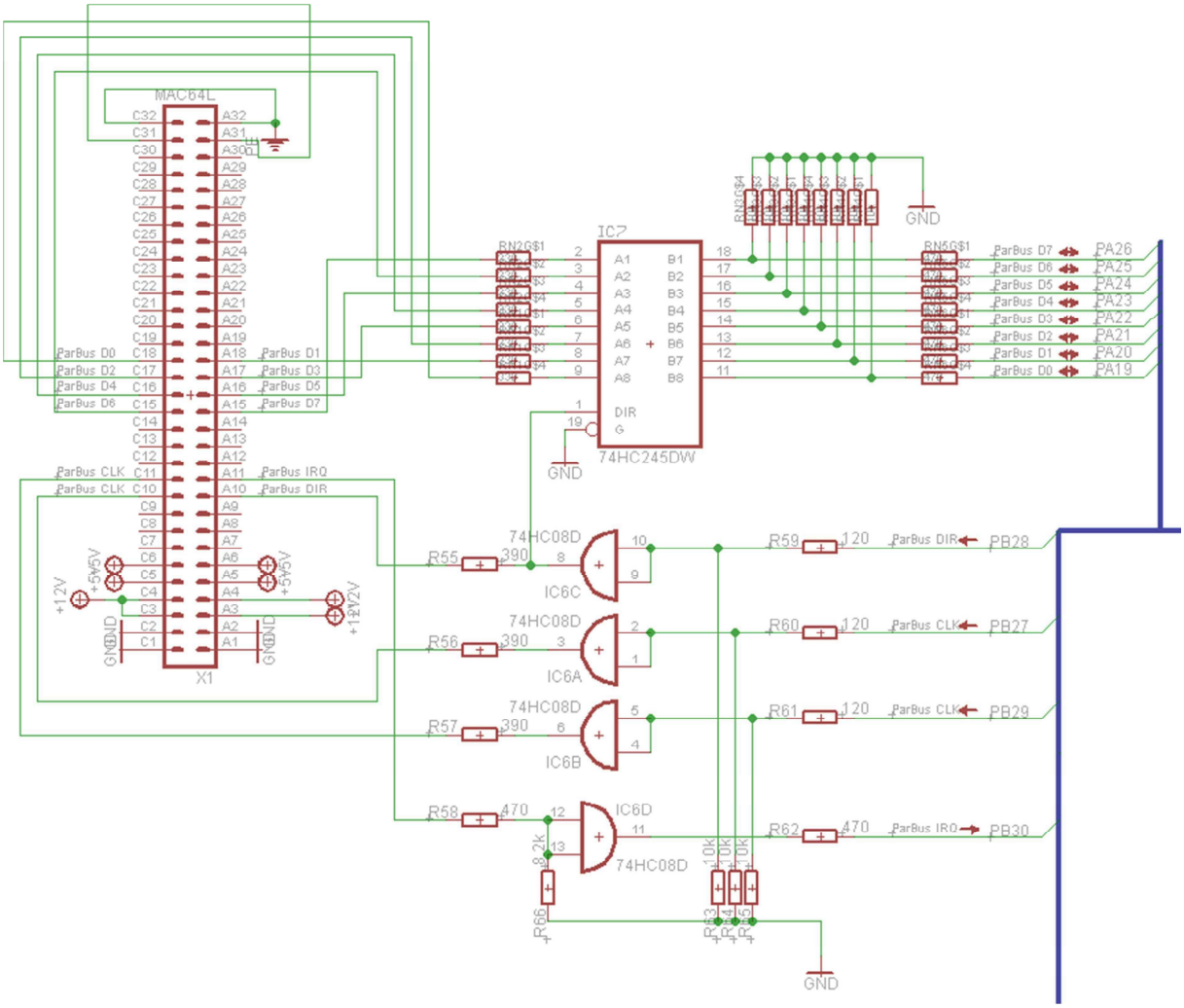
Pro komunikaci se sériovými zařízeními je využíváno komunikace RS-485 realizovanou pomocí dvou obvodů firmy Maxim MAX3070/73/76 (obrázek 8.2). Sběrnice je opatřena ochrannými transcili. Budiče RS-485 jsou připojeny na piny na PA0 a PA1 (USART0) AT91SAM7X512. Z důvodu snížení potencionálních problémů na sériových linkách jsou použity dva budiče, které jsou navzájem multiplexovány prostřednictvím PA2 a PA3 procesoru. Každá RS-485 sběrnice je vyvedená na konektor RJ-45 [10].



Obrázek 8.2: Převodníky pro komunikaci RS-485

8.1.3. Převodník 3,3 V ↔ 5V pro paralelní sběrnici

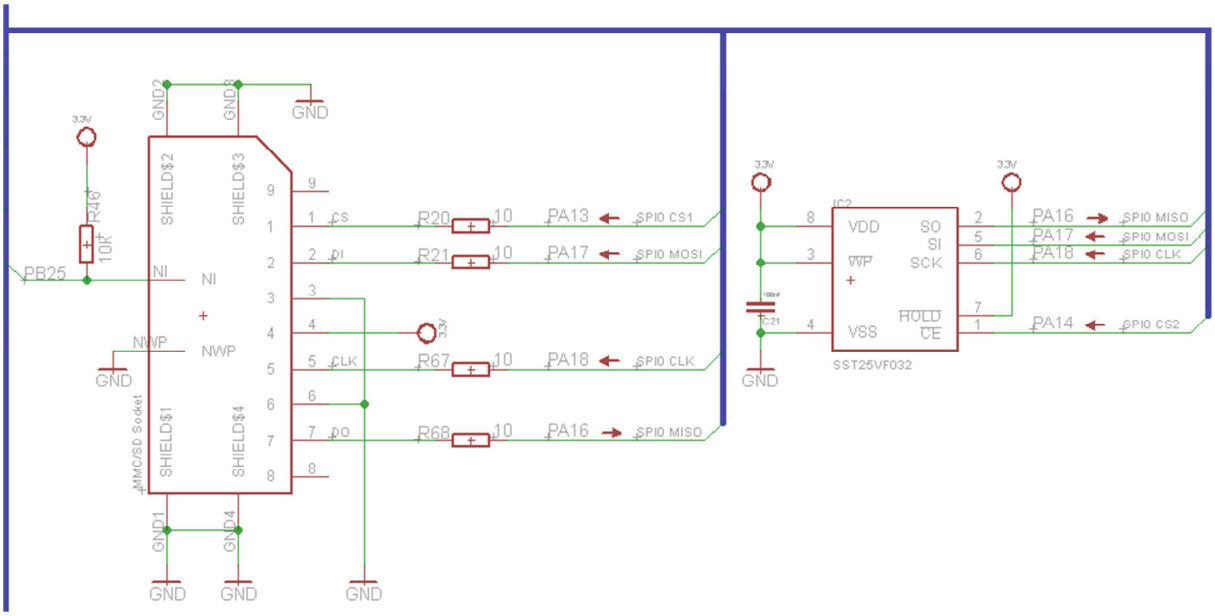
Osmibitová paralelní sběrnice, včetně řídících signálů, na kterou jsou připojeny všechny komunikační karty, komunikuje na úrovni 5V CMOS logiky. Hlavní procesorová karta (mainboard) osazená procesorem AT91SAM7X512 pracuje na úrovni 3,3 V logiky. Z tohoto důvodu je třeba zajistit komunikační převodník (obrázek 8.3). Pro jednoduchost je použit neinvertující obousměrný osmibitový budič sběrnice 74HC245 [7], který akceptuje na svém vstupu napěťové úrovně pro logickou jedničku větší, než 2 V, čímž je možné tento obvod připojit na 3,3 V logiku. Vzhledem k tomu, že řídící signály (ADDRESS/DATA, DIR, CLK, IRQ) jsou jednosměrné, je použita čtveřice neinvertujících hradel 74HC08 [13].



Obrázek 8.3: Převodník 3,3 V ↔ 5V pro paralelní sběrnici

8.1.4. Paměti FLASH

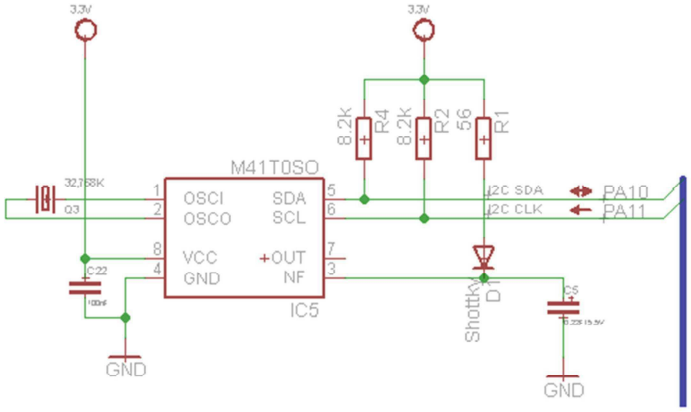
Pro rozšíření paměti jsou použity externí FLASH paměti o kapacitě 32 Mb a standardní SD karta zasunutá do čtečky SD karet o maximální velikosti 2048 MB. Obě paměti jsou připojeny prostřednictvím sběrnice SPI. Schématické zapojení FLASH paměti a pouzdra SD karty jsou znázorněny obrázkem 8.4.



Obrázek 8.4: Paměť FLASH, SD karta

8.1.5. Hodiny reálného času

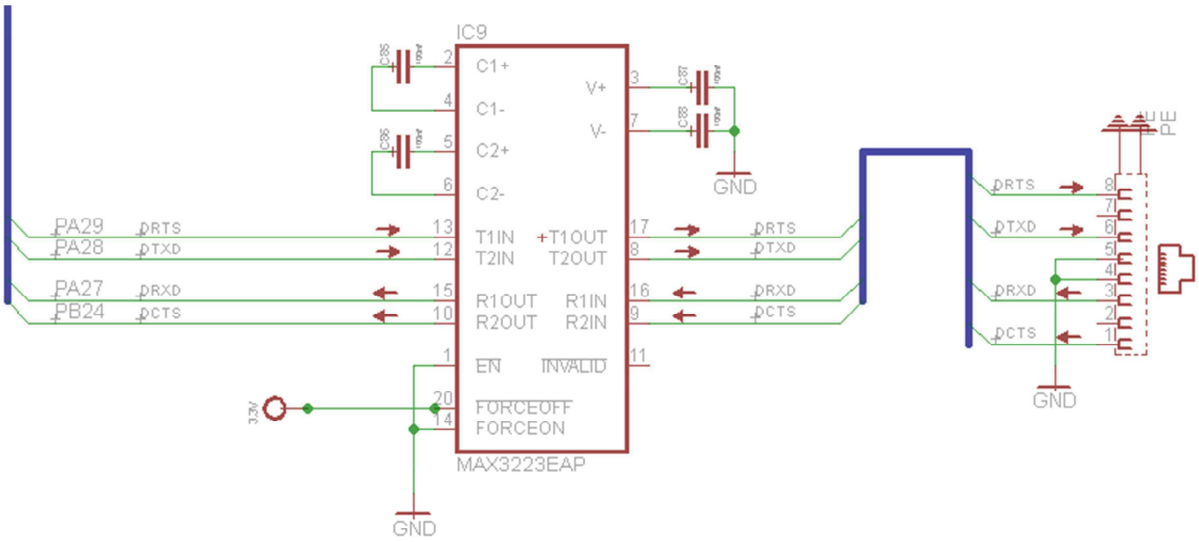
Pro zabezpečení funkce reálného času se využívá knihovna operačního systému EtherNut, která získává reálný čas z internetu. Jako záložní zdroj reálného času je použit interní obvod firmy Maxim (Dallas) DS1802 [6], připojeného k hlavnímu procesoru prostřednictvím sběrnice I^2C. Obvod je řízen hodinovým krystalem 32768 Hz a poskytuje po sériové lince všechny časové údaje (sekundy, minuty, hodiny, dny, měsíce, roky, kalendář, atd.). Schéma zapojení RTC je znázorněno na obrázku 8.5.



Obrázek 8.5: Hodiny reálného času

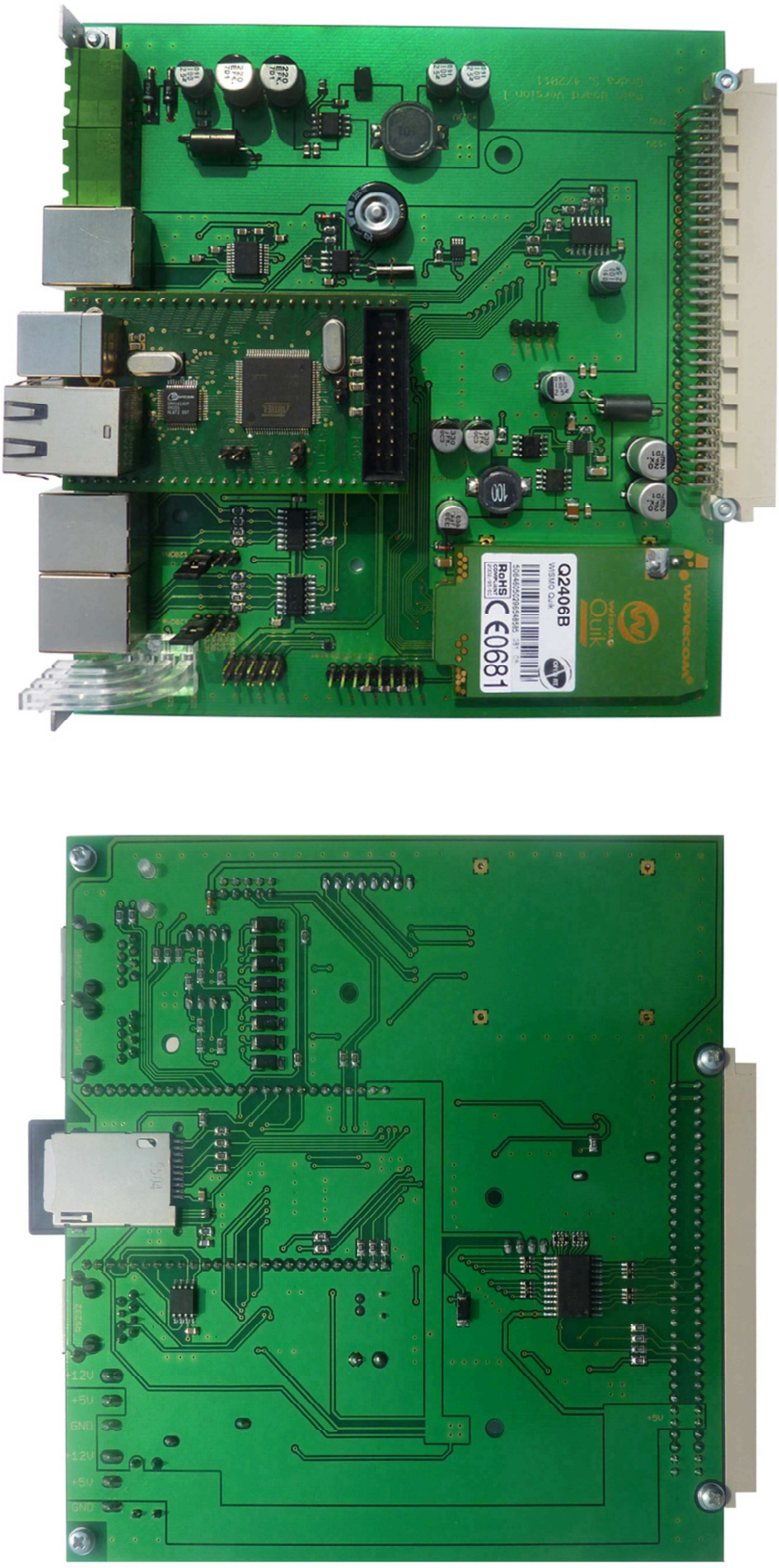
8.1.6. Debugovací rozhraní na RS-232

Pro debugování, nebo pro univerzální použití je jeden ze tří universálních USARTů připojen na převodník TTL RS-232 MAX3223E [11]. Obvod obsahuje dva Tx vysílače a dva Rx přijímače. Obvod je zapojen podle katalogového zapojení výrobce (obrázek 8.6).

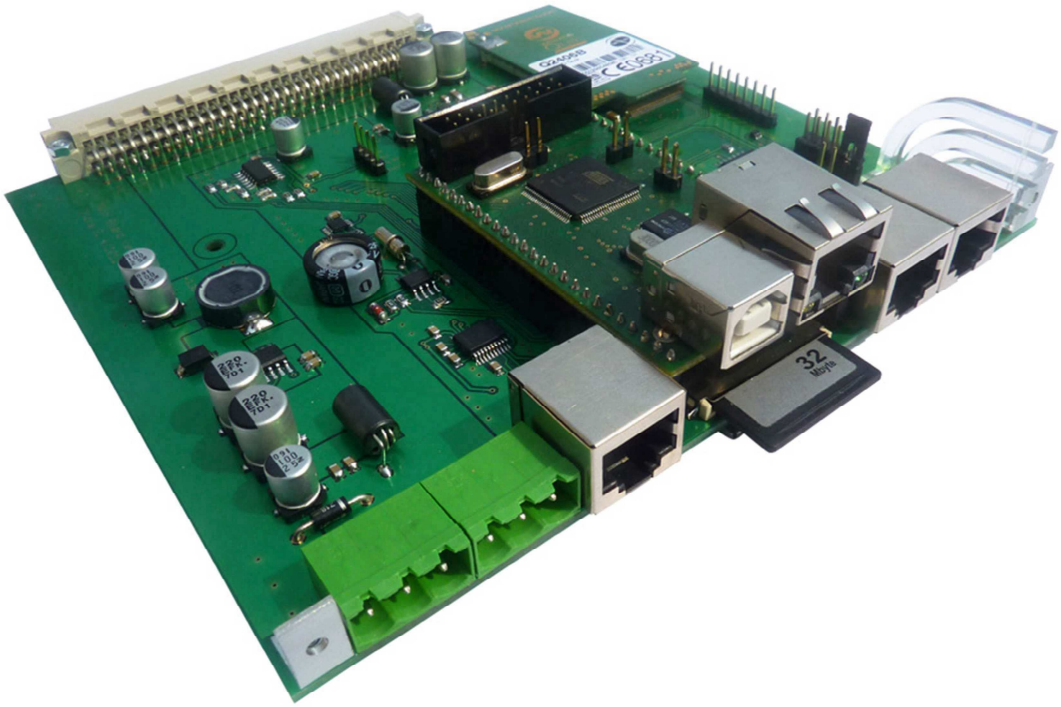


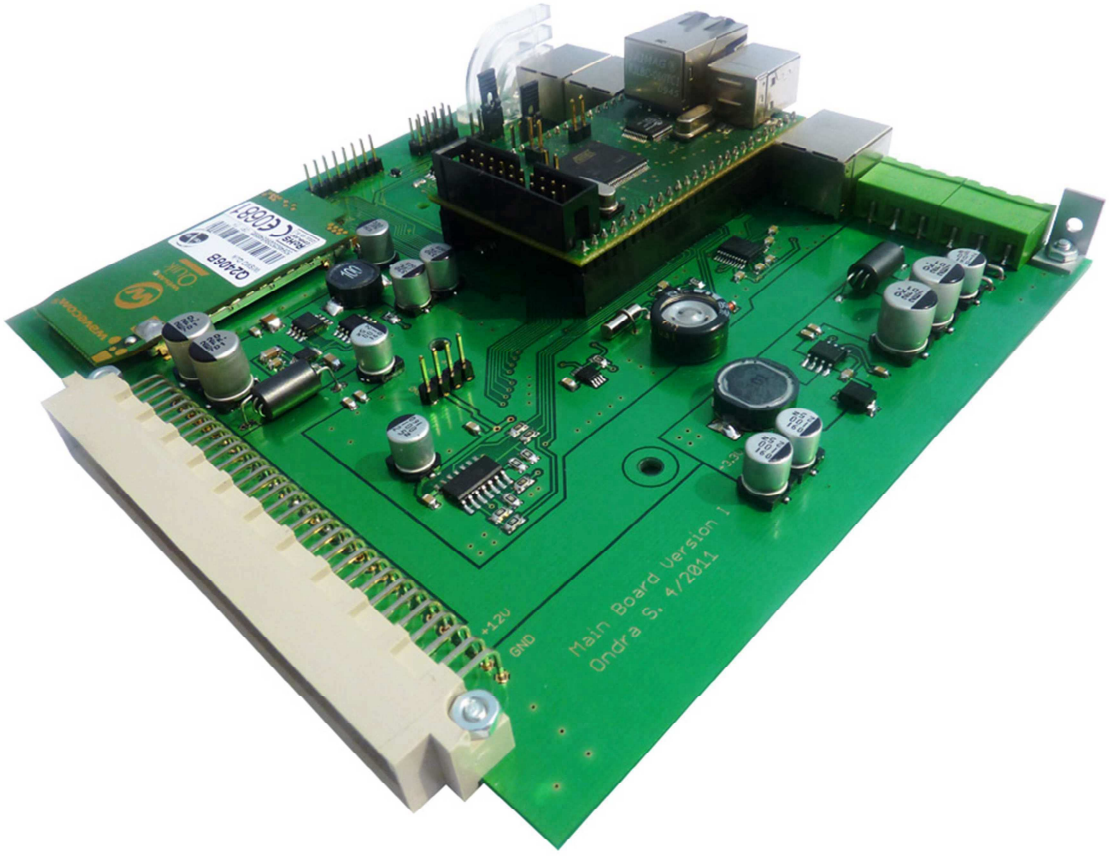
Obrázek 8.6: Debugovací rozhraní na RS-232

Hlavní procesorová deska (mainboard) je vytvořena na oboustranné desce plošných spojů o definovaném formátu 140 x 149 mm, která se zasune do schelfu vybaveného backplainem s paralelní sběrnicí umožňující komunikaci s ostatními paralelními kartami (obrázek 8.7 až 8.10). Deska je vyvinuta v návrhovém prostředí plošných spojů Eagle. Na desce je příprava pro GSM modul G207 , zdroj 3,6 V / 3A MAX1791 [9] a teploměr, komunikující po 1Wire. Tato část není předmětem této práce a je připravena pro další možné rozšíření.



Obrázek 8.7, 8.8: Fotografie osazené a zrealizované hlavní řídící karty





Obrázek 8.9, 8.10: Fotografie osazené a zrealizované hlavní řídící karty

Testování a implementace

Karta hlavní řídící jednotky byla testována s jednou kartou vstupů, kartou výstupů, powermodulem a klávesnicí. Tyto čtyři moduly představují v jádru kompletní a ucelený systém určený pro řízení a ovládání domu.

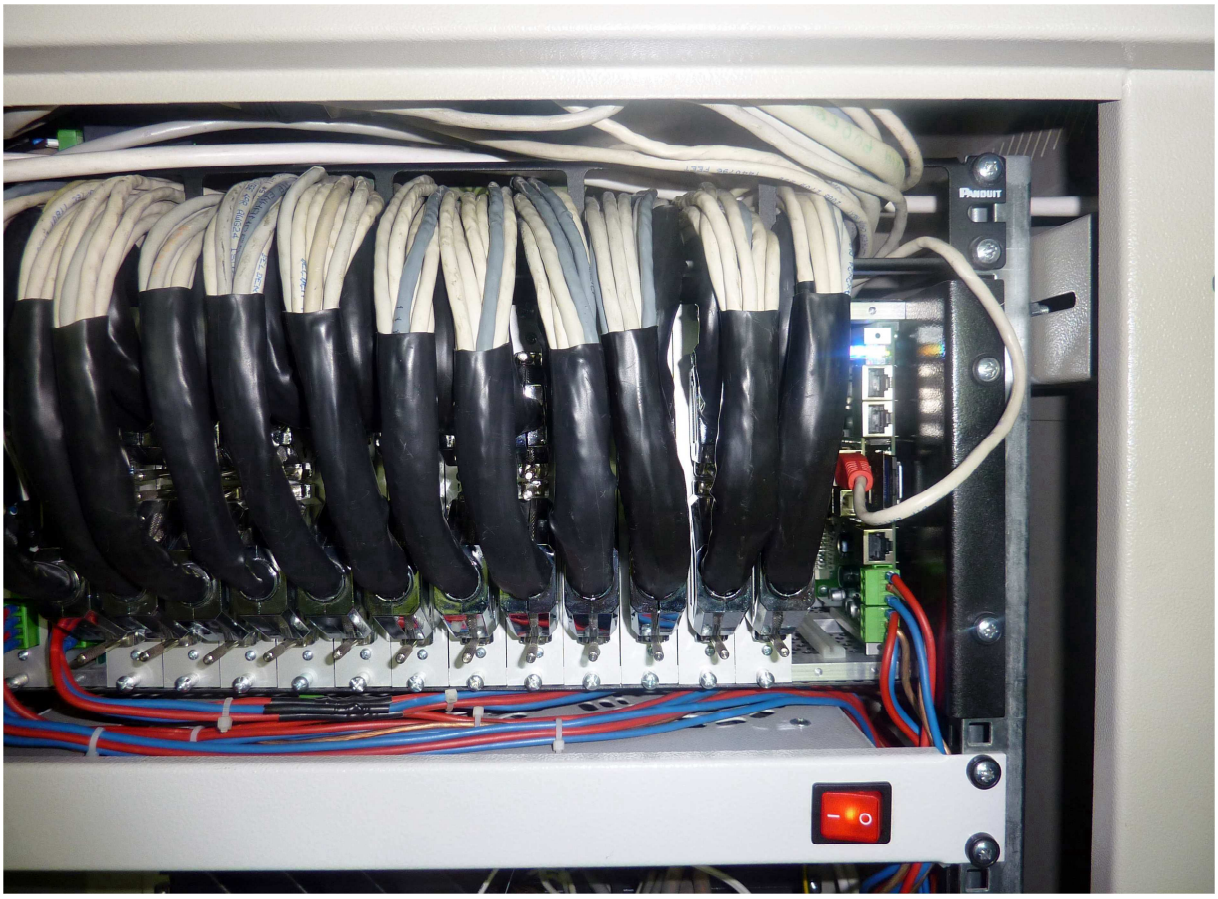
Na uvedeném systému byly provedeny následující měření:

- Komunikace s kartou vstupů
- Komunikace s kartou slaboproudých relé
- Sériová komunikace s kartou silnoproudých relé
- Komunikace s klávesnicí (načtení textů do klávesnice, načtení PINů do klávesnice, autentifikace PINů)
- Ovládání výstupů pomocí vstupů
- Ovládání výstupů pomocí klávesnice
- Komunikace hlavní karty s webovým rozhraním
- Vzdálený přístup přes internet do řídící karty a celého systému
- Konfigurace celého systému přes webové rozhraní
- Ovládání výstupů přes webové rozhraní
- Identifikace nedosažitelnosti a poruchy karty (karta neodpovídá)
- Přepínání mezi sériovými sběrnicemi RS-485

Všechny provedené testy proběhli bez drobných závad a systém je funkční v rozsahu popsané práce.

Následně byla započata implementace nové hlavní řídící karty do již existujícího systému osazeném prozatímním řešením s ATmega 128. Celý systém je implementován v technologických rozvaděčích, viz obrázek 9.1. Bylo provedeno základní měření na tomto rozsáhlém systému a systém pracuje bez technických problémů. Pro finální implementaci nové hlavní řídící jednotky je třeba provést „uživatelskou“ konfiguraci vstupů na výstupy a vyplnit rozsáhlou konfigurační tabulku.

Vzhledem k tomu, že tento systém nelze vypnout na delší dobu než několik hodin a celé nové řešení používá nové komunikační protokoly, bylo přeprogramováno pouze několik karet, aby byla zjištěna souběžná komunikace mezi jednotlivými kartami a hlavní řídící jednotkou.



Obrázek 9.1: Fotografie kompletního systému v 19“ technologickém racku se zasunutou hlavní řídící kartou

Závěr

Navrhl jsem, zrealizoval a naprogramoval kompaktní řídicí systém pro velké objekty (průmyslové budovy, rodinné domy a automatizované celky) umožňující dovyvinout další zařízení komunikující s tímto systémem a umožňující přístup k zařízení prostřednictvím webových stránek. Tohoto cílu se podařilo dosáhnout.

Práce vychází z již existujícího projektu, ze kterého byla převzata pouze hardwarová část komunikačních karet a modulů.

V rámci projektu jsem:

- naprogramoval nový komunikační protokol mezi kartami na paralelní sběrnici
- přeprogramoval všechny karty na paralelní sběrnici
- vytvořil nový komunikační protokol pro sériovou linku
- přeprogramoval všechny powermoduly
- navrhl software klávesnice s displejem a připojil ji po sériové lince RS-485 k systému
- navrhl, zkonstruoval a oživil hlavní řídící jednotku celého systému vč. návrhu PCB
- vytvořil jsem software pro hlavní řídící jednotku pro komunikaci po sériové a paralelní sběrnici, komunikaci po internetu, webové stránky systému umožňující ovládání, konfiguraci a nastavení celého systému

Řídicí jednotku jsem osadil jednočipovým procesorem AT91SAM7X512 s jádrem ARM 7 od firmy Atmel. Jako operační systém jsem použil systém EtherNut. Hlavním srdcem celého systému je modul AT91SAM7X512-KIT obsahující již zmíněný procesor a hardware pro komunikaci po ethernetu. Celý hardware jsem postavil na oboustranné desce realizované prostřednictvím programu Eagle o rozměrech 140x149mm. Tato hlavní řídicí deska je určena k zasunutí do technologického racku obsahující velké množství paralelních vstupních a výstupních karet. Desku jsem dále vybavil budiči RS-485 a RS-232 určenými pro sériovou komunikaci s dalšími připojenými kartami (zařízeními). Deska obsahuje spínaný napájecí zdroj 3,3V, indikační LED a paticí pro SD karty (o maximální kapacitě 2 GB).

Program se skládá z driverů pro komunikaci po sériové a paralelní sběrnici (v tento okamžik jsou podporované paralelní karty vstupů, paralelní karty výstupů, sériové karty powermodulů a klávesnice), konfigurace systému, matrix procesu (vypočítávající všechny kombinační a sekvenční funkce) a z webových stránek.

Paralelní a sériová komunikaci jsem přeprogramoval a použil nový bezpečnější komunikační protokol (přeprogramoval jsem karty celého systému).

Uživatelské rozhraní v externích klávesnicích jsem doplnil a přeprogramoval, aby byla zajištěna lepší uživatelská komunikace. Při programování klávesnice jsem se seznámil s procesorem PIC16F877A od firmy Microchip.

Jako programovací jazyk celého systému jsem použil jazyk C. Pro programování webových stránek jsem použil šablonovací systém XSL.

Byla započata implementace celého systému do již existujícího řídícího systému, skládajícího se z dvanácti karet vstupů (288 vstupů), z třech karet slaboproudých relé (celkem 45 slaboproudých výstupů), osmi powermodulů (celkem 128 výkonových relé) a třech klávesnic.

Z důvodu dalšího možného rozšíření celého systému a zkvalitnění externí komunikace je na desce umístěn zdroj 3,6V a GSM modul (včetně patice na SIM kartu), který je připojen k hlavnímu procesoru prostřednictvím USART. Dále je zde teploměr pro informativní měření teploty systému prostřednictvím 1Wire a externí paměť FLASH o kapacitě 32Mbitů. Tato část je připojena pro další vývoj celého systému.

Seznam použitých zkratek

ARM	Advanced RISC Machine
ARP	Address Resolution Protocol
CGI	Common Gateway Interface
CSS	Cascading Style Sheets
DHCP	Dynamic Host Configuration Protocol
DMA	Dynamic Memory Access
DNS	Domain Name System
EEPROM	Electrically Erasable Programmable Read-Only Memory
FAT	File Allocation Table
FIFO	First Input First Output
GZIP	GNU Zip
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
ICMP	Internet Control Message Protocol
JS	Java Script
MAC	Media Access Control
MIPS	Millions of Instructions Per Second
MMC	MultiMedia Cards
PIO	Programmed Input Output
PPP	Point-to-Point Protocol
RAM	Radnom Access Memory
RTC	Real-time Clock
SD	Secure Digital
SEO	Search Engine Optimization
SLIP	Serial Line Internet Protocol
SNTP	Simple Network Time Protocol
SRAM	Static Random Access Memory
TCP	Transmission Control Protocol

TTL	Transistor Transistor Logic
UART	Universal Asynchronous Receiver and Transmitter
UDP	User Datagram Protocol
USART	Universal Synchronous / Asynchronous Receiver and Transmitter
USB	Universal Serial Bus
W3C	World Wide Web Consortium
WWW	World Wide Web
XHTML	eXtensible Hypertext Markup Language
XML	eXtensible Markup Language
XSL	eXtensible Stylesheet Language

Zdroje

1. **Atmel**. Atmel Comporation. ATmega128. [Online] 2011. [Citace: 21. 4 2011.] http://www.atmel.com/dyn/resources/prod_documents/doc2467.pdf.

2. **Atmel**. Atmel Comporation. Atmel256. [Online] 2011. [Citace: 21. 4 2011.] http://www.atmel.com/dyn/resources/prod_documents/doc2549.pdf.

3. **Atmel**. Atmel Comporation. AT91RM9200. [Online] 2009. [Citace: 21. 4 2011.] http://www.atmel.com/dyn/resources/prod_documents/doc1768.pdf.

4. **Atmel**. Atmel Comporation. AT91SAM7X. [Online] 2008. [Citace: 21. 4 2011.] http://www.atmel.com/dyn/resources/prod_documents/doc6120.pdf.

5. **Atmel**. Atmel Comporation .Atmel Products. [Online] [Citace: 5. 5 2011.] http://www.atmel.com/dyn/products/tools_card.asp?tool_id=3883.

6. **Atmel**. Atmel Comporation. DS1802. [Online] [Citace: 21. 4 2011.] http://datasheets.maxim-ic.com/en/ds/DS1802.pdf.

7. **Catalog, Datasheet**. Datasheet Catalog. Datasheet Catalog.com. [Online] 1998. [Citace: 21. 4 2011.] http://www.datasheetcatalog.org/datasheet/philips/74HC_HCT245_CNV_2.pdf.

8. **Kramára, Petr**. AT91SAM7X512_KIT. Kramara s.r.o. vývoj a výroba elektroniky. [Online] 1. 10 2009. [Citace: 6. 11 2010.] http://www.kramara.com.

9. **MAXIM**. MAXIM Inovation Delivered. MAX1791. [Online] 2001. [Citace: 21. 4 2011.] http://datasheets.maxim-ic.com/en/ds/MAX1791EVKIT.pdf.

10. **MAXIM**. MAXIM Innovation Delivered. MAX3070/73/76. [Online] 2010. [Citace: 21. 4 2011.] http://datasheets.maxim-ic.com/en/ds/MAX3070E-MAX3079E.pdf.

11. **MAXIM**. MAXIM Inovation Delivered. MAX3223E. [Online] 2009. [Citace: 21. 4 2011.] http://datasheets.maxim-ic.com/en/ds/MAX3223EEVKIT.pdf.

12. **MAXIM**. MAXIM Inovation Delivered. MAX5035. [Online] 2010. [Citace: 21. 4 2011.] http://datasheets.maxim-ic.com/en/ds/MAX5035.pdf.

13. **NXP**. NXP Semiconductors. 74HC08. [Online] 2003. [Citace: 21. 4 2011.] http://www.nxp.com/documents/data_sheet/74HC_HCT08.pdf.

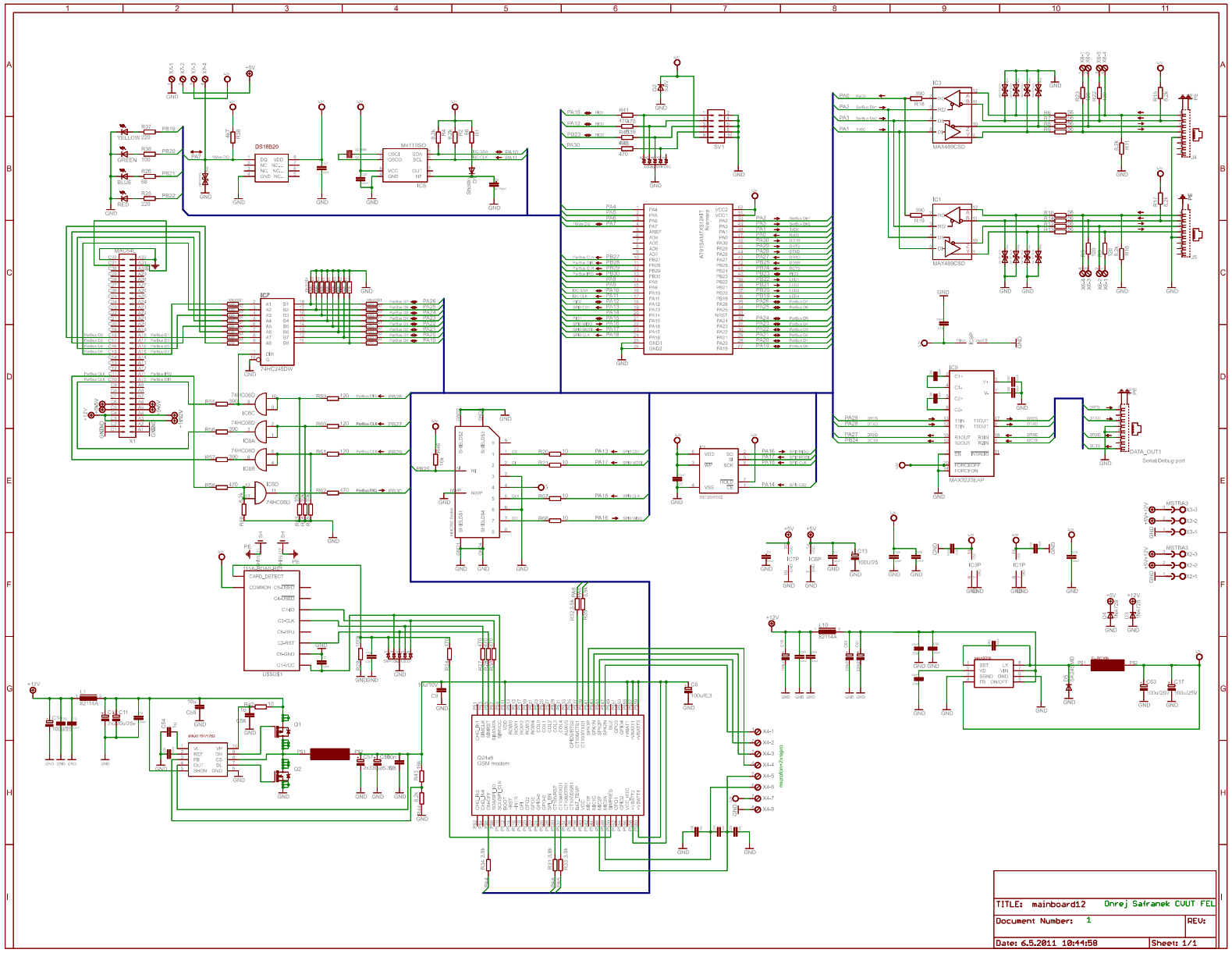
14. **Redakce**. Co to je Ethernut. HW Server. [Online] HW Server, 28. 12 2004. [Citace: 6. 11 2010.] http://www.hw-group.com.

15. **RFC-Ref**. RFC-Ref. [Online] 1988. [Citace: 20. Duben 2011.] http://rfc-ref.org/.

16. **YAGARTO**. YAGARTO. [Online] [Citace: 5. 5 2011.] http://www.yagarto.de/.

Přílohy

1. Schéma zapojení hlavní řídící jednotky
2. Deska s plošnými spoji hlavní řídící jednotky 140x149 mm vytvořená v programu Eagle – TOP
3. Deska s plošnými spoji hlavní řídící jednotky 140x149 mm vytvořená v programu Eagle – BOTTOM


TITLE: mainboard12 Onrej Safranek CUUT FEL
Document Number: 1
REV:
Date: 6.5.2011 10:44:58
Sheet: 1/1