

Embedded systém pro monitorování topných soustav

René Panák

Bakalářská práce
2015

 Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
akademický rok: 2014/2015

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **René Panák**
Osobní číslo: **A12595**
Studijní program: **B3902 Inženýrská informatika**
Studijní obor: **Informační a řídicí technologie**
Forma studia: **kombinovaná**

Téma práce: **Embedded systém pro monitorování topných soustav**
Téma anglicky: **An Embedded Device for Monitoring Heating Systems**

Zásady pro vypracování:

1. Vytvořte literární rešerši na téma problematika HW a SW implementace embedded systémů vhodných pro vzdálené řízení a monitoring technologických procesů.
2. Prozkoumejte aktuální stav (state-of-the-art) dostupných řešení.
3. Analyzujte možnosti využití vhodných embedded procesorů obsahujících HW/SW podporu přenosů dat prostřednictvím protokolů TCP/IP nebo WIFI.
4. Navrhněte a realizujte vzorovou implementaci embedded systému, založeného na vhodném embedded procesoru, umožňujícím vzdálený monitoring měřených reálných veličin (topného systému) s možností vizualizace měřených průběhů prostřednictvím zabudovaného HTTP serveru nebo jiné vhodné mobilní technologie.

Rozsah bakalářské práce:

Rozsah příloh:

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

1. FAI UTB ZLIN. CodeDesigner RAD website [online]. 2015 [cit. 2015-01-28].
Dostupné z: <http://codedesigner.org/>
2. BLIŽŇÁK, Michal, Tomáš DULÍK, Roman JAŠEK a Pavel VAŘACHA. Optimized Production-Ready Source Code Generation Based on UML [online].
INTERNATIONAL JOURNAL OF SYSTEMS APPLICATIONS, ENGINEERING DEVELOPMENT: NAUN Press, 2013, roč. 7, č. 1 [cit. 2014-01-23]. ISSN 2074-1308.
Dostupné z: <http://www.naun.org/main/UPress/saed/16-498.pdf>
3. BLIŽŇÁK, Michal, Tomáš DULÍK a Roman JAŠEK. Production-Ready Source Code Round-Trip Engineering [online]. INTERNATIONAL JOURNAL OF COMPUTERS: NAUN Press, 2012 [cit. 2014-01-23]. ISSN 1998-4308. Dostupné z: <http://www.naun.org/wseas/cms.action?id=3036>
4. BARR, Michael. Programming embedded systems in C and C++. 1st ed. Sebastopol, Calif.: O'Reilly, c1999, xvii, 174 p. ISBN 15-659-2354-5.
5. BERNARDO, Marco a Flavio CORRADINI. Formal methods for the design of real-time systems: International School on Formal Methods for the Design of Computer, Communication, and Software Systems, SFM-RT 2004, Bertinoro, Italy, September 13-18, 2004 : revised lectures. New York: Springer, c2004, vi, 293 p. Lecture notes in computer science, 3185. ISBN 35-402-3068-8.
6. LI, Qing. Real-time concepts for embedded systems. San Francisco: CMP Books, 2003, xii, 294 s. ISBN 15-782-0124-1.

Vedoucí bakalářské práce:

Ing. Michal Bližňák, Ph.D.

Ústav informatiky a umělé inteligence

Datum zadání bakalářské práce:

6. března 2015

Termín odevzdání bakalářské práce:

22. května 2015

Ve Zlíně dne 6. března 2015



doc. Mgr. Milan Adámek, Ph.D.
děkan



prof. Ing. Vladimír Vašek, CSc.
ředitel ústavu

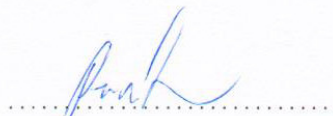
Prohlašuji, že

- beru na vědomí, že odevzdáním bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že bakalářské práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky. Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – bakalářskou práci nebo poskytnout licenci k jejímu využití jen s příjím-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně



podpis autora

ABSTRAKT

Tato práce se zabývá návrhem embedded systému pro monitorování topných soustav. Tato práce má za cíl prozkoumat možnosti (vzdáleného) monitorování systémů v budovách prostřednictvím lokální počítačové sítě a globální sítě internet.

Teoretická část se zabývá problematikou implementace hardwaru a softwaru a zkoumá vlastnosti dostupných mikročítačů použitelných pro tyto účely.

Praktická část navrhuje architekturu systému s ohledem na rozšiřitelnost ve smyslu počtu monitorovaných bodů či rozšíření funkčnosti. Součástí zpracování je i návrh softwarového vybavení jednotlivých prvků a návrh vizualizace stavu monitorované soustavy. Výsledkem praktické části je ukázková implementace základních prvků navrhovaného systému.

Klíčová slova: embedded zařízení, mikročítač, počítačová síť, HTTP protokol.

ABSTRACT

This work describes the design of embedded systems to monitor heating systems. This work aims to explore the possibilities of (remote) monitoring systems in buildings through local area networks and global Internet network.

The theoretical part deals with the implementation of hardware and software, and examines the characteristics of available microprocessors useful for these purposes.

The practical part proposes a system architecture with regard to scalability in terms of number of monitoring points or functionality expand. Part of processing is also the draft of the software for individual components and design of visualization of the monitored system status. The result is an example of the practical implementation of the basic elements of the proposed system.

Keywords: embedded device, microcontroller, computer network, HTTP protocol.

Rád bych na tomto místě poděkoval svému vedoucímu práce, Ing. Michalu Bližňákovi, Ph.D. cenné rady, připomínky a čas, který mi věnoval v průběhu psaní této práce.

OBSAH

ÚVOD	9
I TEORETICKÁ ČÁST	10
1 ÚVOD DO MONITORINGU	11
1.1 PROBLEMATIKA HW IMPLEMENTACE.....	11
1.1.1 Vlastnosti binárních vstupů/výstupů	11
1.1.2 Vlastnosti analogových vstupů (AD převodníku).....	11
1.1.3 Sběrnice a rozhraní.....	12
1.2 PROBLEMATIKA SW IMPLEMENTACE	13
2 DOSTUPNÁ ŘEŠENÍ	15
2.1 PRŮMYSLOVÁ ŘEŠENÍ.....	15
2.2 ELECTRIC IMP	15
2.3 NEST	16
3 NABÍDKA PROCESORŮ	17
3.1 ATMEGA328 (ARDUINO MINI)	17
3.2 STM32F103C8	17
3.3 ESP8266	17
3.4 CUBIEBOARD2	18
3.5 BANANA PI / PRO	18
II PRAKTICKÁ ČÁST	19
4 CÍLE PROJEKTU	20
5 NÁVRH REALIZACE	21
5.1 NÁVRH ZAPOJENÍ DO SÍTĚ.....	21
5.2 NÁVRH ZPŮSOBU SBĚRU DAT	21
5.2.1 Koncové body jako servery	21
5.2.2 Klienti webové služby.....	21
5.2.3 Klienti centrálního prvku.....	22
5.2.4 Smíšená struktura	22
5.3 NÁVRH HARDWARU	23
5.3.1 Hardware koncových bodů.....	23
5.3.2 Terminál.....	25
5.3.3 Hardware centrálního prvku.....	25
5.4 NÁVRH SOFTWARE	26
5.4.1 Software koncových bodů	26

5.4.2	Koncový bod jako klient	27
5.4.3	Koncový bod jako server	28
5.4.4	Software centrálního prvku	28
5.4.5	Uživatelská část centrálního prvku.....	29
5.4.6	Funkční část centrálního prvku	29
5.4.7	Server v internetu.....	31
6	VZOROVÁ IMPLEMENTACE	32
6.1	KONCOVÝ BOD S ETHERNETEM	32
6.2	KONCOVÝ BOD S WIFI	32
6.3	CENTRÁLNÍ PRVEK	33
6.4	NÁVRHY NA ROZVOJ	34
	ZÁVĚR.....	35
	SEZNAM POUŽITÉ LITERATURY	36
	SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK	38
	SEZNAM OBRÁZKŮ	39
	SEZNAM PŘÍLOH	40

ÚVOD

Monitorování různých technologických procesů je staré jako samy tyto procesy. Ovšem způsoby jakými jsou monitorovány se v průběhu času spolu s vývojem techniky mění. Na začátku bylo pozorování lidmi, sledovali proces a snažili se odhadnout jeho stav. Poté se objevily různé měřicí přístroje (teploměry, tlakoměry ...) které již dokázaly člověku říci přesněji co se v procesu děje. Člověk však stále musel být u procesu přítomen aby mohl údaje z měřících zařízení číst. Měřicí přístroje se dalším vývojem změnily z mechanických na elektronické, takže bylo možné přenášet měřené hodnoty pomocí elektronických signálů z různých částí procesu na jedno místo, kontrolní panel. Dalším vývojem se dostaly měřené hodnoty až do řídicího centra, ovšem vytvoření takového monitorovacího systému bylo nejen velmi nákladné ale také náročné, nemluvě o hledání poruch na tomto systému. S nástupem digitálních technologií a počítačů se celý systém podstatně zjednodušil. K přenosu informací se využívá počítačová síť a díky připojení této sítě s internetem je možné proces sledovat i vzdáleně, odkudkoliv ze světa.

To všechno se týkalo průmyslových technologických procesů. Ovšem monitorovat lze i budova jako taková, či její vybavení (například právě vytápěcí systém). V této oblasti bylo vymyšleno mnoho a zabývají se tím velké a výrobci programovatelných automatů. Kdyby snad někoho napadlo využít toto průmyslové řešení v domácnosti, rychle by od svého záměru upustil ihned po zjištění ceny. Ano, tato řešení jsou pro běžné uživatele nepředstavitelně drahá.

S postupujícím časem a vývojem výpočetní techniky se objevují dostupná řešení pro domácnosti která přenáší automatizační prvky z průmyslu do domácnosti („home automation“ nebo „smart home“ - chytrý domov). Tato řešení většinou využívají myšlenku IoT, tedy že zařízení je prostřednictvím internetu připojeno ke cloudovému serveru který zajišťuje onen vzdálený přístup. V jedné z kapitol této práce jsou některá řešení popsána.

Společným problémem těchto řešení je absolutní závislost na přístupu k internetu a na provozu cloudového serveru. To mě přivedlo k myšlence jakým způsobem se této závislosti alespoň částečně zbavit. Nejde ani tak o nalezení nového komunikačního kanálu pro vzdálený monitoring či ovládání, ale o zajištění základních funkcí systému i v případě výpadku spojení. Cílem práce je tedy prozkoumat možnosti a navrhnout takový systém a jeho část realizovat.

I. TEORETICKÁ ČÁST

1 ÚVOD DO MONITORINGU

1.1 Problematika HW implementace

Chceme-li sledovat (monitorovat) nějaký technologický proces pomocí výpočetní techniky, je potřeba použít hardware k tomu určený. Za výpočetní techniku v tomto případě můžeme považovat obecný mikropočítač. Jaké předpoklady by měl splňovat takový mikropočítač? Aby hardware mohl nějakým způsobem interagovat s vnějším prostředím, musí obsahovat vstupní/výstupní piny. Tyto piny, pokud je mikropočítač obsahuje, jsou ve výchozím stavu binární. Můžeme je tedy použít k zjišťování stavu typu „svítí/nesvítí“, což je pro komplexní monitoring nedostačující. V technologických procesech potřebujeme monitorovat i hodnoty vícecestavové (spojité). Aby byl mikropočítač, jakožto diskrétní prvek, schopen tyto hodnoty zaznamenat musí být vybaven analogově-digitálním převodníkem. Moderní mikropočítače jsou již tímto převodníkem vybaveny, avšak ne vždy vlastnosti převodníku splní požadavky které máme při monitorování dané veličiny.

1.1.1 Vlastnosti binárních vstupů/výstupů

Základní vlastností binárních vstupů je maximální rozsah napětí a oblasti napětí pro log0 a log1. Typicky jsou využívány napěťové úrovně TTL logiky (log0: 0V až 0.5V, log1: 2.4V až 5V). Ovšem maximální napětí které jsou schopné binární vstupy snést závisí na konkrétním mikropočítači, typicky jsou rovny napájecímu napětí. Máme-li mikropočítač s napájecím napětím 3.3V, je třeba konzultovat s katalogovým listem daného procesoru, zda je tzv. 5V tolerant. Přivedením většího napětí by mohlo dojít k poškození daného vstupu či zničení celého mikropočítače! Proto je třeba důkladně zvážit zda nepoužít převodník úrovní (5V na 3.3V). Není-li zaručeno že se během provozu na vstup nedostane vyšší napětí, je vhodné vstup vybavit příslušnou ochranou. Jako ochranu je možné použít diodu, tranzistor nebo optočlen).

U binárních výstupů je podstatnou vlastností nejen napětí při stavu „sepnuto“, ale také maximální proud který může být spínaným obvodem odebírán. Přetížíme-li tento binární výstup, dojde k jeho poškození až zničení! Proto je ke spínání větších proudů (> 10mA) vhodné použít tranzistor. Tento tranzistor je vhodné opatřit ochrannou diodou, obzvláště při spínání indukčních prvků jakým je třeba relé.

1.1.2 Vlastnosti analogových vstupů (AD převodníku)

Důležitá vlastnost analogově-digitálních převodníků je rozlišení (bitová hloubka). Počet bitů převodníku určuje na kolik hladin se rozdělí úroveň referenčního napětí. Výsledná číselná hodnota je číslem hladiny do které dosáhne měřené napětí. I u analogově-

digitálního převodníku je podstatnou vlastností maximální napětí. V případě převodníku integrovaného na čipu mikropočítače je maximální napětí shodné s maximálním napětím binárního vstupu, není však zaručeno, že je to nejvyšší napětí které lze změřit. Nejvyšší hodnota napětí které lze převodníkem změřit je shodné s hodnotou referenčního napětí. Snížením hodnoty referenčního napětí lze zvýšit přesnost převodu, referenční napětí se rozdělí na užší úrovně. Zároveň nelze rozlišit velikost napětí nad hodnotou referenčního. Tedy lze-li připojit vlastní referenční napětí, musí splňovat požadavky na maximální povolené napětí a musí mít stabilní přesně známou hodnotu. Běžně používaná bitová hloubka ADC v mikropočítačích je 10 či 12 bitů, což při referenčním napětí 5V tvoří krok přibližně 4.9mV při 10bit a přibližně 1.2mV při 12bit. Pokud by pro danou aplikaci byly tyto parametry nedostatečné (bitová hloubka nebo napěťový rozsah), lze pomocí některého rozhraní připojit převodník externí.

1.1.3 Sběrnice a rozhraní

Samotná existence binárních či analogových vstupů/výstupů ještě nezaručuje možnost připojení všech typů snímačů. Někdy je potřeba připojit specializovaný snímač s jinými napěťovými úrovněmi nebo snímač s AD převodníkem s vyšším rozlišení. Některé „chytré“ senzory poskytují hodnotu nebo i více hodnot najednou pomocí některého rozhraní.

I2C I2C, někdy také TWI (two wire interface), je sériová datová sběrnice využívající dva vodiče. První vodič označovaný jako SDA slouží k přenosu dat jedním nebo druhým směrem. Druhý vodič označovaný jako SCL je nositelem hodinového signálu podle kterého je řízen chod sběrnice. Na sběrnici musí být přítomen alespoň jeden prvek v režimu master. Každé zařízení na sběrnici má svoji adresu o délce 7 bitů nebo 10 bitů.

SPI SPI je taktéž sériová datová sběrnice. Využívá tři společné vodiče a jeden výběrový vodič pro každé připojené zařízení. Na sběrnici musí být právě jeden master prvek. Funkce vodičů jsou následující: SCK (serial clock) - hodinový signál, MISO (master input, slave output) - data směrem k master zařízení, MOSI (master output, slave input) - data směrem z master zařízení, CS (chip select) nebo SS (slave select) - aktivace slave zařízení. Na této sběrnici je možné, díky dvěma datovým vodičům, provozovat synchronní přenos dat mezi masterem a slavem.

RS232 RS232 je sériové asynchronní rozhraní. Toto rozhraní je určeno pro komunikaci s jiným zařízením, pro připojení periferních zařízení se spíše nevyužívá. Základem toho rozhraní jsou tři vodiče: RxD, TxD a GND. Dříve se používaly ještě další vodiče pro hardwarové řízení toku dat, dnes se lze setkat s použitím vodičů RTS (request to

send) a CTS (clear to send). Původní verze rozhraní využívá symetrické napětí, -25V až -3V pro log1 a 3V až 25V pro log0. Při použití nejvyššího napětí lze provozovat komunikaci pomocí tohoto rozhraní až na vzdálenost 20 metrů. Má-li mikropočítač integrováno rozhraní RS232, používá zcela jistě napěťové úrovně TTL. Kvůli tomuto nedokáže dosáhnout takové vzdálenosti komunikace, navíc nelze přímo připojit k běžné sériové lince (hrozí zničení mikropočítače!). Existují však převodníky úrovní pro tento účel. V dnešní době jsou sériové linky obsažené v běžném PC spíše vzácností, používají se tedy převodníky například na sběrnici USB které vytvoří virtuální sériový port. Napětí sériové linky poskytované USB převodníkem dosahují úrovně TTL (5V nebo 3.3V).

1.2 Problematika SW implementace

problematika SW implementace

Při návrhu softwaru je třeba brát v úvahu použitý hardware. Těžko můžeme počítat s využitím vyššího operačního systému na jednoduchých mikropočítačích. Hlavní obtíží se kterou se setkáme je „paralelní“ zpracování. Čekáme-li jako server na klienta, nemůžeme zároveň v pravidelných intervalech zjišťovat hodnoty měřených veličin. Musíme tyto činnosti vhodně přepínat. Zároveň musíme zajistit aby žádná z činností netrvala příliš dlouho a nebrzdila tak ostatní. Existují operační systémy určené i pro drobné mikropočítače, ale jejich použití není jediná možnost jak správného časování a přepínání úloh dosáhnout. Pro jednodušší aplikace, u kterých nejsou požadovány reakce jako u hard real-time systémů lze použít přepínání úloh pomocí stavů a podmínek pro přechod mezi stavy. Je možné tuto přepínací strukturu sestavit ručně, či použít sofistikovaného návrhového softwaru.

Jedním z takových nástrojů je CodeDesigner [1]. Tento program umožňuje sestavit aplikaci graficky pomocí jejího stavového diagramu. Prvním krokem je nadefinování stavů ve kterých se může program ocitnout. Dalším krokem je návrh podmínek pro přechody mezi jednotlivými stavy. Lze také nadefinovat jaká činnost bude prováděna programem během setrvání v určitém stavu. Po dokončení návrhu stavového diagramu program provede kontrolu zda diagram obsahuje právě jeden výchozí stav, zda ze všech stavů vede nejvýše jeden bezpodmínečný přechod, zda obsahuje diagram alespoň jeden koncový stav, zda ze stavu nevedou dvě totožné podmínky na rozdílné stavy a zda splňuje další kritéria [2]. Po rozhodnutí že je diagram validní, provede program jeho optimalizaci. Sloučení paralelních přechodů, negace podmínek které nespouští žádnou akci, sloučení stavů mezi nimiž není přechodová podmínka a podobně. Po dokončení optimalizace je provedeno vygenerování zdrojového kódu kostry programu včetně definice použitých metod a proměnných. V nastavení generování lze zvolit pro jaký programo-

vací jazyk má být zdrojový kód generován a vybrat jaká konstrukce má být použita pro realizaci přechodů mezi stavy. Software podporuje generování tří typů přechodových algoritmů, Go-To, Loop-Case a Else-If.

2 DOSTUPNÁ ŘEŠENÍ

2.1 Průmyslová řešení

Existuje velké množství průmyslových monitorovacích a ovládacích systémů. Příkladem budiž systém CompactRIO společnosti National Instruments [4]. Jedná se o modulární systém určený pro monitorování a řízení libovolného technologického procesu. Ano, je možné jimi monitorovat a řídit i topné soustavy rodinných domů, ale jelikož se pořizovací náklady takového systému se pohybují v desítkách tisíc, jen těžko by je někdo použil tímto způsobem. Tyto systémy jsou jak z důvodů finančních, tak svým průmyslovým zaměřením, pro domácí použití naprosto nevhodné.

Existuje však i mnoho platform zaměřujících se právě na domácnosti.

2.2 Electric Imp

Electric imp přímo neřeší monitorování technologických procesů, ale poskytuje hardwarovou a softwarovou platformu pro IoT. Nabízí několik variant zařízení, které lze snadno připojit k internetu přes WiFi a automaticky se spojí s cloudem platformy, který umožňuje vzdálenou správu tohoto zařízení.

Takovéto zařízení ještě nic neumí. Má pouze „operační systém“ a pojení s cloudem, ale vlastní funkčnost je potřeba naprogramovat. Programování probíhá například přes tzv. WebIDE které je součástí cloudu. Umí napsaný program přes internetové spojení nahrát do zařízení a spustit. Tedy k tomu aby byl programátor schopen nahrát nový program, není nutná jeho fyzická přítomnost u zařízení.

Jak jsem již zmínil, není to hotové řešení, ale hotová platforma pro vývoj. Při použití této platformy není potřeba řešit připojení k internetu, časování procesů, atd. Programátor se zabývá pouze svou přidanou hodnotou a využívá hotové funkce a služby vestavěného operačního systému.

Jedinný zásah do zařízení, při kterém je potřeba přítomnost u zařízení, je nastavení přístupových údajů k WiFi a k účtu na cloudu. K nastavení stačí pouhý tablet nebo chytrý telefon. Electric imp využívá metodu BlinkUp, kde speciální aplikace pro iOS nebo Android „nabliká“ nastavení do zařízení [5]. Vstupem pro tuto aplikaci jsou údaje které chceme vložit do zařízení. Aplikace použije display telefonu nebo tabletu jako optický vysílač pomocí něhož přenesa data do přiloženého zařízení. Pokud se zařízení úspěšně připojí k síti a ke cloudu, není pro softwarové úpravy potřeba žádný kontakt.

Vypadá to skvěle a jednoduše, což pro jednoduché aplikace využívající jednoho zařízení také je také levné. Pokud je však třeba v domě rozmístit více zařízení jakožto chytré senzory, začíná se toto řešení značně prodražovat. Další nevýhodou tohoto řešení je absolutní závislost na připojení k internetu.

2.3 Nest

Dalším řešením domácího monitoringu je Nest [11]. Jedním z produktů pod tímto názvem je Nest Thermostat. Jedná se o takzvaný chytrý termostat. Připojí se k internetu pomocí bezdrátové WiFi sítě a dovolí sledovat či nastavovat teplotu pomocí počítače či chytrého telefonu kdekoli kde je přístup k internetu. Kromě termostatu nabízí i detektor kouře a oxidu uhelnatého (Nest Protect) nebo například kameru (Dropcam). Tato kamera reaguje na podněty jako je pohyb či informace od detektoru kouře a oxidu uhelnatého. Video v HD kvalitě je streamováno přímo na cloudový server. Vše lze sledovat v internetovém prohlížeči či v Nest App v chytrém telefonu.

Se společností Nest spolupracují i další výrobci kteří využívají spojení s Nest Thermostat či s Nest Protect. Jedná se o výrobky jako například chytrý zámek který dá zprávu termostatu že jste opustili dům a ten automaticky sníží teplotu vytápění nebo zastaví klimatizaci.

Společným znakem těchto výrobků je potřeba přístupu k internetu. Například termostat bez přístupu k internetu je schopen funkce, není však schopen zaznamenat průběh teploty v době výpadku. Taktéž Nest Protect (chytrý požární hlásič) vyžaduje připojení k WiFi (a hlavně k internetu). Ocitne-li se v případě problému bez připojení, chová se jako obyčejný požární hlásič, houká ale vy se o tom nedozvíte pokud jej nemůžete slyšet.

3 NABÍDKA PROCESORŮ

3.1 ATmega328 (Arduino Mini)

Arduino Mini je založeno na použití 8-bit Atmel AVR procesoru ATmega328P. Tento procesor může fungovat na frekvenci až 20MHz, avšak v tomto modulu je taktován pouze na 16MHz. Flash paměť pro program má celkový objem 32kB, ale přibližně 2kB zabírá Arduino bootloader. Operační paměť dostupná v tomto mikropočítači činí 2kB. Tento mikropočítač má vestavěnou EEPROM paměť o velikosti 1kB kterou je možné využít například pro uložení různého nastavení.

Procesor obsahuje 8kanálový 10bit AD převodník, ovšem na modulu nemusí být vyvedeny všechny (typicky schází vstupy A6 a A7). Procesor obsahuje rozhraní RS232 v úrovni TTL - lze využít pro programování. Dalším rozhraním je SPI a sběrnice I2C. Na některých výstupních pinech lze provozovat 8-bit PWM.

Výhodou jsou malé rozměry, jednoduchost programování pomocí Arduino IDE a dostupná cena.

3.2 STM32F103C8

Toto je sice samostatný procesor, ale lze najít různé moduly které jej používají. Procesor je 32-bit, založen na jádře ARM Cortex-M3. Maximální frekvence procesoru je 75MHz. Flash paměť pro program má objem 64kB, operační paměť mikroprocesoru činí 20kB. K napájení procesoru je zapotřebí napětí 3.3V, avšak vstupní porty jsou 5V tolerantní. Obsahuje komunikační rozhraní 2 x SPI, 2 x I2C, USB, 3 x UART a CAN. V procesoru jsou vestavěny dva 12-bit AD převodníky pro 10 analogových vstupů. Až 37 GPIO, některé z nich podporují 16-bit PWM.

Díky projektu Maple [6] je možné programovat tento procesor v prostředí podobném Arduino IDE a využít tak knihovny pro Arduino. Pro projekt Maple byl sice použit procesor STM32F103RB, ale od STM32F103C8 se liší pouze velikostí dostupné flash paměti a počtem GPIO.

3.3 ESP8266

Tento modul je primárně použitelný jako WiFi modem pro libovolný mikropočítač. Komunikace probíhá přes rozhraní UART (RS232 TTL) pomocí sady „AT“ příkazů. Základními prvky tohoto modulu je procesor Espressif ESP8266, který obsahuje kompletní WiFi přijímač/vysílač, a 512kB SPI flash paměť s firmwarem.

Tento modul se vyrábí ve více hardwarových variantách označovaných ESP-xx. Liší se např. anténou (bez antény, vytištěná na desce, keramická, konektor na externí) nebo počtem vyvedených GPIO. Např. základní verze ESP-01 má vyvedeny pouze dva GPIO

pinů a má tištěnou anténu. Verze ESP-07 má vyvedeno 9 GPIO pinů a jeden ADC vstup a má keramickou anténu i konektor na anténu externí. Procesor pracuje na frekvenci 80MHz, celý modul je napájen napětím 3.3V, vstupy nejsou 5V tolerantní.

Okolo tohoto modulu existuje komunita uživatelů která se zabývá úpravou firmware a tvorbou vlastních aplikací. Existuje pro tento modul SDK (software development kit) díky kterému lze tyto vlastní aplikace psát a využít tak potenciál tohoto kontroléru.

Hlavními výhodami tohoto modulu jsou jeho rozměry (20x14mm), výkon, velká paměť, WiFi a velmi nízká cena (přibližně 5 dolarů).

3.4 Cubieboard2

Cubieboard2 je minipočítač postavený na procesoru AllWinner A20. Procesor má dvě jádra ARM Cortex-A7 na frekvenci 960MHz. Operační paměť: 1GB DDR3. Počítač je vybaven ethernetem, 2 x USB2, SATA, IR přijímačem, HDMI výstupem, audio in / audio out. Na desce je vyvedeno 96 GPIO, některé z nich mají určenou roli. Lze mezi nimi najít vývody sběrnice I2C, SPI, UART, LVDS, VGA výstup, vstupy AD převodníku a další. Jako úložiště pro operační systém lze využít vestavěnou 4GB NAND flash nebo microSD kartu, případně libovolný pevný disk připojený k SATA. Napájení tohoto počítače je třeba napětí 5V. Lze tedy napájet z USB počítače nebo pomocí univerzální USB nabíječky.

Jako operační systém pro tento počítač lze použít systém Android nebo nějakou Linuxovou distribuci (například Cubian - Debian přizpůsobený k provozu na Cubieboard2).

3.5 Banana Pi / Pro

Tento minipočítač je založený na stejném procesoru jako Cubieboard2. Z hlediska hardwaru je vybaven stejně jako Cubieboard2, avšak nemá NAND Flash a liší se celkovým rozložením prvků na desce. Rozložení jednotlivých konektorů je kompatibilní s rozložením konektorů minipočítače Raspberry Pi. Tak jako existují dvě hw verze Raspberry Pi, tak existují i dvě hw verze Banana Pi, eventuelně Banana Pro. Banana Pi (jako Raspberry Pi model A) má slot pro SD kartu a kompozitní video výstup. Banana Pro (jako Raspberry Pi model B) má slot pro microSD kartu a více vyvedených pinů. Co má však Banana Pro navíc je gigabitový ethernet a vestavěný WiFi modul.

Na tomto minipočítači lze provozovat tentýž operační systém jako na Cubieboard2.

II. PRAKTICKÁ ČÁST

4 CÍLE PROJEKTU

Cílem projektu je navrhnout embedded systém pro vzdálené monitorování nejen topných soustav. Všechna mnou zkoumaná řešení která se zabývají se běžným monitorování domácnosti a domácí automatizací jsou absolutně závislé na internetu. Nastane-li výpadek, nejsou schopné informovat o tom co se během výpadku dělo, natož upozornit na případný problém. Internet je dobrý sluha a přináší nové možnosti v různých oblastech lidského života nelze však spoléhat na jeho bezpodmínečnou dostupnost a nevědomě si tak budovat závislost na této službě. Podle [12] se budou útoky na internet množit s rozšiřováním inteligentních zařízení v domácnostech. Úkolem je počítat v návrhu s dočasnou nedostupností vzdálených služeb.

Součástí projektu je vytvoření vzorové implementace tohoto systému. Vzorová implementace bude zahrnovat vytvoření zařízení které bude přímo spojeno s technologickým procesem (topnou soustavou) a naznačení možnosti prezentace naměřených dat pomocí webového prohlížeče PC připojeného k počítačové síti.

- robustnost (odolnost, spolehlivost)
- škálovatelnost
- univerzálnost / přizpůsobitelnost
- nenáročnost na údržbu
- ekonomičnost provozu
- nezávislost

5 NÁVRH REALIZACE

5.1 Návrh zapojení do sítě

Budeme-li brát v úvahu si typický topný systém například rodinného domu. Tedy kotel umístěný v technické části domu a jednotlivé radiátory rozmístěné v budově. Současně je možné použití více tepelných agregátů (kotel na tuhá paliva / biomasu, elektrický kotel, plynový kotel) v různé kombinaci. Tedy vzniká nám mnoho měřících bodů neblízko od sebe. Možnost realizace všeho měření jedním prvkem je tedy nevhodná, jednak kvůli vzdálenosti jednotlivých měřících bodů a také kvůli nutnosti instalace nové kabeláže k snímačům. Bude tedy vhodné rozmístit více zařízení vybavených snímači, které jsou v daném místě potřebné. Dalším problémem je připojení těchto zařízení do sítě. V moderních stavbách se již počítá s možností zavedení počítačové sítě, tedy společně např. s televizní přípojkou je dostupná přípojka k síti LAN na více místech domu. Tedy první možností je použití existující metalické sítě či její částečné dobudování. Existují-li v domě místa kde připojení neexistuje či by bylo z nějakého důvodu náročné jeho dobudování, je možné použít bezdrátové připojení k síti (WiFi). Tady je zase důležité dostatečné pokrytí celého prostoru bezdrátovou sítí. Konkrétní řešení připojení závisí vždy na lokálních možnostech.

5.2 Návrh způsobu sběru dat

Z předchozího vyplývá, že je nutné v návrhu počítat s oběma variantami připojení zařízení v měřících bodech. Máme tedy skupinu „koncevých bodů“ připojených k síti. Jakým způsobem budeme data nyní sbírat? Nabízí se několik možností.

5.2.1 Koncevé body jako servery

První možnost je, že všechny koncevé body budou pracovat v režimu server. Způsob jak z takového prvku dostat data je připojit se k němu jako klient. Vzhledem k potřebné jednoduchosti by mohly poskytovat jen aktuální hodnoty, případně průměr za určitý čas. V této konfiguraci by nebylo možné zobrazit průběhy hodnot za časový interval, neboť by si je zařízení kvůli nedostatku paměti nemohlo pamatovat, či by nemělo dostatečné prostředky k poskytnutí přehledného grafu (grafů). Další nevýhodou by byla skutečnost, že pro zjištění různých hodnot by bylo potřeba navštívit různé servery. V neposlední řadě by neexistoval rozumný způsob vzdáleného přístupu k hodnotám.

5.2.2 Klienti webové služby

Druhá možnost je stav kdy jsou všechny prvky klienty webové služby. Způsob na kterém je založena „technologie“ IoT. Lze použít jednu z veřejných služeb pro zaznamenávání

dat, například ThingSpeak.com. Data jsou ukládána do tzv. kanálů. Každé zařízení musí mít vytvořen vlastní kanál. Uživatel může vlastnit více kanálů a rozhoduje zda budou veřejné nebo pouze soukromé. Tato služba poskytuje i vytváření pohledů na data a vykreslování grafů. Avšak tyto grafy mají jistá omezení. Jedním z omezení je 15vteřinový interval mezi vkládáním dat, což je ve většině případů dostačující. Tuto službu by bylo možné použít pro základní monitorování, ale z důvodů slabého zabezpečení, přístupový klíč kanálu, lze tyto údaje lehce podvrhnout. Přístupový klíč lze jednoduše odchytnout z nešifrovaného HTTP požadavku a vkládat libovolné data. Díky minimálnímu časovému intervalu tak může útočník zcela zastínit skutečná data. Řešením je vygenerování nového klíče a přenastavení zařízení, toto řešení bude mít účinnost jen do doby než si útočník změny všimne.

Využití existující služby nemusí vyhovovat všem našim potřebám. Jednou z potřeb by mohlo být zasílání oznámení formou e-mailu nebo SMS o přílišném poklesu, nárůstu, změně měřených hodnot nebo denní report s průběhy apod. Tyto a další funkce by mohla poskytnout vlastní nová webová služba. Tato služba by mohla splnit všechny požadavky na sběr dat a oznamování, ale zůstává zde otázka bezpečnosti. Jelikož by koncové body měly být realizovány pomocí málo výkonných mikropočítačů, bylo by zde použití zabezpečeného připojení nemalou komplikací. Další potíží je absolutní závislost na spojení s internetem. Při výpadku by celý systém zůstal nemonitorován a jediné oznámení které by bylo možné odeslat by bylo oznámení o výpadku spojení.

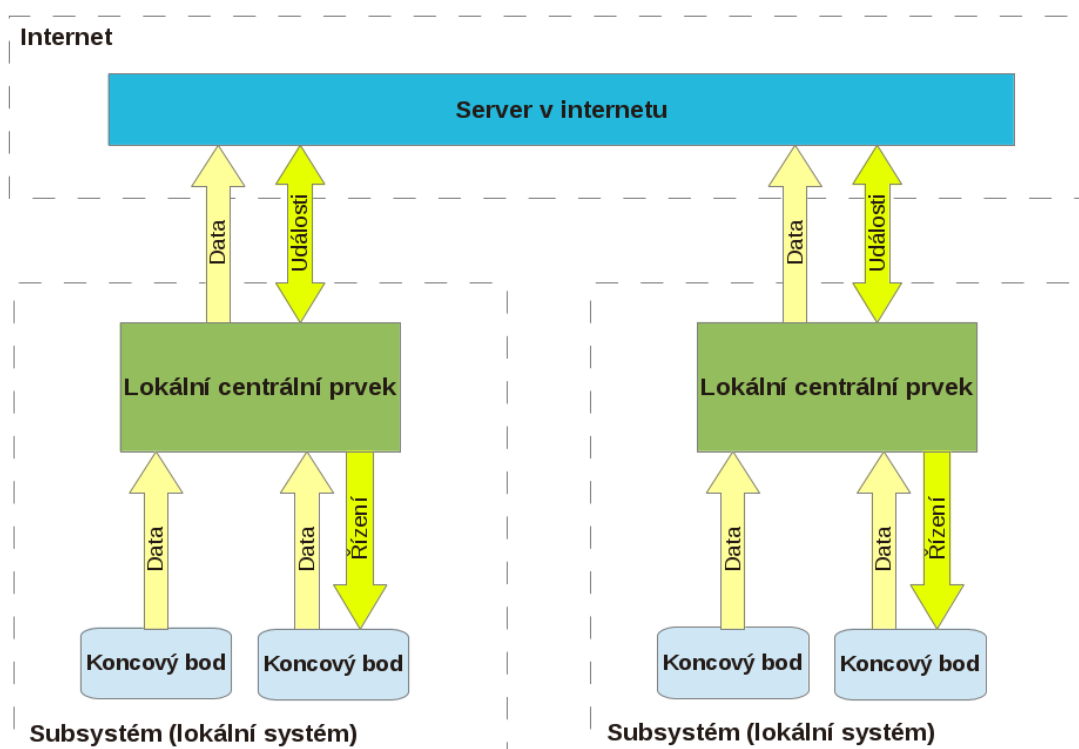
5.2.3 Klienti centrálního prvku

Právě kvůli, alespoň částečnému, omezení závislosti na internetu navrhuji použít „centrální prvek“. Jako centrální prvek by byl použit jeden z navrhovaných minipočítačů. Tento prvek by byl umístěn ve stejné lokální síti jako koncové body a působil by jako lokální služba pro záznam dat. Prováděl by kromě sběru date také jejich zabezpečené odesílání na vlastní webovou službu a zároveň by mohl působit jako lokální server pro prezentaci dat. Takový prvek by značně zvýšil nezávislost na spojení se světem. Při výpadku by logování probíhalo bez přerušení a po obnově spojení by chybějící data mohla být zpětně donahrána na webovou službu. Oznamování pomocí e-mailu by bylo během výpadku stále nedostupné, ale pokud by byl centrální prvek vybaven GSM modulem, mohl by v nutných případech zaslat SMS přímo on. Výpadek připojení sice omezí možnosti systému jako celku, ale neochromí ho zcela jako v předchozích případech.

5.2.4 Smíšená struktura

V předchozích strukturách byly koncové body vždy v režimu klient. Zasílání dat prováděly v pravidelných intervalech, nebo při změně sledované veličiny (záleží na určení

konkrétního prvku). Pokud by koncových zařízení byl větší počet nebo by zasílaly data ve stejný moment, nárazově by zatěžovaly centrální prvek. Toto uspořádání by nebylo vhodné z důvodu možného pozdějšího rozšíření o zpětnou vazbu, ovládání koncových bodů. Proto se logicky nabízí koncové body v režimu server čekající na klienta. Jedním z povelů je i příkaz pro vrácení dat. Tak jsem se opět dostal do situace kdy jsou koncové body servery. Narozdíl od prvního řešení máme v systému centrální prvek, samostatný počítač, který za nás bude obcházet jednotlivé servery a číst z nich data. Na centrálním prvku tedy kromě lokální prezentace poběží démon který bude v nastavených intervalech číst z koncových bodů a předávat data lokálnímu serveru (jinému procesu v rámci stejného stroje). Klientské koncové body bych ale úplně nezavrhoval, protože je-li třeba zasílat v nepravidelných intervalech či v závislosti na vstupech, je serverové řešení nevhodné. Je tedy potřeba dobře zvolit mód zařízení v závislosti na činnosti kterou má plnit. Výsledná logická struktura systému je znázorněna na obr. 5.1.



Obr. 5.1 Schéma výsledné struktury

5.3 Návrh hardwaru

5.3.1 Hardware koncových bodů

Jak bylo zmíněno, hardwarové provedení bude řešeno minimalisticky. Co nejmenší rozměry (v prototypovém provedení až tak malé nebudou), výkon mikrokontroléru jen

takový aby dokázal zvládnout své poslání. Jako výchozí mikroprocesor poslouží ATmega328 (Arduino (compatible) Mini) společně s ethernetovým kontrolérem ENC28J60 nebo s ESP8266 pro bezdrátovou verzi. Všechny tyto komponenty je možné zakoupit v ceně jednotek dolarů, což činí tuto volbu prakticky nejlevnější. Další náklady na hardware budou tvořeny cenou použitých snímačů a senzorů.

Použitý mikroprocesor ATmega328 poskytuje dostatek vstupně/výstupních pinů včetně osmi analogových vstupů, sběrnici I²C, rozhraní SPI a UART. Je tedy možné připojit prakticky libovolný snímač.

Pokud by se mělo jednat o koncový bod který by měl činit nějaký akční zásah, i zde by se dala tato konfigurace použít. Byl by-li výkon ATmega328 nedostatečný, je možné použít například STM32F103. Tento typ mikropočítače poskytuje větší výpočetní výkon, větší prostor pro program a větší počet I/O. Díky projektu Maple [6] by po úpravách mohl být použit stejný základní software vytvořený pro koncové body s ATmega328.

Ethernet Jako ethernetový kontrolér se k použití nabízí ENC28J60 [7] od firmy Microchip či W5100 [8] firmy Wiznet. Oba kontroléry využívají připojení přes rozhraní SPI. K oběma kontrolérům existují knihovny pro vývojový pro Arduino IDE takže není zásadní problém který z kontrolérů využít. Pro W5100 existuje knihovna podporovaná komunitou Arduino, takže je dostupná v Arduino IDE. Pro kontrolér ENC28J60 je dostupných více knihoven například UIPEthernet [10] (rozhraní shodné s Arduino knihovnou pro ethernet) či EtherCard.

WiFi Při výběru WiFi modulu není co moc vybírat. Existuje vícero modulů poskytujících připojení k bezdrátové síti, ale ceny těchto modulů jsou příliš vysoké na to, aby byly v systému využity ve větším množství. Je však jeden modul který lze zakoupit za cenu jednotek dolarů. Jedná se o modul s procesorem ESP8266. Vyrábí se ve více verzích lišících se například typem antény, počtem a rozmístěním vývodů či rozměry. Komunikace s tímto modulem probíhá pomocí rozhraní UART za pomoci „AT“ příkazů (AT-commands).

Napájení Typické napájecí napětí používané v oblasti mikropočítačů je 5V nebo 3.3V. Konkrétně procesor ATmega328 existuje ve verzích pro 3.3V i 5V, ale ve verzi s nižším napájecím napětím pracuje na frekvenci pouze 8MHz. Je možné v rámci jednoho koncového zařízení použít moduly s různým napájecím napětím. Aby byla zajištěna jednoduchost instalace a mohly být moduly napájeny univerzálním napáječem, bude každý z modulů obsahovat svoji zdrojovou část kde si vyrobí potřebná napětí 5 a 3.3V. Bude možné použít i nestabilizovaný napáječ s výstupním napětím v mezi 6V a 24V.

Tato vlastnost je vhodná zejména pro koncové body ve verzi s ethernetem protože je možné využít ethernetový kabel zároveň jako nosič napájení. Tato technologie se nazývá PoE (Power over Ethernet). Existují dva základní přístupy injektování napětí do ethernetového kabelu - aktivní a pasivní. Zatím co aktivní přístup vyžaduje speciální napájecí switche a koncové body musí být vybaveny modulem schopným toto napájení využít, pasivní přístup k vedení napětí využívá nepoužité páry vodičů ethernetového kabelu. Existují k tomuto účelu speciální redukce injector (přivede do kabelu napětí ze zdroje) a splitter (odebere z kabelu napětí a přivede do zařízení). Při použití tohoto typu napájení je třeba myslet na následující skutečnosti: nelze použít gigabit ethernet a rychlejší; napájený kabel nelze bez splitteru připojit do žádného zařízení jelikož by mohl způsobit jeho poškození! Napájecí napětí doporučené pro pasivní PoE je alespoň 15V a vyšší kvůli ztrátám na vedení, což našim koncovým zařízením díky jejich integrovanému stabilizátoru nikterak nevádí.

5.3.2 Terminál

Terminál bude speciální koncové zařízení. Bude obsahovat zobrazovací jednotku a tlačítka. Toto zařízení nebude odesílat žádná data, ale bude je přijímat a zobrazovat. Bude zobrazovat aktuální stav celého subsystému (případně jen některých součástí). Co bude zobrazeno bude moci nastavit hlavní uživatel skrze centrální prvek.

Výběr mikroprocesoru, který bude použitý pro tento koncový bod, závisí na požadavcích na zobrazovací jednotku. Bude-li použit obyčejný znakový LCD display 16x2 nebo 20x4 znaky a podobné, lze bez problémů použít ATmega328. Pokud by však byl požadován bodový display s grafickými prvky bylo by nutné použít jiný, výkonnější mikroprocesor například zmíněný STM32F103.

Tento terminál by mohl být rozšířen o možnost zadávání příkazů, ovládání subsystému. Tyto příkazy by směřovaly k centrálnímu prvku, který by je vyhodnotil a provedl příslušnou akci.

5.3.3 Hardware centrálního prvku

Jako centrální prvek použijí jeden z vyjmenovaných minipočítačů. Jelikož jsou tyto minipočítače dostatečně hardwarově vybavené, je třeba vyřešit jen úložiště pro operační systém a nasbíraná data. Jako primární úložiště dat se nabízí paměťová karta (micro)SD. V případě Cubieboard je možné využít i vestavěnou NAND Flash paměť.

Použití operačního systému Debian požaduje okolo 2GB prostoru podle zvolené verze a instalovaného softwaru. Serverová verze (bez grafického systému) požaduje do 500MB diskového prostoru. Při předpokládané velikosti paměti 4GB (interní NAND flash Cubieboardu má právě 4GB) zůstane minimálně 2GB prostoru pro zaznamenané

hodnoty měření. Zdá se to dostatečné, ale z dlouhodobého hlediska, při zachování celé historie hodnot, by tento prostor nemusel být dostačující. Dalším aspektem použití paměťové karty je rychlost čtení/zápisu a počet zápisů do jedné buňky. Jelikož bude tento prvek sloužit i jako databázový server, je rychlost čtení významná vlastnost. Co se zápisů týče, nebudou přenášeny až tak velké objemy dat, ale zápisy budou velmi časté. S tím souvisí další vlastnost paměťové karty - počet přepisovacích cyklů. Obecně - karty typu SLC mají, z hlediska počtu zápisů, delší životnost než MLC, dosahují však menších kapacit a jsou nesrovnatelně dražší. Při použití Raspberry Pi 2 není jiné alternativy, než hledat vhodnou paměťovou kartu. Je potřeba vybírat skutečně pečlivě, neboť sázka na špatnou kartu dříve nebo později skončí nefunkčním systémem.

Rozhodneme-li se použít minipočítač jako je Cubieboard nebo Banana Pi, můžeme použít jako úložiště klasický pevný disk s rozhraním SATA. Řešit spolehlivost záznamu z hlediska nárazů a vibrací není třeba, jelikož centrální prvek bude umístěn stabilně na jednom místě. Nejvhodnější je použít 2,5palcový disk kvůli jeho malým rozměrům, nižší spotřebě energie a napájení 5V. Spotřeba energie nebude tak malá jako při použití paměťové karty, za to získáme mnohem větší kapacitu a spolehlivost zápisů. Z hlediska spolehlivosti je třeba pamatovat na výpadky napájení. Aby nedocházelo ke ztrátě dat vlivem náhodného výpadku napájení, je vhodné vybavit centrální prvek malým záložním zdrojem. Vzhledem k potřebnému napájecímu napětí 5V by stačil malý li-ion akumulátor nebo USB powerbank. Je téměř jisté že vypadne-li napájení centrálního prvku, jsou i koncové body bez elektřiny. Úkolem záložního zdroje napájení není udržet zařízení v chodu po celou dobu výpadku, ale jen zajistit zaznamenání události a řádné ukončení operačního systému.

Jak bylo naznačeno v kapitole 5.2.3, centrální prvek by měl být vybaven GSM modulem v zájmu omezení závislosti na připojení k internetu. Primárně by tento kanál sloužil k zasílání upozornění s vyšší prioritou v případě nedostupnosti vzdáleného serveru. Případně by mohl sloužit jako alternativní připojení k internetu (mobilní internet), ale primární účel tohoto modulu je zaslání oznámení v případě problému (záloha).

5.4 Návrh softwaru

5.4.1 Software koncových bodů

Software koncových bodů bude programován v jazyce C / C++. S výhodou lze při vývoji využít Arduino IDE a knihovny které obsahuje. Při návrhu softwaru je nutné myslet na časování úloh. Například každou sekundu změřit teplotu, každé půl minuty odeslat data na server, nebo čekat na připojení klienta. Tyto úlohy se málokdy vyskytují zcela samostatně tedy nelze na časování použít funkci delay (po zadaný čas nic nedělej) a nelze spoléhat ani na to, že intervaly jednotlivých akcí budou své násobky (každý

třetí interval prověď činnost) vzniká problém.

Kostru softwaru koncových bodů tvoří časovač událostí. Je to jednoduchá statická třída kterou jsem si napsal. Třída je navržena jako statická, protože mít více instancí tohoto časovače nemá význam. Při startu zařízení je potřeba do časovače vložit tasky (události) a nastavit jejich interval opakování. Task se vkládá jako ukazatel na funkci vykonávající tento úkol. Metoda která vloží task vrací jeho identifikátor pomocí kterého lze za běhu programu pozastavit či znova spustit vykonávání tohoto tasku. Výběr a spuštění tasku probíhá na začátku „otočky cyklu“ zavoláním metody *loopIntervals()*. Během provádění metody je vybrána úloha jejíž interval již uběhl, je jí nastaven čas dalšího spuštění a je okamžitě provedena. Během jednoho volání metody (během jedné otočky cyklu) je provedena právě jedna úloha, aby se zbytečně neprodlužovalo trvání otočky cyklu. Doporučeno je proto vkládat do časovače tasky seřazené dle intervalu od nejdelšího k nejkratšímu, aby se nestalo že bude v každém cyklu spuštěn task s nejkratším intervalem. Tento časovač je určen k časování v řádu sekund. Lze nastavit i interval například 100ms, ale nelze zaručit že takto krátký čas bude dodržen.

K přenosu dat bude použit protokol HTTP ve verzi 1.1. Celá výměna dat bude založena na architektuře webových aplikací REST. Formát zvolený pro přenos dat bude JSON. Formát JSON volím kvůli potřebě menšího objemu dat při přenášení, také předpokládám snadnější parsování na mikropočítači. Formát XML jsem zavrhl z důvodů nadbytečných ukončovacích tagů a velké spotřebě paměti pro parsování. Osobně mi přijde jednodušší parsovat JSON, protože „řídící prvky“ jsou pouhé znaky („{“, „}“, „[“, „]“, „:“, „““, „““, „,“) kdežto „řídící prvky“ XML jsou celé řetězce (tagy).

Klientské koncové body budou vlastně klienty REST API které poběží na centrálním prvku. Serverové koncové body budou v miniaturní verzi implementovat REST API taktéž.

5.4.2 Koncový bod jako klient

Základem tohoto softwaru je právě zmíněný časovač. Jsou vytvořeny tasky „načti hodnoty“, „odešli data“ a „čekej na odpověď“. Task „načti hodnoty“ je spuštěn každou sekundu a načítá hodnoty ze snímačů a tvoří jejich průměrné hodnoty za interval odesílání a tyto hodnoty vypíše na debugovací rozhraní (sériovou linku). Task „odešli data“ je spuštěn v intervalu půl minuty, při přípravě dat k odeslání vynuluje průměrné hodnoty za interval. Task „čekej na odpověď“ je ve výchozím stavu pozastavený. Tento task je aktivován po odeslání dat a k jeho spuštění dojde za stanovený interval pouze v případě že do té doby nebude přijata odpověď od serveru. Ve funkci která se stará o vyřízení odpovědi je deaktivace tasku „čekej na odpověď“. Pokud přeci jen dojde ke spuštění tohoto tasku, sám se deaktivuje a spustí funkci „inicializuj spojení“. Tím je

zajištěno že v případě náhodného odpojení dokáže zařízení znovu navázat spojení s routerem. Funkce „inicializuj spojení“ se jinak spouští pouze při startu zařízení.

Tato kostra programu je společná pro Ethernetovou i WiFi verzi. Drátová a bezdrátová verze se liší použitou knihovnou síťového rozhraní, jiným obsahem funkce pro odeslání dat a obsahem funkce inicializace spojení. Části programu zajišťující samotný sběr dat jsou prakticky identické.

5.4.3 Koncový bod jako server

Základem i serverové verze je časovač událostí s tím rozdílem, že task na obsluhu síťové komunikace je spouštěn s nejkratším intervalem. Tento běží kdykoliv kdy neběží jiný task, takže téměř neustále. Sám tento task má své vnitřní stavy. Může se nacházet v jednom ze stavů: čekání na klienta, čtení požadavku, vyhodnocení požadavku/tvorba odpovědi, zápis odpovědi, ukončení spojení, chyba. Přejít do stavu „chyba“ způsobí znovupřipojení k síti („inicializuj spojení“). Opět základ programu bude shodný pro ethernetovou i WiFi verzi. Lišit se budou pouze části specifické pro použitý síťový kontrolér. Co se rozhodně měnit nebude, bude stav „vyhodnocení požadavku/tvorba odpovědi“. Právě zde bude řešena implementace REST API pro klienta (centrální prvek).

Tak jak je zde navrhnuo obslužení klientského požadavku, lze obsloužit pouze jeden požadavek. Snad by bylo možné tuto obsluhu upravit tak aby dokázala vyhovět více klientům, ale nejspíš by bylo naraženo na problém nedostatku operační paměti (ATmega328 má 2kB). Ovšem tato vlastnost, obslužení více klientů, není zapotřebí, jelikož jediným oficiálním klientem bude centrální prvek a ten bude vytvářet vždy jen jeden požadavek.

5.4.4 Software centrálního prvku

Centrální prvek jakožto minipočítač bude poháněn operačním systémem Debian. Debian je svobodný operační systém založený na jádře Linux. Jeho hlavními výhodami jsou: dostupnost systému pro mnoho procesorových architektur (jednou z nich je i ARM - architektura procesoru AllWinner A20); obrovské množství softwarových balíčků připravených k použití na vybrané architektuře (v základním repozitáři jsou všechny balíky naprosto svobodné - licencí není nikterak omezeno jejich použití); vynikající stabilita systému (využíván na serverech - roky běhu bez restartu). Jedním z důvodů proč jsem si pro centrální prvek vybral tento operační systém je fakt, že jej k naprosté spokojenosti využívám na svém počítači pro osobní potřebu i pro výkon zaměstnání.

V případě využití minipočítače Cubieboard2 existuje možnost stažení obrazu operačního systému Debian přizpůsobeného tomuto počítači přímo z webových stránek

výrobce. Jsou nabízeny dvě verze systému: DESKTOP a SERVER. Verze SERVER sice nabízí předinstalovaný webový server a další software, neobsahuje však grafické prostředí. Přestože grafické prostředí není pro tuto aplikaci podstatné, vybral jsem si verzi s prostředím LXDE. V běžném provozu grafické prostředí nepoběží, ale lze ho po připojení monitoru spustit a výhodně použít při nastavování - využití „klikacího“ režimu je někdy pohodlnější.

Na minipočítači kromě samotného systému poběží databázový server MySQL a webový server Apache s podporou PHP. V neposlední řadě bude na centrálním prvku dostupné běhové prostředí pro programovací jazyk Java.

5.4.5 Uživatelská část centrálního prvku

Jak je lehce naznačeno v předchozí části, prezentační (uživatelská) část lokálního systému bude webová aplikace v jazyce PHP. K sestavení prezentační aplikace lze použít jeden z PHP ORM frameworků schopných přizpůsobit se existující databázi. Jedním z nich je například QCubed [9] který obsahuje generátor ORM tříd podle existující databáze. Velkou výhodou tohoto frameworku je událostí řízené uživatelské rozhraní. Do frameworku je plně integrován AJAX, díky čemuž je možné si od serveru vyžádat jen část obsahu stránky která se má změnit. Tato skutečnost velmi sníží zátěž centrálního prvku způsobenou přístupem k prezentační aplikaci.

Prezentační aplikace bude mít dvě hlavní části, veřejnou a uživatelskou. Veřejná část bude dostupná komukoliv v síti a bude prezentovat jen to co povolí hlavní uživatel k veřejné prezentaci. Hlavní uživatel, správce subsystému, má práva měnit nastavení, vytvářet další uživatele a přiřazovat jim práva ke čtení, ovládání koncových bodů. Změnou nastavení je míněno i přidávání a konfigurace koncových bodů, přiřazení a pojmenování dat které poskytují. Uživatel bude mít možnost zadat IP adresu koncového bodu, nastavit interval získávání dat, nastavit oznámení v závislosti na hodnotě jednotlivých datových položek které koncový bod poskytuje. Toto nastavení bude automaticky odesláno i na server v internetu. Každý z uživatelů má možnost vytvořit si z dostupných zdrojů dat pohled.

5.4.6 Funkční část centrálního prvku

Na centrálním prvku kromě uživatelského rozhraní poběží i důležitá funkční část jakou je REST API. API bude poskytovat služby jako „schránka“ pro zasílání dat z klientských koncových bodů a fronta zpráv. Vzhledem ke koncovým bodům v módu server je třeba aby existoval klient který z nich bude číst, případně jim posílat příkazy. K tomuto účelu bude na koncovém prvku spuštěn démon, který z API získá informace o koncových bodech (IP adresu, interval čtení, příkazy které je třeba poslat na zařízení) a poté se

bude k zařízením připojovat, získávat od nich data a vkládat je do datové schránky API. Jako nejschůdnější možnost se jeví využít k vývoji tohoto démona programovací jazyk Java díky nezávislosti hotového programu na hardwarové či softwarové platformě kterou zajišťuje virtuální stroj JVM jakožto součást běhového prostředí JRE.

Aplikace bude využívat vícevláknovou architekturu a bude sestávat z několika komponent. První komponentou bude vlákno které získá z API seznam koncových bodů a jejich nastavení a předá tyto informace druhé komponentě - seznamu zařízení. Seznam zařízení bude obsahovat objekty reprezentující jednotlivá zařízení. Každý tento objekt bude mít za úkol vytvářet ve stanoveném intervalu pomocí časovače požadavek na zařízení a jeho odpověď zasílat do API. Aby byl celý systém interaktivní, musí se změna nastavení provedená v uživatelském prostředí projevit ve funkci systému. Změní-li uživatel interval získávání dat z jednoho ze zařízení nebo změní-li se IP adresa zařízení, musí se tato změna skutečně projevit aniž by proběhl restart celého systému nebo některé z jeho komponent. Z tohoto důvodu bude komponenta zajišťující získání informací o zařízení provádět svoji činnost opakovaně. Změní-li se některá vlastnost objektu reprezentující koncový bod, automaticky se zastaví proces vytváření požadavků a následně se spustí s novými parametry. Další komponentou bude vlákno získávající z API zprávy určené pro koncové body. Vlákno rozdělí zprávy příslušným objektům které mají zařízení na starost a tyto objekty zařídí samotné zaslání zprávy.

Úkolem další komponenty démona, případně samostatného démona bude zajištění komunikace se serverem v internetu. Jde o dvě operace které je třeba provést: odeslání dat a výměna zpráv. To lze uskutečnit více způsoby. Jedním ze způsobů je volání PHP skriptů které budou součástí aplikace poskytující API. V případě přenosu dat tento skript zajistí jak výběr dat k zaslání, tak samotné zaslání do vzdáleného API. V případě výměny zpráv skript zajistí nejen výběr a odeslání zpráv do vzdáleného API, ale také zapsání zpráv do lokální schránky. Druhý způsob je využití lokálního API k získání dat a zpráv a samotné odesílání na vzdálený server by prováděl sám démon. V případě že by se ze vzdáleného serveru vrátily nějaké zprávy, zapsal by je do lokálního API. Druhý způsob se zdá být zbytečně komplikovaný, ale ve výsledku by mohl produkovat menší zatížení centrálního prvku vzhledem k tomu, že vytváření a odesílání požadavků bude provádět téměř nativní aplikace v Javě místo interpretované aplikace v PHP. Ve výsledku by použití tohoto způsobu jen zvýšilo modularitu systému jelikož by nezáleželo na skutečnosti v jakém jazyce je ono API realizováno. Bylo by možné v rámci budoucího vývoje možné PHP API zaměnit například za API v Javě. Cílem těchto změn by bylo zrychlení uživatelské aplikace (soustředila by se pouze na prezentaci dat) a snížení zátěže centrálního prvku.

5.4.7 Server v internetu

Na server v internetu poběží webová PHP aplikace rozšířená verze lokální webové aplikace. Rozšíření bude spočívat v implementaci zabezpečeného spojení (HTTPS), omezení přístupu do API (identifikace jménem a heslem, případně přístupovým klíčem), správa více subsystému (subsystém - centrální prvek a jeho koncové body), více nezávislých uživatelů.

Zabezpečené připojení bude použito z důvodu aby nebylo možné odposlouchávat, případně podvrhovat, přenášené informace. Možnost podvržení informace jdoucí libovolným směrem (z lokálního do internetového či naopak) by znamenala snížení důvěryhodnosti ve smyslu správnosti prezentovaných dat, či spolehlivosti zda bude provedena přesně ta operace kterou jsem zadal. Šifrování zamezí útoku typu „man-in-the-middle“, nezamezí však neoprávněnému čtení/zápisu z/do API cizím klientem.

Přístup do API bude chráněn přístupovým klíčem. Tento přístupový klíč bude jednoznačně identifikovat centrální prvek ze kterého požadavek přichází. Všechny akce v API musí být prováděny v rámci daného subsystému tedy další informace kterou klient musí poslat je identifikátor subsystému. API rozhodne zda jsou zadané informace platné, tedy je-li daný klíč přiřazen k vybranému subsystému a pokud ano, teprve s ním začne komunikovat.

API bude zahrnovat dvě hlavní součásti: schránku pro data a frontu zpráv. Fronta zpráv bude fungovat obousměrně, klient (centrální prvek lokálního systému) pošle zprávy pro nadřazený prvek (změna nastavení, oznámení o události) a v odpovědi od serveru budou obsaženy obdobné zprávy pro lokální systém. Schránka dat bude jednosměrná, bude pouze pro vkládání dat lokálním zařízením.

Volání API budou iniciovány lokálním centrálním prvkem v zadaném časovém intervalu. Časový interval volání API je na dalším zvážení. Je třeba brát v potaz faktory jako čas reakce (čas od zadání příkazu na webu po akci koncového bodu), zpoždění dat (čas od změření údaje na koncovém bodě po jeho uložení do databáze vzdáleného serveru), zátěž sítě a serveru (neopakovat volání zbytečně často), množství dat k přenesení (synchronizace dat na vzdálený server po výpadku spojení musí probíhat rychleji - může být ve frontě nahromaděno velké množství záznamů). Interval volání bude nastaven na určitý čas, řekněme 10 sekund. Aplikace by měla možnost za běhu změnit délku intervalu odesílání dat dle aktuálních podmínek. Interval výměny zpráv by měl být co možná nejkratší, aby se zbytečně neprodlužoval reakční čas.

6 VZOROVÁ IMPLEMENTACE

Vzorová implementace systému je spíše taková demoverze neschopná skutečného nasazení. Tato vzorová implementace má za cíl naznačit základní funkci navrhnutého systému. Bude implementována jen lokální část systému, tedy centrální prvek a vzory koncových bodů s připojením pomocí ethernetu a pomocí WiFi.

6.1 Koncový bod s ethernetem

Mikropočítač použitý ke stavbě tohoto zařízení je Arduino Mini kompatibilní modul. Jeho jádrem je Atmel ATmega328P. Jako ethernetový kontrolér jsem použil modul s obvodem ENC28J60 ve verzi mini. Zdrojovou část tohoto zařízení tvoří stabilizátor napětí LM7805 pro napětí 5V a AMS1117 pro napětí 3.3V. Použitý teplotní senzor - DHT21. Celé zařízení je složeno na prototypovací desce o rozměrech 5x5cm. Procesorový i ethernetový modul je nasazen na konektorech kvůli zjednodušení montáže ostatních prvků a případné výměně modulu. Teplotní senzor je připojen taktéž konektorem na prodlouženém kabelu.

Software pro toto zařízení jsem napsal v Arduino IDE. Původně jsem ho implementoval jako klienta, ale pro testovací účely byl přepsán na serverovou verzi. V této verzi není implementováno kompletní API, ale pouze vrácení naměřených hodnot. Struktura softwaru je taková jak je popsána v kapitole 5.4.3. Pro ovládání ethernetového kontroléru jsem si vybral knihovnu EtherCard [13] kvůli nižší velikosti zkompilevaného programu.

6.2 Koncový bod s WiFi

Procesorový modul tohoto koncového zařízení je shodný s ethernetovou verzí, tedy Arduino Mini kompatibilní pro napětí 5V. Jako WiFi modul je v tomto příkladě použit modul s procesorem ESP8266 v hardwarové verzi ESP-01. Osazení zdrojové části je shodné s ethernetovou verzí i rozměr desky je týž, avšak rozložení jednotlivých komponent se liší. Důvodem jiného rozložení bylo přiblížení prvků k pinům procesoru které využívají. Jelikož ESP8266 je navrženo pro 3.3V systémy a jeho vstupy nejsou 5V tolerantní, musí být mezi procesor a WiFi modul zařazen převodník úrovní.

Další hardwarovou změnou oproti ethernetové verzi je použitý teploměr. Zde je přímo na desce namontován modul s převodníkem MAX6675 který slouží k připojení termočlánku typu K díky němuž lze měřit teploty v rozsahu 0-1023°C.

Software tohoto zařízení z něj dělá klienta centrálního prvku. Opět je použita struktura programu jaká byla popsána v kapitole 5.4.2. Interval odesílání dat je nastaven na 30 sekund. Pro ovládání WiFi kontroléru jsem si musel napsat vlastní funkce. Na-

šel jsem sice existující řešení této komunikace, ale neodpovídalo mým požadavkům na funkčnost. Vzhledem k dobré dokumentaci AT příkazů [14] to nebyl zásadní problém.

6.3 Centrální prvek

Jako centrální prvek byl vybrán Cubieboard2. Prvek je poháněn operačním systémem Debian běžícím z pevného disku. Prvek poskytuje HTTP server, MySQL server a běhové prostředí pro Java. Prakticky implementuje vše popsané v 5.4.4 až na několik omezení. Prvním z omezení je nekompletní uživatelské prostředí. Prakticky je z uživatelského hlediska dostupná pouze vizualizace průběhu veličiny za den. Pro zobrazení lze vybrat jeden z předešlých třiceti dnů a lze měnit zobrazovaný časový úsek. Konkrétně lze zvolit počátek zobrazování (kolik hodin od půlnoci) a kolik hodin se bude zobrazovat (24, 12, 6, 3, 2 a 1). Z funkční části není implementovat démon pro synchronizaci dat se vzdáleným serverem. Démon který zajišťuje získávání dat serverových koncových bodů je funkční. PHP API obsahuje pouze vložení dat a návrat seznamu koncových bodů s jejich nastavením. Struktura databáze použitá pro ukládání dat je následující zobrazena na obrázku 6.1 . Obsahuje jen základní informace o připojených zařízeních pro démona. Samotná data jsou záměrně rozdělena do několika tabulek podle charakteru dat, logické hodnoty, celá čísla, desetinná čísla. V případě potřeby není problém tabulky rozšířit o větší počet sloupců či přidat tabulku pro ukládání například textových hodnot. Jakou hodnotu ze zařízení uložit do které tabulky a kterého sloupce rozhoduje API podle nastavení pro dané zařízení.



Obr. 6.1 Schéma části databáze

6.4 Návrhy na rozvoj

Hlavním cílem pokračující práce na tomto projektu je zprovoznění systému v takovém stavu jak je popsáno v návrhu. Konkrétně implementace vzdálené části systému s přívětivým uživatelským rozhraním a funkční frontou zpráv, synchronizace dat a výměna zpráv mezi lokálním centrálním prvkem a serverem v internetu a vytvoření finálního koncového bodu. Možností je i rozšíření o sledování jiných veličin tedy úkolem bude připravit pro toto sledování potřebný hardware.

Dalším cílem je rozšíření celého systému o možnost posílat příkazy z internetu do koncových zařízení. K tomu je také nutné navrhnout zařízení které bude schopno na tyto povely reagovat. Vizí je i mobilní aplikace která bude prezentovat vizualizace, zobrazovat upozornění a nabídne ono nové zasílání povelů.

ZÁVĚR

Cílem této práce bylo navrhnout systém vhodný k monitorování topných soustav budov. Navrhnutý systém měl mít následující vlastnosti:

- rozšířitelnost
- univerzálnost
- nezávislost

Vlastnost rozšířitelnost je zajištěna schopností systému přijmout zcela nové koncové zařízení (jiné snímačem jiný procesor a podobně) bez nutnosti hardwarových nebo softwarových úprav systému. Za předpokladu že toto zařízení splňuje požadované vlastnosti na komunikaci. Zařazení tohoto zařízení by proběhlo jeho připojením k počítačové síti a nastavením parametrů zařízení v centrálním prvku.

Vlastnost univerzálnost je splněna v tom, že při návrhu systému nebylo kalkulováno s konkrétní topnou soustavou v konkrétní budově. Bude existovat sada měřících zařízení které lze nainstalovat libovolně a data která poskytují pojmenovat tak aby popisovaly skutečné umístění. Neodpovídalo-li by žádné zařízení požadavkům (například nedostatečný rozsah měření, málo vstupních kanálů), lze vytvořit nové a do systému jej přidat (viz. předchozí odstavec).

Navrhnutý systém není závislý na žádné konkrétní službě jejíž nedostupnost by zcela znemožnila jeho provoz (snad kromě elektřiny). Měřící zařízení - lze použít libovolný mikropočítač, důležité je aby dokázal komunikovat po síti a obsahoval domluvené komunikační rozhraní. Centrální prvek - lze použít libovolný minipočítač (dostatečného výkonu) schopný rozběhnout Linux s Apache a MySQL.

Vzorová implementace je skutečně jen ukázkou že lze tento systém složit z dostupných levných komponentů a že využití počítačové sítě není problém ani pro 8-bit mikropočesory. Při řešení jsem se setkal se skutečností že problémem není sestavení hardwaru ale vytvoření přívětivé uživatelské aplikace. Protože nejsem webový designer, opustil jsem od jakékoliv grafiky s výjimkou grafu zobrazující průběh naměřených hodnot. Při řešení jsem se spíše zaměřil na demonstraci funkčnosti navrhnuté struktury systému.

SEZNAM POUŽITÉ LITERATURY

- [1] FAI UTB ZLIN. *CodeDesigner RAD website [online]. [cit. 2015-01-28]. Dostupné z WWW: <http://codedesigner.org/>.*
- [2] BLIŽŇÁK, Michal, Tomáš DULÍK, Roman JAŠEK a Pavel VAŘACHA. *Optimized Production-Ready Source Code Generation Based on UML[online]. INTERNATIONAL JOURNAL OF SYSTEMS APPLICATIONS, ENGINEERING DEVELOPMENT: NAUN Press, 2013, roč. 7, č. 1 [cit. 2015-01-23]. Dostupné z WWW: <http://www.naun.org/main/UPress/saed/16-498.pdf>.*
- [3] BLIŽŇÁK, Michal, Tomáš DULÍK a Roman JAŠEK. *Production-Ready Source Code Round-Trip Engineering [online]. INTERNATIONAL JOURNAL OF COMPUTERS: NAUN Press, 2012 [cit. 2015-01-23]. ISSN 1998-4308. Dostupné z WWW: <http://www.naun.org/wseas/cms.action?id=3036>.*
- [4] NATIONAL INSTRUMENTS. *What Is NI CompactRIO? [online]. [cit. 2015-04-01]. Dostupný z WWW: <http://www.ni.com/compactrio/whatis/>.*
- [5] ELECTRIC IMP. *BlinkUp technology [online]. [cit. 2015-03-01]. Dostupný z WWW: <https://electricimp.com/product/blinkup/>.*
- [6] LeafLabs. *Maple-Arduino Compatibility [online]. [cit. 2015-03-15]. Dostupný z WWW: <http://leaflabs.com/docs/ide.html>.*
- [7] Microchip. *ENC28J60 [online]. [cit. 2015-02-11]. Dostupný z WWW: <http://www.microchip.com/wwwproducts/Devices.aspx?dDocName=en022889>.*
- [8] Wiznet. *W5100 [online]. [cit. 2015-02-14]. Dostupný z WWW: <http://www.wiznet.co.kr/product-item/w5100/>.*
- [9] QCubed. *Welcome to QCubed! [online]. [cit. 2015-03-02]. Dostupný z WWW: <http://qcu.be/content/welcome>.*
- [10] Norbert Truchsess. *UIPEthernet [online]. [cit. 2015-01-12]. Dostupný z WWW: https://github.com/ntruchsess/arduino_uip.*
- [11] Nest Labs. *Works with Nest [online]. [cit. 2015-02-14]. Dostupný z WWW: <https://nest.com/works-with-nest/>.*
- [12] VŠETEČKA, Roman. *Skončil obří útok na evropský internet. Hrozba i pro internet věci [online]. [cit. 2015-05-02]. Dostupný z WWW: http://technet.idnes.cz/utok-ntp-evropa-nejmasivnejsi-internetovy-utok-ukazuje-narust-riziko-139-sw_internet.aspx?c=A140212_161619_sw_internet_use.*

-
- [13] *Jean-Philippe Lang. EtherCard is a driver for the ENC28J60 chip, compatible with Arduino IDE. [online]. [cit. 2015-01-05]. Dostupný z WWW: <http://jeelabs.net/pub/docs/ethercard/index.html>.*
- [14] *NURDspace. ESP8266 (AT commands) [online]. [cit. 2015-03-11]. Dostupný z WWW: <https://nurdspace.nl/ESP8266>.*

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

ADC	analogově-digitální převodník
AVR	architektura procesoru založená na Harvardské architektuře použitá firmou Atmel
CPU	Central Processing Unit (výkonná výpočetní jednotka - procesor)
GPIO	General Purpose Input/Output (vstupně/výstupní vývod [pin] pro obecné použití)
JRE	Java Runtime Environment (běhové prostředí Java)
JVM	Java Virtual Machine (virtuální stroj Java)
MCU	MicroController Unit (mikropočítačová jednotka)
MLC	multi-level cell (více bitů na paměťovou buňku)
ORM	object-relational mapping (objektově-relační mapování)
PLC	Programable Logic Controller (programovatelný logický automat)
RAM	Random Access Memory
SLC	single-level cell (jeden bit na paměťovou buňku)
SoC	System on Chip (obdobně jako MCU, je na jednom čipu vše, avšak může používat exte
SPI	Serial Pheriperal Interface (sériová komunikační sběrnice)
UART	asynchronní sériová linka

SEZNAM OBRÁZKŮ

Obr. 5.1	Schéma výsledné struktury	23
Obr. 6.1	Schéma části databáze	33

SEZNAM PŘÍLOH

P I. Obsah přiloženého CD

PŘÍLOHA P I. OBSAH PŘILOŽENÉHO CD

CD obsahuje zdrojové kódy softwaru koncových bodů, démona lokálního serveru a webové aplikace centrálního prvku.

/	
/demon/	zdrojové kódy démona v Javě
/koncove_body/	
/koncove_body/libraries/	knihovny použité ve zdrojových kódech
/koncove_body/wifi_klient/	Arduino sketch wifi klienta
/koncove_body/ethernet_klient/	Arduino sketch ethernetového klienta
/koncove_body/ethernet_server/	Arduino sketch ethernetového serveru
/lokalni_api/	PHP ORM třídy a lokální api