UNIVERZITA PARDUBICE
Fakulta elektrotechniky a informatiky

Inteligentní dům

Radek Smejkal

Bakalářská práce
2013

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: **2012/2013**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Radek Smejkal**
Osobní číslo:	**I10311**
Studijní program:	**B2612 Elektrotechnika a informatika**
Studijní obor:	**Komunikační a mikroprocesorová technika**
Název tématu:	**Inteligentní dům**
Zadávající katedra:	**Katedra elektrotechniky**

Zásady pro vypracování:

Moderní inteligentní domy umožnují základní řízení prostředí domu s podporou vzdáleného přístupu, monitoringu a nastavení pomocí datových sítí. Zadáním bakalářské práce je návrh řešení jednoduchého modulu inteligentního domu. Teoretická část práce rozebere možnosti řešení, vhodnost využití různých typů senzorů, datových sítí, výběr vhodné platformy. Praktická část se pak bude zabývat návrhem a ukázkovou implementací modulu se vzdáleným přístupem schopných měřit alespoň teplotu, snímat a řídit stavy několika digitálních vstupů/výstupů, provádět jednoduchou regulaci nebo signalizaci na základě měřené hodnoty.

Rozsah grafických prací:

Rozsah pracovní zprávy:

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

Seznam odborné literatury:

Kreidl, M.; Měření teploty - senzory a měřicí obvody, BEN, ISBN 80-7300-145-4
RIPKA, P.; TIPEK, A.: Master Book of Sensors. Praha : BEN, 2003. ISBN 0-12-752184
ĎAĎO, S.; KREIDL M.Senzory a měřicí obvody; Praha : Vydavatelství ČVUT, 1999 ISNB 80-01-02057-6

Vedoucí bakalářské práce: **Ing. Pavel Rozsíval**
Katedra elektrotechniky

Datum zadání bakalářské práce: **21. prosince 2012**
Termín odevzdání bakalářské práce: **10. května 2013**

Univerzita Pardubice Česká republika

L.S.

prof. Ing. Simeon Karamazov, Dr.
děkan

Ing. Zdeněk Němec, Ph.D.
vedoucí katedry

V Pardubicích dne 29. března 2013

Prohlášení autora

Prohlašuji, že jsem tuto práci vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Souhlasím s prezenčním zpřístupněním své práce v Univerzitní knihovně.

V Pardubicích dne 20. 5. 2013 Radek Smejkal

Poděkování

Tímto bych rád poděkoval vedoucímu práce panu Ing. Pavlu Rozsívalovi, za poskytnuté odborné rady a připomínky během zpracování této práce a všem dalším lidem, kteří mě podporovali.

ANOTACE

Cílem práce je navrhnout systém pro inteligentní dům. Ten bude obstarávat webové rozhraní, pomocí kterého bude možné ovládat výstupy, či sledovat vstupy. Dále bude možné sledovat několik teplot. Ty bude systém ukládat do paměti pro pozdější zobrazení historie. Systém bude také obstarávat jednoduchou regulaci výstupů. Obsahovat bude i další doplňkové funkce, jako je nastavení aktuálního času ze sítě, či odeslání emailu. Teoretická část rozebere použité senzory, ethernetové protokoly a platformu, na které je systém postaven.

KLÍČOVÁ SLOVA

inteligentní, dům, ethernet, C#

TITLE

Smart Home

ANNOTATION

The aim of this thesis is to outline a system for intelligent house. The system will be accessible via web interface, in which user will be able to set outputs or watch inputs. Moreover, it will be possible to watch a number of temperatures. They will be stored in system's memory for later history view. System will ensure simple outputs regulation. More functions will be available, like setting current time from network or sending an e-mail. The theoretical part will focus on used sensors, ethernet protocols and platform, which the system is built on.

KEYWORDS

Smart house, ethernet, C#

Obsah

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

ÚVOD

Zařizování domu jako inteligentní je dnes běžnou činností. Takto zařízené domy nám potom zpříjemňují a usnadňují obývání. Bakalářská práce se bude zabývat návrhem jednoduchého systému pro ovládání a monitorování takového domu. Hlavní funkcí bude sledování teplot, ovládání výstupů a sledování vstupů. Další funkce by měla být schopna obstarávat jednoduchou regulaci na základě nastavených hodnot. Vše by mělo být možné sledovat a ovládat na webové stránce, kterou bude zprostředkovávat samotný systém.

Práce je rozdělena do několika částí. V první se seznámíme s pojmem inteligentní dům a přínosem takového systému v domě, Druhá část bude zaměřena na popis dvou komerčních zařízení podobající se svými vlastnostmi a parametry navrhovanému systému. Třetí část práce je věnována výběru vhodné platformy. V následující čtvrté části budou rozebrány použité síťové protokoly a následovat bude kapitola s informacemi o sběrnici zprostředkovávající komunikaci s teplotními senzory. V šesté části bude popsán použitý teplotní senzor a způsob odečítání spotřebovávané energie z elektroměru. Sedmá část pojednává o softwarovém řešení a přibližuje nejdůležitějších částí zdrojového kódu. Poslední část je věnována webovým stránkám, informacím o jejich návrhu a použitých technologiích.

1 Inteligentní dům

Realizací inteligentního domu je spousta, ale vždy se zejména jedná o systémy, které nám usnadňují a zpříjemňují obývání nebo ovládání domu. Především jde o uživatele, jaké funkce bude od celého domu požadovat. Před tím, než se tyto systémy začaly používat v rodinných domech, byly využívané ve velkých centrech či administrativních budovách. Hlavní a asi nejvíce žádanou funkci je vzdálené sledování, případně ovládání domácnosti. Předchůdcem těchto inteligentních domů se stalo používání vypínačů a koncových spínačů propojených sběrnicí, kde se díky těmto rozvodům a vzájemnému propojení všech prvků zajistila maximální flexibilita. Díky tomu je možné v pár krocích změnit světlo, které se má daným spínačem rozsvítit.

V případě inteligentních domácností je nutné přidat centrální prvek, systém, který bude mít přístup k této sběrnici a bude moci osvětlení v budově automaticky měnit, nebo sledovat jeho stav. Avšak sběrnicové rozvody nejsou nutností a každý koncový spotřebič může být ovládán individuálně.

Dalším hodně skloňovaným slovem se stává úspora. Dům automaticky sleduje teploty a další jevy z okolí domu a na základě těchto veličin efektivně reguluje teplotu topení nebo ovládá okenní rolety. Návrh takovéto regulace ovšem není předmětem této bakalářské práce.

Toto je pár základních vlastností, kterými se vyznačují inteligentní domy. Ale jedná se jen o holý základ a dnešní používané nejmodernější systémy umějí mnohem více, ovšem se značným nárůstem nutných financí na jejich pořízení.

Díky velkému rozvoji elektroniky zejména v oblasti výpočetní techniky, se v nových stavbách stávají inteligentní rozvody samozřejmostí. Díky velké konkurenci se snižují náklady na pořízení různých komponentů potřebných k vybavení inteligentního domu. Hlavním hnacím motorem těchto instalací se stává lenost uživatelů a vidina ovládání celého domu skoro odkudkoli, což je samozřejmě v dnešní době velmi příjemnou a čas šetřící věcí. Po velkém nástupu smart-phonů (chytrých telefonů) je tato možnost ještě zpříjemněna díky možnosti použit vlastní mobilní zařízení. Díky efektivnímu řízení se dá uvažovat i o pozdější návratnosti prvotní vetší investice do těchto systémů.

1.1 Odhady

Podle odhadů je 50% novostaveb v USA budováno jako inteligentní domy. Dále je podle odhadů „Strategy Analytics" 6% amerických domácností klasifikovány jako "inteligentní" a výnosy z hardwarových a montážních poplatků dosáhly v roce 2011 4.9 miliard dolarů. Tyto výnosy se podle předpovědí budou zvyšovat o 3 miliardy ročně.

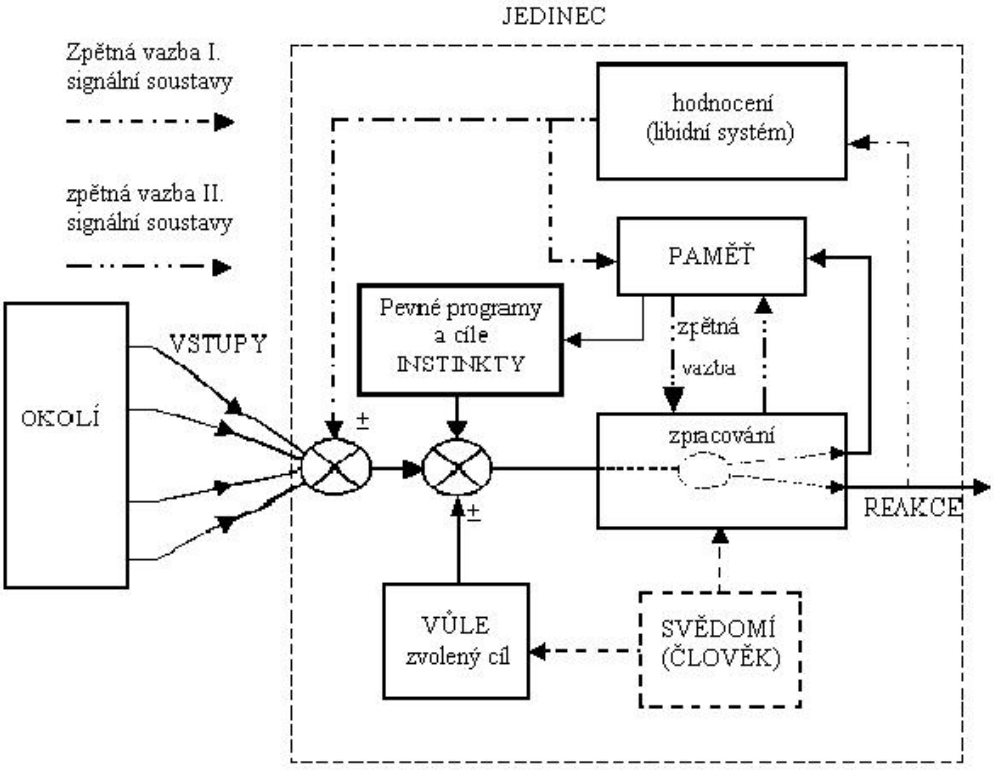
Podle dalšího článku z „businesswire.com“ bude v roce 2015 přes 5 milionů domácností zařízených jako inteligentní (Smart).

V Evropě je tento trend zhruba poloviční. V západní Evropě se dají jako inteligentní domácnosti klasifikovat 3% staveb.

Rozhodně se jedná o velký byznys, který se v příštích letech může dočkat velkého nárůstu zákazníků požadujících tyto služby. Proto můžeme do budoucna čekat snížení ceny propracovaných a kvalitních systémů, které jsou nyní stále ještě drahé.

1.2 Pojem inteligence

Dále bych se pozastavil nad samotným názvem bakalářské práce „Inteligentní dům“, a tím dospěl k tomu, že inteligentní dům vlastně žádnou inteligenci neobsahuje, možná jen částečnou. Inteligence (z lat. inter-legere, rozlišovat, poznávat, chápat) je totiž schopnost, řešit situace, ať už známé nebo nově vzniklé, dále se učit ze zkušeností či se přizpůsobit novým okolnostem. A můžeme jí ohodnotit podle toho, jak dobře byla situace vyřešena. Na následujícím obrázku (Obr. 1) je znázorněné blokové schéma inteligentního systému.[8]



Obrázek 1Blokové schéma inteligentního systému[1]

Můžeme si všimnout důležitého prvku, kterým je vůle či svědomí. Ať už bude náš řídicí systém v domě sebevíc složitý, pravděpodobně tyto dva bloky obsahovat nebude. Vždy se bude jen jednat o algoritmus, který bude podle vstupních hodnot nastavovat výstupní, a to podle přesně zadaných pravidel. Tady vidíme, že nebude mít tu důležitou vlastnost inteligence, se kterou by byl schopný se vypořádat s nově vzniklými situacemi, aniž by na ně byl připravený.

Proto se spíše hodí pro pojmenování „digitální domácnost, digitální dům“ nebo „automatizovaný dům“. Ale protože se v České republice pojem „inteligentní dům“ velmi rozšířil, bude se pravděpodobně dále používat toto pojmenování a žádné jiné. Anglický název „Smart House“ je přesnější protože si ho můžeme vyložit jako luxusní či bystrý dům. Samozřejmě se pracuje, a také se už i velmi často používají, řídicí systémy s umělou neuronovou sítí, u kterých se více přiblížíme inteligentnímu systému a však do rodinných domů se ještě nedostaly. Jedná se spíše o úzce určené aplikace, do kterých se tyto neuronové sítě aplikují.

[1] Zdroj obrázku http://cs.wikipedia.org

2 KOMERČNÍ SYSTÉMY

Na trhu je velké množství systémů pro inteligentní domácnost, každý se svými výhody či nevýhody. Pro výběr budou nejdůležitější naše představy o tom, co by měl dům ve výsledku zvládat a podle toho se zvolí potřebný ovládací systém. Já zde uvedu dva z té nižší cenové kategorie. Zmíněné budou pouze řídicí systémy.

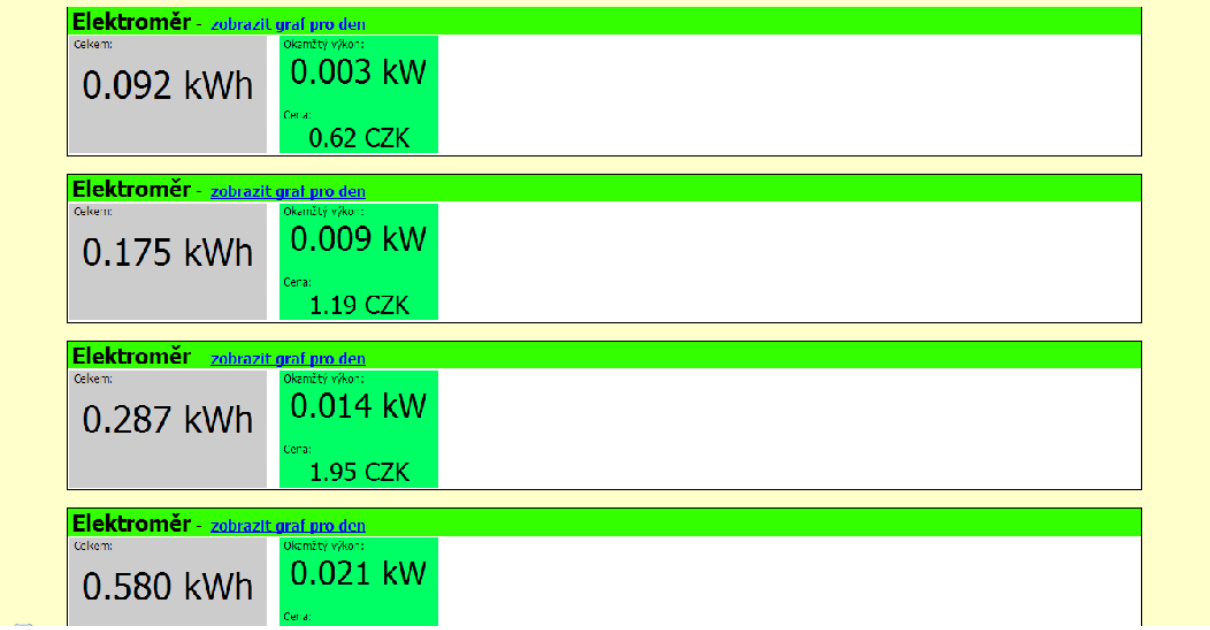
2.1 SDS MACRO

Prvním zmíněným komerčním systémem je SDS MACRO. Jedná se o český výrobek vyvíjený od roku 2005. Zařízení se dá spíše zařadit do PAC/PLC (programovatelný automat). V nabídce mají několik variant modulů, každý s přístupem přes webové rozhraní. Jedná se hlavně o systém pro realizaci online odečtu energie, proto u těchto modulů není podpora připojení sběrnicových rozvodů se standardizovanými protokoly. Modul se dá využít jako PLC, v tomto případě spíše PAC (Programmable Automation Controller) s možností programování ve speciálním jazyku SDS-C.

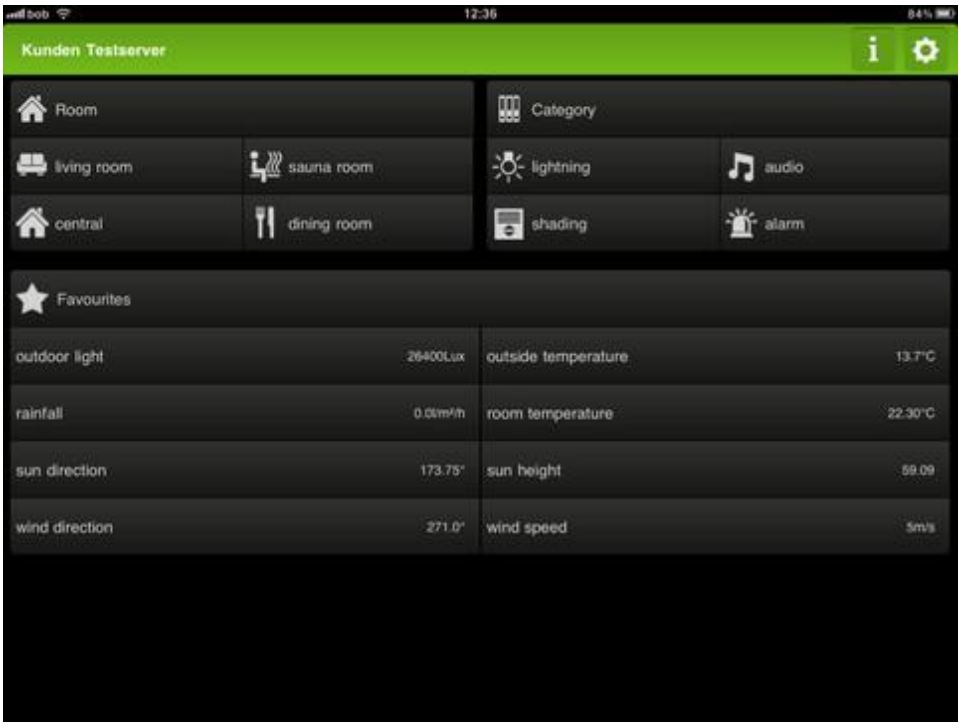
Modul je dodáván zakrytý v ABS plastu s možností uchycení na DIN lištu.
K dispozici jsou vstupy pro měření odběru el. energie, vody, plynu, dále analogové vstupy pro měření napětí a sledování vlhkosti vzduchu. Výstupy jsou digitální, reléové a PWM. Obsažen je také převodník na RS485 sběrnici. Další nejdůležitější sběrnicí je 1-Wire pro připojení teplotních senzorů Dalas 18B20.

Sledování dat může být zprostředkováváno přímo SDS modulem na jeho IP adrese, nebo je využit vzdálený server, který shromažďuje data a může tak obstarat jejich historii.

V nabídce má firma několik variant modulu, například SDS MICRO s menším počtem vstupů a integrovaných relé. Moduly z cenově vyšší kategorie obsahují i vestavěný LCD pro zobrazování naměřených hodnot. S dodatečným naprogramováním v jazyku SDS-C je možné rozšířit funkce nad základní, kterou je měření a zobrazování teplot na webové stránce.



Obrázek 2 Ukázka webové stránky SDS



Obrázek 3 Ukázka ovládacího softwaru Lonxone

2.2 Loxone

Tento druhý komerční systém má více možností použití a obsahuje také propracovanější webové prostředí, proto je také ve vyšší cenové kategorii. Je hlavně zaměřen na efektivní ovládání topení a rolet.

Systém vyvíjí rakouská společnost Loxone se zaměřením na domácí automatizaci. Celý modul je opět řešen pro přichycení na DIN lištu. Další připojitelné moduly jsou například pro rozšíření vstupů a výstupů, rozšíření o RS485 sběrnici, 1-Wire sběrnici, ovládání světel a další.

Hlavním srdcem je Loxone Miniserver modul. Ten obsahuje reléové výstupy, digitální vstupy, D/A a A/D převodníky. Programování funkcí probíhá v propracovaném softwaru Loxone Config.

Výhodu u tohoto modulu vidím v možnosti propojení se standardizovanou *KNX/EIB* sběrnicí. Díky tomu je možné využít koncové moduly například od ABB, Siemens, a tím nebýt závislý jen na sortimentu firmy Loxone. Za další přednost považuji propracované webové prostředí, a také software pro mobilní zařízení. Ovšem hlavní nevýhoda tohoto modulu je v připojení teplotních senzorů, samotný Miniserver neobsahuje rozhraní pro 1-Wire sběrnici a je nutné dokoupit rozšiřovací modul.

2.3 Porovnání

Tabulka 1 Porovnání komerčních systémů

	MACRO 485	**Loxone**
připojení k PC	RJ45/Ethernet	RJ45/Ethernet
Provedení	modul na DIN lištu	modul na DIN lištu
napájení	12 až 24V AC/DC	24V DC
výstup pro připojení kabelu	ARK svorky	ARK svorky
Paměť	interní 2MB flash	Micro SD
webový teploměr	ano	ne (s 1-Wire modulem)
digitální vstupy	8x	8x
integrované relé	4x	8x
D/A výstup	ne	4x (0-10V)
A/D vstup	4x (0-30V)	4x (0-10V)
S0 vstup	ano	ano (vlastní program)
PWM výstup	ano	ne (ano s PWM modulem)
vlastní program	ano	Ano
vlastní HTML stránky	ano	ne (vše již připraveno)
připojení k Eportálu	ano	/
převodník RS485	ano	Ne
KNX/EIB sběrnice	ne	Ano
1-Wire sběrnice	ano 2x	ne (ano s 1-Wire modulem)
Cena (s DPH) serveru	4 220 Kč	12 499 Kč
Cena (s DPH) 1-Wire modulu	již obsažen	4 249 Kč
Cena (s DPH) celkem	**4 220Kč**	**16 748 Kč**

K systému Loxone jsem do ceny přidal 1-Wire modul, aby jak můj návrh systému, tak i tyto dva zmíněné obsahovali tuto sběrnici. Cenu Loxonu navršuje především propracovanější webové rozhraní, výkonnější systém, než jakým disponuje SDS MICRO, dále určitě podpora KNX/EIB sběrnice a další aspekty. Můj systému bude mít srovnatelné parametry se systémem SDS a navíc grafické webové rozhraní. Ovšem nebude k němu vytvořen software, přes který by bylo možné vytvářet další program pro řízení a ovládání dalších funkcí. Oba zmíněné komerční systémy tento software osahují, program se vytváří podobně jako pro PLC.

3 VÝBĚR PLATFORMY

K vytvoření mého systému pro inteligentní dům, bylo důležité vybrat správný mikroprocesor, který by zvládal obstarávat požadované funkce a byl schopný využívat ethernetových služeb. Nebo se dále nabízela možnost pořídit některou z vývojových desek.

U první možnosti by přicházel v úvahu některý procesor z řady ATmega, u kterého by ale mohli nastat potíže s jeho výkonem, či dostatečně velkou RAM pamětí při zpracovávání velkých webových stránek. Lepším kandidátem by proto bylo využití procesoru s ARM architekturou. Ale díky nezkušenosti s tímto procesorem, jsem od této možnosti vlastního návrhu desky a použití některého z těchto procesorů ustoupil.

V druhém případě bylo uvažováno o využití vývojové desky, ty mají velké zastoupení a je jich proto velký výběr. Při rozhodování byl hlavní faktor ten, aby byl na desce osazen i ethernetový modul, nebo k ní byl alespoň bezproblémově připojitelný.

3.1 Arduino

Prvním velmi rozšířeným je platforma Arduino, který využívá procesory z řady mega a je k němu možné připojit další moduly, (které se nazývají shieldy) a to například i onen požadovaný ethernetový. Programování je samozřejmě možné přes klasické ISP, nebo JTAG. Nejpoužívanější se však stává vývojové prostředí od výrobců Arduina, které s deskou komunikuje přes USB-seriový převodník. Procesor proto obsahuje předehraný framework. Pro psaní kódu je určen zjednodušený Arduino jazyk se syntaxí odvozenou od jazyku C.

3.2 Raspberry Pi

Druhou možností je využití platformy s názvem Raspberry Pi. Jedná se o platformu co má vše, co je potřeba, ethernetové rozhraní a výkonný procesor s ARM architekturou. Jedná se spíše o malý počítač s OS Linux a jeho hlavní přednost, kvůli které je asi i nejvíce kupován, je optimalizace pro přehrávání HD videa. Pro programování vlastní aplikace je v OS vývojové prostředí používající jazyk Python.

3.3 Panda II

Poslední zmíněnou platformou je ta, kterou jsem použil pro moji práci. Proto zde o ní budou uvedeny podrobnější informace. Jde o desku od GHI Electronics sídlící v USA, přímo o typ s názvem **Panda II**. Srdcem je mikrokontrolér NXP's LPC2387 s předehraným .NET Micro Frameworkem, rozložení konektorů na desce je stejné jako u Arduino platformy. Programování probíhá v prostředí Microsoft's free Visual C#., a tím vzniká nové využití jazyka C# pro programování mikrokontrolérů. Deska je s PC propojena pomocí USB. Také je zde plná podpora ladění (debugging) kódu, hlavně krokování a zkoumání proměnných. Pro ladění velkých, nevyzkoušených kódů je to velmi prospěšná a čas šetřící funkce. Prostředí a styl programování zůstal stejný tak, jak je typické pro Visual Studio.

3.3.1 Specifikace základní desky

- založena na NXP's LPC2387 micro-controlleru s GHI komerečním USBizi firmware/software balíčkem,
- 72MHz. 32- bit ARM7 procesor,
- 512 KB Flash (148KB pro vlastní aplikaci),
- 96 KB RAM (62KB pro vlastní aplikaci),
- kompatibilní se spousty Arduino moduly (shields),
- USB připojení pro ladění při běhu zařízení,
- speciální knihovny pro nastavení USB portu k emulaci virtuálního COM, myši nebo klávesnice,
- USB ladění a virtuální COM mohou pracovat současně,
- USB Host s podporou dalších modulů,
- vestavěná Micro-SD zásuvka s detekcí karty (S podporou vysokorychlostní SDHC, bez omezení karet nad 2GB),
- 54x digitálních I/O portů,
- 6x 10bitový A/D převodník,
- 10bit D/A (S možností přehrávání WAV),
- 6x Hardwarových PWM výstupů,
- 2x CAN sběrnice,
- s baterií záloha 2KB RAM,

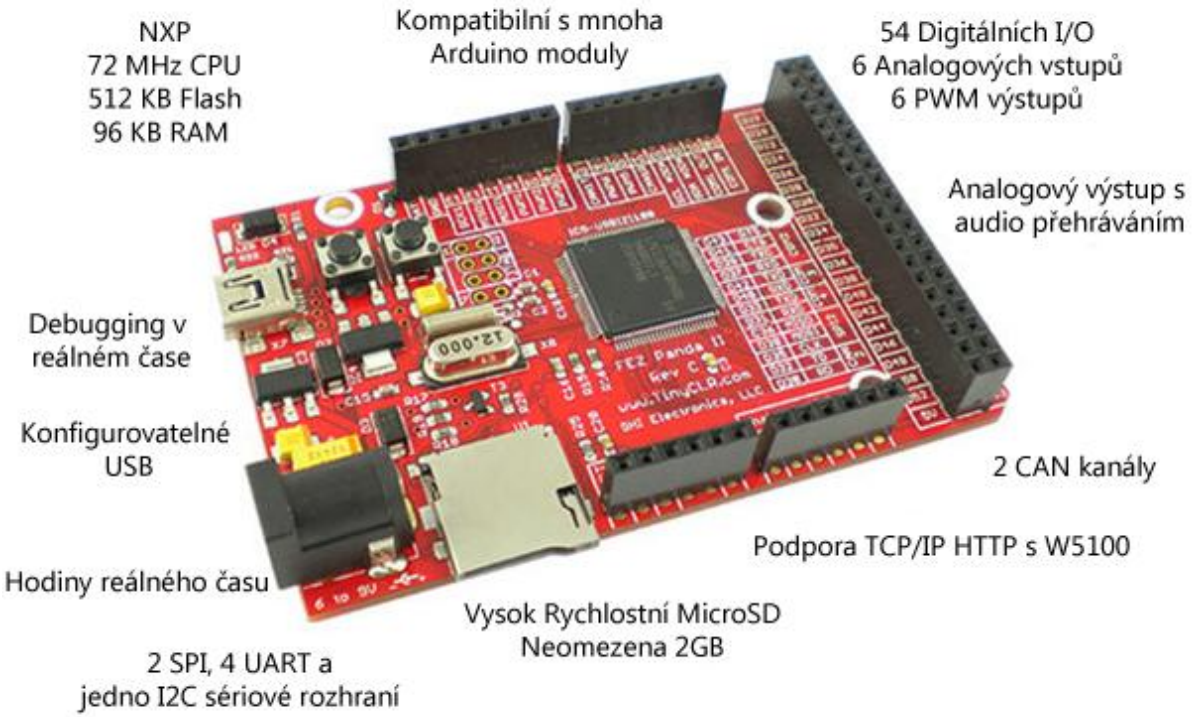
- na desce 1x LED a tlačítko,
- 4x UART sériové porty (jeden s možností hardwarového řízení toku),
- OneWire rozhraní (na každém I/O portu),
- vestavěné Real Time hodiny (RTC) s krystalem 32.768kHz,
- přistup k registrům procesoru,
- OutputCompare pro generování signálu s vysokou přesností,
- RLP umožňuje uživateli nahrát nativní (C/Assembler) kód pro požadavky v reálném čase,
- ethernetové rozhraní s W5100 a podporou TCP, UDP, HTTP, DHCP a DNS protokolů. Propustnost 500Kbps. Připraveno pro použití FEZ Connect modulu,
- rozšíření pro dvojitou přesnost matematických operací,
- paralelní port (ideální pro barevné displeje),
- vyvedený JTAG (použití jen pokud není obsažen originální Firmware),
- možnost využití více vláken najednou,
- XML, FAT souborový systém, podpora USB hosta,
- cryptografie (AES a XTEA),
- aktualizace z SD nebo sítě,

Napájení:

- prostřednictvím USB portu, nebo externího DC 6-9V napájecího zdroje (připojení obojího je možné),
- k dispozici je 3.3V stabilizovaný DC výstup,
- k dispozici je 5.0V stabilizovaný DC výstup,
- digitální I/O jsou 3.3V, ale 5V je také tolerováno,
- low power a hibernate módy,
- aktivní spotřeba 103 mA,
- Idle spotřeba 65 mA,
- spotřeba v hibernaci 3.75mA,

Prostředí:

- Vyhovující RoHS směrnici s použitím bezolovnaté pájky
- Provozní teplota: -20 to 65°C



Obrázek 4 Panda II

3.3.2 Ethernetový modul

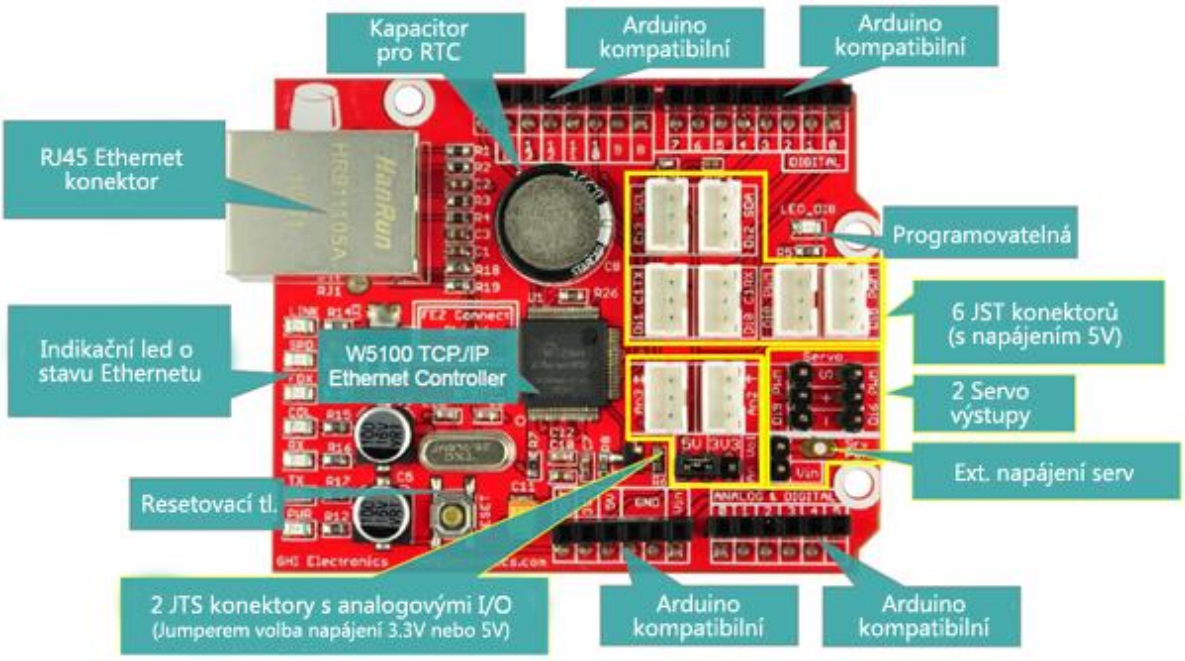
Důležitý shield, o kterém se zmíním je ethernetový modul. Díky nutnosti propojení „se světem“ je tento modul důležitou součástí. K obsluze linky je použit ethernetový řadič WIZnet TCP/IP s označením W5100 s podporou 10BASE-T a 802.3u 100BASE-TX.

3.3.3 Specifikace ethernetového modulu[2]

- Ethernet,
- komunikace po SPI sběrnici,
- 8 JST konektorů pro senzory,
- 2x standardní výstupy pro servomotory s možností využití odděleného napájení,
- 0.22 F kondenzátor pro udržení reálného času při odpojení napájení,
- resetovací tlačítko,
- LED připojená na Di8 (PWM),

[2] Specifikace převzaty ze stránky výrobce
http://www.ghielectronics.com/catalog/product/256

- 7 LED pro indikaci (Napájeni, Ethernetový provoz, rychlost a stav připojení),
- jumperem výběr napětí (5V nebo 3.3V) přivedené na JTS konektory An2 a An3(A/D i D/A převodník),
- 5 Voltový stabilizátor (pro kompatibilitu s Arduino),
- s Arduino kompatibilní rozestupy konektorů,
- rychlost 500Kbps (62.5KB/s),
    - (Změřená rychlost 40KB/s),



Obrázek 5 Ethernetový modul (shield)

3.3.4 Důvody výběru

Zde bych chtěl poznamenat důvody vybrání této desky. Určitě rozhodujícím faktorem bylo, aby deska obsahovala relativně výkonný mikrokontrolér, a proto jsem od Arduino platforem odstoupil. Potom se nabízela možnost využít zmíněný počítač Raspberry Pi, který byl silným kandidátem díky výkonnému procesor i dalšími parametrům. Ale protože pro programování vlastní aplikace bylo nutné použít jazyk Python, a též bylo potřeba dobré orientace v Unixovém systému, vyhrála třetí varianta GHI Panda II, kde se stal rozhodujícím faktorem programovací jazyk, a také již známé vývojové prostředí Microsoft Studio C#.

4 SÍŤOVÉ PROTOKOLY APLIKAČNÍ VRSTVY

V práci bude využito několik aplikačních protokolů, zprostředkovávající komunikaci klient – server. Proto zde některé uvedu, aby hlouběji přiblížili, jak a s jakou syntaxí probíhá komunikace. Půjde například o protokol sloužící mikrokotroléru k nastavení aktuálního času či protokol umožňující odeslání emailu.

4.1 NTP

Protokol NTP (Network Time protokol) je služba pro synchronizování hodit počítačů a dalších zařízení připojených k síti. Jedná se o velmi přesný způsob nastavení hodin, hlavně díky uvažování proměnlivého zpoždění paketů. Pro transport dat je použit protokol UDP s naslouchacím portem serveru 123. Protokol v sobě nese několik informačních hodnot pro nastavení co nejpřesnějšího času, při správném algoritmu a využití dat od několika NTP serveru se dosáhne času s odchylkou maximálně +/- 10ms, v lokální sítí při ideálních podmínkách může být dosaženo přesnosti až 200 mikrosekund.

NTP vznikl hned po příchodu TCP/IP komunikačního protokolu, navrhl ho Dave Mills z univerzity v Delaware. Protokol je stále vyvíjen a současná verze je NTP 4, kterou popisuje RFC 5905.

Přesný čas server dostává od cesiových hodin. Základním časem je tzv. TAI (Internation Atomic Time), kdy jedna sekunda odpovídá 9 192 631 770 oscilací cesiového atomu. TAI je dále upravován, podle zeměpisných jevů, kdy dochází občas k drobným korekcím.

Přijatý paket v sobě nese několik informací, důležitými jsou, čas kdy žádost opustila klientův počítač a čas kdy byla žádost přijata a poté čas odeslání odpovědi serverem. Z těchto informací a dalšího klient dopočítá co nejpřesnější čas s ohledem na zpoždění paketů.

Zjednodušenou formou je tzv. SNTP (Simple NTP). Klient nepoužívá tak složitý algoritmus, nepropočítává zpoždění, proto se dosahuje menší přesnosti, ta však postačuje pro většinu dnešních systému, kde čas hraje pouze informační roly pro uživatele.

4.1.1 Formát přijatého času

Přijatý čas nemá formu takovou, jakou bychom běžně čekali. Hodnota je uložena v 8 bajtech a představuje počet sekund od 1. 1. 1900. Přesněji jde o číslo s pevnou desetinou čárkou, 32bitů pro sekundy a zbylých 32bitů pro desetinou část sekund. Maximální hodnota je tedy 2^32, což je přibližně 136 let, s teoretickým rozlišením 233 pikosekund. Roku 2036 dojde tedy k přetečení časového rámce a bude potřeba výchozí rok 1900 změnit na 2036.

Tabulka 2 Formát času u NTP

b.	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
40	Vteřiny																															
72	Desetinná část vteřin																															

Blok zobrazený v tabulce je část odpovědi od serveru. Nese v sobě přesný čas odeslání paketu. Blok začíná od 40. bajtu celé odpovědi. Celá datová část paketu je zobrazena v příloze A.

Čas je v UTC formátu, proto je nutné po přijetí provést úpravu, podle našeho časového pásma. V našem případě se jedná o přičtení 1 hodiny.

Algoritmus v mém zařízení pracuje tak že z odpovědi od serveru převezme 64 bitovou informaci o času, kterou převede na milisekundy a přičte ji k datu 1. 1. 1900 dále výsledný datum a čas s přičtenou jednou hodinou zapíše do vnitřních hodin mikrokontroléru.

České NTP server jsou například **ntp.eri.cz** a **tik.cesnet.cz.**

4.2 SMTP

SMTP je protokol pro přenos elektronické pošty, klient pomocí něho komunikuje se serverem obstarávající přenos správ do schránky adresáta. První norma popisující tento protokol byla vydána už v roce 1982. Poslední norma byla vydána roku 2001 pod označením RFC 2821. Využívá spojovaného protokolu TCP na portu 25. Ukázka komunikace klienta se serverem je zobrazen v následující tabulce.

Tabulka 3 Průběh komunikace s SMTP serverem seznam.cz

	K:	Navázání TCP spojení s **smtp.seznam.cz** na port **25**
	S:	**220 2.0.0 Seznam SMTP server waiting for your HELO/EHLO**
1	K:	HELLO seznam.cz\r\n
	K:	AUTH LOGIN\r\n
	K:	“přihlašovací jméno (BASE64)“\r\n
	K:	“přihlašovací heslo (BASE64)“\r\n
	S:	**334 VXNlmb5hbWU6** **334 UGFzbxdvcmQ6** **235 2.2.0 Authentication succeeded**
2	K:	MAIL FROM:<odesilatel@seznam.cz>r\n
	S:	**250 2.1.0 Ok <smssmejk@seznam.cz>**
3	K:	RCPT TO:<příjemce@seznam.cz>r\n
	S:	**250 2.1.5 Ok <smssmejk@seznam.cz>**
4	K:	DATA\r\n
	S:	**354 Enter message, ending with <CRLF>.<CRLF>**
5	K:	Subject: “předmět“\r\n
	K:	From: odesilatel@seznam.cz\r\n
	K:	To: příjemce@seznam.cz\r\n\r\n
	K:	“text“\r\n
	K:	.\r\n
	S:	**250 2.0.0 Mail 473507169 queued for delivery in session 7fdf00000000.**
6	K:	QUIT\r\n
	S:	**221 2.0.0 Thanks for your visit, have a nice day.**
7	K:	Ukončeni TCP spojení

K: klient, S: server

Znaky **\r** a **\n** jsou používané značky k zalamování řádků.

V předchozí tabulce je zobrazena reálná komunikace. Pro komunikaci byl náhodně vybrán emailový server seznam.cz. Program probíhá ve smyčce While() po přijetí odpovědi od serveru se přečtou první 3 bajty a převedou na textový řetězec, poté na číslo a pokud souhlasí s hodnotou potvrzující správné přijetí, odešle se odpovídající blok dat a znovu se čeká na odpověď. Přihlašovací jméno a heslo musejí být zakódovány algoritmem BASE64 pro tento převod je v .NET frameworku funkce. Odpověď serveru „**334 VXNlmb5hbWU6**“ je opět BASE64 zakódovaný text „334 UserName:“ a následující odpověď „**334 UGFzbxdvcmQ6**“ je „334 Password:“.

Kódování BASE64 je převod binárních dat do tisknutelných ASCII znaků a nejedná se šifrování. Text k převodu, je v binární podobě spojen do jednotného bloku, který je pak rozdělen na 6 - ti bitové úseky které určují pořadí znaku v abecedě: ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/.

Díky použití 6 - ti bitových úseků a poté jejich převodu na 8 bitové znaky dojde k navýšení výsledného řetězce o zhruba 33%.

5 SBĚRNICE

5.1 1- wire

Sběrnice byla vyvinuta a je nejvíce používána firmou Dallas Semiconductor.
Pro její realizaci jsou za potřebí jenom dva vodiče, signálový a zemnící. Může být vždy připojen jeden master a jeden, nebo více slave zařízení. V klidu je ve stavu logické 1, na kterou ji zvedá odpor 4.7kΩ na napětí 3.6 – 5V.

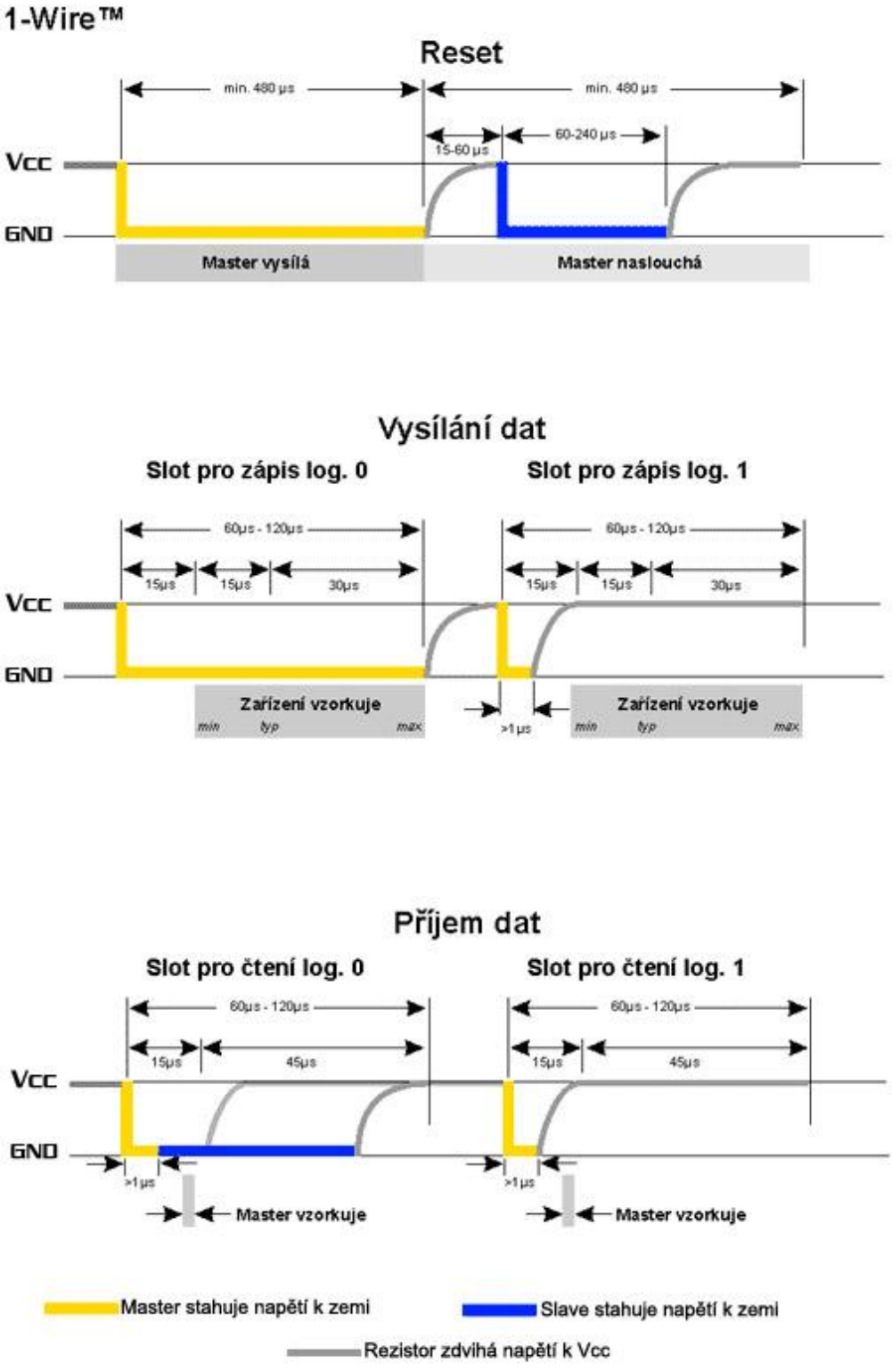
5.1.1 Fyzická vrstva

Komunikaci zahajuje vždy master reset pulsem. Nejprve "stáhne" datový vodič do log. 0 a drží ho na této úrovni minimálně 480 mikrosekund. Pak sběrnici uvolní a naslouchá. Odpor zatím vrátí sběrnici zpět do log. 1. Pokud je na sběrnici připojené nějaké 1-Wire zařízení, tak detekuje tuto vzestupnou hranu a po prodlevě (15 - 60 µs) stáhne sběrnici na 60 - 240 µs k log. 0.

Pokud se zařízení správně ohlásí, může master začít vysílat a přijímat data. Data jsou vysílána v tzv. "time slotech", česky bychom řekli nejspíš v "časových úsecích", nebo v "okénkách". Slot je dlouhý 60 až 120 µs a během jednoho slotu je vyslán, nebo přijat jeden bit informace. Mezi jednotlivými sloty musí být minimálně 1 µs mezera, kdy je sběrnice v klidu.

Existují 4 druhy slotů: Zápis 1, Zápis 0, Čtení 1 a Čtení 0. Zápisové sloty slouží k tomu, aby master vyslal data do zařízení. Zápis log. 1 probíhá tak, že master stáhne sběrnici k nule minimálně na 1 µs a nejpozději do 15 µs od začátku, ji opět uvolní a ponechá uvolněnou. Zdvihací odpor ji tedy vytáhne k log. 1. Zápis log. 0 je o něco jednodušší: Master stáhne sběrnici k 0 a ponechá ji tak po celý slot, tedy min. 60 µs. Zařízení vzorkuje stav na datovém vodiči zhruba 30 µs po začátku timeslotu.

Čtecí sloty opět inicializuje master tím, že stáhne sběrnici k nule na minimálně 1 µs, a opět ji uvolní. Po tomto zahájení může zařízení vyslat 1 bit buď tím, že ponechá sběrnici v klidu (log. 1) nebo ji stáhne (log. 0). Podrobnosti snad osvětlí následující obrázek.[5]



Obrázek 6 Průběhy na 1- Wire sběrnici

5.1.2 Časování

Reset:

- stáhnout sběrnici na 0, počkat 480 μs,
- uvolnit sběrnici, počkat 70 μs,
- přečíst sběrnici. Je-li ve stavu log. 0, je na ní připojeno nějaké zařízení. Pokud je v log. 1, žádné zařízení připojeno není,
- počkat 410 μs,

Zápis log. 1:

- stáhnout sběrnici na 0, počkat 6 μs,
- uvolnit sběrnici, počkat 64 μs,

Zápis log. 0:

- stáhnout sběrnici na 0, počkat 60 μs,
- uvolnit sběrnici, počkat 10 μs.

Čtení:

- stáhnout sběrnici na 0, počkat 6 μs,
- uvolnit sběrnici, počkat 9 μs,
- přečíst sběrnici. Její stav udává přečtený bit,
- počkat 55 μs,

5.1.3 Komunikace s více slave zařízení

Pokud je na sběrnici více zařízení, je potřeba je od sebe nějakým způsobem rozlišit. Pro tento účel obsahuje každý slave svojí jedinečnou 64bitovou adresu uloženou v ROM paměti.

Nejdříve je potřeba vyslat příkaz MATCH_ROM[55h], ten připraví zařízení pro naslouchání, pokud se vyslaná adresa neshoduje s adresou zařízení, odmlčuje se a přestává po sběrnici komunikovat. Proto na ní zůstane aktivní jenom jediné zařízení, které má správnou adresu. Více o jedinečné adrese v kapitole o senzoru DS18B20.

6 PERIFERIE

Při výběru teplotních senzorů bylo rozhodující kritérium to, aby bylo možné k mikrokontroléru připojit přes desítku senzorů. Použití měřícího čidla jako samotný termistor, PN přechod nebo termoelektrický článek, by přinášelo nutnost využití tolik A/D převodníků, kolik je samotných čidel. Při nedostatku A/D převodníků by byla možnost použít multiplexer, který by časově řadil signály z čidel a přiváděl je na jediný převodník. Jelikož mikrokontrolér obsahuje pouze 6 A/D převodníků, nepřichází první řešení v úvahu. S další možností by se tento problém vyřešil, ovšem bylo by potřeba vyřešit použití a řízení multiplexeru. Všechny tyto problémy vyřeší integrované senzory, které komunikují po sběrnici, a tím pádem přinášejí i největší výhodu redukci kabeláže, a také minimální náchylnost na rušení. Z těchto důvodů byly pro měření teploty zvolené dále rozebírané senzory DS18B20.

6.1 Senzor 18B20

Jedná se o digitální teplotní senzor komunikující po zmíněné 1-Wire sběrnici. Senzory jsou vyráběny ve třech pouzdrech TO-92, μSOP a SO, poslední dvě jsou SMD provedení. Senzor má vždy použité jen 3 vývody a to +, GND a data. Nejsou určeny do náročných prostředí, zejména do vlhkého, zde je velká pravděpodobnost, že se při delším působení dostane vlhkost dovnitř pouzdra. Proto je vhodně používat senzory zalité ve speciální hmotě.

6.1.1 Základní parametry

- Sběrnice pro komunikaci 1-Wire,
- neměnná 64 bitová adresa zařízení,
- pracující bez dalších vnějších součástek,
- napájecí napětí od 3V do 5.5V,
- rozsah měřené teploty od -55°C do +125°C (-67°F do +257°F),
- maximální chyba změřené teploty,
  ±0.5°C pro rozsah (-10°C to +85°C),
  ±2°C pro (-55°C to +125°C),
- nastavitelné rozlišení převodníku 9bit,10bit,11bit a **12 bitů (nastaveno od výroby)**.

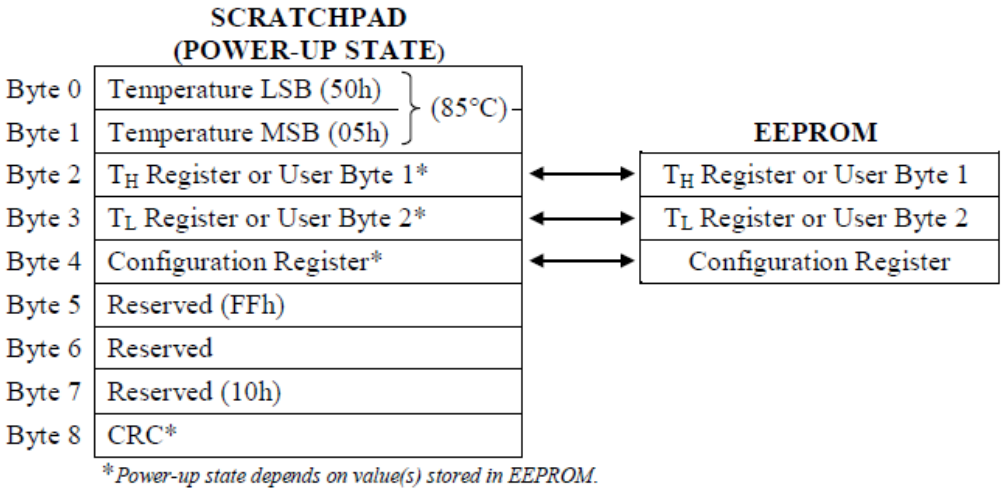
Tabulka 4 Časy převodu teplot senzoru 18B20

Rozlišení převodníku	9bitů	10bitů	11bitů	12bitů
Čas převodu	93.75ms	187.5ms	375ms	750ms

Tabulka 5 Proudový odběr senzoru 18B20

	VDD	Typicky	Maximum
Standby mód	5V	750 nA	1 μA
Aktivní mód	5V	1 mA	1.5 mA

Senzor obsahuje dvě paměti, SRAM (SCRATCHPAD) a nonvolatilní EEPROM. V paměti SRAM je na prvních dvou pozicích uložena teplota. Na prvním místě je LS (nižší) byte a na druhém (MS) vyšší. Důležité je uvést, že záporná teplota je uložena jako dvojkový doplněk. Dále zde jsou dva registry pro nastavení teploty při použití senzoru jako termostatu. Poté nastavovací registr senzoru a poslední byte je CRC počítaný z prvních 56bitů SRAM paměti. Pro lepší znázornění je na následujícím obrázku uvedeno rozdělení paměti.



Obrázek 7 Rozdělení paměti senzoru 18B20

Prezentace teploty v paměti už byla uvedena, a tak dalším významným registrem je nastavovací (Configuration Register). Slouží pro nastavení rozlišení převodníku, jak je uvedeno v následující tabulce. (Zbývající bity nejsou použity).

Tabulka 6 Nastavovací registr senzoru 18B20

Bit 6	Bit 5	Rozlišení [Bitů]
0	0	9
0	1	10
1	0	11
1	1	12

Tabulka 7 Ukázka reprezentace teplot senzoru

Teplota [°C]	Binárně	Hexadecimálně
+125	0000 0111 1101 0000	07D0h
+85*	0000 0101 0101 0000	0550h
+10.125	0000 0000 1010 0010	00A2h
+0.5	0000 0000 0000 1000	0008h
0	0000 0000 0000 0000	0000h
-0.5	1111 1111 1111 1000	FFF8h
-10.125	1111 1111 0101 1110	FF5Eh
-55	1111 1100 1001 0000	FC90h

*Po zapnutí senzoru je v paměti uložena teplota +85°C

Převod z binární podoby do stupňů Celsia je následující, pro kladné hodnoty bude číslo vždy v rozmezí od 0-2048, potom ho stačí vydělit 16 pro získání stupňů Celsia. Při hodnotě nad stanovenou mez je číslo záporné, proto je nutné po vydělení 16 ještě odečíst 4096, čímž se dostaneme s číslem do záporných hodnot.

6.1.2 Připojení senzoru

Jsou dva způsoby jeho zapojení, jedná se o klasické připojení třemi vodiči, nebo jenom dvěma. První případ je nejjednodušší a není zde potřeba nic jiného řešit. Při připojení dvěma vodiči je nutné mít na datový vodič přes tranzistor případně mosfet přivedeno napájecí napětí. Pokud je potřeba přečíst data spínací prvek je nutno zavřít a senzor z uložené energie ve vnitřním kondenzátoru obslouží dotaz o přečtení teploty.

6.1.3 Příkazy pro komunikaci

- **[44h]** Důležitý příkaz, slouží pro spuštění převodu teploty u senzoru.
- **[4Eh]** Zapsání dat do SCRATCHPAD (první se odešle Th, poté Tl a nakonec nastavovací byte).
- **[BEh]** Přečtení SCRATCHPAD.
- **[55h]** Po tomto příkazu naslouchá 64bit adresu.
- **[48h]** Uložení SCRATCHPAD do EEPROM.
- **[B8h]** Obnovení SCRATCHPAD z EEPROM.
- **[F0h]** Hledání ROM (s identifikační adresou). Pro přečtení adres všech zařízení na sběrnici
- **[33h]** Přečtení ROM (s identifikační adresou). Možnost použít jen pokud je na sběrnici jeden slave.
- **[CCh]** Tento příkaz slouží k oslovení všech slave zařízení najednou, bez nutnosti zasílání adresy konkrétního zařízení.

Celý průběh komunikace bude uveden v kapitole s obsluhou senzorů, kde budou řešeny funkce pro čtení teplot a pro hledání nových zařízení.

6.2 Elektroměr a výstup S0

Elektroměr je zařízeni pro měření odebrané elektrické energie. Integrací odebíraného výkonu získává spotřebovávané energie. Nové digitální elektroměry mají impulzní výstup pro odečet aktuálního přenášeného výkonu, tento výstupy je definované normou DIN 43864 a označuje se S0. Používají ho nejenom elektroměry ale také jiné měřiče jako například plynoměry, vodoměry a jiné. Díky tomuto výstupu můžeme mikrokontrolérem lehce měřit impulzy a z jejich četnosti zjišťovat velikost měřené veličiny.

Při využití těchto funkcí na vlastním zařízení není žádný problém, ale pokud by byl elektroměr ve vlastnictví distribuční společnosti je potřeba o využití výstupu S0 požádat, ta poté za poplatek nainstaluje k výstupu oddělovací prvek, nejspíše optočlen, který zajistí logický výstup pro připojení mikrokontroléru. Výstupem nejsou napěťové impulzy, ale je spínán polovodičovým prvek, proto je dobře dodržovat polaritu přivedeného napětí. Piny

mají označení S+ a S-. Napětí přivedené na kontakt by nemělo přesáhnout 27V a maximální přípustný proud kontaktem je udáván 27mA.

Každé zařízení má definovaný počet impulzů na jednotku měřené veličiny. Pro co nejaktuálnější hodnotu příkonu je nutné měřit čas mezi impulzy. Poté získaná hodnota po přijetí impulzu bude střední hodnotou od předchozího do tohoto impulzu. Nevýhoda je toto zpoždění, kde při malém počtu impulzů na jednotku veličiny dostáváme průměrnou hodnoty z velkého časového úseku, při malém odběru se může jednat i o několik minut. Snímání spotřebované energie je řešeno ta, že po 10 zaznamenaných impulzech se přičte jim odpovídající energie.

Vztahy:

$$K = \frac{imp.}{kWh},$$

Kde: ***K*** určuje vztah mezi počtem impulzů a měřenou veličinou a je udávána na měřicím zařízení.

$$P = \frac{3600}{K} \cdot \frac{1}{t} \qquad (kW; s),$$

$$P = \frac{3600000}{K} \cdot \frac{1}{t} \qquad (W; s).$$

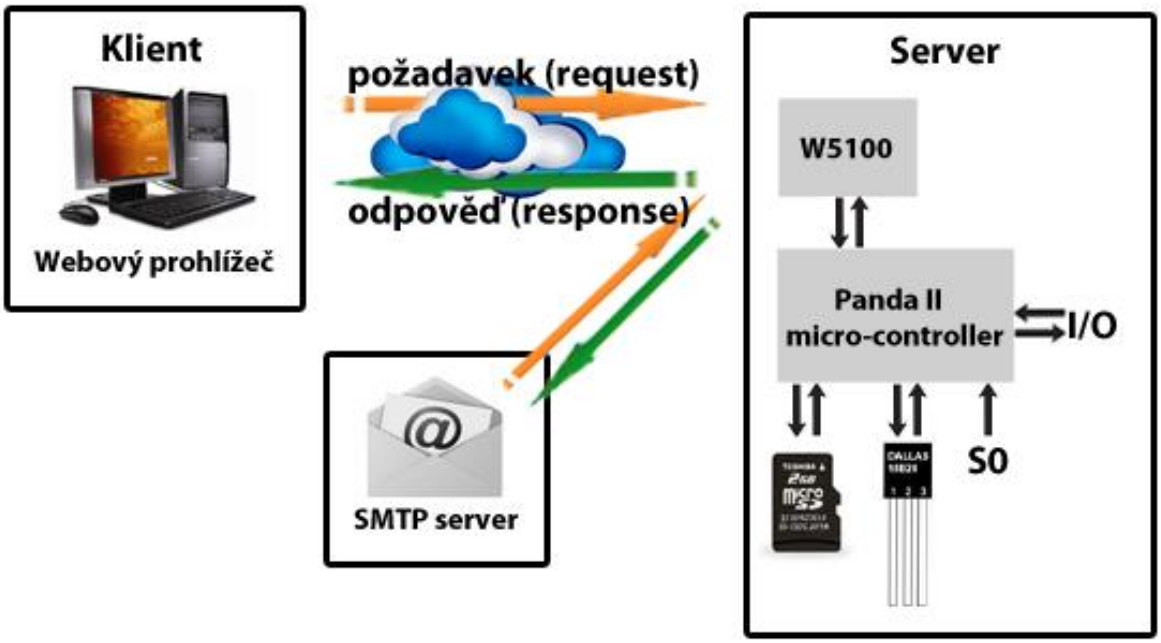
Vztah pro velikost aktuálního příkonu ve wattech.
Kde: ***t*** je čas mezi jednotlivými impulzy v sekundách.

Přírůstek energie po přijetí 10 impulzů má velikost ΔE podle vztahu.

$$\Delta E = \frac{10}{K} \qquad (kWh).$$

7 VLASTNÍ ŘEŠENÍ

Pro návrh mého systému inteligentního domu byly zvoleny tyto základní funkce, které by zařízení mělo zvládat. V první řadě se jedná, o vzdálený přistup ke všemu, co systém sleduje. Půjde o řešení pomocí ethernetového rozhraní, které bude zprostředkovávat přístup k webovým stránkám, na kterých bude vše v přehledné grafické podobě. Jednou ze základních funkcí bude sledování teplot pomocí několika senzorů, například pro sledování teplot v místnostech, na kamnech či akumulační nádrži. Další sledovanou veličinou je elektrický příkon domu, pro zjištění této veličiny budou použity digitální elektrické hodin domu. Ty mají výstup s frekvencí pulzů odpovídající průměrnému příkonu. Dále se bude jednat o sledování, zda je aktivní nižší tarif či nikoli (tzv. HDO). A také samozřejmou věcí je ovládání výstupů mikroprocesoru a sledování vstupů. Také bude možnost ve webovém rozhraní nastavit jednoduchou regulaci. Sledovány budou čas, teplota či napětí na zvoleném převodníku a podle nastavených mezních hodnot bude ovládán vybraný výstup. Další bude funkce pro dálkové informování uživatele o výjimečných situacích či aktuálních teplotách. Pro tuto funkci byl zvolen způsob informování pomocí e-mailu. K tomuto řešení jsem přistoupil z důvodu placených SMS brán. O SMS správu však nebudeme ošizeni, převod bude řešen přímo u operátora, kteří nabízejí služby pro přeposílání emailů do mobilního telefonu.



Obrázek 8 Schéma komunikace serveru

7.1 Ethernetová komunikace

Síťové připojení je řešeno pomocí již zmíněného ethernetového modulu propojeného s hlavním micro-kontrolerem pomocí SPI sběrnice. Tento modul je díky W5100 schopen používat tyto základní protokoly TCP, UDP, HTTP, DHCP a DNS. Pro moji práci postačí ruční obsluha TCP a HTTP protokolů pro webové rozhraní a UDP protokolu pro odesílání emailů.

Prvotní nastavení spočívá v inicializaci. Je nutné nastavit PINy, které jsou použity pro komunikaci po SPI sběrnici. Ty jsou v tomto případě pevně dány díky propojení modulu pomocí konektoru s přesně danými roztečemi. K inicializaci patří nastavení adres (IP, MAC atd.).

Pro komunikaci s modulem je využita knihovna `GHIElectronics.NETMF.Net`

Ve třídě obsluhující WEB server jsou tyto důležité objekty:

```
HttpListenerResponse response = null;    //odpověď
HttpListenerRequest request = null;      //žádost
HttpListenerContext context = null;
```

Inicializační funkce má takovýto tvar:

```
private void ZapnoutSit()
{
WIZnet_W5100.Enable(SPI.SPI_module.SPI1, (Cpu.Pin)FEZ_Pin.Digital.Di10,
(Cpu.Pin)FEZ_Pin.Digital.Di7, true);

            byte[] ip = { 192, 168, 1, 200 };
            byte[] maska = { 255, 255, 255, 0 };
            byte[] gateway = { 192, 168, 1, 254 };
            byte[] mac = { 0x00, 0x1F, 0x16, 0xC8, 0x31, 0xBE };

            NetworkInterface.EnableStaticIP(ip, maska, gateway, mac);
            NetworkInterface.EnableStaticDns(new byte[] { 86, 61, 244, 1 });
}
```

Třída WIZnet_W5100 s funkcí Enable obsahuje tyto vstupní parametry:

- `SPI.SPI_module.SPI1` – je určení že komunikace proběhne pro SPI1 rozhraní
- `(Cpu.Pin)FEZ_Pin.Digital.Di10` – je parametr který určí na jaký pin procesoru je připojen CS(Chip Select) zpřístupňující komunikaci s modulem.
- `(Cpu.Pin)FEZ_Pin.Digital.Di7` – je další parametr určující pin pro externí resetování W5100
- `Poslední parametr` – jedná se o logickou hodnotu, která určuje, zda se má v paměti vyčlenit jeden socket (ze čtyř) pro zajištění plynulosti DNS a DHCP služeb.

Další nezbytnou věcí je nastavení adres. První v pořadí je IP adresa určující adresu zařízení. Dále je maska, která slouží k oddělení síťové části. Gateway je výchozí braná. Jedná se například o router, na který jsou odesílána data, pokud nepatří do vnitřní sítě.

MAC adresa je jedinečná adresa zařízení. Při komunikaci je tato adresa vkládána do linkového rámce. Při prvních testech programu nefungovala s modulem komunikace, poté se ukázalo, že chyba je v MAC adrese. Protože adresa musí být ve správném rozsahu, není možné použít náhodně vymyšlenou posloupnost čísel. Mnou použitá MAC je adresa PC, a aby nedocházelo ke komplikacím, adresu jsem o jedničku inkrementoval. Tím byl problém s nefunkční komunikací vyřešen.

V další funkci už dojde ke spuštění serveru, je nutné přiřadit číslo portu, na kterém bude server naslouchat a také realizovat nekonečnou smyčku pro zpracování přijatých požadavků.

```
public void StartWebServer()
        {
            HttpListener listener = new HttpListener("http", 80);
            listener.Start();

            while (true)
            {
                try
                {
                    response = null;
                    request = null;
                    context = null;

                    context = listener.GetContext();
                    if (context == null) continue;
                    response = context.Response;
                    request = context.Request;

                    ProcessRequestData(request);

                    string path = CestaZurl(request.RawUrl);
                    SendResponse(response,path);
                }
                catch (Exception x)
                {
                    Debug.Print("chyba " + x.ToString());
                }
                finally
                {
                    if (context != null) context.Close();
                    Debug.GC(true);
                }
            }
        }
```

Třída `HttpListener` vytvoří obsluhu http protokolu. V dalším řádku dojde ke spuštění naslouchání na zvoleném portu. Čísla portů jsou všeobecně známá, u HTTP jde o port 80 a u šifrovaného HTTPS o číslo 443. A však není nutnost se tohoto doporučení držet. Je možné použít i jiné číslo. V tom případě se docílí lehkého zabezpečení. Přístup k serveru bude mít jenom ten, kdo zná použité číslo portu. A však opravdu se jedná jen o

velmi primitivní zabezpečení, při použití scanneru portů se rychle zjistí, které porty jsou na dané IP adrese využity.

V dalších krocích dojde k obsluze žádosti od klienta (webového prohlížeče) na náš server. Jako první se obslouží přijaté instrukce, které může dotaz obsahovat. Obsluha je provedena funkcí `ProcessRequestData(request).`

Vstupním parametrem je `HttpListenerRequest`, objekt s názvem request. V tomto objektu jsou obsažena všechna data http protokolu. Z nich je pro nás důležitá takzvaná dotazovací metoda POST. Proto je jako první zjištěno, zda přijatý požadavek obsahuje tuto metodu, pokud ano z objektu se převede funkcí GetRequestContent(request) na text. Tento text je velmi důležitý, obsahuje protokol s jednoduchou syntaxí. Po určení pro jaké nastavení je určen, dojde k jeho zpracování. Nejčastěji se v něm budou nacházet nastavovací data systému. Může se například jednat o požadavek na změnu některého z výstupu.

```
private void ProcessRequestData(HttpListenerRequest request)
        {
       if ((request.HttpMethod != "POST") || (request.ContentLength64 == 0))
       return;       //Kontrola zda je v dotazu obsažena metoda POST

            String requestContent = this.GetRequestContent(request);
            bool vystupy = requestContent.IndexOf("vystupy") != -1;
            bool elektromer = requestContent.IndexOf("elektromer") != -1;
            bool restart = requestContent.IndexOf("restart") != -1;

            if (vystupy)
            {
                dataVystupy = requestContent;
                if (zmenaVystupu != null) zmenaVystupu(requestContent);
       //Vyvolá přerušení a ve funkci zmenaVystupu() nastaví data
            }
            if (restart)
            {
              if(requestContent.Split(';')[1] == heslo)
              PowerState.RebootDevice(true);
       //Pokud odpovídá heslo, dojde k restartování procesoru
            }
       … zkrácený kód
        }
```

7.1.1 Přehled nastavovacích protokolů a jejich syntaxí

Přehled dat, kterými je možné nastavovat mikrokontrolér. Data se odesílají na IP sítového rozhraní v podobě HTTP dotazu, s daty obsaženými v POST dotazovací metodě. Dá se říct, že se jedná o jednoduché protokoly. Jako oddělovací znak byl zvolen středník.

Tabulka 8 Syntaxe nastavovacích příkazů

	Text v POST metodě
1	`elektromer ; heslo ; kWh ; imp./kWh` příklad: (elektromer;123;9999;800)
2	`cas ; heslo ; Rok,Měsíc,Den,hodina,minuta,vteřina` příklad: (cas;123;2013,1,1,12,00,5)
3	`Vystupy ; heslo ; 1 = (1/0) ; 2 = (1/0) ; … ; 8 = (1/0)` příklad: (vystupy;123;1=0;2=1;3=1)
4	`tabulka \ typ ; ton[h] | Ton ; ton[m] ; toff[h] | Toff ; toff[m] ; senzor ; inverzně? ; výstup \ …\ …` příklad: (tabulka\1;12;00;15;00;0;0;1\2;25;0;20;0;3;0;2)
5	`adresy ; heslo ; adresa(1) \n adresa(2) \n adresa(n)` příklad: (adresy;123; 40,151,254,155,3,0,0,191\n40,7,246,189,3,0,0,73)
6	`adresyRefresh ; heslo` příklad: (adresyRefresh;123)
7	`restart ; heslo` příklad: (restart;123)

Vysvětlení:

1) *kWh* - hodnota ke které se připočítává spotřeba
   imp./kWh - údaj o počtu impulzů na kWh (podle elektroměru).
2) Nastavení hodin mikrokontroléru.
3) Nastavení výstupů, číslo před rovnítkem udává kolikátý výstup má být nastaven a číslo 1 nebo č. 0 za rovnítkem určuje logickou hodnotu výstupu.
4) Nastavení hodnot v tabulce, parametry stejné jako v poli pro řízení automatické regulace.
5) Nastavení adres senzorů 18B20. Pořadí adres odpovídá pořadí přijatých teplot. Adresy se zapíší do souboru *adresy.home* na paměťové kartě.
6) Spustí se funkce prohledání sběrnice. Pokud se nalezne nová adresa senzoru, zařadí se nakonec seznamu.
7) Odesláním tohoto příkazu se mikrokontrolér restartuje.

7.1.2 Odeslání dat

Poslední důležitou částí kódu v tomto bloku je tento:

```
string path = CestaZurl(request.RawUrl);
SendResponse(response,path);
```

Funkce `CestaZurl()` převádí adresu zadanou v prohlížeči na tvar, se kterým pracují metody pro obsluhu souborů na paměťové kartě. Standardní adresa zadaná v internetovém prohlížeči je rozdělena pravými lomítky ‘/‘, a však k určení cesty souboru jsou potřeba opačná lomítka. Například adresu */SD/index.html* na tvar *\SD\index.html*.

Poté je hned zavolána funkce `SendResponse(response,path)` se dvěma vstupními parametry, první je objekt `HttpListenerResponse` a druhý naše zmíněná cesta k souboru.

Část funkce vypadá takto:

```
private void SendResponse(HttpListenerResponse response, string strFilePath)
        {
            if (File.Exists(strFilePath))
            {
                SendFileOverHTTP(response, strFilePath);
                #region Data pro odeslání
            }
            else if (strFilePath == "\\SD\\info.cas.home")
            {
                response.StatusCode = (int)HttpStatusCode.OK;
                response.ContentType = "text/html";

string data = RealTimeClock.GetTime().ToString("dd.MM.yyyy") + "\nEND";
response.ContentLength64 = data.Length;
response.OutputStream.Write(UTF8Encoding.UTF8.GetBytes(data),0, data.Length);
            }
            else if (strFilePath == "\\SD\\vystupy.home")
            {
                …
            }
        }
```

V první podmínce se zjišťuje, zda se na paměťové kartě nachází soubor podle zadané cesty, pokud takový soubor existuje, přejde se na funkci `SendFileOverHTTP(response,cesta)`, která nejdříve podle přípony souboru nastaví odpovídající index do HTTP hlavičky, aby přijímající strana věděla, o jaký druh souboru se jedná (jpg, avi, html, atd.).

Poté se přejde k postupnému načítání a odesílání souboru po 2048B částech. Tuto velikost jsem zvolil až po občasných chybách při odesílání, kdy při větších blocích nebyl mikroprocesor schopen v RAM paměti najít dostatečně velký blok.

Ukázka odeslání dat pomocí HTTP:

Z předchozí funkce jsou vidět čtyři základní kroky pro správné odeslání dat.

```
response.StatusCode = (int)HttpStatusCode.OK;
```

Zde se nastaví status kód v http hlavičce. Status OK má dekadickou hodnotu 200.

```
response.ContentType = "text/html";
```

Dále následuje již zmíněný index s informací o formátu přenášených dat.

```
response.ContentLength64 = data.Length;
```

Poté určení délky dat.

```
response.OutputStream.Write(UTF8Encoding.UTF8.GetBytes(data), 0,
data.Length);
```

V posledním kroku dojde k odeslání dat. Prvním parametrem jsou data k odeslání, které je nutné převést z textového řetězce na sekvenci bajtů. Používá se kódování UTF8. Jedná se o podobný způsob kódování textu, jako je ASCII. Znak může být zakódován do jednoho bajtu, pak je kompatibilní s ASCII, nebo až do 6 bajtů. Další parametry určují pořadí od kolikátého a do jakého bajtu se mají data odeslat.

7.1.3 Přehled a struktura odesílaných informací

Tabulka 9 Přehled a struktura odesílaných informací

	Adresa	**Vrácený text**
1	teploty.home	`T1;T2;…;Tn;END`
2	elektromer.home	`kWh;W;HDO;END`
3	vstupy.home	`vstup(1);vstup(2);…;vstup(8);END`
4	vystupy.home	`výstup(1);výstup(2);…;výstup(8);END`
5	info.cas.home	`Den.Měsíc.Rok;hodina:minuta:vteřina;END`
6	info.system.home	`verze;Ubat;teplota_desky;imp./kWh;END`
7	data.home	*`typ;ton[h]|Ton;ton[m];toff[h]|Toff;toff[m];senzor; inverzně;výstup& nastavení 2. & …`*

1) T_1 až T_n teploty od senzorů se stejným pořadím jako adresy v souboru adresy.home.
2) `kWh` – aktuální spotřebovaná energie v kWh.
   `W` – aktuální příkon počítaný podle frekvence impulzů.
   `HDO` – Vrátí true pokud je sepnuto HDO, v opačném případě false.
3) Přítomnost log. 1 na vstupu vrací True, v opačném případě je vráceno False.
4) Kontrola nastavených výstupů, opět log.1 = True, log.0 = False.
5) Vrátí nastavení RTC hodni mikrokontroléru.
6) `Ubat` – napětí zálohovací baterie pro RTC v mV.
   `teplota_desky` – Hodnota desetinásobek °C.
7) Hodnoty pro automatickou regulaci.

7.2 Obsluha senzorů 18B20

Všechny senzory na sběrnici se chovají jako slave, proto je nutná periodické dotazování na teplotu od mikrokontoléru (masteru). Nejdříve je potřeba ve všech senzorech na sběrnici spustit převod aktuální teploty. Tento převod trvá určitou dobu, jak už bylo zmíněno v kapitole o senzorech 18B20. Poté se přejde k postupnému vyčítání teplot od senzoru. Protože je na sběrnici více jak jeden senzor, vyčítaní musí probíhat individuálně podle adres senzorů. Tyto adresy si při spuštění načte mikrokontrolér z paměťové karty a uloží si je do vlastní paměti. Vyčítání teplot probíhá v cyklu, který je volán každé 2 sekundy. Na konci každého cyklu dojde k odeslání požadavku na opětovné převedení aktuálních teplot. Teploty vždy budou maximálně dobu trvání periody staré.

7.2.1 Načtení teploty

```
public float Teplota(byte[] adresa)
        {
            ow.Reset();                 //Resetování sběrnice
            ow.WriteByte(0x55);         //Upozornění senzorů na jejich výběr
            ow.Write(adresa, 0, 8);     //odeslání adresy vybraného senzoru
            ow.WriteByte(0xBE);         //Požadavek o přečtení teploty

            teplota = ow.ReadByte();    //přečtení prvního Bajtu
            teplota += (ushort)(ow.ReadByte() << 8);
            //K prvnímu LSB bajtu se na konec přičte druhy MSB bajt.
            teplotaf = ((float)teplota / 16);   //Pro °C vydělit 16
            if (teplota > 2048)
            {
//pokud je teplota větší jak 2048 (to znamená záporná) odečte se od ní 4095
                teplotaf -= 4096;
            }
            ow.Reset();                 //reset sběrnice
            return teplotaf;
        }
```

Ze zdrojového kódu je vidět, že je nejdříve na sběrnici odeslán Byte s hodnotou 0x55, kterým jsou senzory převedeny do stavu naslouchání 8 Bytové adresy. Po jejím odeslání je poslaný další příkaz pro vyčtení 2 Bytu s teplotou. Složením těchto Bytů do 16bitové proměnné a vydělením 16, získáme teplotu ve stupních Celsia. Kvůli formátu

dvojkového doplňku pro záporné teploty, je nutné od teploty větší jak polovina 12bitového rozsahu odečíst 4096 (2^12). Teplota je obsažena jenom v prvních 11 bitech a zbylých 5 obsahuje pro zápornou hodnotu jedničku. Díky posledním 5 bitům s log1. nemůže nikdy záporné číslo být menší jak 63488 (1111 1000 0000 0000), pro případ 12bitového rozlišení převodníku.

Po obsloužení všech senzorů se odešlou dva bajty pro spuštění převodu nové aktuální teploty.

```
ow.Reset();
ow.WriteByte(0xCC); //Další příkaz je určen všem senzorům
ow.WriteByte(0x44); //Příkaz pro spuštění převodu teploty
```

7.2.2 Zjištění ID senzorů

Každý senzor má svoje identifikační číslo, svoji adresu, tu je možné z každého senzoru přes sběrnici vyčíst. V začátcích byl program řešen tak, že nejdříve bylo potřeba tyto adresy vyčíst z každého senzoru zvlášť, s tím že senzor musel být na sběrnici sám. Po zjištění všech adres použitých senzorů byly adresy zapsány přímo ve zdrojovém kódu pro mikrokontrolér. To však neslo značné nevýhody při výměně senzoru, v tom případě by muselo dojít k přeprogramování. Proto byla využita funkce senzorů, postupného odesílání adres a to i pokud se na sběrnici vyskytuje více jak jeden. Přes webové stránky se dá spustit prohledávání sběrnice, při kterém jsou načítány adresy a porovnávány s již dříve nalezenými. Pokud se aktuálně přečtená adresa neshoduje ani s jednou z adres ze záznamu, přidá se na jeho konec. Poté je na webu ještě možnost editace, kde se dají podle potřeby adresy seřadit. Celý kód hledání a porovnávání adres se záznamem, zde nebudu uvádět, jediné co stojí za zmínku, je řešení prohledávání sběrnice. To je ukázáno na následujícím kódu.

```
ow.Search_Restart();
while (ow.Search_GetNextDevice(adresa))
{
	if (adresa[0] == 40)
	{
	//Tato podmínka bude platit, pokud se jedná o senzor 18B20
	}
}
```

Po restartování sběrnice se volá funkce `Search_GetNextDevice(adresa)`, ta po vykonání uloží do 8x8Bajtové proměnné adresu senzoru který se jako první ozval. Smyčka WHILE bude platit dokut na sběrnici budou senzory, které nestihly svoji adresu odeslat. Prvním bajtem v adrese se rozlišují druhy zařízení, pro senzory 18B20 je tato hodnota rovna 40 (0x28), jde o takzvaný FAMILY CODE.

7.3 Automatická regulace

Automatická regulace je prováděna algoritmem, který 2x za vteřinu kontroluje hodnoty v 2D poli s aktuálním časem nebo teplotou vybraného senzoru. Jde o dvourozměrné Bytové pole s 8 sloupci a 6 řádky, tím jsme omezeni na maximálně 6 možných regulovaných výstupů. Rozsah hodnot je 0 až 255 proto jsme u nastavování teploty omezeni na kladné teploty, tento problém by se dal vyřešit vhodným offsetem hodnoty. Každému nastavení události odpovídá jeden řádek, ten muže mít 5 podob podle toho jaká je hodnota v prvním bajtu, jedná se o typ události. Dalšími hodnoty jsou čas, teplota nebo napětí na AD převodníku, čas je rozdělen do dvou bajtů, první jsou hodiny (24h formát) a další bajt následují minuty. Teplota je umístěna na pozici hodin a hodnota v poli pro minuty je nulová. Následující bajt je vyplněn, pokud se jedná o regulaci pomocí teploty nebo ADC a hodnota odpovídá číslu senzoru. Sedmý bajt obsahuje číslo 1, pokud má být nastavení provedeno inverzně. Posledním bajtem nastavíme číslo spínaného výstupu. Vše je shrnuto v následující tabulce.

Tabulka 10 Struktura dat pro regulaci

Bajt 1.	**Bajt 2.**	**Bajt 3.**	**Bajt 4.**	**Bajt 5.**	**Bajt 6.**	**Bajt 7.**	**Bajt 8.**
Typ	Hodiny Teplota ADCdat.	Minuty	Hodiny Teplota ADCdat.	Minuty	Senzor	Inverzně?	Výstup
0- nic 1- čas 2- teplota 3- ADC 4- mail	Stav při kterém dojde na výstupu k nastavení log.1		Stav při kterém dojde na výstupu k nastavení log.0		Číslo senzoru.	1- Nastavení bude invertované.	Číslo výstupu.

7.4 Vstup S0

Měření impulzů je řešeno ve třídě Elektroměr, uvnitř je deklarovaný interrupt port, který vyvolává přerušení. Při každém vyvolaném přerušení je od aktuálního času (počtu tiků) odečtena hodnota zaznamenána při předchozím přerušení, z této hodnot je při znalosti počtů tiku za sekundu zjištěn čas mezi impulzy. Jako mód přerušení je zvolena sestupná hrana, tím se bude detekovat začátek impulzu, protože je signál díky optočlenu a pullupu invertován.

```
public Elektromer(Cpu.Pin impPin, int impulsuNaKWh)
      {
       this.impPin = new InterruptPort(impPin, true,
       Port.ResistorMode.PullUp, Port.InterruptMode.InterruptEdgeLow);
       this.impPin.OnInterrupt += new NativeEventHandler(impPin_OnInterrupt);
       this.impulsuNaKWh = impulsuNaKWh;
       konstantaP = 3600000/impulsuNaKWh;
       konstantaW = 10/(float)impulsuNaKWh;
      }
```

Deklarace interrupt portu má několik nastavení. První parametr je vstupní PIN, dále logická hodnotu zda má být zapnut glitchFilter, třetím parametrem je nastavení Pull rezistoru s možností (vypnut, PullUp a PullDown) a poslední určuje jakou změnou signálu je vyvoláno přerušení. GlitchFilter řeší problém s několikanásobným přerušením při zákmitu signálu.

```
      private void impPin_OnInterrupt(uint data1, uint data2, DateTime time)
            {
prikonW = (int)(konstantaP * (TimeSpan.TicksPerSecond / (int)(time.Ticks - ticks)));
//Násobitel konstantyP je čas mezi impulzy v sekundách.
            ticks = time.Ticks;        //Počet tiků pro následující přerušení

            if (i == 10)
            {
            energieKWh += konstantaW;  //Po 10 imp. přičte spotřebovanou energii
            i = 0;
            }
            i++;
            }
```

8 WEB

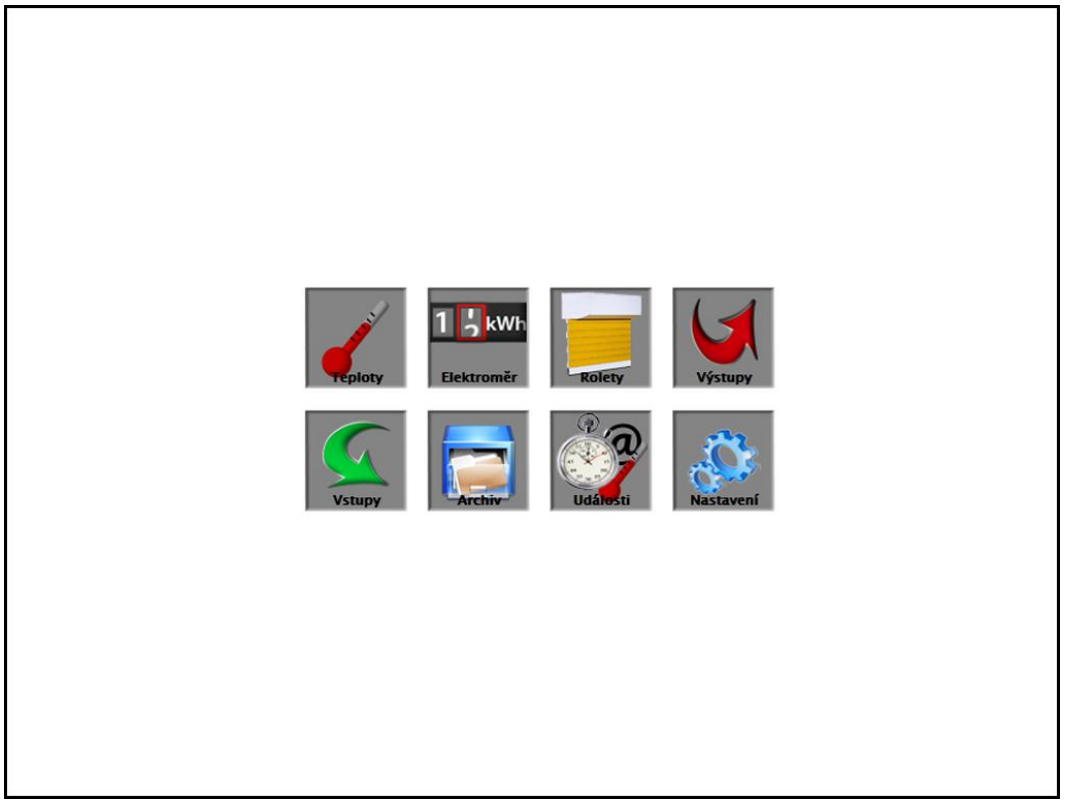
Vzhled a struktura webových stránek, procházela postupným vývojem spolu s tím jak jsem objevoval nové technologie pro jejich návrh. Prvním krokem k vytvoření stránek bylo nalezení způsobu aktualizování teplot, bylo potřeba nalézt způsob, kdy hlavní stránky bude periodicky načítat textová data z jiné adresy, aniž by ona sama musela být znovu celá aktualizována. Dále se musela přijatá data nějakým způsobem zpracovávat, k tomu bylo nevyhnutelné použít skriptovací jazyk javascript. Bohužel pro získávání dat z jiné webové stránky nemá přímo tento programovací jazyk funkce, proto byla zvolena technologije s názvem AJAX, ta je dnes velmi důležitým nástrojem pro tvorbu interaktivních webových aplikací. Tato technologie například obstarává načítání map v oblíbené službě Google Maps čí načítání dat v Gmailu čí spousty jiných webových aplikacích.

Další důležitá věc stránek je vzhled, v prvních návrzích byli vyžívány pouze možnosti jazyku HTML, ty ovšem pro vytvoření webu s grafickým prostředím nestačí. Proto jsem se musel seznámit s využitím takzvaných kaskádních stylů CSS. Ty se ukázaly důležité pro rozmístění grafických, či textových prvků, dále díky nim bylo možné využít různé efekty či styly písma.

Teoretická datová rychlost ethernetového modulu je sice 62KB/s ale reálná se pohybuje kolem 35KB/s. S touto rychlostí by načtení například pozadí trvalo několik desítek sekund, proto je většina obrázků umístěna na jiném serveru. Dojde tím k razantnímu urychlení načítání stránek a hlavně k odlehčení ethernetového modulu, tím předejdeme i jeho zahlcení.

8.1 Hlavní menu

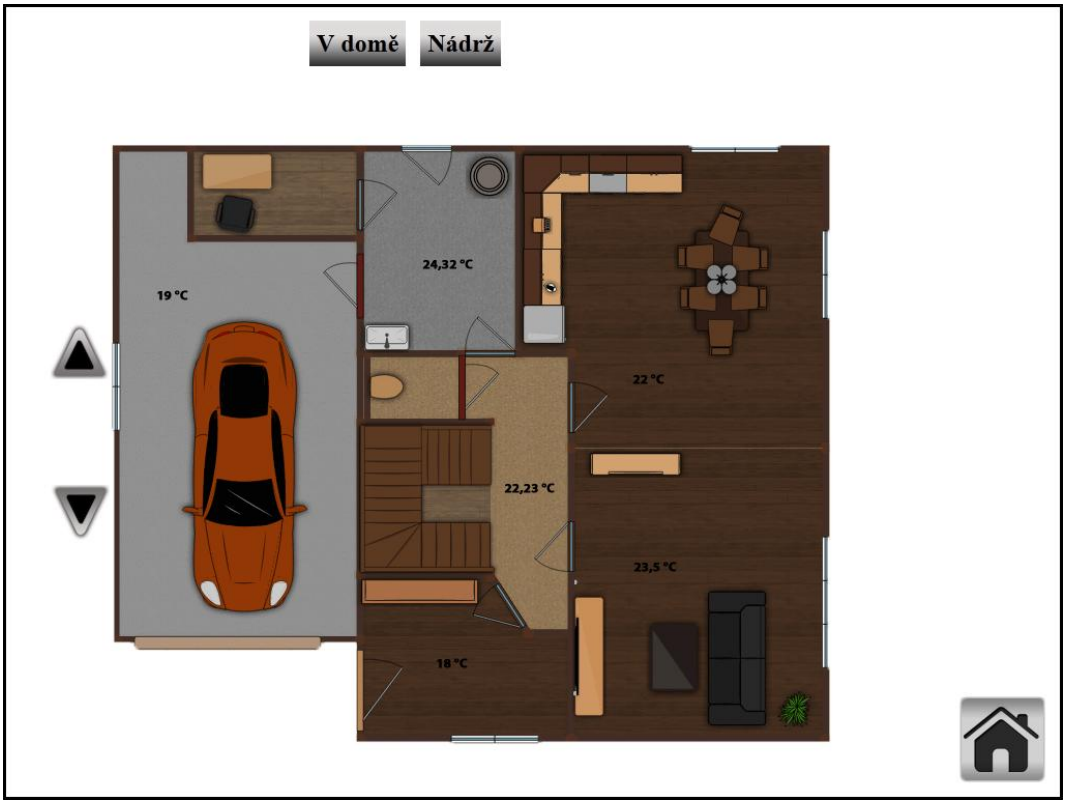
Tato první úvodní stránka nezobrazuje žádné důležité informace, jde pouze o rozcestník k dalším webům.



Obrázek 9 index

8.2 Teploty

Tato stránka nám poskytuje informace o teplotách, ty jsou zobrazeny na modelu domu. Šipkami na levé straně se mění podlaží a vrchními tlačítky se dostaneme na teploty vody v nádrži. Web v sobě obsahuje časovač, který každé 2 sekundy volá funkci, ta pomocí již zmíněného AJAXU načte aktuální teploty se stránky „teploty.home“ a doplní je do pro ně určených polí.



Obrázek 10 Stránka teplot

8.2.1 AJAX

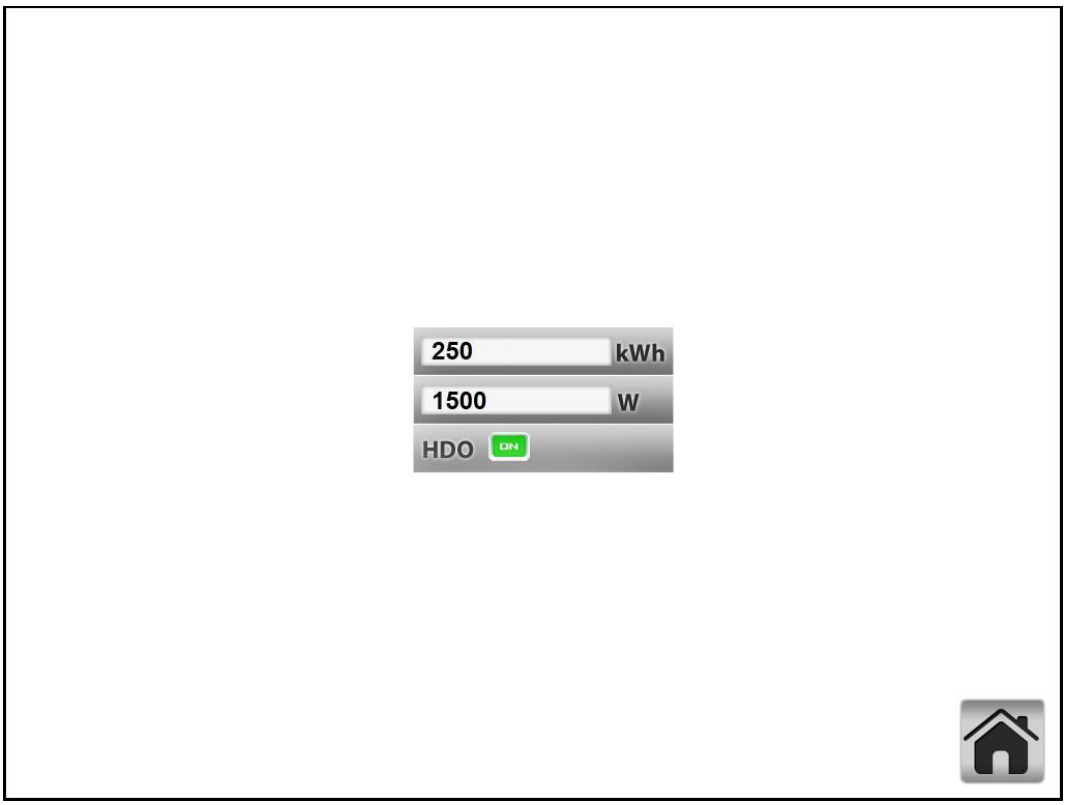
Zde je ukázka Ajax funkcí pro načítání dat z jiné webové stránky. Tato funkce je využita ve všech stránkách, kde je nutné získávat aktuální informace. Důležité je říct, že kvůli bezpečnosti není v některých prohlížečích možné pomocí Ajaxu načíst obsah webové stránky, která je na jiné doméně, než stránka, ze které je funkce volaná.

```
function Nacist(adresa) {
            if (window.XMLHttpRequest) { // pro IE7+, Firefox, Chrome, Opera, Safari
                xmlhttp = new XMLHttpRequest();
            }
            else {                       // pro IE6, IE5
                xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
            }
            xmlhttp.open("POST", adresa, true);
            xmlhttp.onreadystatechange = function () {
                if (xmlhttp.readyState == 4) {
                    source = xmlhttp.responseText;
                }
            }
            xmlhttp.send(null);
            return source;
        }
```

Po zavolání `xmlhttp.send(null)` dojde k odeslání požadavku na zadanou webovou adresu, po přijetí odpovědi se vykoná blok **onreadystatechange** a pokud vyhoví podmínka, zapíše se do proměnné source obsah webové stránky. Adresu je nutné funkci **open()** předávat ve tvaru "http://www.adresa.xx".

8.3 Elektroměr

Na této stránce dostáváme informace o spotřebované energii v kWh a o aktuálním příkonu. Spodní ukazatel značí, zda je aktivní HDO (hromadné dálkové ovládání).



Obrázek 11 Stránka elektroměru

8.4 Výstupy/Vstupy

Na stránce výstupy je možné ovládat jeden z 8 výstupů mikrokontroléru. Po načtení stránky se jako první vykoná funkce pro zjištění aktuálních hodnot. Pro korektní nastavení výstupů je nutné zadané správné heslo. Pokud je špatné, server sice správu příjme, ale k žádné změně na výstupech nedojde.

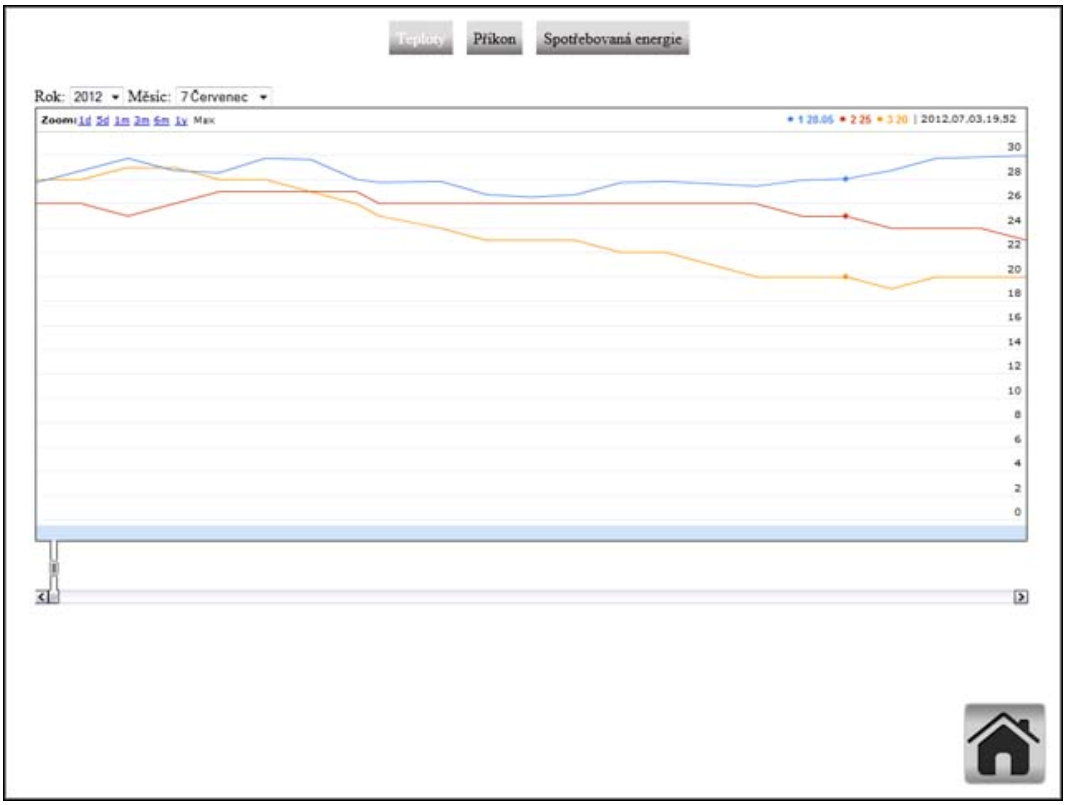
Stránka vstupů pracuje podobně jako teplot, každé 2 sekundy je zjišťována aktuální situace na vstupech.

Obrázek 12 Stránka výstupů

8.5 Archiv

Na této stránce si je možné nechat zobrazit archiv teplot a spotřeby. Mikrokontrolér každých 15 minut přidá do textového souboru aktuální data. Soubor v sobě nese hodnoty za 1 měsíc. Název je vytvářen podle aktuálního roku a měsíce ve tvaru "archiv-rok-mesic(dek.).txt". Na stránce se vybere rok a měsíc, po tomto výběru je na server odeslán dotaz o načtení souboru s tímto názvem.

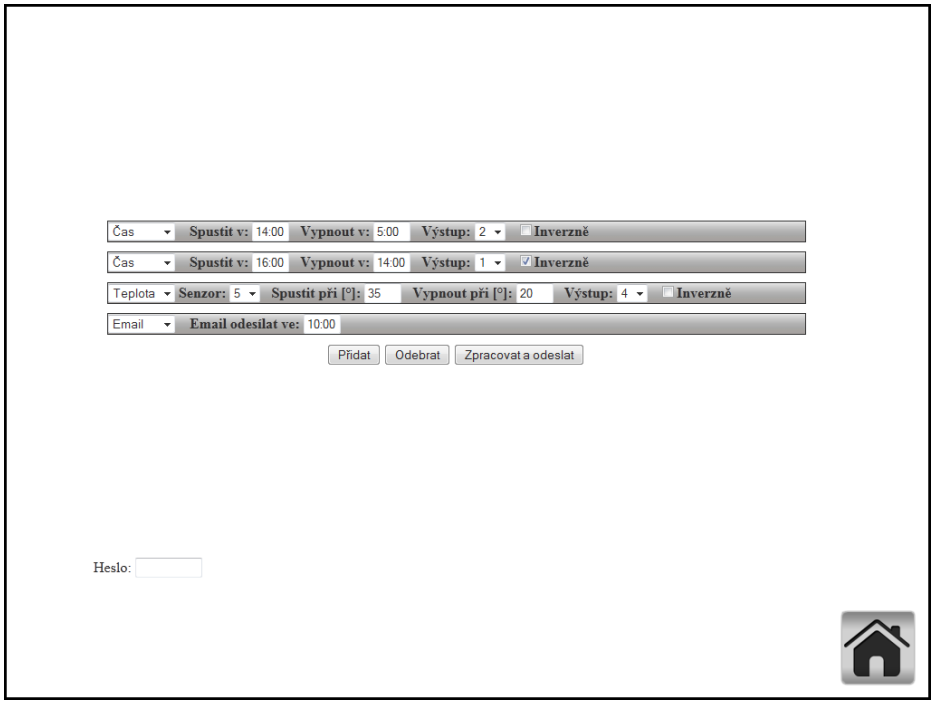
Zdrojový kód grafu je převzat z volné databáze API knihoven Google code. Na stránkách Code Playground je velký výběr různých předpřipravených grafů čí jiných grafických celků. Použitý graf má název Annotated Time Line.



Obrázek 13 Stránka archivu

8.6 Regulace

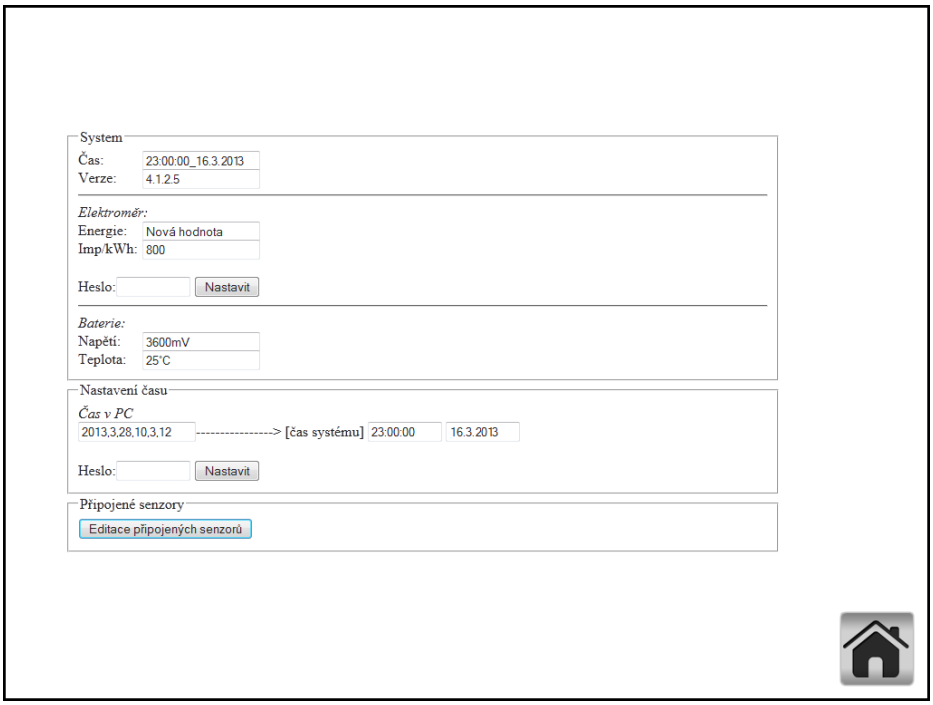
Tato stránka slouží k nastavení jednoduché regulace. Tlačítkem přidat vložíme nový regulovaný výstup, ten může být regulován pomocí teploty, časového úseku či napětím na A/D převodníku. V poslední možností se dá nastavit čas odeslání informačního emailu.



Obrázek 14 Stránka regulace

8.7 Nastavení

Na této stránce se dá nastavit nebo sledovat několik hodnot. K nastavením patří čas, počet impulzů na kilowatthodinu, spotřebovaná energie nebo pořadí teplotních senzorů. Je zde možnost restartovat mikrokontrolér nebo spustit prohledávání sběrnice pro přidání nového čidla. Zjistit je možné čas systému, verzi a napětí na záložním kondenzátoru.



Obrázek 15 Stránka nastavení

ZÁVĚR

Dokončená práce odpovídá prvotním představám. Ovšem začátky nebyly jednoduché, bylo třeba se vypořádat s řešením způsobu, jakým bude server obstarávat aktuální teploty a celkově data, pro webové stránky. To vše se podařilo a myslím, že vše funguje přesně podle prvotních představ. Samozřejmě že teď, při dokončování práce už by se opět našlo několik věcí na vylepšení, hlavně asi vzhled webových stránek, kde se vždy s novou dokončenou verzí začala objevovat vize nového vzhledu. Další by se nabízelo rozšíření funkcí systému, přidání PWM výstupů, nebo využití D/A převodníku, přidání bezdrátového modulu pro komunikaci s dalšími periferiemi. Možnost komunikace po průmyslové sběrnici by byla určitě také k užitku, pokud by po budově byla rozvedena sběrnice ze strukturované kabeláže, došlo by k efektivní expanzi vstupů a výstupů. Byla by možnost buď implementovat některé standardní protokoly například KNX/EIB, nebo vytvořit vlastní a využít k tomu velmi odolnou průmyslovou sběrnici RS458. Implementováním této sběrnice by celé zařízení dostalo nový rozměr díky ovládání jakéhokoli světla v celém domě, ovšem samozřejmě by to vyžadovalo vytvoření ovládacích modulů obsažených u každého světla či zásuvky. Pro úplnost by musely být k této sběrnici připojeny i veškeré vypínače. Každý koncový modul by obsahoval svoji jedinečnou identifikační adresu, podobně jak je typické pro zmíněné senzory 18B20. Dalším vylepšením systému bude nejspíše schopnost ovládání okenních rolet. Možná si v hlavním menu najde místo i stránka s informacemi o solárních panelech, například o napětí či dodávaném proudu. Pro sledování proudu se jistě hodí integrované obvody využívající Hallův článek. Mikrokontrolér by sledoval výstupní napětí tohoto článku úměrné procházejícímu proudu. Další úpravy si zaslouží i samotný zdrojový kód, který své chyby ukáže až delším testováním. Většiny těchto úprav a vylepšení se systém nejspíše dočká díky tomu, že bude použit v bratrově novostavbě, proto snad tato Bakalářská práce není koncem vývoje tohoto systému, ale jeho začátkem.

POUŽITÁ LITERATURA

[1] VALEŠ, Miroslav. *Inteligentní dům*. 1. vyd. Brno: ERA, 2006, 123 s., il. (část barev.). ISBN 80-736-6062-8.

[2] GHI ELECTRONICS. *GHI Electronics* [online]. GHI Electronics LLC, © 2003-2013 [cit. 2013-04-06]. Dostupné z: http://www.ghielectronics.com

[3] ISSA, Gus. Beginners Guide to .NET Micro Framework. In: *Beginners Guide to C# and the .NET Micro Framework* [online]. 2.1, 2012, September 24th 2012 [cit. 2013-04-06]. Dostupné z: http://www.ghielectronics.com/downloads/FEZ/Beginners%20guide%20to%20NETMF.pdf

[4] Codeshare. GHI Electronics [online]. GHI Electronics LLC, © 2003-2013 [cit. 2013-04-06]. Dostupné z: http://www.ghielectronics.com/community/codeshare

[5] MALÝ, Martin. Sběrnice 1-Wire™. *HW.cz: Vše o elektronice a programování* [online]. 2004 [cit. 2013-04-06]. Dostupné z: http://www.hw.cz/navrh-obvodu/rozhrani/sbernice-1-wiretm.html

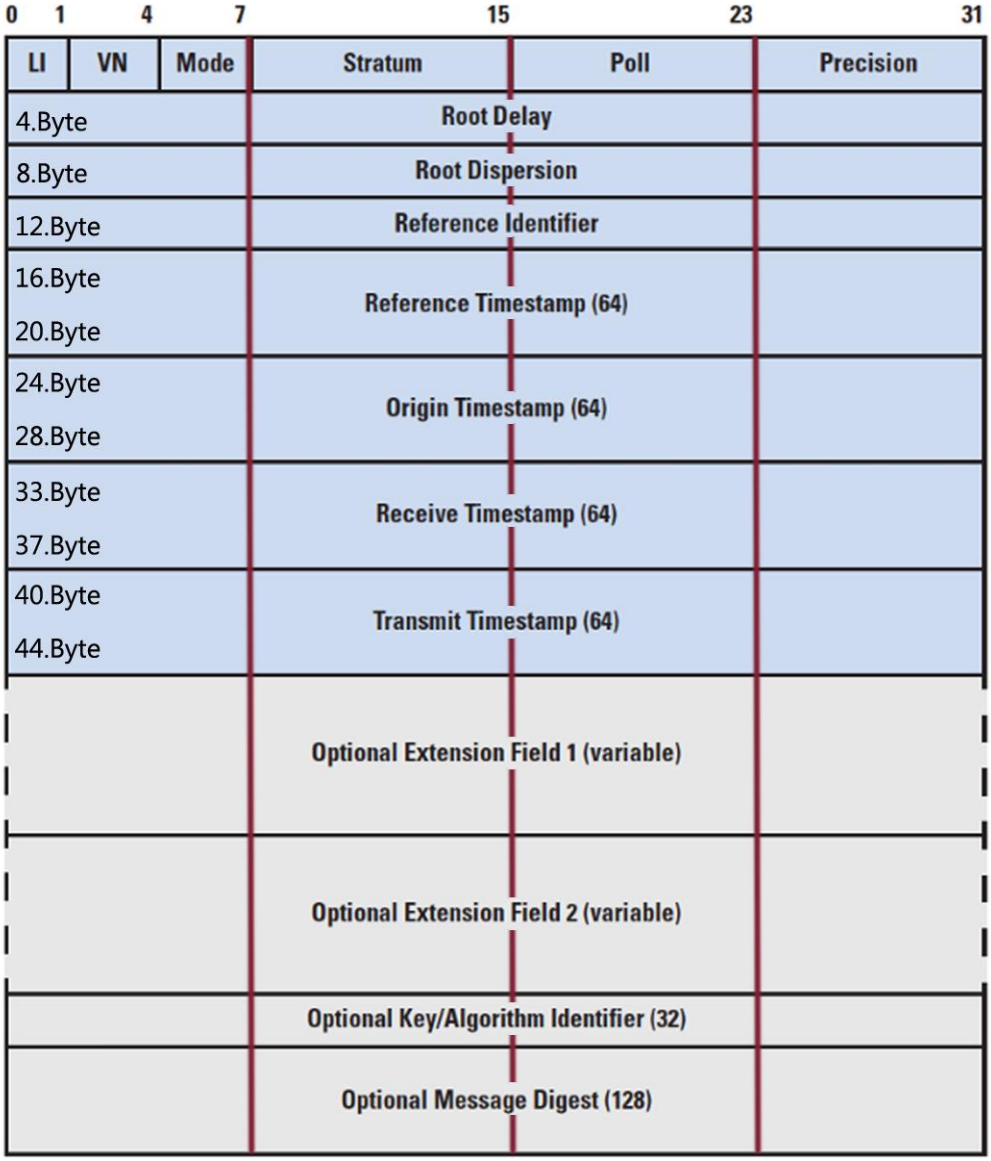
[6] B. POSTEL, Jonathan. RFC 821. *SIMPLE MAIL TRANSFER PROTOCOL*. University of Southern California: Information Sciences Institute, August 1982, 68 s. Dostupné z: http://tools.ietf.org/pdf/rfc821.pdf

[7] Network Time Protocol. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001-2013, 8. 3. 2013 [cit. 2013-04-06]. Dostupné z: http://cs.wikipedia.org/wiki/Network_Time_Protocol

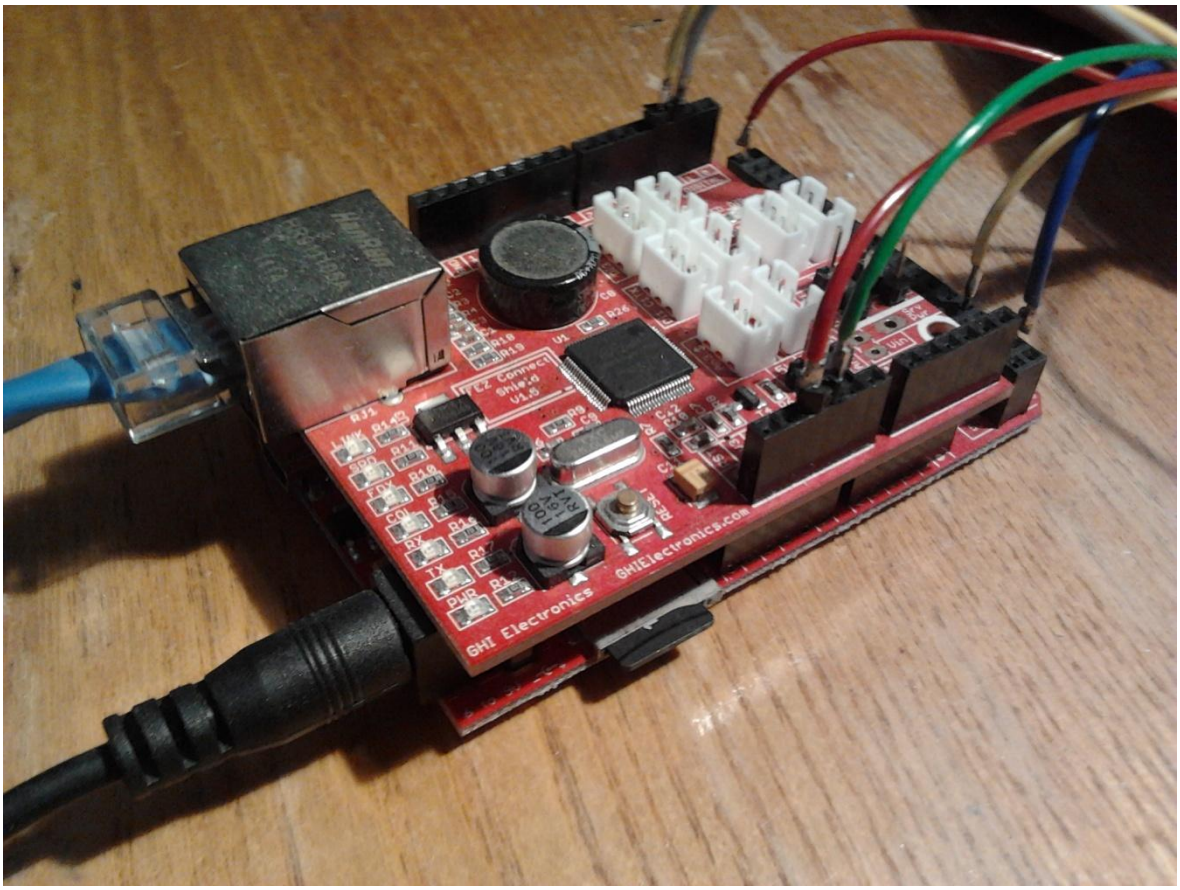
[8] Inteligence. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): WikimediaFoundation, 2001- [cit. 2013-05-19]. Dostupné z: http://cs.wikipedia.org/wiki/Inteligence

PŘÍLOHY

Příloha A - *Struktura NTP protokolu*



Příloha B – *Foto Pandy II s ethernetovým modulem*



Příloha C - *CD*

Obsah CD:

SmejkalR_InteligentniDum_PR_2013.pdf – vlastní text práce

adresář **MCU**: vlastní program pro MCU (panda II) (projekt Visual Studio 2010)

adresář **WEB**: webové stránky a ostatní soubory z SD karty

adresář **PANDAII**:

- FEZ_Panda_II_UserManual.pdf - uživatelský manuál
- FEZ_Internet_of_Things_Book.pdf - přehled komponent
- Beginners guide to NETMF.pdf - průvodce základy programování
- Beginners guide to NETMF2.pdf - průvodce základy programování II
- datasheet_DS18B20.pdf - datasheet k senzorům DS18B20