FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Návrh a implementace chytrého domu
Student:	Jakub Kyzek
Vedoucí:	Ing. Martin Daňhel, Ph.D.
Studijní program:	Informatika
Studijní obor:	Počítačové inženýrství
Katedra:	Katedra číslicového návrhu
Platnost zadání:	Do konce letního semestru 2019/20

Pokyny pro vypracování

Analyzujte možnosti návrhu chytrého domu založeného na drátové technologii.
Pro chytrý dům/domácnost zvolte vhodnou platformu např. AVR, Arduino, PIC apod.
Pro zvolenou platformu navrhněte systém, který bude implementovat následující funkčnosti:
Sledování teploty a vlhkosti v jednotlivých místnostech.
Systém bude dále umožňovat ovládání jednotek klimatizací.

Ovládání bude dostupné přes webové rozhraní s možností vykreslit grafy za posledních 24 hodin.
Navržené řešení implementujte v reálných podmínkách (na skutečném domě).
Vytvořený systém otestujte.

Seznam odborné literatury

Dodá vedoucí práce.

doc. Ing. Hana Kubátová, CSc.
vedoucí katedry

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
děkan

V Praze dne 5. února 2019

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Bakalářská práce

Návrh a implementace chytrého domu

Jakub Kyzek

Katedra číslicového návrhu

Vedoucí práce: Ing. Martin Daňhel, Ph.D.

13. května 2019

Poděkování

Rád bych poděkoval Ing. Martinu Daňhelovi, Ph.D., který vedl tuto bakalářskou práci, za jeho věcné připomínky a ochotu, se kterou mě vedl ke kýženému cíli. Dále děkuji majiteli domu, že mi umožnil vytvořit takto rozsáhle dílo na jeho domě. A na závěr bych chtěl poděkovat celé mé rodině za podporu během studia.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou a veškeré jejich dokumentace (dále souhrnně jen „Dílo"), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, avšak pouze k nevýdělečným účelům. Toto oprávnění je časově, teritoriálně i množstevně neomezené.

V Praze dne 13. května 2019 ……

České vysoké učení technické v Praze

Fakulta informačních technologií

Odkaz na tuto práci

Kyzek, Jakub. *Návrh a implementace chytrého domu.* Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2019.

Abstrakt

Práce se zabývá analýzou možností, návrhem a následnou implementací chytrého domu na vybrané platformě AVR. Systém je navržený tak, aby umožňoval přístup k informacím a ovládání domu pomocí internetu. Síť senzorů je postavena na sběrnici RS-485. Webové rozhraní je implementováno ve Flasku – to je propojeno s chytrým domem pomocí databáze PostgreSQL. Práce se dále věnuje detailnímu popisu tvorby bootloaderu pro procesor AVR a nastavení serveru pro dům na linuxové distribuci Debian.

Klíčová slova chytrý dům, bootloader AVR, Arduino, python, Flask, vestavný systém, Debian

Abstract

This thesis deals with the analysis of possibilities, design and implementation of SmartHome on selected AVR platform. The system is designed to allow access to information and control of the house via the Internet. The sensor network is built with the use of the RS-485 bus. The web interface is implemented using Flask, which is connected to the SmartHome using PostgreSQL. Moreover, this thesis also detailly describes the creation of the bootloader for the AVR processor and the server setting for the house on the Debian Linux distribution.

Keywords SmartHome, bootloader AVR, Arduino, python, Flask, embedded system, Debian

Obsah

Seznam obrázků

Seznam tabulek

Seznam zdrojových kódů

Úvod

V dnešních domácnostech je spousta nemoderních elektronických zařízení, která musí být obsluhována ručně. S rychlým rozvojem technologií se otevírají cesty k tomu, aby si lidé usnadnili život a aby byly tyto rutinní činnosti automatizovány. V zimě regulovat výkon topení, v létě výkon klimatizace. Pokud je v bytě příliš vlhko – větrat, pokud je venku zima – nevětrat. Tyto jednoduché činnosti zaberou lidem spoustu času a starostí – například zapnuté topení, když nikdo není doma. To je důvod, proč se tato práce zabývá návrhem a implementací jednoduchého chytrého domu. Na trhu lze najít velké množství výrobců, kteří se zabývají jak výrobou hotových systémů, tak produkcí nepřeberného množství čidel a komponent. Proto se práce zaměřuje i na to, aby byl výsledný dům cenově lépe dostupný a tedy zajímavější pro širší okruh lidí.

Moje osobní motivace spočívala hlavně v tom, že by výsledky této práce mohly pomoci při boji s vlhkostí v našem domě. S vlhkostí bojujeme již delší dobu, důvodem je vysoké stáří budovy. Proto jsme pořídili rekuperaci vzduchu[1], ale příliš nám nevyhovovalo její ovládání. Teplotu a vlhkost v jednotlivých částech domu jsme měřili jednoduchými moduly s LCD dodávanými z Číny, u kterých stále docházely baterie. Pokud bylo třeba zjistit aktuální hodnoty senzorů v domě, musel někdo obejít postupně jednotlivá čidla v místnostech. Po rozmluvě s majitelem domu jsem postavil první jednoduchý systém, který sloužil ke sledování hodnot v domě. V této bakalářské práci se zabývám vývojem daleko pokročilejšího systému a snažím se vyvarovat chybám, na které jsem narazil během mého prvního pokusu.

Práce je rozdělena do sedmi kapitol včetně úvodu a závěru. Součástí práce jsou čtyři přílohy, za zmínku stojí příloha B – která je uživatelskou příručkou

[1]Zařízení fungující obdobně jako „vzduchotechnika“ ale navíc využívá tepla vzduchu opouštějícího dům k ohřívání vzduchu vstupujícího do domu. Podrobněji na straně 5.

k vytvořenému systému chytrého domu a příloha C, ve které lze nalézt krátkou fotodokumentaci částí implementovaného systému. V první kapitole jsou detailně rozepsány cíle práce. Druhá kapitola – „Analýza“ – seznámí čtenáře s teoretickým základem potřebným k porozumění danému tématu a rozebere některá existující komerční řešení. Diskutuje různé možnosti chytrých domů, dále se věnuje popisu potřeb domu, na kterém bude provedena implementace. Také jsou v ní diskutovány možnosti různých platforem. Ve třetí kapitole jsou z možností navržených v analytické části vybrány ty nejvhodnější a ty jsou dále detailněji rozebrány. V této kapitole je také popsán návrh softwaru pro jednotlivé jednotky. Programové vybavení jednotek se skládá ze dvou částí – z bootloaderu a firmwaru. Kapitola čtyři rozebírá implementaci práce. Pátá se věnuje testování systému a ukázkám zajímavých dat vzniklých během testovacího provozu. Následuje kapitola „Závěr“.

Kapitola 1

Cíle práce

Práce se zabývá návrhem a implementací chytrého domu za použití drátové technologie. Implementovaný systém bude poskytovat následující funkcionality:

- Sledování a ukládání dat získaných z jednotlivých místností a jiných měřených míst (primárně teplotu a vlhkost).
- Zprostředkování přístupu k důležitým údajům pomocí webového rozhraní, které umožní:
  - Zobrazit aktuální hodnoty čidel v jednotlivých místnostech.
  - Vykreslit grafy z čidel v místnostech za posledních 24 hodin.
- Ovládání klimatizace v různých místnostech, a to bez větších úprav původního hardwaru klimatizace.

Aby bylo možné implementovat vyjmenované funkcionality, je nejprve potřeba analyzovat dostupné možnosti. V kapitole věnované analýze budou teoreticky shrnuté možnosti hardwaru – procesory, desky, sběrnice – a softwaru pro použití během implementace chytrého domu. Tím bude připravena živná půda pro návrhovou část a výběr HW pro konečný systém. Z možností navržených v analytické části budou v kapitole „Návrh“ vybrány nejvhodnější dostupné technologie pro jednotlivé konkrétní části navržené v analýze (konkrétní platforma, čidla, atd.), na kterých bude dále celý systém postaven.

V implementační části budou popsány jednotlivé kroky implementace chytrého domu a jejich aplikace na skutečný dům pomocí technologií zvolených v teoretické části. Konečný vytvořený systém bude nakonec otestován.

Práce se nevěnuje zabezpečení softwaru ani hardwaru proti manipulaci uvnitř domu (pokud útočník projde dveřmi), to je velká výhoda oproti bezdrátovým systémům. Bezpečnost ošetřuje pouze na rozhraních dostupných vzdáleně po

internetu – šifrování spojení pomocí SSL a přihlašování. I když by šifrování na vybraném procesoru bylo technicky možné, je to mimo rozsah této práce.

Dále se práce nevěnuje celkové automatizaci domu. Shromažďuje ovládání domu na jedno místo a některé nejvíce důležité činnosti automatizuje – například ovládání klapky v případě mrazu, kdy by hrozilo poškození rekuperační jednotky. Dále se na rozdíl od velké většiny komerčních chytrých domů nevěnuje automatickému zatahování žaluzií. Autor zásadně vyznává obyčejné závěsy.

Kapitola 2

Teoretická část a analýza problému

Následující kapitola se věnuje rozboru možností pro návrh chytrého domu. Jsou v ní popsány i zřejmě špatné možnosti a slepé uličky. Nejprve je nutné blíže se podívat na dům, na kterém bude na závěr provedena implementace. Od toho se bude dále odvíjet analýza. Dům pochází ze začátku dvacátého století. I přes neustálé větrání jsou v něm problémy s vlhkostí. Bylo by tedy vhodné monitorovat vlhkost.

Aby se zabránilo problémům spojeným s vlhkostí, byla do domu nainstalována rekuperace. Ta vyměňuje vlhký vzduch z bytu za sušší vzduch zvenku. Lidově řečeno „větrá“. Výhodou proti běžnému větrání je, že rekuperace obsahuje tepelný výměník, který se stará o předání tepla vzduchu přicházejícího zvenku pomocí tepla vzduchu odcházejícího z domu a tím šetří náklady na vytápění. Také obsahuje vzduchový filtr, takže odstraní hrubší nečistoty, které běžně oknem projdou. Tepelný výměník je bohužel náchylný na teploty pod nulou. Proto byla na přívodu vzduchu (zvenku) do rekuperace nainstalována klapka, která přepne sání zvenku na sání z chodby (kde není mráz). Sice dům nevětrá, ale alespoň nedojde k poškození tepelného výměníku. Bylo by vhodné kontrolovat teplotu vzduchu na sání a v případě nízkých teplot automaticky zavírat klapku.

Dalšími součástmi domu jsou dvě klimatizace značky Panasonic[2]. Ty bude třeba nějakým způsobem ovládat. K topení (kromě klimatizací[3]) dále slouží elektrická akumulační kamna. Ty bývají ručně ovládána obyvateli domu pokud teplota venku klesne pod určitou teplotu, kdy už nelze klimatizacemi topit na dostatečně efektivní úrovni.

[2]Konkrétně modely KIT-UE18RKE (5 kW) a KIT-UE12RKE (3,5 kW).
[3]Ano i s nimi se dá celkem efektivně topit.

2.1 Možnosti chytrých domů

Chytré domy nabízejí velké množství možností. Lze například:

- Zatahovat a roztahovat žaluzie v závislosti na venkovních světelných podmínkách.
- Regulovat teplotu – elektrická topení, radiátory s kotlem, klimatizace.
  - „Hloupé“ řízení teploty topení (konstantní teplota).
  - Regulace teploty pro každou místnost – předvídat a topit s předstihem.
- Obstarávat funkci zvonku.
- Zobrazovat informace o domě.
  - Počítat cenu spotřebované energie elektrických spotřebičů.
  - Zobrazovat a ovládat teploty v domě.

A různě provazovat tyto funkce. Například topit elektrickými přímotopy (které mají nízkou tepelnou kapacitu), než se nahřeje kotel, který zásobuje teplou vodou radiátory (které mají vyšší kapacitu). Tato funkcionalita by byla velmi přínosná pro správce bytových domů.

2.2 Současný stav trhu

Na internetu je dostupné vysoké množství nabídek. Ovšem nenašel jsem žádnou společnost zabývající se ultra levnými chytrými domy. Většina z firem nabízí služby, které v této práci implementovány nebudou. Například dálkové ovládání sauny nebo automatické vyhřívání bazénu. Také ve většině nabídek požadovali změnu elektroinstalace, aby bylo možné ovládat zařízení přímo z rozvaděče. To během implementace této práce také nepřipadá v úvahu.

Požadavky pro výběr společností: ovládání topení, ovládání klimatizace, měření teploty v místnostech, ovládání vzduchotechniky (rekuperace). Ovládání pomocí telefonu nebo notebooku (ideálně obojí – webové rozhraní).

Z dohledaných firem byla vybrána jedna ukázková. Společnost TotalISB uvádí na svých stránkách možnost vyzkoušet si jejich chytrý dům[4]. S jejich webovým rozhraním měl problém i středně výkonný počítač. Bohužel to vypadá, že ani tato firma neposkytuje podobné služby, kterými se zabývá tato práce, odhad minimální ceny[5] produktů je 100.000 Kč. Proto se již této společnosti není třeba věnovat více do hloubky.

[4] Dostupné dne 2019-04-20 z `http://www.inteligentni-dum.eu/`.
[5] Ceny jsou uvedeny včetně DPH.

2.3 Komunikační sběrnice a protokoly

Dá se předpokládat, že systém bude složen z jednotlivých jednotek – senzory, ovládací prvky, atd. Ty musejí být mezi sebou nějakým způsobem propojeny, aby mohly spolupracovat a poskytovat požadované informace nebo případně reagovat na požadavky od člověka. Propojením bude nějaká forma sítě. Dále jedna jednotka bude hlavní (dále označovaná jako server), zodpovědná za komunikaci s člověkem.

Nyní nastává otázka, kolik by měly mít jednotlivé jednotky zodpovědnosti. Pokud je jednotka inteligentnější – není to pouze holý senzor, ale je u něj pro obsluhu například mikroprocesor – lze rozhodovat o tom, jak moc funkcionality bude na straně jednotky a jak moc na straně serveru. Například: lze udělat naprosto „hloupé“ jednotky, které rozumí pouze příkazům od serveru typu *řekni teplotu*, *nastav pin* a chytrý server, který bude pomocí jednoduchých příkazů skládat složitější. Nebo složitější jednotky a jednodušší server, kde příkazy posílané serverem mohou vypadat jako například: *udržuj teplotu v místnosti*.

V následujících pár odstavcích je popis různých, převážně sériových, komunikačních sběrnic a protokolů používaných u vestavěných systémů.

2.3.1 Asynchronní sériová komunikace (USART)

„Univerzální sériový asynchronní vysílač / přijímač“ je zařízení umožňující full-duplex komunikaci mezi dvěma zařízeními. Je hojně využívané pro ladění zařízení. Například uvnitř osciloskopu Keysight 1000X se nacházelo jedno takovéto ladící rozhraní. To posloužilo během úpravy osciloskopu pro zpřístupnění placených funkcí zdarma [1].

Obrázek 2.1: Serial – přenos 8 bitů, bez parity

Před vysíláním se musí obě strany shodnout na pár základních údajích:

- **Baudrate** určuje přenosovou rychlost (počet bitů za sekundu).
- **Počet bitů vysílání** mezi pěti a devíti[6].
- **Parita** – žádná / sudá / lichá.
- **Počet stop bitů** – jeden nebo dva.

[6]Nejběžněji 8 bitů, 9 bitů pouze u MPCM.

Na rozdíl od jiných (např. I^2C) nepotřebuje USART další vodič pro přenos hodin. Synchronizace probíhá za pomoci start bitu. Na obrázku 2.1 je zobrazena ukázka přenosu jednoho rámce.

Pro full-duplex komunikaci mezi dvěma zařízeními jsou potřeba právě dva vodiče – RX (přijímací) a TX (vysílací). U obou dvou zařízení jsou označeny stejně a je nutné propojit je „křížem". Vodič RX u prvního zařízení musí být připojen na vodič TX u druhého zařízení a naopak.

2.3.2 Sběrnice I^2C

Často využívaná jednoduchá multimaster sériová sběrnice navržená firmou Philips Semiconductor. Mezi její základní vlastnosti patří detekce kolizí, arbitrace a rychlosti až do 100 kbit/s v normálním režimu. Existují specifikace pro rychlejší komunikaci – například fast-mode, který dosahuje až 400 kbit/s. Původní návrh sběrnice počítal s použitím při komunikaci na desce – velmi krátké vzdálenosti (do metru). Komunikace probíhá po dvou vodičích SDA (data) a SCL (hodiny), na principu „open drain". Je ale třeba mít další společný vodič pro zem. Každé zařízení na sběrnici má svoji unikátní adresu (nebo by alespoň mít mělo). Součástí standardu je i komunikační protokol. Ten diktuje formát datového přenosu mezi zařízeními. Veškeré informace o sběrnici jsou zdarma a volně přístupné ve standardu [2].

Další výhodou sběrnice je její široká podpora v hardwaru procesorů a čidel. Napětí se kterým pracuje je běžně 3,3 V nebo 5 V. Nevýhodou je krátký dosah sběrnice – bez úpravy sběrnice (přidávání bufferů).

2.3.3 Sběrnice OneWire

Sériová sběrnice navržená firmou Dallas Semiconductor, která se používá pro komunikaci zařízení po jednom vodiči (+ zem). Používají je například teplotní čidla DS18B20. Sběrnice dosahuje pouze nízkých rychlostí (16,3 kbp/s) a má vždy právě jednoho mastera. Poslední zmíněná vlastnost prakticky vylučuje kolize. Většinou pracuje s napětími 3,3 V nebo 5 V – vhodné pro přímé propojení s procesorem. Vyžaduje jeden externí odpor s hodnotou 4,7 kΩ. Standard specifikuje formát paketu. Není třeba mít HW podporu, lze snadno emulovat metodou „bit banging".

Sběrnice je vhodná pro použití na kratší vzdálenosti, nebo v případě nedostatku pinů u procesoru nebo čidla. Nevýhodou je krátký dosah sběrnice a náchylnost na rušení.

2.3.4 CAN

Sériová sběrnice používaná hojně v automobilovém průmyslu. Například automobil značky Škoda Octavia (první generace) používá minimálně 2 sběrnice CAN [3]. Jedna pro přenos informací o motoru (např. sdílení rychlosti otáček s centrální jednotkou) a druhá pro komfortní funkce (např. signál pro ote-

vření okének). Data na sběrnici mají formát daný standardem. Oproti I^2C nebo OneWire se hodí k přenosům na středně dlouhé vzdálenosti.

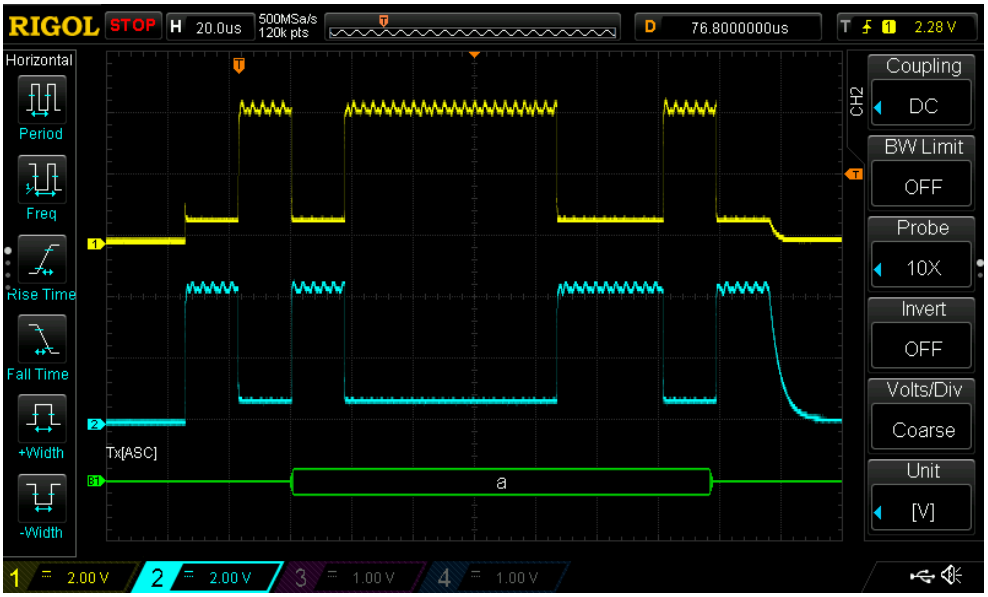
Výhodami sběrnice je především odolnost proti rušivým vlivům. Některé procesory mají dokonce HW podporu (ovšem bez budičů).

2.3.5 RS232

Tento standard umožňuje komunikaci na střední vzdálenosti (většinou pod 20 metrů). Řeší primárně standardizaci napětí na lince. Pro komunikaci se používá protokol popsaný v 2.3.1.

2.3.6 Sběrnice RS-485

RS-485 označuje standard pro komunikace na mnohem delší vzdálenosti. Limit vzdálenosti je 1200 metrů s možností komunikace až 32 zařízení na sběrnici. Rychlost sběrnice může dosahovat až 10 mbit/s, ovšem s přibývajícími zařízeními se maximální rychlost snižuje. Společnost Texas Instruments prodává galvanicky oddělené čipy (ISO-485), které umožňují připojení většího množství zařízení a zvyšují maximální možnou rychlost. Tyto čipy jsou oproti původním MAX-485 až desetkrát dražší.

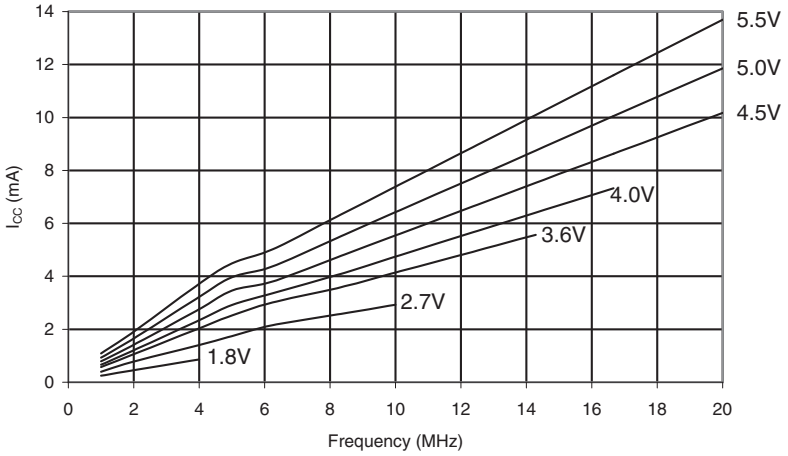


Obrázek 2.2: RS-485 komunikace (záznam z osciloskopu)

Jsou 2 možnosti konfigurace. Buď sběrnice pracuje na dvou vodičích (A, B) v režimu half-duplex, nebo na čtyřech vodičích v režimu full-duplex. Označení vodičů A a B není bohužel součástí standardu [4], takže mezi výrobci nastává zmatek. Občas se vodiče také značí + a −.

2.4 Různé mikroprocesory

Existuje nezměrné množství různých procesorů. Kritéria výběru pro tuto práci jsou hlavně rychlost, velikost paměti programu, velikost RAM, hardwarová podpora pro UART, alespoň jeden PWM kanál a alespoň jeden časovač.



Obrázek 2.3: Graf spotřeby – ATmega328P (převzato [5])

Rychlost nesmí být příliš nízká, aby zařízení dokázalo v rozumné době zpracovat požadavky. Naproti tomu nesmí být ale příliš vysoká, protože vyšší frekvence způsobuje vyšší spotřebu (lze pozorovat na obrázku 2.3).

Vybral jsem si následující výrobce: Microchip, AVR (nyní již součástí Microchipu), Espressif, ST, Texas Instruments, s jejichž produkty mám již předchozí zkušenosti. Od každého výrobce byl vybrán vzorek procesorů:

- **Atmel (Microchip)** ATtiny85, Atmega8, ATmega168, ATmega328, ATmega32, ATmega2560
- **Microchip** PIC24
- **Espressif** ESP8266, ESP32
- **Texas Instruments** MSP432
- **ST** STM32

Z nichž byly rovnou vyřazeny čipy poskytující bezdrátovou konektivitu, tzn. všechny ESP. Dále byly odstraněny nedostatečně výkonné nebo zastaralé čipy: ATtiny85, Atmega8, …

Pro porovnání jsou vybrány následující 3 procesory: ATMEGA328P-AU, PIC-24FJ256GB106-I/PT a STM32F103C8T6. Různá pouzdra zásadně ovlivňují cenu procesoru – proto se u všech uvedených procesorů počítá s pouzdrem TQFP. Procesor STM32 se v tomto pouzdře nevyrábí u něj byla vybrána

alternativa (LQFP48). Procesor ATmega328 bude podrobněji rozebírán podrobněji v sekci 2.5.

Vlastnosti	ATmega328P	PIC24FJ…	STM32F…
Jádro a architektura	AVR (8 bit)	PIC (16 bit)	ARM (32 bit)
Rychlost [MIPS]	0–20	0–16	0–72
Velikost programu [KB]	32	256	64/128
SRAM [KB]	2	16	20
EEPROM [B]	1024	0	0
Timery	2x8, 1x16	5x16	3x16
UART	1	4	3
Napětí [V]	1,8–5,5	2,0–3,6	2,0–3,6
Teplota [°C]	−40 až +80	−40 až +85	−40 až +85
Cena[7] [Kč]	47,47	132,56	132,57

Tabulka 2.1: Porovnání procesorů

STM32F103C8T6
Spotřebou je na tom tento procesor velice podobně (pokud je vyrovnána frekvence) jako další kandidát ATmega328. Obsahuje RTC, které může (s externí baterií) běžet i když zbytek procesoru stojí. I přes nižší napájecí napětí ($< 3.6\,\mathrm{V}$) jsou některé piny tzv. „5V-tolerant“, to je vhodné pro připojení zařízení, která pracují pouze s 5 volty. Na procesoru se velmi snadno ladí chyby (pomocí ladícího zařízení ST-LINK/V2). Procesor dále nabízí velké množství komunikačních rozhraní včetně CAN a USB. Procesor poskytuje HW podporu pro počítání kontrolních součtů.

Bohužel nenabízí takovou podporu ze strany komunity jako AVR a na programové vybavení dodávané výrobcem si vývojáři stěžují, většina to řeší napsáním vlastních funkcí. Dále nemá procesor zabudovanou paměť EEPROM pro ukládání ne-volatilních informací.

2.5 Procesor ATmega328P

V následujících odstavcích jsou rozebrány hardwarové možnosti poskytované procesorem – ATmega328. Procesor může s okolním světem komunikovat pomocí různých komunikačních rozhraní. Většina zajímavých rozhraní, které jsou v práci využity (UART, I^2C) jsou popsány v části 2.3. Nahrávání programu do procesoru pomocí bootloaderu je popsáno v části 5.3.

Procesor ATmega328 má přijatelnou cenu, dostatečný počet GPIO pinů a lze ho pořídit levně včetně desky Arduino. Vývojová deska Arduino Pro Mini obsahuje pouze součástky potřebné ke správné funkci procesoru – oscilátor, regulátor a pár diskrétních součástek, žádné zbytečné součástky navíc.

[7] Ceny za kus převzaté z `tme.eu` k datu 2019-03-29

ATmega328 má dostupný otevřený toolchain založený na gcc, takže lze používat standardní linuxové nástroje jako třeba Make. Tím lze získat plnou kontrolu nad procesem sestavování a tedy i nad výsledným produktem. Navíc nad těmito nástroji je postavené Arduino, které je také open source a obsahuje velké množství knihoven pro různá externí zařízení, jako například senzory. Bližší informace o nástrojích pro vývoj budou rozepsány dále (sekce 4.1).

Časovače (timery/countery) poskytuje procesor celkem 3 (číslované od nuly), tedy pokud nepočítáme Watchdog – to je technicky vzato také časovač. Všechny mají možnost generovat interrupty při různých událostech (například přetečení) a dále v nějaké formě umožňují použít PWM. Časovač číslo 1 je jako jediný 16bit. Ostatní jsou pouze osmibitové. Arduino core (rozebírané později) využívá časovač 0 pro počítání času od startu programu, tedy počet použitelných časovačů při použití Arduina se snižuje.

Bezpečnost dat v procesoru zajišťují různá protiopatření. Jako první: Watchdog. Jeho hlavní funkcí je reset procesoru při zaseknutí programu. Resetu lze předejít jeho pravidelným nulováním. Délka časového intervalu pro reset lze nastavit v rozmezí 16 ms až 8 s. BOD je ochranou proti podivnému chování procesoru při nižším napájení (nestandardní provádění instrukcí[8], neúplné zapsání dat do paměti a další). Ochrana aktivně sleduje napájecí napětí a pokud napájení klesne pod jistou mez (nastavitelnou pomocí pojistek), tak BOD „drží“ reset.

Přerušení na pinu slouží k tomu, že při změně (může jich být více druhů) přeruší běh programu a skočí na tzv. obsluhu přerušení. Lze tak asynchronně reagovat na události zvenčí procesoru. Přerušení nejsou omezeny pouze pro GPIO, ale lze pomocí nich pracovat s událostmi od časovačů nebo třeba i s událostmi sériové komunikace (např. přijetí bytu dat).

Procesor umožňuje několika způsoby docílit snížení nároků na energii. Lze odpojovat od napájení nepotřebné periferie. Například, pokud není v programu potřeba SPI rozhraní, není třeba ho udržovat aktivní, jednoduše lze v programu odpojit pomocí funkce `power_spi_disable(void)`. Další možností je použít různé úsporné režimy. Nejhlubší režim je „Power-down“ – hodiny stojí, procesor se probudí pouze při přerušeních INT0 a INT1, nebo při shodě adres v I^2C. Pokud je Watchdog aktivní může procesor resetovat.

Při vývoji software je velice praktické mít možnost zastavit běh programu, vypsat si paměť, nebo upravit hodnotu proměnné. ATmega328 podporuje ladění pomocí protokolu DebugWire. Ten je specifický pro jednodušší procesory firmy Atmel (nyní již Microchip).

[8] Toto chybné chování procesoru lze zneužít případným útočníkem například pro získání soukromých klíčů z paměti.

2.5.1 Druhy pamětí procesoru Atmega328P

V procesoru [5] se nachází několik druhů pamětí. Paměť dat, neboli SRAM, což je obdoba paměti RAM u běžných počítačů, slouží k uchovávání dat za běhu programu. Při každém resetu je smazána a není spustitelná. To je dáno tím, že použitý procesor je postaven na Harwardské architektuře – oddělení paměti dat a paměti instrukcí. Do SRAM se zapisuje běžným používáním programu a není třeba se o ní v bootloaderu nijak starat.

Dalším druhem paměti je EEPROM. Stálá paměť, která se nemaže ani při odpojení napájení čipu[9]. Na vybraném čipu je velikost této paměti 1024 bytů. Předpokládaná životnost je omezená počtem zápisů (smazání a zapsání) – nejméně sto tisíc. Počet čtení je neomezený. Firmware v této paměti může uchovávat informace o ID jednotky, verzi FW, ale může i další. Proto je žádoucí, aby bootloader dokázal tuto paměť přepsat. EEPROM má svůj oddělený adresní prostor. Lze do ní přistupovat po jednom bytu. Paměť je rozdělena do 256 stránek, kde každá má 4 byty – je vhodné provádět zápis po jednotlivých stránkách, aby se předešlo zbytečným zápisům.

Jako další druh paměti by se daly považovat pojistky (fuses) a lock bity. Fuses slouží pro nastavení parametrů procesoru. Mezi nastavované parametry patří: výběr krystalu (s dobou čekání na stabilizaci), nastavení ochran (Watchdog a BOD) a další. Lock bity umožňují zamknout paměť procesoru a zabránit neautorizovanému přístupu.

Poslední pamětí je flash (paměť programu). Oproti EEPROM má menší počet možných přepisů (minimálně deset tisíc). U vybraného procesoru je velká 32768 bytů a rozdělená do 256 stránek (každá stránka o velikosti 128 bytů). V této paměti je uložen jak FW (uživatelský program), tak i bootloader. Z tohoto důvodu se paměť dále dělí na 2 sekce – RWW a NRWW [6]. První zkratka znamená Read-While-Write – z paměti lze během zápisu číst a druhá zkratka No-Read-While-Write – paměť je během zápisu uzamčená a nelze z ní číst.

2.5.2 Zdroje resetu

Procesor může být resetován několika způsoby. Po startu je poslední způsob resetu uložen v registru MCUSR.

- **PORF (Power-on)** Normální spuštění.
- **EXTRF (External)** Na pin RESET byla přivedena logická 0 – manuální reset (používá se například při nahrávání softwaru).
- **BORF (Brown-out)** Reset byl způsoben snížením napájení za běhu po určitou hranici, tak procesor zabránil „glitchům“ tím, že procesor udržel ve stavu resetu.

[9] Výjimkou je selhání napájení během zápisu s vypnutou funkcí Brown-Out Detection.

- **WDRF (Watchdog)** Došlo k resetu z důvodu přetečení Watchdog timeru – čítač nebyl včas resetován, nejspíše se program zasekl a bylo třeba provést reset. Lze „zneužít“ k manuálnímu resetu pomocí softwaru.

2.5.3 Bootloader

Důvod proč používat bootloader [7] je možnost nahrávání FW vzdáleně, bez nutnosti být fyzicky připojen na SPI rozhraní. Tato vlastnost se hodí v případech když je procesor umístěn na nepřístupném místě nebo když je třeba nahrát více procesorů najednou. Od bootloaderu je vyžadováno, aby zvládl zápis do EEPROM i flash paměti.

Umístění bootloaderu v paměti a jeho velikost je dána nastavením bitů BOOTSZ1 a BOOTSZ0 v EFUSE (Extended fuse) – ukázka v tabulce 2.2.

BOOTSZ[1:0]	boot reset address	Velikost
00	0x3800	2048 words
01	0x3C00	1024 words
10	0x3E00	512 words
11	0x3F00	256 words

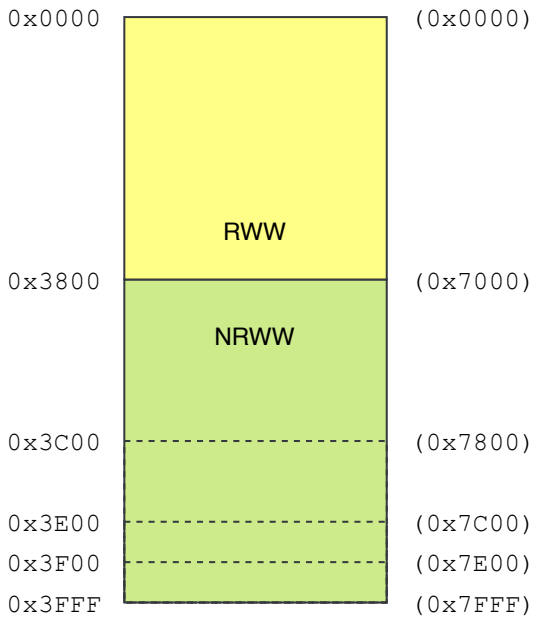
Tabulka 2.2: Nastavení velikosti bootloaderu

V aplikační části programu je možné volat funkce z bootloaderu. Tím lze ušetřit paměť pro společné části (například komunikace nebo obsluha nějakého zařízení). Ovšem nikdy by to nemělo být naopak [7]. Tedy nikdy nevolat části z aplikace pomocí zavaděče. Důvodem je případ, kdy dojde k selhání nahrávání aplikace a bootloader se pokusí zavolat funkci v poškozené části paměti.

Zápis do paměti probíhá pomocí instrukce SPM (store program memory). Zápis probíhá v následujících krocích – kroky 1 a 2 lze prohazovat podle požadované funkcionality:

1. Načtení dat do dočasného bufferu v SRAM.
2. Vymazání požadované stránky v paměti flash.
3. Zápis bufferu do flash.

Na obrázku 2.4 je zobrazeno rozložení flash paměti. Adresy vyobrazené vlevo jsou jako v datasheetu [5], adresovány pomocí slov (16 bitů). Na pravé straně (v závorkách) jsou uvedeny adresy vždy pro byte. Pro gcc je nutné uvádět adresu bytu (například při linkování), naproti tomu v procesoru jsou většinou použity adresy slov, což může dosti mást. Bootloader musí být umístěn v NRWW části. Lze si vybrat ze čtyř možností, vždy od dané adresy v NRWW, až do konce paměti.



Obrázek 2.4: Sekce flash

2.6 Vývojové desky založené na platformě AVR

Lze jistě navrhnout vlastní desku. Její návrh rozhodně není náročný úkol. V datasheetu [5] jsou poskytnuty veškeré potřebné informace. Ovšem to spadá mimo rozsah této práce. Je tedy vhodné vybrat co nejjednodušší desku a přidat k ní moduly poskytující kýženou funkcionalitu.

Arduino je výhodné zvolit jako primární platformu právě díky dostupnosti knihoven na různá zařízení a velké podpoře ze strany komunity. Lze dohledat veškerou dokumentaci k deskám od schémat až po software a to je veliké plus. Také je poskytováno pod velice výhodnou licencí.

2.6.1 Podmínky používání Arduina

Produkty Arduino, jak desky, tak software (Arduino core a knihovny) lze libovolně využít v komerčních produktech za následujících podmínek[10]:

- Na fyzicky zabudované Arduino se nevztahují žádná omezení (ohledně hardwaru).
- Upravená schémata (např. desky) se musí zveřejnit pod stejnou licencí jako originál (Creative Commons).

[10]Informace čerpány z `https://www.arduino.cc/en/main/FAQ` (2019-03-07).

- Použití Arduino core a knihoven pro firmware komerčního produktu nevyžaduje zveřejnění zdrojových kódů produktu. LGPL nicméně vyžaduje alespoň zveřejnění objektových souborů pro relinkování firmwaru proti novějším verzím Arduino core a knihoven.
- Jakákoli úprava zdrojových kódů Arduina, nebo knihoven musí být zveřejněna pod LGP.

2.6.2 Výběr desky

Existuje velké množství druhů různých desek pro platformu Arduino [8]. Jejich cena bývá velice přijatelná. Na oficiálních stránkách Arduina je rozsáhlý přehled[11]. Deska musí splňovat pár základních předpokladů. Musí být levná, nesmí mít zbytečně vysokou spotřebu a bylo by vhodné kdyby, byla menších rozměrů.

Bez zbytečných USB konektorů, bez převodníků USART na USB, zůstanou pouze tyto desky: LilyPad, Pro, Pro Mini, Mini. LilyPad má nevhodný tvar desky a špatně by se k němu přichytávaly moduly. Arduino Pro je příliš velké. Obě dvě zbylé desky vyhovují požadavkům. Arduino Mini má tu výhodu, že má na desce navíc dva analogové piny, ovšem 5 analogových pinů je bohatě dostačujících.

Vybrané desky obsahují minimum hardwaru – pouze součástky nutné k běhu procesoru – filtrační kondenzátory, odpor na resetu, krystal nebo rezonátor a jejich podpůrné obvody (další kondenzátory). Dále deska obsahuje 2 LED diody. První signalizuje přívod napájení – je „zbytečná“ a pouze zvyšuje spotřebu. A druhou LED diodu, kterou lze ovládat programově (jde použít místo napájecí a zároveň pro signalizaci stavu). Poslední součástka se stará o převod napětí z RAW pinu na 5 voltů. V případě, že desky dostanou zvenčí regulované napájení, není součástka třeba a lze ji pro snížení spotřeby odstranit.

2.6.3 Ukázka Arduino core API

Arduino core poskytuje řadu užitečných funkcí [9], které usnadňují vývoj a poskytují jistou vrstvu abstrakce. Na ukázku jsem sepsal krátký kus kódu pro porovnání s „čistým“ AVR C a abstrakcí, kterou poskytuje Arduino core. Čisté C a Arduino core lze libovolně kombinovat. Člověk ale musí mít představu, co dělá.

Seznam funkcí použitých v ukázce kódu 2.2:

- `void pinMode(pin, mode)` Nastavuje režim pinu (vstup, výstup, …), podle tabulky 2.3.
- `int digitalRead(pin)` Přečte logickou úroveň z digitálního pinu a vrátí jednu z hodnot: `HIGH` nebo `LOW`.

[11] `https://www.arduino.cc/en/products.compare`

- `void digitalWrite(pin, value)` Zapíše logickou hodnotu `value` na pin podle tabulky 2.3.

`value`	`mode`	Fyzický výstup
LOW	OUTPUT	GND
LOW	INPUT	Vysoká impedance
LOW	INPUT_PULLUP	Vysoká impedance
HIGH	OUTPUT	VCC
HIGH	INPUT	PULL UP
HIGH	INPUT_PULLUP	PULL UP

Tabulka 2.3: Arduino – režimy pinů

```
// PB0 je pin D8
// nastavení směru pinu D8 na výstup
DDRB |= (1 << DDB0);
// zapsání logické 1 na pin D8
PORTB |= (1 << DDB0);
// pozastavit program na 100 ms
_delay_ms(100);
// zapsání logické 0 na pin D8
PORTB &= ~(1 << DDB0);
```

Kód 2.1: Ukázka čistého AVR libc

```
pinMode(D8, OUTPUT);
digitalWrite(D8, HIGH);
delay(100); // čas v ms
digitalWrite(D8, LOW);
```

Kód 2.2: Ukázka nadstavby – Arduino core

Z porovnání je jasné, který kód vede v přehlednosti. Na druhou stranu kód 2.2 je o několik desítek cyklů pomalejší než čisté C v ukázce kódu 2.1. Tedy pro časově kritické operace je výhodnější použít přímý přístup (v extrémním případě lze využít i assembler).

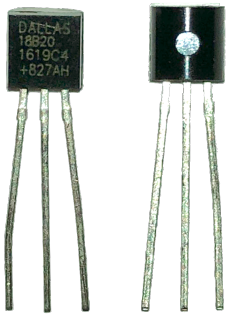
2.7 Měřící čidla

K měření podmínek v místnostech budou jistě třeba nějaké senzory. V následující sekci jsou popsány různé senzory sloužící k měření různých fyzikálních veličin.

Čidla je nejvýhodnější nakupovat z Číny. Důvodem jsou nesrovnatelně nižší ceny (oproti tuzemskému trhu) a poštovné zdarma. Je ovšem důležité součástky před užitím řádně otestovat. Originální (a dražší) součástky bývají méně poruchové a přesnější.

2.7.1 Teplotní čidlo DS18B20

Jednoduchý senzor na měření teploty [10]. Komunikace s ním probíhá po sběrnici OneWire. Umožňuje měřit teplotu s různým rozlišením. Od devíti do dvanácti bitů. S vyšším rozlišením se zpomaluje rychlost měření – pro dvanáct bitů již celkem citelných 750 ms.



Obrázek 2.5: Čidlo DS18B20

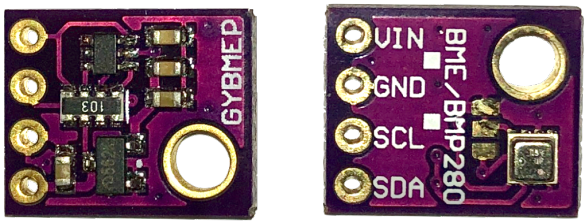
Každé čidlo má unikátní sériové číslo (64 bit). To je z továrny „vypálené“ v ROM paměti.

Čidlo má další pokročilejší vlastnosti – „alarm“ v případě, že teplota překoná uživatelsky nastavené hranice, vnitřní paměť EEPROM a tak dále. Těmto „pokročilejším“ vlastnostem se již dále nebude věnováno.

Senzor lze napájet ve dvou režimech. Pro normální režim potřebuje 3 vodiče, z toho 2 jsou napájení (VCC a zem) a jeden datový (sběrnice OneWire). V parazitním režimu stačí vodiče dva. K zajištění napájení potom slouží sám datový vodič. Pin VCC, který v normálním režimu slouží k napájení je propojen se zemí (GND).

2.7.2 Kombinovaný senzor teploty, tlaku a vlhkosti BME280

Kombinované čidlo od firmy Bosch sloužící k měření vlhkosti, tlaku a teploty [11]. Na komunikaci lze použít jak I^2C, tak SPI. Při porovnání různých čidel bylo BME280 vybrané jako čidlo s nejlepšími vlastnostmi [12].



Obrázek 2.6: Čidlo BME280

Je důležité zvolit správnou desku (ne alternativní BMP280). A aby bylo podporované správné napájení: tedy 1,8–5,0 V. Ceny začínají na přibližně dvou a půl eurech za čidlo (za desku na obrázku 2.6).

2.7.3 Čidlo vlhkosti a teploty DHT22

Kombinované čidlo na snímání teploty a vlhkosti. Jinak označované také jako AM3202. Výrobce uvádí v datasheetu vysokou spolehlivost [13], i když testy naznačují přesný opak [12].

2.7.4 Bezkontaktní IR teplotní čidlo GY906

Infračervené bezkontaktní teplotní čidlo, označované také jako MLX90614 [14]. Lze vybírat ze dvou variant napájení (3 V nebo 5 V). Operační teplota (ambientní teplota okolo čidla) neprůmyslové verze musí být v rozmezí od −40 °C do +80 °C. Čidlo komunikuje pomocí sběrnice SMBus (pod 100 kHz zaměnitelná za I^2C [15]).

Lze měřit buď ambientní teplotu okolo senzoru, nebo teplotu objektu před čidlem. Teplota objektu, který je měřen vzdáleně může dosahovat až 300 °C. Rozlišení senzoru je 0,02 °C, přesnost na intervalu 0–50 °C je podle datasheetu [14] ±0,5 °C.

Na zapojení je třeba jeden kondenzátor (napájení) a dva odpory pro SMBus. Jsou prodávány hotové desky se všemi potřebnými součástkami. Cena[12] za takovou desku s čidlem je 98 Kč za kus.

2.7.5 Senzor pro měření vysokých teplot – MAX6675

Měřící zařízení se skládá ze dvou částí: čidlo a čip zprostředkující měření a komunikaci s procesorem (MAX6675). Toto čidlo je vhodné pro měření vyšších teplot. Poskytuje měřený teplotní rozsah 0–1024 °C [16].

Cena za desku s čipem včetně čidla se pohybovala okolo 50 Kč za kus[13].

[12] Nejnižší nabídka z `https://www.ebay.com` k datu 2019-05-05
[13] Z obchodu `https://www.aliexpress.com` k datu 2019-05-05.

2.8 Klimatizační jednotky a analýza protokolu

Jedním z požadavků majitele domu bylo zprovoznit ovládání jednotek tak, aby nebyla narušena případná záruka. Tedy neinvazním způsobem a v ideálním případě vůbec do nainstalovaných jednotek nezasahovat. Všechny klimatizace jsou od stejného výrobce a ovladače na první pohled vypadají stejně.

U klimatizačních jednotek, které jsou nainstalovány v domě, je třeba udělat hlubší analýzu komunikace mezi nimi a ovladačem, aby bylo možné později vytvořit dálkové ovládání. Tím budou splněny podmínky pro neupravování klimatizace, protože bude pouze nahrazen ovladač (za vlastní).

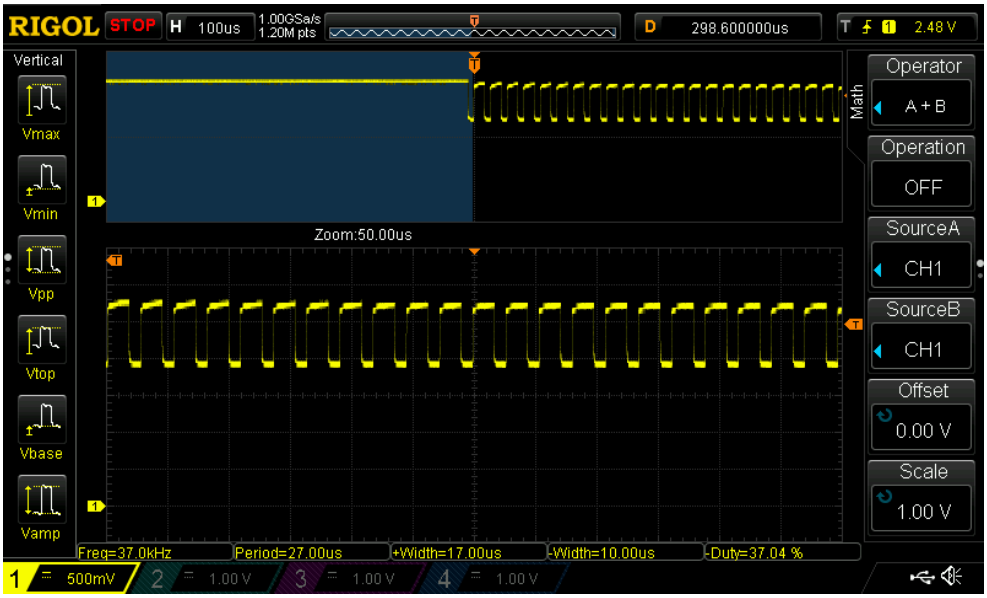
Z konstrukce jasně vyplývá, že se jedná o IR technologii – v čele ovladače se nachází zabudovaná IR dioda a ovládání nefunguje v případě, že není přímý dohled ovladače na klimatizaci. Aby bylo možné napodobit vysílaný signál, je třeba znát 3 údaje:

1. nosnou frekvenci vysílání,
2. vlnovou délku na které vysílá IR dioda,
3. použitý protokol na předávání povelů.

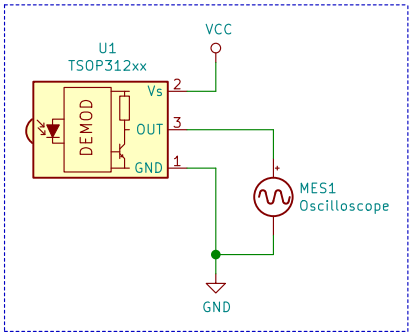
Protože není žádoucí ovladač rozebírat (podmínky záruky), měření nosné frekvence probíhalo jako měření napětí na bateriích, které jsou snadno přístupné bez složitého rozebírání. Výsledky měření lze vidět na obrázku 2.7. Při vysílání dochází k poklesu napětí na bateriích z důvodu zatížení (tzv. vnitřní odpor). Měření předpokládalo, že největší spotřebu v době vysílání má právě vysílací IR dioda. Z měření vyplynulo, že frekvence je okolo 37 kHz, což je běžná nosná frekvence pro IR vysílače.

Další potřebná informace k zajištění přenosu je vlnová délka vysílače. Do blízkého infračerveného záření (NIR) se řadí elektromagnetické záření s vlnovou délkou od 0,75 μm až do 1,4 μm. Nakonec byla zvolena běžně dostupná 5mm IR dioda (IR333) s vlnovou délkou 940 nm. Dioda pokrývá alespoň dvaceti procenty intenzity rozsah (přibližně) 900–990 nm, což by mělo (při zajištění dobrého výhledu) na testování stačit.

Po zjištění všech potřebných parametrů byl při dekódování velmi nápomocný tento článek [17]. I když dálkové ovládání nebylo stejné, tak způsob kódování jednotlivých bitů přibližně odpovídal. Pomocí logického analyzátoru Saleae (popsaném dále v sekci 4.1) byla nahrána data z ovladače pomocí kterých jsem analýzou v Pythonu vytvořil kodér a dekodér. Zapojení logického analyzátoru je vyobrazené na obrázku 2.8.



Obrázek 2.7: Odběrová analýza – nosná frekvence



Obrázek 2.8: Schéma přijímače IR signálu

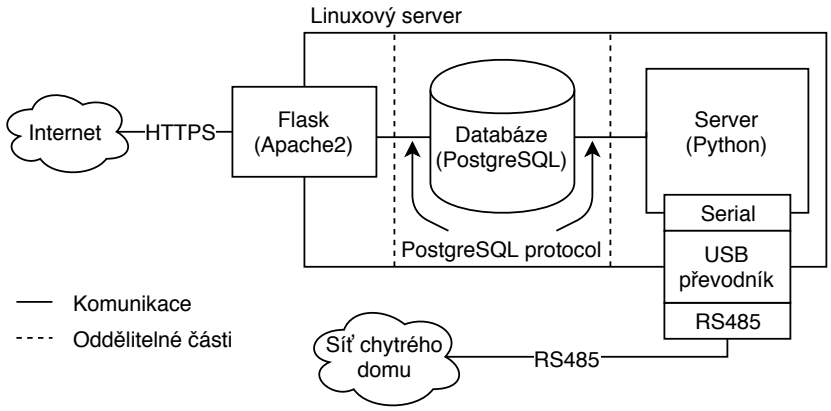
Kapitola 3

Návrh

V první části následující kapitoly je rozepsán celkový návrh systému, schémata rozmístění senzorů podle jednotlivých měřených míst. Dále se věnuje detailům návrhu jednotlivých částí jako jsou firmware, bootloader a webové rozhraní.

3.1 Topologie systému

Uživatelské rozhraní formou webové aplikace je velice vhodný kandidát. Umožňuje připojit se odkudkoli z různých platforem (telefon, počítač, tablet). Existují kvalitní webové servery, které jsou dostupné zdarma. Příklady takových serverů jsou například Nginx, Apache.



Obrázek 3.1: Schéma systému

Na obrázku 3.1 je vidět výsledný návrh celkového systému. Webové rozhraní zprostředkované pomocí Flasku propojené pomocí databáze s backendem. O propojení do internetu se stará webový server Apache2, který zajišťuje spo-

jení s uživatelem pomocí protokolu HTTPS. Backend, na obrázku označený jako Server, zajišťuje propojení databáze se sítí chytrého domu.

Návrh umožňuje jednoduché rozdělení jednotlivých částí. Ovšem výhodnější je udržovat části u sebe co nejblíže kvůli nižší latenci propojených částí.

Nejjednodušší možností návrhu sítě chytrého domu je použít nějakou sběrnici senzorů. To by znamenalo, zvolit místa měření v jednotlivých místnostech. Ideálně co nejblíže k sobě, kvůli vedení. A na tyto body umístit senzory.

3.2 Návrh jednotky chytrého domu

Návrh sběrnice senzorů je dobrý, ale výhodnější je přidat k senzoru další elektroniku, která bude obsluhovat senzor a zajišťovat komunikaci. Potom je možné obsluhovat více senzorů pomocí jednoho procesoru propojeného s ostatními. Taková množina senzorů bude v práci dále nazývána jako „jednotka“.

Základ každé jednotky se skládá z desky Arduino a modulu RS-485. Desky se dodávají s 16MHz krystalem. ATmega je na této rychlosti schopna provádět bez problémů všechny požadované funkce, takže není třeba provádět úpravy taktu jednotky.

Arduino je propojeno s RS-485 naletovanými vodiči. Kromě RX a TX obsahuje RS-485 ještě dva kontrolní piny. Jeden slouží k zapnutí vysílání a druhý k ovládání přijímání. Jeden z pinů je invertován, lze tedy oba kontrolní vodiče propojit a najednou ovládat vysílání a přijímání na sběrnici pomocí jednoho pinu na procesoru.

Dále je vybrán jeden pin pro sběrnici OneWire – standardně pin D9. Ten je propojen pomocí 4,7kΩ rezistoru k zemi. Na tento pin se v případě potřeby připojí čidla DS18B20. Jednotky, které mají zároveň měřit vlhkost musí obsahovat čidlo BME280. To je připojeno na I^2C rozhraní mikročipu – implicitně piny A4 (SDA) a A5 (SCL).

Běžné jednotky v síti jsou všechny napájeny ze stejného místa (po síti směrem od serveru). Nepoužívají zabudovaný lineární regulátor napětí, tím je dosaženo snížení spotřeby. Tento styl napájení je vhodný pouze pro menší sítě. U velkých sítí by bylo možné používat vyšší napětí (např. 12 V) nebo napájet všechny jednotky ze sítě (220 V) přímo u nich.

Tím je daná základní kostra nejběžnějších jednotek, která by se již neměla měnit. Pokud by byl vyráběn plošný spoj pro jednotku, obsahoval by právě tuto základní kostru. Ostatní funkčnost by byla řešena pomocí modulů

3.2.1 Jednotka v kamnech

U jednotky v kamnech je s napájením problém. Obsahují totiž ovládací relé s vysokou spotřebou. V tom případě je jednotka stále napájena pomocí 5V sítě, ale výkonný spotřebič – v tomto případě relé – je napájen z kamen (220V

větev). U zdroje je ovšem třeba hlídat, aby nevznikla tzv. „zemní smyčka“. Toho lze docílit použitím vhodného zdroje, nebo zajistit galvanické oddělení ovládací a výkonné části jednotky. Naštěstí některé moduly z Číny obsahují již zabudované optické oddělovače. Ty se postarají o oddělení jednotlivých částí. To přináší další výhodu – napájení serveru je spolu s napájením sítě chytrého domu zálohováno pomocí UPS, při výpadku elektrické energie lze stále monitorovat teplotu kamen.

Pro měření teploty v kamnech, kde je třeba sledovat teplotní rozsahy, které jsou mimo provozní schopnosti čidel DS18B20 nebo BME280 je třeba využít čidla GY906 a MAX6675.

3.2.2 Ostatní jednotky

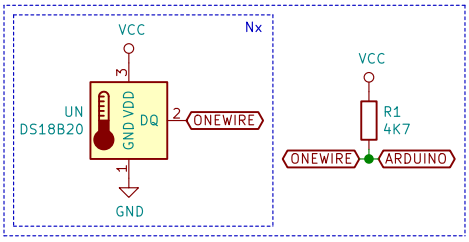
Kromě běžných a kamnových jednotek bude ještě několik dalších druhů zastupujících specializované funkce.

Jednotka v bojleru bude mít navíc dvojité snímání 220 V. První čidlo snímá napájení přivedené k bojleru, aby šlo určit zda je aktivní „noční proud“. Druhé čidlo bude snímat napětí na topných cívkách. Tím se do systému dostane informace o tom, zda bojler nahřívá vodu nebo ne. Na standardní OneWire rozhraní jednotky bude připojeno 5 senzorů DS18B20 sloužící k určení teploty v různých částech bojleru.

Další speciální jednotkou je jednotka obsluhující ovládání rekuperace. Ta obsahuje navíc modul s optickými oddělovači ovládající jednotlivá tlačítka rekuperace. A podobná jednotka s relé se bude starat o otevírání a zavírání protimrazové klapky.

3.2.3 Zapojení teplotních senzorů DS18B20

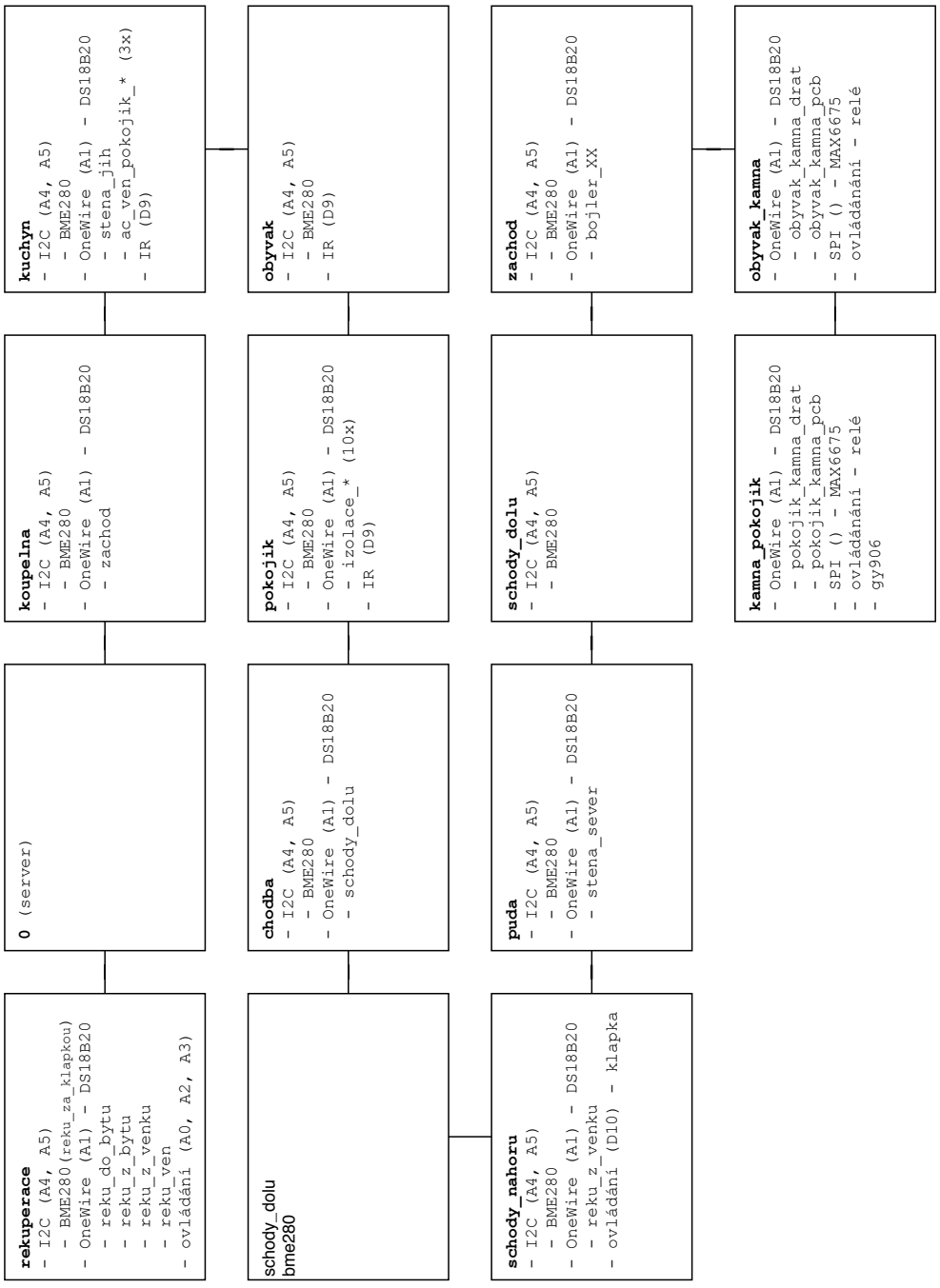
U senzoru DS18B20 bude používán normální režim napájení. Důvodem je vyšší stabilita během měření s větším množstvím čidel najednou. Schéma zapojení sběrnice OneWire a senzorů DS18b20 je na obrázku 3.2. Čidel lze na takovou sběrnici připojit více.



Obrázek 3.2: Zapojení senzorů DS18B20

Konkrétní přiřazení senzorů k jednotlivým jednotkám je popsáno ve schématu na obrázku 3.3.

3.3 Rozmístění jednotek na sběrnici RS-485

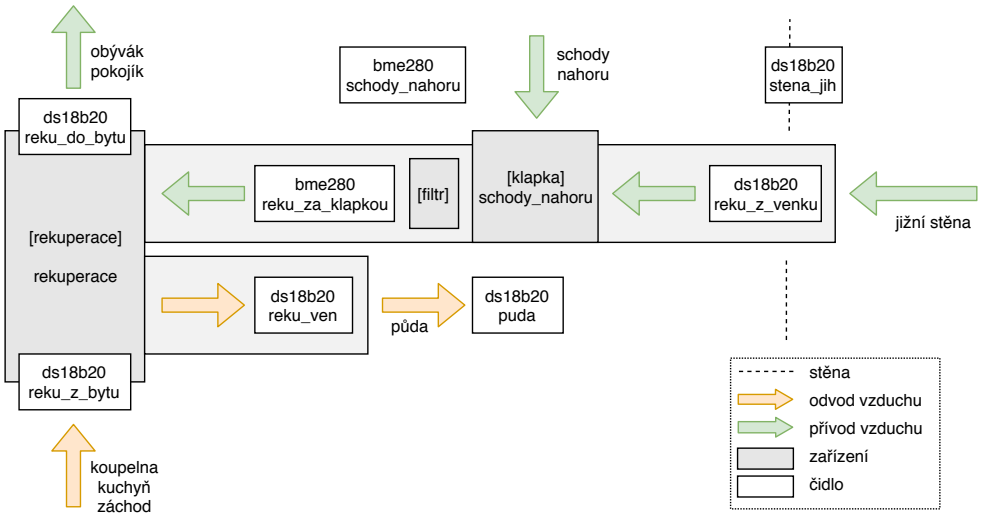


Obrázek 3.3: Rozmístění jednotek v síti

3.4 Rozmístění senzorů – vzduchotechnika

Na obrázku 3.4 je vidět budoucí rozmístění senzorů obsluhujících vzduchotechniku. Senzory zatím kromě sbírání dat a ovládání protimrazové klapky nebudou mít další funkce.

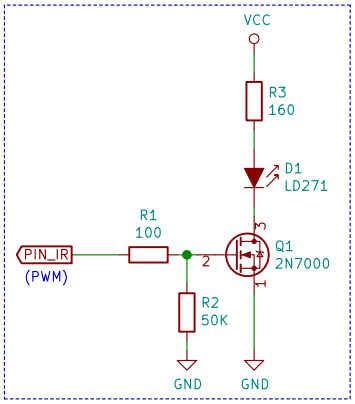
Přívodní izolovaná roura vede přibližně 5 m domem, kde je teplota během zimy vyšší než venku a i přes izolaci roury může být rozdíl mezi čidlem `reku_z_venku`, které je nainstalované hned za vstupem a `reku_za_klapkou` až 5 °C.



Obrázek 3.4: Rozmístění senzorů – vzduchotechnika

3.5 Obvod pro řízení klimatizace

Na obrázku 3.5 je zobrazeno schéma zapojení vysílací IR diody. Rezistor R3 omezuje proud procházející skrz diodu tak, aby nedošlo k jejímu poškození. Výstupní piny procesoru ATmega328P mají omezený maximální (dlouhodobý) proud na 20 mA. Je třeba použít externí tranzistor. Ve schématu je vybrán 2N7000 [18], jeden z běžně používaných N-channel MOSFET tranzistorů. Odpor R1 slouží k omezení proudové špičky mezi procesorem a tranzistorem. R2 slouží jako „pull-down“ rezistor. V případě, že by `PIN_IR` byl v režimu vysoké impedance R2 zajistí „uzavření“ tranzistoru Q1. Jinak by se za jistých okolností mohla dioda chovat nepředvídatelně, například během resetu procesoru (kdy jsou všechny GPIO piny v režimu vysoké impedance).



Obrázek 3.5: Schéma navrženého IR vysílače

Bylo třeba určit hodnoty rezistorů. Na ukázku následuje výpočet odporu R3. Budeme předpokládat že IR dioda potřebuje k plní funkcí 20 mA s tím, že hodnota z datasheetu pro „forward current“ je 1,2 V.

Tedy $U = VCC - U_{led} = 5 - 1{,}2 = 3{,}8$ a podle Ohmova zákona platí:

$$R_3 = \frac{U}{I_{led}} = \frac{3{,}8}{0{,}02} = 190\Omega \tag{3.1}$$

Hodnota rezistoru R2 není až tak důležitá. Nesmí být ovšem příliš malá, aby se zbytečně nezvyšovala spotřeba, ale ani příliš velká, aby rezistor stále fungoval jako pull-down i ve více zarušeném prostředí (rozsah 10 kΩ – 50 kΩ je vyhovující).

Obvod je tedy navržen. Nyní je třeba vyřešit jak bude procesor generovat signál. Naštěstí jednou z funkcí procesoru ATmega328 je generování PWM. Takže procesor dokáže velice jednoduše provádět modulaci signálu – není tedy nutné žádné další externí zařízení na modulaci. Po nastartování procesoru tedy stačí nastavit generování PWM a přerušit přívod hodin. Pro zapnutí výstupu signálu stačí nastavit pin na výstup a nastavit před-děličku (spustit hodiny).

3.6 Navržený komunikační protokol

Díky nízké ceně a HW podpoře ze strany procesoru ATmega328 (UART0) bylo rozhodnuto, že bude použita sběrnice RS-485, která se bude starat o komunikaci na fyzické vrstvě. Je ovšem třeba zajistit nějaký jednoduchý protokol, který bude postaven nad RS-485, pomocí kterého jednotky budou komunikovat se serverem. Bude to master-slave protokol, aby se zabránilo kolizím během komunikace.

Na výběr bylo více možností. Lze používat buď binární nebo ASCII protokol. Protože odstraňování chyb z ASCII protokolu je daleko jednodušší než z binárního (není třeba psát další nástroje na dekódování do čitelné podoby), byl vybrán právě ASCII protokol.

ATmega podporuje MPCM, což je také velká výhoda. Snížilo by se tím zatížení ostatních jednotek (a tím i spotřeba). Bohužel linuxové ovladače nepracují s 9bit hodnotami příliš dobře (vlastně vůbec). Proto musel být MPCM prozatím z návrhu vynechán.

Baudrate komunikace byl prozatím experimentálně určen na 57600. Hodnoty vyšší než 115200 způsobovaly chyby v přenosu ve větší síti. Tak byl baudrate stanoven na polovinu stabilní hodnoty.

Formát paketu

První znak paketu musí být dolar. Jde o signalizaci jednotce, že bude probíhat komunikace. To dá jednotkám čas, aby se pomocí přerušení probudily ze spánku. Následuje začátek paketu a *id*. To si každá jednotka načte během startu z EEPROM. Pokud *id* není její, vrátí se do spánku. Pokud ano, pokračuje v parsování a ukládání paketu. Aby byl popis srozumitelnější následuje popis paketu v EBNF:

```
packet    = dummy, ">", id, "|", data, "<";
dummy     = "$";
id        = master | broadcast | word;
master    = "0";
broadcast = "all";
data      = command, {argument};
argument  = " ", word, {argument};
word      = ? libovolý tisknutelný ASCII znak kromě: " |<" ?;
```

Dvě ukázky validního paketu:

- `$>kuchyne|bme280<`
- `$>puda|ds18b20<`

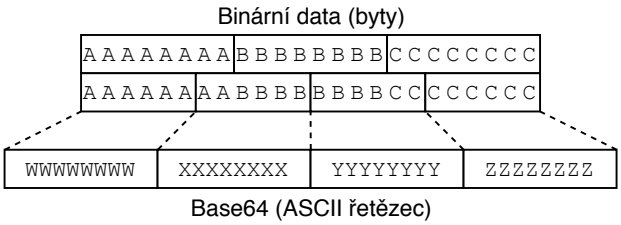
3.6.1 Popis hlavních příkazů v protokolu

Protokol specifikuje následující sadu příkazů, ovšem není nutné, aby každá jednotka uměla všechny příkazy.

- `ping` – Odpoví jednotka jednoduše `pong`, je to signál pro ověření, zda jednotka je na síti a zda komunikuje.
- `led ...` – Lze ovlivnit chování led diody, ta explicitně signalizuje komunikaci a tímto příkazem lze diodu vypnout.
- `delay <ms>` – Jednotka čeká specifikovaný počet milisekund a během toho nepřijímá žádný vstup.
- `pin ...` – Slouží k manuálnímu ovládání GPIO u jednotek. Lze použít k ovládání kamen.
- `ds18b20 ...` – Příkazem lze vypsat všechna čidla na sběrnici OneWire připojené k jednotce, aktualizovat jejich hodnoty nebo selektivně vybrat konkrétní čidlo.
- `reset` – Použije Watchdog k resetování jednotky.
- `bootloader` – Stejné jako předchozí příkaz, ovšem postará se o to, že po resetu zůstane jednotka v režimu bootloaderu.
- `bme280 ...` – Vrátí všechny 3 hodnoty ze senzoru nebo lze vybrat kterou konkrétní hodnotu odeslat (teplota, tlak, vlhkost).
- `ir ...` – Slouží k ovládání klimatizace.
- `temperature` – Jednotka odešle interní teplotu čipu ATmega328 – detekce přesáhnutí provozních podmínek.
- `voltage` – Vrátí interní napájecí napětí vhodné například k předvídání budoucích chyb v napájení.
- `flash_dump` – Vrátí dump FLASH paměti – pouze pro ladění!
- `mem_info` vypíše aktuální stack pointer a adresu posledního alokovaného bloku na haldě. Pouze pro ladění!

3.6.2 Base64

Protože je pro komunikaci využit ASCII protokol, je třeba zajistit přenos binárních dat pomocí textového protokolu. Na to je vhodné kódování Base64 [19] – umožňuje totiž přenos dat v poměru 3 binární byty : 4 textové byty. Umožňuje daleko efektivnější přenos než posílání jednotlivých bytů jako čísla. Existuje ještě úspornější kódování, Base85, to umožňuje kódovat v poměru 4:5, nicméně to používá některé speciální znaky, které jsou využity pro označování začátku a konce paketu v komunikačním protokolu.



Obrázek 3.6: Kódování Base64

Kódování probíhá tak, že se vstup rozdělí na bloky po 3 bytech. Každý 3 bytový blok rozdělíme na 4 hodnoty po 6 bitech a každá hodnota je použita jako index do tabulky 3.1. Dekódování je přesně opačný proces.

Index	Znak
0–25	A–Z
26–51	a–z
52–61	0–9
62	+
63	/
(padding)	=

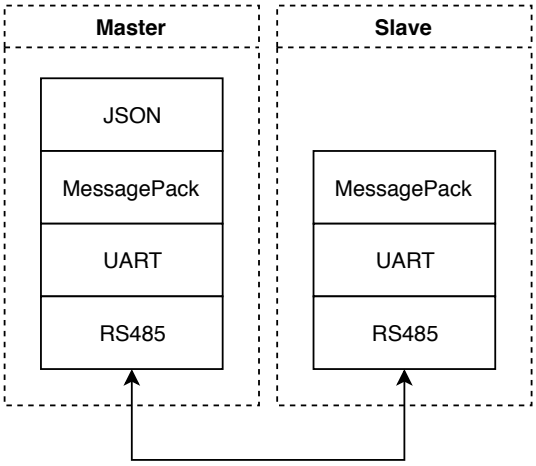
Tabulka 3.1: Kódování Base64

3.6.3 Možnosti budoucího rozšíření

V budoucnu lze uvažovat o přechodu na efektivnější binární protokol. Testovací ASCII protokol je velice vhodný pro ladění celého systému a také pro názornost. Ovšem je značně neefektivní na délky přenosů, tzn. i na paměťovou náročnost. V protokolu jsou používány pouze tisknutelné znaky ASCII tabulky. Z toho plyne, že z 255 možných hodnot, které je možné použít při osmibitovém UART přenosu je efektivně využita méně než polovina potenciálních hodnot. Konkrétně 95/255 (62,75%) odeslaného bytu není nikdy nijak využito. Během nahrávání FW, kdy jsou přenášena binární data zakódovaná v Base64 je promarněna další 1/5 potenciálních dat.

Strukturální návrh modernějšího protokolu vyobrazen na obrázku 3.7. Následuje popis možného tvaru paketu protokolu v EBNF:

```
paket     = dummy, id, data, checksum;
dummy     = 0x00;
id        = master | broadcast | 0x01-0xfe;
master    = 0x00;
broadcast = 0xff;
data      = ? příkaz ve formátu MessagePack ?;
checksum  = ? kontrolní součet celého paketu ?;
```



Obrázek 3.7: Možný návrh rozvrstvení modernějšího protokolu

3.7 Návrh bootloaderu

Nabízí se možnost pro přenos citlivých dat snížit baudrate komunikační sběrnice. To sníží množství výskytu určitého typu chyb. Při testování se ukázalo, že na takto nízkém baudrate (57600) je zbytečné snižovat rychlost a daleko výhodnější je provádět nějakou formu kontroly integrity dat (např. CRC16). Poškozená data následně odeslat znovu.

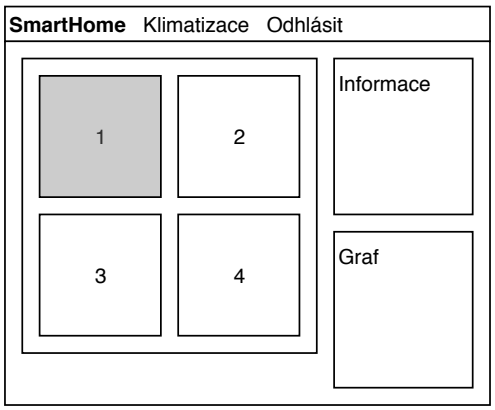
Pro bootloader bylo vybráno největší možné místo. Určitě by bylo možné bootloader zmenšit. Například vynecháním vektorů přerušení – v bootloaderu nejsou třeba (s výjimkou resetu). Případně optimalizací kódu – hlavně přepsání největších částí do assembleru.

Během implementace bootloaderu by byla opravdu užitečná funkce `sscanf`. Načítání formátovaného vstupu přímo z bufferu do proměnných by hodně usnadnilo práci. Bohužel tato funkce je pro použití v bootloaderu příliš velká a i přes zapnuté optimalizace je velikostně nepřípustná.

Bootloader bude po stránkách zapisovat data přicházející po síti a odesílat zpět jejich CRC. Protože data budou přenášena kanálem, který byl navržen jako ASCII, je třeba přenášená data kódovat pomocí Base64 (blíže popsáno v sekci 3.6.2).

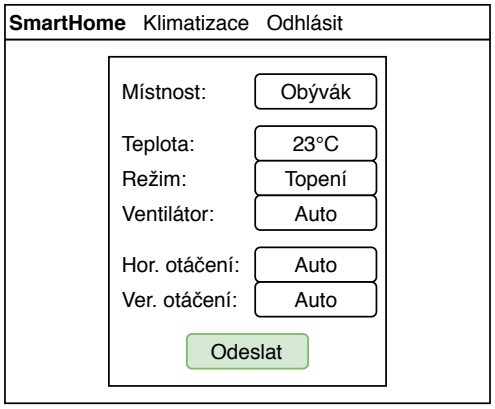
3.8 Webové rozhraní – Wireframe

Podle zadání musí webové rozhraní poskytovat přístup k aktuálním hodnotám čidel v domě, dále vykreslovat data za posledních 24 hodin a umožnit ovládání klimatizace. Návrh takového webu lze vidět na obrázku 3.8.



Obrázek 3.8: Návrh hlavní stránky webového rozhraní

Na obrázku 3.8 se nahoře nachází navigační lišta, která umožňuje měnit aktuální zobrazenou stránku. V levé části obrazovky je graficky znázorněn půdorys bytu. V něm se nachází jednotlivé místnosti uvnitř kterých bude aktuální výpis ze senzorů. Z těchto místností půjde vybrat myší jednu místnost ke které se v pravé dolní části stránky zobrazí historie formou grafu za posledních 24 hodin. Vpravo nahoře jsou zobrazeny globální informace o domě.



Obrázek 3.9: Návrh rozhraní ovládání klimatizace

Z popisu chování ze zadání chybí už jen ovládání klimatizace (obrázek 3.9).

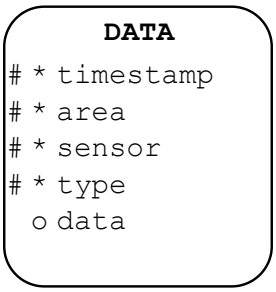
3.9 Výběr a návrh databáze

První možnost kterou je třeba zvážit je žádnou databázi nepoužívat a záznamy ukládat do obyčejného souboru. Takový systém by zajisté fungoval, ovšem při zvýšení požadavků – například zmenšení intervalů mezi měřeními nebo nutnost pamatovat si informace delší dobu než 24 hodin by zcela jistě selhal. Takových příkladů, kdy je uložení dat v souboru neefektivní je více. Například dostupnost během hromadného přístupu více uživatelů najednou nebo i obyčejný obtížný přístup. Přidání databáze je tedy dalším logickým krokem.

V této práci není třeba využívat nějakých pokročilejších vlastností moderních databází. Bylo by vhodné mít datový typ pro čas a datum, možnost pomocí něj efektivně řadit. Dále typ pro desetinné číslo a textový řetězec. Tyto požadavky hravě splní libovolná dnešní databáze.

PostgreSQL má za sebou již přes 30 let vývoje a testování. Stále vyvíjená open source relační databáze – multiplatformní, dostupná pro Windows i Linux a je zdarma. Protože u tohoto projektu není třeba nějaká rozsáhlejší komerční podpora ze strany databáze, lze celkem snadno využít.

Na ukázku byla vybrána jedna tabulka z databáze (obrázek 3.10). V databázi bude tabulek daleko více – ovládání klimatizace, logování bojleru a další.



Obrázek 3.10: Návrh tabulky pro ukládání dat

Kapitola 4

Implementace

V této kapitole jsou rozebírány bližší informace ohledně implementace práce. V první části jsou popsány nástroje s jejichž pomocí byla celá práce vyvíjena – kompilátory, debuggery a další. V další části je popsána práce s HW a jakým způsobem musela být upravena deska pro ladění chyb. Následuje detailní popis sestavení programu na platformě AVR. Zbytek kapitoly se věnuje implementaci backendu a webového rozhraní.

Většina SW částí práce je psaná v jazyce Python pro jeho úspornost a jednoduchost. FW v jednotkách je kombinací C++ (Arduino core), C (avr-libc) a assembleru.

4.1 Vývojové nástroje a používané technologie

Vývoj je prováděn na počítači s operačním systémem Mac OS, ale práce bude nasazena na Linuxový stroj s operačním systémem Debian. To lze provést díky tomu, že většina používaných nástrojů je multiplatformní a díky návrhu který umožňuje oddělení jednotlivých komponent.

Při vyvíjení softwaru pro platformu AVR jsou na výběr dvě hlavní možnosti. Jít proprietární cestou, tedy používat Atmel Studio (současná[14] verze je 7). Nebo open source cestou a používat GNU toolchain.

Arduino IDE (postavené nad open source toolchainem) je vhodné pro začátečníky, ovšem neumožňuje využití mnoha funkcí, které by se mohly hodit: například manuální linkování, změny úrovně optimalizací Arduino core, atd. Proto byla během vývoje snaha nikdy s ním nepracovat přímo. Arduino core poskytuje sadu „standardních funkcí“ a možnost použití knihoven, proto před začátkem implementace bylo z Arudino IDE vyextrahováno Arduino core, ze kterého byly odstraněny zbytečné funkce pro jiné procesory. A linkování

[14] K datu 2019-04-05.

FW neprobíhalo v rámci Arduino IDE, ale samostatně pomocí GNU toolchainu.

Největší výhoda proprietárního softwaru je plná podpora ze strany výrobce čipu. Pro práci nakonec byl zvolen GNU toolchain. Ten nabízí klasické nástroje pro vývoj:

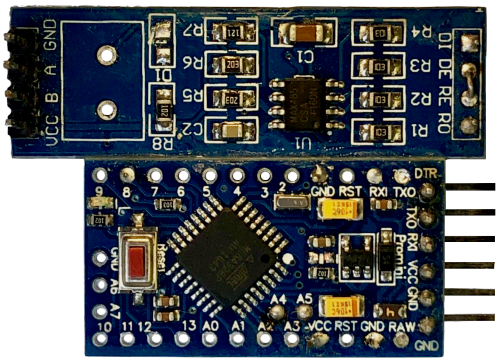
- **Make** – Nástroj na sestavování projektů a řešení závislostí zdrojových souborů.
- **Avrdude** – Čtení, zápis, ověřování kódu a dat z/v/do procesoru.
- **avr-gcc** – Kompilátor pro platformu AVR. Umožňuje vybrat procesor a na rozdíl od kompilátoru v Arduino IDE je častěji aktualizován.
- **avr-libc** – Standardní knihovna pro jazyk C pro AVR platformu.
- **avr-gdb** – Běžný linuxový debugger.
- **Arduino core** – Statická knihovna poskytující funkce pro aplikace Arduino.
- **Python** – Programovací jazyk ve kterém bude provedena implementace webové aplikace a serverové části komunikace s jednotkami.

Z hardwaru jde hlavně o měřící přístroje nebo nástroje propojující Avrdude s mikroprocesorem. Tedy:

- **AVR Dragon** – Oficiální nástroj pro práci s AVR čipy – umožňuje vysokonapěťové programování a další pokročilé funkce.
- **Bus Pirate (v4)** – Otevřená a levnější alternativa k AVR Dragon – umožňuje komunikovat po velkém množství protokolů.
- **Rigol DS1054Z** – Levný 100MHz osciloskop. Vhodný k hledání chyb v HW částech práce.
- **Saleae** – Logický analyzátor – oproti osciloskopu je „křehčí“ (pouze 0–5 V), ale k načtení signálu do počítače je vhodnější.

4.2 Výroba hardwaru

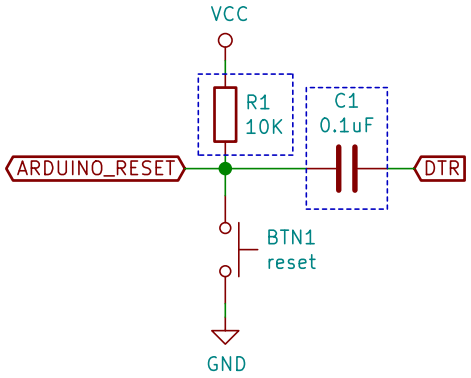
„Výroba“ jednotek je díky používání modulů velmi snadná. Nejprve je nutné nahrát do ATmegy bootloader (do paměti flash) a její ID (do paměti EEPROM). Oba dva kroky lze provést jedním příkazem z Avrdude. Nahrání probíhá pomocí ICPS konektoru na desce – potom lze přistoupit k osazení dalšími moduly. Rozhraní RS-485 je propojeno s pomocí tří vodičů, stejně jako BME280. Následně stačí naletovat rezistor pro sběrnici a prototyp desky je hotov. Prototyp takové desky lze vidět na obrázku 4.1, vodiče a senzor BME280 jsou přiletovány zespodu.



Obrázek 4.1: Prototyp vyrobené jednotky

4.2.1 Úprava desky pro ladění

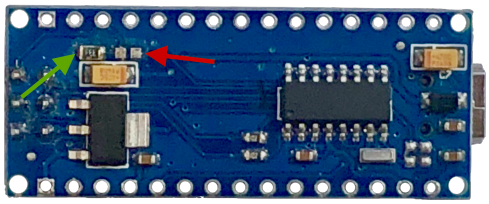
Na ladění byla použita deska Arduino Nano, nicméně bylo třeba desku upravit. Ve specifikaci je dáno, že na pinu RESET nesmí být připojeno nic kromě pull-up rezistoru, který nesmí být menší než 10 kΩ (viz. datasheet [5]).



Obrázek 4.2: Schéma resetu (Arduino NANO)

Na obrázku 4.2 je původní zapojení pinu RESET u Arduina. Signál DTR slouží k resetování pomocí USB převodníku. V tomto zapojení je třeba odebrat kondenzátor C1. A vyměnit 10kΩ rezistor za nějaký s vyšším odporem (v tomto případě 33 kΩ). Protože tlačítko je ve výchozím stavu rozpojené, není třeba ho odebírat, pouze během ladění nemačkat.

Tato úprava (obrázek 4.3) umožní používat DebugWire, ovšem znemožní automatický reset zařízení při nahrávání aplikace pomocí USB – tato funkce není u desky na ladění potřebná (nahrávání probíhá přes ICSP nebo DebugWire). Červená šipka označuje místo odstraněného kondenzátoru a zelená ukazuje na vyměněný rezistor.



Obrázek 4.3: Upravená deska pro ladění

4.3 Komunikační sběrnice

Nejprve byl systém implementován pomocí jedné řídící jednotky s velkým množstvím čidel DS18B20 na jedné OneWire sběrnici. Tato implementace nebyla úspěšná. Rozhodně může spolehlivě fungovat pro menší sběrnici, například do pěti metrů, ale vždy při prodloužení vedení nebo přidání dalších čidel se razantně snížila spolehlivost přenosu dat až k hranici naprostého selhání. DS18B20 byly v režimu normálního napájení – napájeny byly pomocí 5 V. Částečně pomohlo snížit pull-up odpor na sběrnici nejprve z 4,7 kΩ na 3,3 kΩ následně až na 2,2 kΩ.

Podobně byl otestován stejný návrh s jinými čidly postavený na I^2C, který také nedokázal zvládnout množství čidel a vzdálenost.

V kapitole 3 byla vybrána sběrnice RS-485. Sběrnice byla po domě rozvedena pomocí kabelů UTP (kategorie CAT5 a CAT6). Kabely byly rozváděné společně s trubkami od rekuperace. To poskytlo možnost schovat jednotky do sacích otvorů. Přes čidla u jednotky potom stále proudil vzduch z dané místnosti.

4.4 Implementace bootloaderu

Aby procesor vždy po resetu jako první spustil bootloader (a nikoli FW), je třeba nastavit správně pojistky. Konkrétně v EFUSE bit BOOTRST. Po nastavení tohoto bitu se vždy po resetu nastaví PC na adresu začátku BLS (Boot Loader Section). Ten se mění podle nastavení BOOTSZ. Tedy bootloader je první kód, který se spustí po resetu mikroprocesoru. Protože zaštiťuje nahrávání firmwaru do zařízení je třeba zajistit, aby v něm bylo co nejméně chyb a aby byl co nejvíce spolehlivý. Ideálně by se měl do zařízení nahrát pouze jednou – „v továrně“.

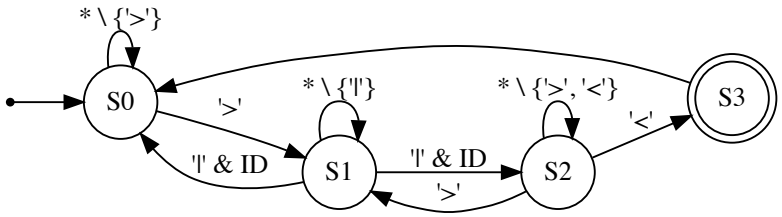
Bootloader je nahráván pomocí externího SPI programátoru. Takže pokud je třeba měnit bootloader musí se k němu člověk fyzicky dostat (problém u špatně dosažitelných jednotek). Proto je třeba zmenšit prostor pro chyby. Toho lze efektivně docílit omezením funkcionality pouze na nejvíce nutné funkce a zbytek dodělat v SW, který je pomocí bootloaderu nahráván.

Je výhodné aby bootloader byl nahráván pomocí stejné sběrnice, jako komu-

nikuje zbytek jednotek. Tím se zachová základ komunikace (například formát paketu), ale bootloader má jen své omezené příkazy.

Protokol obsahuje pouze příkazy popsané v tomto odstavci. První příkaz `f` slouží k zapsání dat do paměti flash. Jako první argument přijímá příkaz číslo stránky za ním následují data zakódovaná pomocí Base64. Všechny příkazy bootloaderu vrací kontrolní součet přijatého paketu (od „>“ po „<“ – včetně). Druhý příkaz `e` slouží k zápisu bloku do paměti EEPROM. Prvním argumentem je adresa, druhým data v Base64 (samo kódování určuje délku zapsaných dat).

Postup spuštění programu (firmware) pomocí bootloaderu je jednoduchý. V paměti EEPROM je na začátku (adresa `0x0000`) uložen C řetězec (zakončen NULL bytem). To je ID jednotky, které si bootloader načte a dál s ním pracuje. Na posledním bytu EEPROM je uložený flag, který označuje, zda má bootloader startovat FW nebo čekat na další instrukce. Pokud se flag shoduje s předem domluvenou hodnotou `0xAB` je hned po startu proveden skok do FW (aby byl HW předán SW ve stavu co nejvíce podobnému resetu). Pokud je hodnota flagu jiná, je to signál, že je třeba setrvat (případně nahrát nový FW).



Obrázek 4.4: Načítání vstupu v zavaděči

Na obrázku 4.4 je vidět zpracovávání vstupu bootloaderem. Následuje popis jednotlivých stavů:

- **S0** – Reprezentuje jednotku připravenou k načítání vstupu.
- **S1** – Bootloader právě načítá ID jednotky pro kterou je určen paket.
- **S2** – Právě je načítáno tělo příkazu.
- **S3** – Je přijat validní příkaz (je spuštěna konkrétní funkce).

U funkce na zapisování do paměti (`eeprom_write_block()`) je důležité dát si pozor na pořadí argumentů. To se liší od většiny standardních funkcí [20], kde se nejdříve píše cíl a pak teprve zdroj.

```
void eeprom_write_block(const void * __src, void * __dst, size_t
↪  __n)
void *memcpy(void* dest, const void* src, size_t len)
```

```
    if (eeprom_read_byte(EEPROM_FLAG_PTR) == EEPROM_FLAG_VAL)
    {
        // nastavení adresy vektorů přerušení na 0x0000
        MCUCR = (1 << IVCE);
        MCUCR = 0;
        __asm__ volatile ("jmp 0000\n"); // skok na FW
    }
```

Kód 4.1: Spuštění firmwaru – tento kód je umístěn na začátku bootloaderu a rozhoduje o tom, zda má spustit FW, nebo zůstat v bootloaderu připraven k nahrávání FW.

```
void write_flash_page(uint16_t page, uint8_t *buf)
{
    // přepočítání čísla stránky na její adresu
    uint16_t address = page * SPM_PAGESIZE;
    uint8_t sreg = SREG; // uložení stavu přerušení
    cli();               // vypnutí přerušení

    boot_page_erase_safe(address); // vymazání stránky

    for (uint16_t i = 0; i < SPM_PAGESIZE; i += 2)
    {
        uint16_t w = *buf++; // příprava slova na zapsání
        w += (*buf++) << 8;

        boot_page_fill_safe(address + i, w); // uložení slova
        ↪  do bufferu
    }

    boot_page_write_safe(address); // zapsání bufferu do
    ↪  stránky
    boot_spm_busy_wait();   // čekání na konec operace
    boot_rww_enable_safe(); // obnovení RWW sekce (potřeba pro
    ↪  skok)
    SREG = sreg;            // obnovení přerušení
}
```

Kód 4.2: AVR – zápis stránky – Kód převzat a upraven z datasheetu [5]. Slouží k nahrání obsahu bufferu do paměti flash na stránku specifikovanou argumentem.

4.4.1 Postup nahrávání nového firmwaru

Pokud je třeba nahrát software do nové desky je třeba postupovat následujícím způsobem:

1. Nahrát pomocí externího SPI programátoru (např. AVR Dragon) bootloader do paměti flash.
2. Stejným programátorem nahrát jméno jednotky do paměti EEPROM.
3. Nahrát pomocí bootloaderu firmware.
4. Spustit aplikaci (ta zapíše do EEPROM flag o nespouštění bootloaderu).

Celková velikost implementovaného bootloaderu:

```
$ avr-size --format=Berkeley --mcu=atmega328p ./main.elf
   text    data     bss     dec     hex filename
   1634       8     562    2204     89c main.elf
```

4.5 Firmware

Firmware zabírá lehce nad polovinu dostupné paměti programu, to dává dostatek prostoru pro implementaci další případné funkcionality. Statické struktury v paměti SRAM zabírají 42% celkové kapacity. To znamená, že pro fungování programu zbývá více než dost místa.

```
$ avr-size --format=avr --mcu=atmega328p ./firmware.elf
Device: atmega328p

Program:   18196 bytes (55.5% Full)
(.text + .data + .bootloader)

Data:        865 bytes (42.2% Full)
(.data + .bss + .noinit)
```

4.5.1 Postup sestavení

První část překladu probíhá stejně jako u běžné C aplikace. Jen je třeba specifikovat druh procesoru a frekvenci. Díky tomu může v avr-gcc fungovat funkce typu `_delay_ms()`. Po prvním překladu vznikne ELF soubor. Ten obsahuje kód, data a případně i ladící symboly. Tím by běžný proces sestavení (například na běžném Linuxu) končil [21].

Dále následuje vykopírování sekcí z ELF souboru do dalších dvou souborů v formátu IHEX. Ten je výhodný oproti obyčejné surové binárce, protože obsahuje informace o adresách. Sekce `.text` (spustitelný kód) a `.data` (proměnné, …) jsou vykopírovány do souboru `firmware.hex`, a sekce která pojímá obsah EEPROM je překopírována do souboru `firmware.eep`. Vzniklé IHEX soubory je možné nahrát pomocí Avrdude. Výsledný ELF soubor lze použít k ladění.

Během sestavování všech částí je důležité mít zapnuté optimalizace. Pokud je vynuceno jejich vypnutí (`-O0`), bude v programu docházet k chybám s časováním (avr-libc s nimi počítá). Optimalizace je ovšem vhodné vypnout – v rámci hledání chyb (dojde k přesnější specifikaci chyby).

Nejprve je nutné sestavit Arduino core. Ve verzi avr-gcc pro MacOS byla naneštěstí chyba, která znemožňovala použití avr-ar a vytvoření „statické knihovny“ se zapnutým LTO. Proto je FW linkován přímo proti objektovým souborům Arduino core. Použití LTO přispělo razantně ke snížení velikosti kódu (úspora téměř 20%).

4.5.2 Funkce programu

FW musí implementovat všechny funkce popsané v návrhu protokolu 3.6. Funkce s předponou `f_` typicky v mém programu označují funkce které budou dostupné jako příkaz. Všechny takové musejí mít stejnou signaturu: `void (*)(int argc, char **argv)`. V FW je statická tabulka těchto příkazů které jdou spouštět. Byla vybrána jedna zajímavá funkce na ukázku.

```
float internal_temperature(void)
{
    ADMUX = (_BV(REFS1) | _BV(REFS0) | _BV(MUX3)); // vybrání
    ↪ kanálu
    ADCSRA |= _BV(ADEN); // zapnout ADC
    delay(20);           // počkat 20 ms (po zapnutí ADC)
    ADCSRA |= _BV(ADSC); // zahájit konverzi
    while (bit_is_set(ADCSRA, ADSC)); // počkat na dokončení
    ↪ konverze
    return (ADCW - 324.31 ) / 1.22; // vrátit teplotu
}

void f_temperature(int argc, char **argv)
{
    rs485_start(NULL);  // odeslání hlavičky paketu
    Serial.print(internal_temperature()); // odeslání teploty
    rs485_end();        // konec odesílání paketu
}
```

Kód 4.3: Měření vnitřní teploty procesoru ATmega328

4.5.3 Ovládání klimatizace

Jednou z HW funkcí procesoru ATmega328 je PWM. Takže procesor dokáže velice jednoduše provádět modulaci signálu – není tedy nutné externí zařízení. Pomocí odběrové analýzy byla zjištěna frekvence kterou je výsledný IR signál modulován (obrázek 2.7). Následná implementace již byla lehká. V paměti je

umístěn buffer do kterého se zpracují (podle příkazu) data. Buffer je potom postupně odeslán pomocí IR diody do klimatizace.

4.6 Server

Každých 15 sekund je provedeno načtení aktuálních hodnot ze senzorů. V pauzách mezi měřeními se kontroluje vstup od uživatele (nastavení klimatizace a rekuperace). Bylo by vhodné využít funkcionalitu poskytovanou databází PostgreSQL – `LISTEN/NOTIFY`. Bohužel SQLAlchemy, které je používáno pro komunikaci s databází s ní neumí pracovat.

4.6.1 Popis konfiguračního souboru

Aby bylo možné změnit parametry programu, byl vytvořen souhrnný konfigurační soubor. Je v něm naznačena strukturou topologie sítě a podle něj server vybírá, kterých jednotek se má ptát na jaké údaje. Pro jednoduchost je použit přímo kód v jazyku Python.

```
# připojení do databáze (pro SQLAlchemy)
DATABASE = 'postgresql://postgres:@127.0.0.1/test'
SERIAL = '/dev/ttyUSB0' # RS-485

# seznam všech jednotek
nodes = {
    "koupelna":
    {
        "bme280": True # značí že jednotka obsahuje BME280
    },
    "kuchyn": # tato jednotka neobsahuje BME280
    {}
}

# seznam čidel DS18B20 pro přístup pomocí adresy
ds18b20 = {
    # označení v DB       adresa čidla DS18B20
    "stena_jih":          "0x28FF93C4611604AF",
    "bojler_05":          "0x28FF1D3CA416034E",
}
```

Kód 4.4: Ukázka konfiguračního souboru pro server

4.6.2 Vzdálené logování

Protože systém je postaven nad Debianem (linuxová distribuce se systemd), lze využít pro logování právě systemd. To standardně ukládá výstup „servisu“ do logu. Přistupovat k logům tak lze pomocí standardních systémových nástrojů například takto: `systemctl -u smarthome`. Při vývoji aplikace je

vhodné, když na sebe systém upozorní v případě problémů a proto byl vytvořen logovací systém, který záznamy z logu automaticky odesílá prostřednictvím komunikační aplikace Telegram. Díky tomu mohou být obyvatelé domu včas upozorněni pokud je v domě nějaký problém. Například teploty mimo stanovený rozsah.

```
journalctl -f -o cat --since=now -u smarthome.service |
↪  sendlogs.py
```

Kód 4.5: Přeposílání logů aplikací telegram

4.7 Webové rozhraní

Obrázky finální implementace jsou součástí uživatelské příručky (příloha B). První testovací verze webového rozhraní byla spojením bash skriptů, HTML a JavaScriptu. I když to spolehlivě fungovalo, o nějaké bezpečnosti se nedalo mluvit. Samozřejmě čitelnost takového kódu byl také problém. Po probrání se různými alternativami bylo rozhodnuto pro použití Flasku na Apache2.

V prvním návrhu se nepočítalo s přístupem k systému z internetu. Ovládání chytrého domu by bylo dostupné pouze z lokální sítě a přihlašování uživatelů by nebylo třeba řešit. Dálkový přístup k takto navrženému systému by byl zprostředkován pomocí VPN. To by zajišťovalo celkem dobrou bezpečnost. Takto navržený systém by ale rapidně snížil kompatibilitu – platformy bez možnosti připojení k VPN by nemohly být použity pro vzdálený přístup.

Nakonec bylo implementováno jednoduché přihlašování pomocí šifrovaných cookies – o tuto funkcionalitu se stará Flask sám.

4.7.1 Konfigurace Apache2

Debian používá jiný styl konfigurace než ostatní Linuxové distribuce. Veškerá konfigurace webového serveru se nachází v adresáři `/etc/apache2/`. Pro pomoc s nastavením existuje několik nástrojů – například `a2ensite` a `a2dissite`.

Je výhodné v konfiguraci serveru nastavit možnosti: `ServerSignature off` a `ServerTokens Prod`. Tím přestane server dávat příliš mnoho (zbytečných) informací, které by mohly usnadnit případnému útočníkovi útok.

Protože na serveru běží více tzv. „virtual hosts“ a je žádané, aby veškeré požadavky na server které nejsou určené jinému virtual hostovi, vyřizoval chytrý dům, je třeba zajistit, aby byl ze zapnutých stránek načtený jako první. Toho lze docílit jednoduchým přejmenováním souborů v konfiguraci (jsou načítány v abecedním pořadí).

Dále je vhodné zapnout na serveru kompresi. Tím sice dojde k mírnému zvýšení nároků procesoru, ovšem dojde ke snížení datové náročnosti. To se hodí

zejména pokud je člověk například na dovolené a má omezený tarif na přenos dat.

4.7.2 Vykreslování grafů

Na vykreslení grafů za posledních 24 hodin je využit nástroj Gnuplot. Ten je nastaven, aby nevracel obrázek, ale text – Gnuplot Canvas (JavaScript). To společně s kompresí zapnutou u webového serveru snižuje náročnost na přenos dat (na rozdíl od bitmapových obrázků u kterých je kompresní poměr dost špatný).

4.7.3 Let's Encrypt a Certbot

Let's Encrypt je svobodná certifikační autorita, která zdarma poskytuje certifikáty [22]. Vše probíhá téměř automaticky. Není tedy již nutné složitě získávat certifikáty osobně. K získání certifikátů slouží snadno použitelný nástroj: `Certbot`. Podmínkou je, že je třeba mít přístup na server na který ukazují DNS záznamy.

```
certbot certonly --standalone -d example.com -d www.example.com
```

Kód 4.6: Získání/obnovení certifikátů

Protože je tato úloha takto snadná, lze jí jednoduše automatizovat. Například pomocí systémového nástroje `cron`. V něm lze nastavit automatické obnovování certifikátů po jejich vypršení a tím ještě více snížit administrativní nároky.

Kapitola 5

Testování

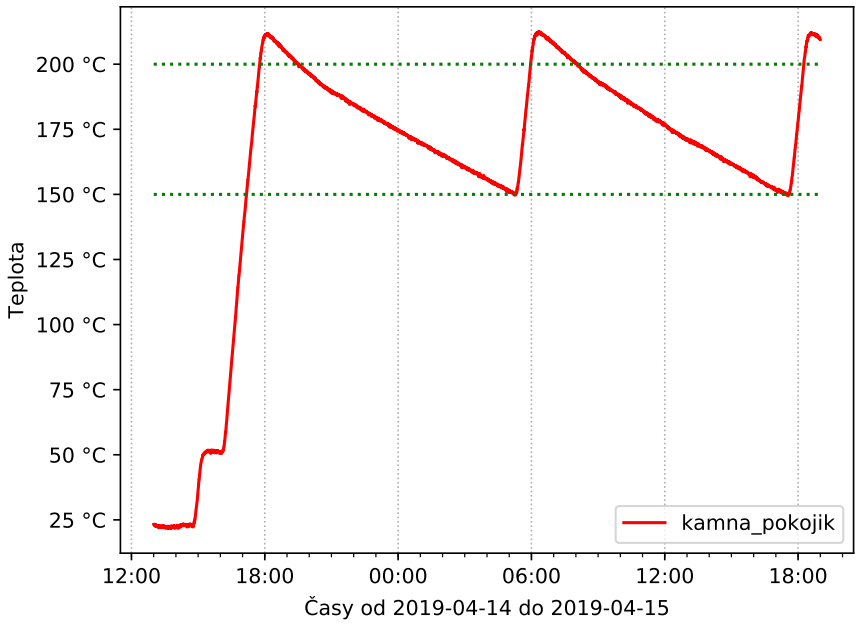
V následující kapitole se nachází popis testování implementovaného systému a ukázky získaných dat. Fotografie implementace systému na skutečném domě se nacházejí v samostatné příloze C. Testování základního systému probíhalo poslední 2 roky. Za tuto dobu se systém ukázal být vysoce spolehlivý. Trpěl pouze dvěma druhy chyb. Problém s komunikací po sběrnici (rozebíráno podrobněji v sekci 5.6) a problém s načítáním hodnot z čidel DS18B20 (viz. 5.5).

5.1 Měření teploty v kamnech

Protože se v budoucnu budou osazovat elektronikou všechny kamna, bylo by přínosné zjistit jaké v kamnech panují podmínky a kde by bylo nejvhodnější umístění jednotky tak, aby teplota nepřekračovala její provozní maximum. Do kamen byla do spodní části pod řídící kabel umístěna jedna jednotka s čtyřmi čidly. Dvě DS18B20, jedno měřící teplotu u řídící jednotky, druhé teplotu u ovládacího panelu. Čidlo MAX6675 bylo umístěno za cihlu u topných těles (zde je očekávána nejvyšší teplota, která bude později použita k řízení kamen). Poslední teplotní čidlo (GY906) mělo měřit teplotu železného plátu na kterém jsou umístěny cihly. Zde se předpokládala také vysoká teplota, proto bylo použito infračervené čidlo. Bohužel lesklý plech odrážel infračervené záření z okolí a čidlo tedy vracelo ambientní teplotu. Měření tímto čidlem bylo tedy neprůkazné.

Na obrázku 5.1 je vidět graf z akumulačních kamen (čidlo MAX6675). Systém chytrého domu byl nastaven tak, že při teplotě nad 200 °C vypnul topná tělesa a při teplotě pod 150 °C je opět zapnul. Hranice jsou symbolizovány zelenými tečkovanými čárami. Na obrázku je dále vidět dočasné vypnutí kamen během prvního nahřívání – jsou nastavena na nahřívání pouze pokud je dostupný noční proud. Jak je vidět, nahřívání probíhalo kolem tří hodin a to včetně výpadku kvůli nočnímu proudu. Pro potřeby návrhu řídícího algoritmu pro

ovládání kamen v závislosti na teplotě v místnosti by šlo odhadnout nahřívání pomocí lineární funkce.



Obrázek 5.1: Nahřívání a udržování teploty kamen

Po otestování nahřívání kamen bylo také testováno chladnutí. Jak je vidět na grafu 5.2, tak chladnutí kamen i z relativně nízké teploty jako je 200 °C může trvat i několik dní. Při nízkých teplotách venku během zimy, naměřená teplota uvnitř kamen přesahovala až 430 °C. Funkce pro modelování chladnutí kamen byla odhadnuta jako:

$$f(x) = a^{x+b} + c \tag{5.1}$$

Jednotlivé parametry byly odhadnuty pomocí Pythonu – konkrétně funkce `scipy.optimize.leastsq(...)`. Ta používá na metodu nejmenších čtverců [23]. Výsledné vypočítané hodnoty:

$$a = 0{,}999989929$$

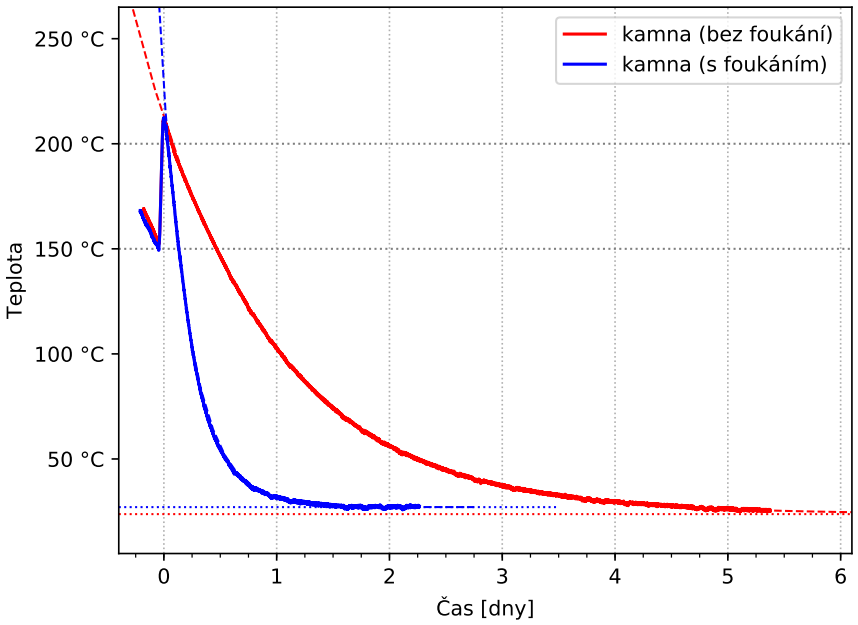
$$b = -520339{,}34$$

$$c = 23{,}4244671$$

A jak je vidět na grafu 5.2, tak odhadnutá funkce k modelování postačuje dostatečně. Dále platí:

$$\lim_{x \to \infty} f(x) = c \approx 23{,}4 \tag{5.2}$$

Tato limita přibližně odpovídá pokojové teplotě. U testování kamen s vyfukováním je skrz kamna prohánět vzduch pomocí zabudovaného ventilátoru. Jak je na grafu vidět, tak kamna chladnou daleko rychleji a limitně se blíží k jiné teplotě. To je způsobeno tím, že vyfukováním z kamen se zvedla teplota v místnosti. Odhad vyšel přibližně 27,75 °C, což je až překvapivě přesné u čidla s rozlišením 0,25 °C a přesností ±3 °C [16].



Obrázek 5.2: Chladnutí kamen – porovnání

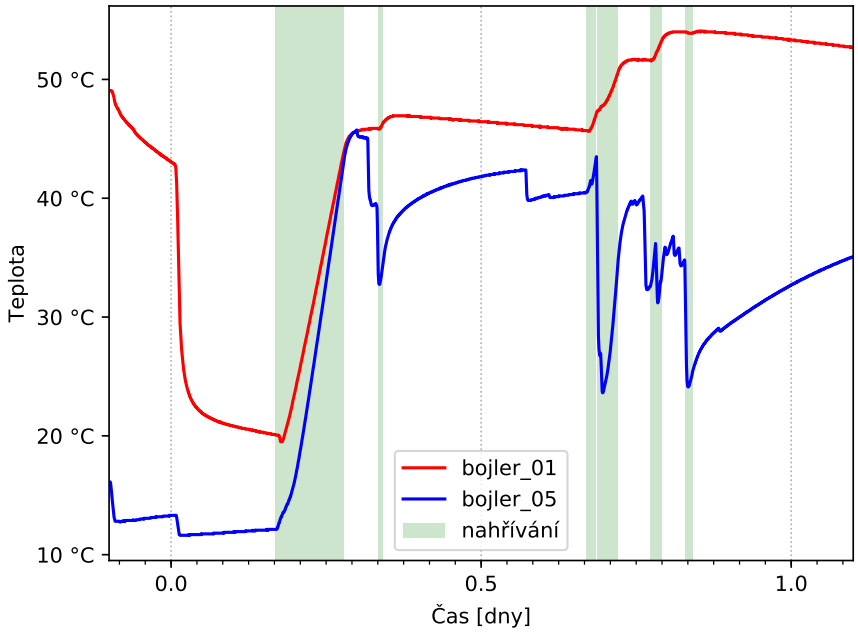
5.2 Využitelné informace z bojleru

S bojlerem v domě jsou také jisté problémy. Brzy ráno dochází k nahřátí a k večeru, kdy je největší spotřeba teplé vody – bojler již nemá dostatečně vysokou teplotu, aby teplá voda pokryla spotřebu. Tedy taková jednoduchá věc jako změna času – kdy se zapne nahřívání – dojde k obrovskému zlepšení situace. Samozřejmě bojler tuto funkci nemá zabudovanou. Díky snímání přívodu elektřiny do bojleru je v systému zaznamenáváno zapínání a vypínání nahřívání. Tato data za jeden den ukazuje tabulka 5.1.

Začátek	Trvání
2019-04-27 04:02:18	2:39:10
2019-04-27 08:00:08	0:10:42
2019-04-27 16:03:57	0:20:56
2019-04-27 16:30:01	0:47:28
2019-04-27 18:32:00	0:26:33
2019-04-27 19:53:31	0:17:14

Tabulka 5.1: Doby nahřívání bojleru

Díky čidlům umístěným na těle bojleru lze snímat i přibližnou teplotu v jednotlivých „vrstvách“. Na obrázku 5.3 je vidět vliv nahřívání na teplotu v bojleru při běžném používání. Je zobrazené nejvrchnější (to s vyšší teplotou) a nejspodnější čidlo. Pro vykreslení nahřívání jsou použity hodnoty z tabulky 5.1.



Obrázek 5.3: Nahřívání a teplota bojleru

V budoucnu je tedy v plánu optimalizovat nahřívání tak, aby v době největší spotřeby teplé vody byl bojler vždy nahřátý na maximální kapacitu.

5.3 Funkčnost bootloaderu

Zavaděč byl testován na zkušební sestavě opakovaným nahráváním firmwaru a měněním ID jednotky. V následující ukázce jsou detailně rozepsány jednotlivé kroky testu.

1. `$>test|<` – Zkouška spojení (prázdný paket), jednotka by měla odpovědět příslušným kontrolním součtem.
2. `flash.py /dev/ttyUSB0 ./firmware.hex 'test'`
   Použití nahrávacího skriptu, který načte vstupní soubor, převede ho z formátu IHEX do čistého binárního souboru a postupně ho po stránkách nahraje do jednotky.
3. `$>test|r<` – Tento příkaz zajistí spuštění firmwaru. Během něj FW automaticky nastaví flag v EEPROM, aby nedošlo k opětovnému spuštění bootloaderu.
4. `$>test|ping<` – Otestování spojení, jednotka by měla být v FW a měla by odpovědět standardním paketem obsahujícím `pong`.
5. `$>test|bootloader<` – Přepnutí z FW do bootloaderu. Dojde ke smazání flagu v EEPROM a resetu.
6. `$>test|<` – Otestování spojení, zda je jednotka ve správném režimu.
7. `echo -en 'obyvak\0' | base64@`
   Protože FW přijímá data zakódovaná v Base64, je nejprve třeba převést jméno jednotky do tohoto formátu.
8. `$>test|e b2J5dmFrAA==<` – Přepsání ID v EEPROM na „obyvak“.
9. `$>test|r<` – Až do resetu nebo spuštění FW platí staré ID.
10. `$>obyvak|ping<` – Otestování spojení s jednotkou s novým jménem.

5.4 Problémy s firmwarem

Během vývoje FW se vyskytlo mnoho problémů, většinou snadno odstranitelných. Jedna z více zákeřných chyb byla nalezena během testování nové verze FW. Určité jednotky se po čase resetovali, nebo to tak alespoň vypadalo. Čas události se lišil mezi jednotkami a i v rámci jedné jednotky, od hodiny po téměř den. Problém, který se vyskytuje takto náhodně se velice špatně ladí. Pokus s laděním byl prováděn pár hodin, ale během doby kdy byla jednotka připojena k debugeru se problém nevyskytl. Nejprve bylo třeba vyšetřit zdroj resetu. První nápad byly výkyvy v napájení a následný restart přes BOD. Testovaná jednotka byla připojena ke stabilnímu a otestovanému zdroji, kde problémy ustaly. Po vrácení na své místo jednotky opět začaly dělat problémy. Po kontrole napájení osciloskopem nebyly odhaleny, alespoň během měření, žádné anomálie. Žádné z měření bohužel neproběhlo během výskytu chyby.

Teorii s BOD se tedy nepodařilo zprvu vyvrátit a také chybou trpěly pouze jednotky s novým FW.

Reset lze také docílit pomocí Watchdogu. Proto byl přidán příkaz, který vrátil zdroj resetu 2.5.2 (po odstranění chyby byl tento příkaz z FW odstraněn). Jednotky po chybě vracely stejnou hodnotu zdroje resetu jako během prvního spuštění. To by znamenalo, že nebyl proveden „čistý reset“. Chováním to hodně připomínalo skok do FW z bootloaderu (4.1). Nakonec byl odhalen únik paměti v části FW starající se o ovládání klimatizace. Únik byl zpozorován pomocí příkazu `mem_info` (blíže popsán na straně 30). Protože v místě výskytu chyby nebyla dynamická paměť třeba, byl problém vyřešen pomocí statické alokace struktury obsahující IR paket.

5.5 Spolehlivost čidla DS18B20

Sběrnice OneWire s čidly DS18B20 se ukázala být značně nespolehlivá. Za rok testovacího provozu způsobila nejvíce problémů. Problém se nepravidelně objevoval a mizel. Naprosto bez chyb čidla fungovala až 3 měsíce v kuse. Následně se začaly objevovat občasné chyby (1–3 denně). V nejhorším případě selhávala až 1/5 měření.

Problém spočívá v tom, že čidlo neodpoví, nebo odpoví špatně a informace se nedostane zpět do jednotky (a do serveru už vůbec).

Problém se povedlo zlepšit zkvalitněním drátových spojů mezi jednotkami a čidly. Tím se snížil výskyt závažnější formy chyby (ztráta v 20% měření). Ovšem vzhledem k množství prováděných měření by nebyl problém i kdyby se ztratila polovina údajů.

Další provedené pokusy o zlepšení problému: kvalitnější a stabilnější napájení (nemělo prakticky žádný vliv na množství chyb). Dále snížení pull-up odporu na OneWire sběrnici – snížilo množství chyb, ale neeliminovalo úplně (také zvýšilo spotřebu celého systému a proto byla vrácena původní hodnota rezistoru).

5.6 Chyby na sběrnici RS-485

Na sběrnici se objevil krátký impulz, který byl nesprávně identifikován jako „start bit“ přenosu (viz. 2.3.1). To způsobilo přijetí bytu, který nebyl nikdy odeslán. Není to závažný problém, protože se tato chyba vyskytovala na sběrnici pouze pokud nikdo zrovna nevysílal. Nikdy za dobu provozu nebyl zaznamenán vadný bit během přenosu, který by nějak poškodil paket.

Závěr

Cílem práce bylo implementovat chytrý dům. Aby bylo možné provést implementaci, bylo třeba provést analýzu a navrhnout systém chytrého domu. Nejdelší část celé práce je rozbor různých technologií v analýze. To je způsobeno velmi širokým rozsahem zaměření práce. Bylo třeba zabývat se všemi vrstvami vývoje tzn. od hardwaru, bootloaderu a FW, přes back-end až po vytvoření webové aplikace (front-end).

Výstupem implementační části práce je funkční a nyní již uživateli používané webové rozhraní. Dále kompletní software k jednotlivým jednotkám pro systém chytrého domu (bootloader a firmware). Web zobrazuje hodnoty teplot v místnostech a vykresluje grafy teploty a vlhkosti. Dále rozhraní umožňuje ovládání obou dvou klimatizačních jednotek v domě. Hardware klimatizací nebyl nijak upraven.

První pokusy implementace chytrého domu probíhaly již před více než třemi lety, proto bylo možné provést dlouhodobé testování hardwarových komponent (hlavně čidel). Software byl průběžně několikrát přepsán.

Vývoj práce se potýkal s řadou problémů. Za zmínku stojí například problémy se spolehlivostí čidla DS18B20 a sběrnicí OneWire. Ty nebyly úplně eliminovány, ovšem byly omezeny na přijatelnou úroveň pro použitelnost v běžných podmínkách. Díky dlouhodobému testování byla odhalena nespolehlivost čidel DHT22, která začala po roce používání selhávat. Nová čidla BME280 se naopak ukázala být velmi spolehlivá. Za tyto cenné zkušenosti s ne/spolehlivostí elektronických komponent jsem velmi vděčný, neboť mi umožnily lépe proniknout do oboru spolehlivosti a díky tomu jsem se naučil klást větší důraz na výběr kvalitnějších a spolehlivějších komponent.

Během vývoje chytrého domu jsem šel cestou open source softwaru. S databází, webovým serverem, operačním systémem a kompilátorem pro AVR nebyly žádné problémy, dokonce vývoj v některých fázích usnadnily. Největším

kamenem úrazu se ukázaly být nástroje pro ladění mikroprocesoru. Ty byly velmi zastaralé, většinou je ani nešlo zprovoznit. Většina open source nástrojů byla špičková, ale pro vývoj embedded zařízení je určitě lepší jít proprietární cestou (například Atmel Studio).

Práce se bude v budoucnu jistě rozšiřovat, je potřeba plně automatizovat ovládání topení a implementovat precizní regulaci teploty v jednotlivých místnostech. Celková automatizace a integrace všech prvků na regulaci teploty by také jistě byla velkým přínosem.

Práce bohužel dosti přerostla přes zadání a už v ní nezbyl prostor pro popis velkého množství zajímavých částí. Systém nakonec obsahoval více než 10 různých jednotek. Mimo zadání je v práci rozepsáno například pár detailů ohledně přípravy budoucí implementace automatické regulace topení na které by jistě šlo navázat nějakou další prací. Dále proběhlo zapojení senzorů na plášť bojleru, což v budoucnu umožní další pokročilejší automatizaci.

Tato práce se množstvím funkcionality nemůže rovnat firmám s mnoha zaměstnanci, které jsou na trhu desítky let, ovšem základní funkcionalitou a hlavně cenou jistě konkurovat může. Pokud by práce měla směřovat někam dále, určitě se zaměřím na bezpečnost systému, která byla prací (až na nezbytné zabezpečení) opomíjena.

Navržený systém je v domě pravidelně využíván obyvateli a hlavně majitelem domu. Protože se částečně také podílel na vývoji a mohl ovlivnit jednotlivé části, aby vyhovovaly hlavně jemu je s prací odvedenou na domě spokojen. Osobně jako občasný obyvatel jsem také spokojen a mám z odvedené práce velmi dobrý pocit, hlavně z praktického využití.

Literatura

1. JONES, David L. *EEVblog #978 - Keysight 1000X Hacking* [online]. 2019 [cit. 2019-05-11]. Dostupné z: `http://www.eevblog.com/forum/blog/eevblog-978-keysight-1000x-hacking/`.

2. *I2C-bus specification and user manual* [online]. 2014. Verze 6 [cit. 2019-05-12]. Dostupné z: `https://www.nxp.com/docs/en/user-guide/UM10204.pdf`.

3. *Systém sériových sběrnic pro použití v autech* [online] [cit. 2019-05-11]. Dostupné z: `http://sevcikpeta.wz.cz/downloads/24_CAN%20Bus.pdf`.

4. *RS-485 Reference Guide* [online]. 2014 [cit. 2019-05-12]. Dostupné z: `http://www.ti.com/lit/sg/slyt484a/slyt484a.pdf`.

5. *ATmega328P DATASHEET* [online]. 2015. 7810D–AVR [cit. 2019-05-12]. Dostupné z: `http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf`.

6. *AVR109: Self Programming* [online]. 2004. 1644G–AVR [cit. 2019-05-12]. Dostupné z: `http://ww1.microchip.com/downloads/en/AppNotes/doc1644.pdf`.

7. SCHICK, Brad. *AVR Bootloader FAQ* [online]. Ed. LAWSON, Cliff. 2009 [cit. 2019-05-12]. Dostupné z: `https://www.avrfreaks.net/sites/default/files/bootloader_faq.pdf`.

8. VODA, Zbyšek. *Průvodce světem Arduina*. Bučovice: Martin Stříž, 2015. ISBN 978-80-87106-90-7.

9. *Language Reference* [online] [cit. 2019-04-10]. Dostupné z: `https://www.arduino.cc/reference/en/`.

10. *DS18B20* [online]. 2018 [cit. 2019-05-12]. Dostupné z: https://datasheets.maximintegrated.com/en/ds/DS18B20.pdf. Rev.: 19-7487.

11. *BME280* [online]. 2018. Verze 1.6 [cit. 2019-05-12]. Dostupné z: https://ae-bst.resource.bosch.com/media/_tech/media/datasheets/BST-BME280-DS002.pdf.

12. SMITH, Robert J. *Wide range of Hygrometers: …* [online]. 2018 [cit. 2019-04-18]. Dostupné z: http://www.kandrsmith.org/RJS/Misc/Hygrometers/calib_many.html.

13. *AM2302 Product Manual* [online] [cit. 2019-05-05]. Dostupné z: http://akizukidenshi.com/download/ds/aosong/AM2302.pdf.

14. *MLX90614 family* [online]. 2017. Verze 3901090614 [cit. 2019-05-12]. Dostupné z: https://www.melexis.com/-/media/files/documents/datasheets/mlx90614-datasheet-melexis.pdf.

15. *Comparing the I2C Bus to the SMBus* [online] [cit. 2019-04-23]. Dostupné z: https://www.maximintegrated.com/en/app-notes/index.mvp/id/476.

16. *MAX6675* [online]. 2014. 19-2235 [cit. 2019-05-05]. Dostupné z: https://datasheets.maximintegrated.com/en/ds/MAX6675.pdf.

17. *Reverse Engineering Air Conditioner IR Remote Control Protocol* [online] [cit. 2019-05-05]. Dostupné z: https://www.instructables.com/id/Reverse-engineering-of-an-Air-Conditioning-control/.

18. *2N700* [online]. 2013. E041009 [cit. 2019-04-20]. Dostupné z: http://ww1.microchip.com/downloads/en/DeviceDoc/2N7000.pdf.

19. JOSEFSSON, Simon. *RFC4648* [online]. 2006. Dostupné také z: https://tools.ietf.org/pdf/rfc4648.pdf.

20. *AVR Libc Reference Manual* [online] [cit. 2019-04-19]. Dostupné z: https://www.microchip.com/webdoc/AVRLibcReferenceManual/.

21. PATTERSON, David A.; HENNESSY, John L. *Computer organization and design: the hardware/software interface*. 4. vyd. 2012. ISBN 978-0-12-374750-1.

22. *About Let's Encrypt* [online] [cit. 2019-04-19]. Dostupné z: https://letsencrypt.org/about/.

23. KLOUDA, Karel. *Metoda nejmenších čtverců: řešení rovnic, které nemají řešení* [online] [cit. 2019-05-12]. Dostupné z: https://marast.fit.cvut.cz/cs/blog_posts/19.

Dodatek A

Seznam použitých zkratek

BOD Brown-Out Detector

DB Database

EBNF Extended Backus-Naur form

FW Firmware

GPIO General-purpose input/output

HW Hardware

ICSP In Circuit Serial Programming

ID Identifier

LTO Link Time Optimalization

MPCM Multi-Processor Communication Mode

NRWW No-read-while-write

PWM Pulse Width Modulation

RWW Read-while-write

SRAM Static Random Access Memory

SW Software

UART Universal Asynchronous Receiver and Transmitter

UPS Uninterruptible Power Supply

Dodatek B

Uživatelská příručka

V této příručce jsou shrnuty základní postupy pro práci s mnou navrženým chytrým domem.

B.1 Instalace systému chytrého domu

SmartHome je v otázkách požadavků na systém celkem flexibilní. Není problém provést instalaci na libovolném stroji s potřebnými knihovnami (Linux, MacOS i Windows). Také umožňuje oddělit od sebe jednotlivé části. Například backendová část (starající se o sériovou komunikaci) může být na Raspberry PI – propojená s ostatními částmi pomocí internetu (PostgreSQL) a vše bude fungovat. Ovšem systém byl navržen, vyvíjen a hlavně testován pro operační systém Debian (Linux x86-64). Také počítal (kvůli databázi) s jedním výkonnějším strojem. Pro Debian byl vytvořen instalační skript „INSTALL.sh“, který stáhne potřebné závislosti a nainstaluje všechny části na svá místa (popis skriptu není součástí práce).

B.2 Přihlášení do systému

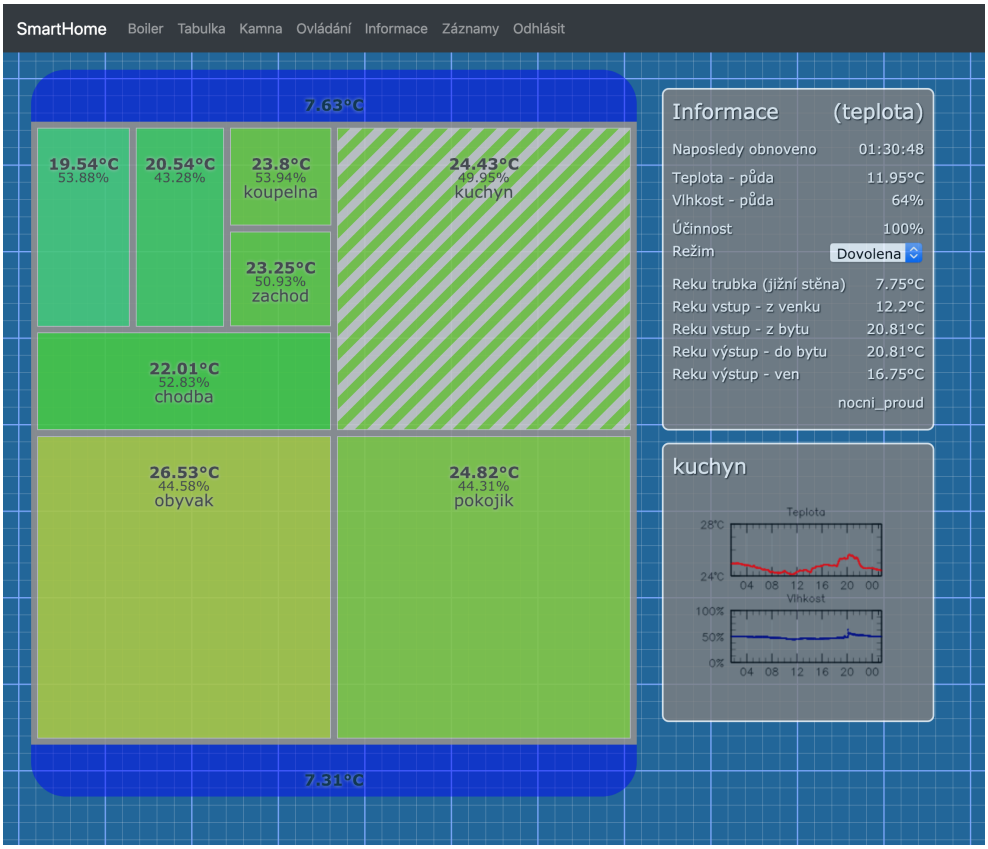
Přístup do systému je chráněn jménem a heslem (obrázek B.1). Je tedy nutné se před jeho používáním přihlásit.

B.3 Hlavní obrazovka

Na obrázku B.2 je zobrazena hlavní obrazovka chytrého domu. Na ní je vyobrazen přibližný půdorys domu. Při kliknutí na místnost se vyšrafuje a v pravém spodním okně se zobrazí graf teplot a vlhkostí (pokud jednotka podporuje měření vlhkosti) pro vybranou místnost za posledních 24 hodin.

Na hlavní obrazovku je možné se vždy vrátit pomocí kliknutí na „SmartHome“ v navigačním řádku.

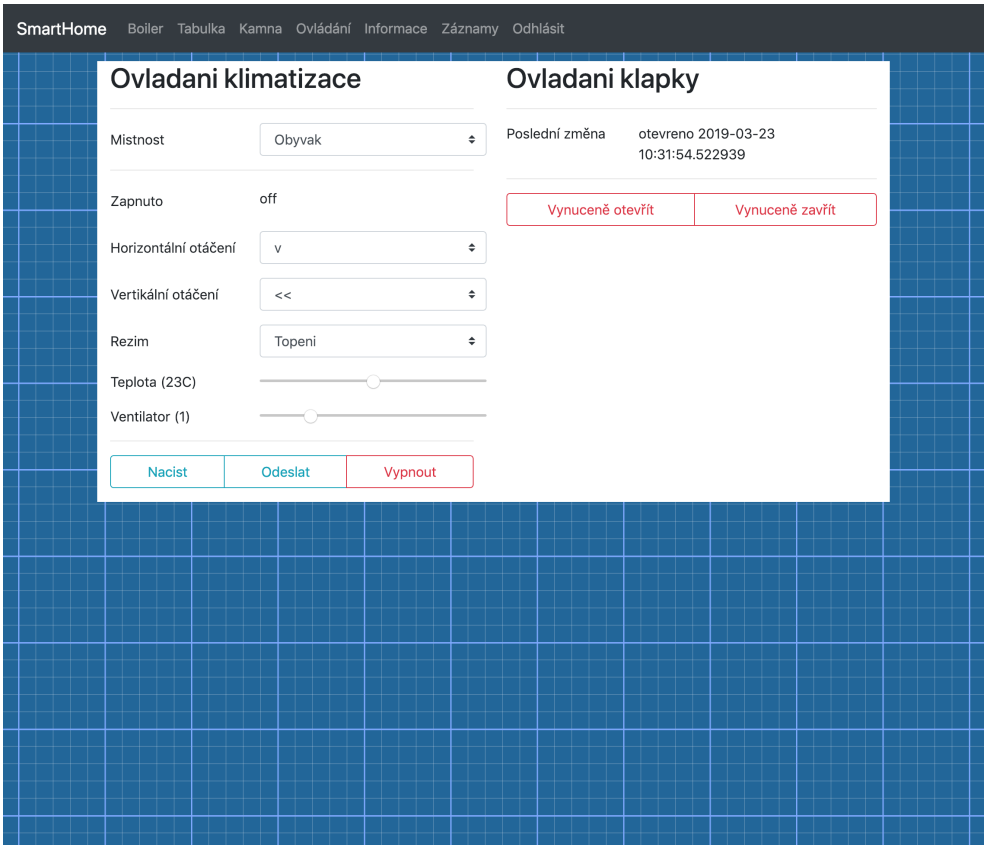
Obrázek B.1: Přihlašovací obrazovka



Obrázek B.2: Hlavní obrazovka webu

B.4 Ovládání chytrého domu

Klimatizace je ovládána pomocí položky „Ovládání“. Jak je vidět na obrázku B.3, nejprve je třeba vybrat místnost, pak lze nastavit parametry a odeslat povel do klimatizace. Tlačítko „Načíst“ slouží k obnovení hodnot pro danou místnost na poslední konfiguraci z databáze. Hodnoty se neobnovují samy, protože případné přenastavení jiným uživatelem by resetovalo aktuální nastavení.

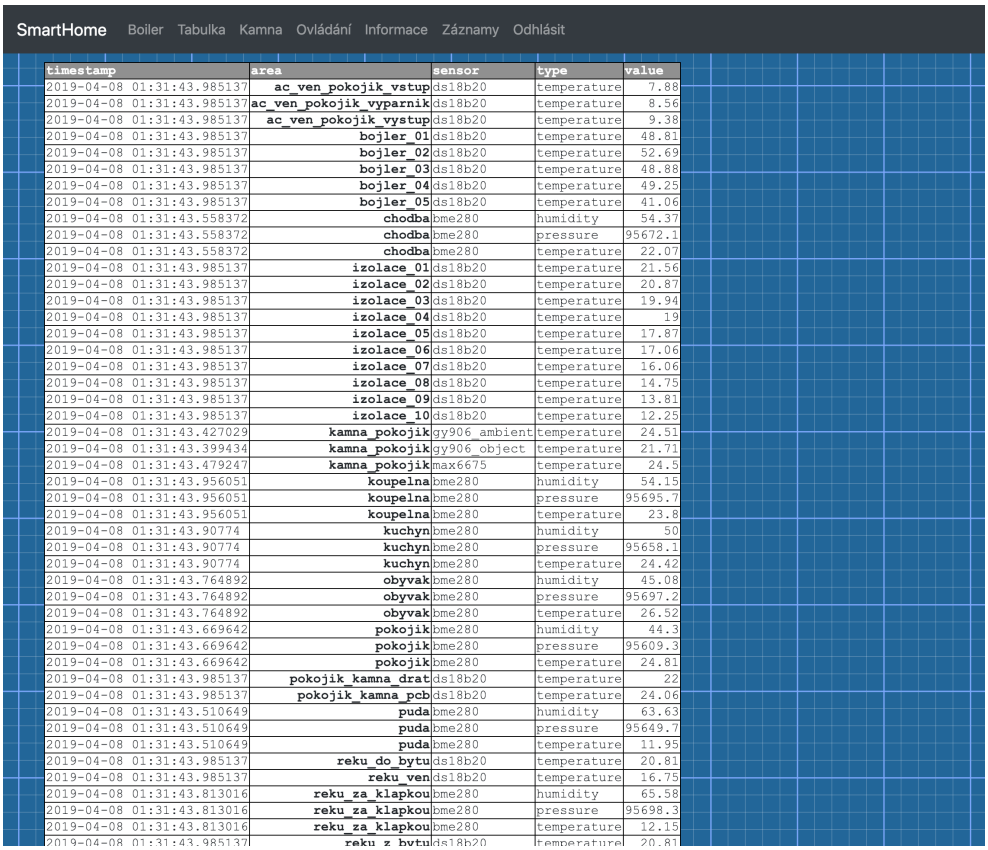


Obrázek B.3: Ovládání klimatizace a klapky

Na obrázku B.3 je v pravé části vyobrazeno nucené ovládání klapky pro odstínění studeného vzduchu zvenku. Klapka je ovládána automaticky tak, aby bylo zamezeno poškození výměníku vzduchu v rekuperaci. Pokud ale uživatel domu chce vynutit otevření nebo zavření této klapky – jde to pomocí tohoto menu. Pokud je zapojeno automatické ovládání klapky – systém po třiceti minutách převezme zpět kontrolu, pokud ne, klapka zůstane ve stavu nastaveném uživatelem.

B.5 Strojový přístup k datům

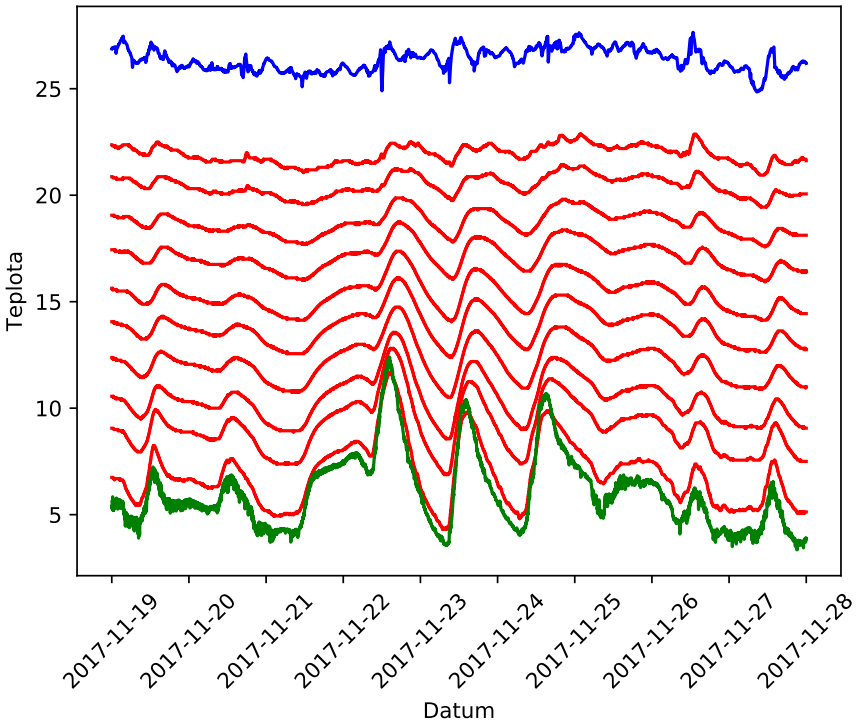
Nejjednodušší přístup k datům je pomocí webu, jak bylo popsáno v sekci B.3. Na webu je dostupná tabulka posledních údajů ze senzorů v sekci „Tabulka“, ta je zobrazena na obrázku B.4. To bohužel neumožní přístup ani k historii ani k datům, které reprezentují něco jiného než teplotu, tlak nebo vlhkost. Další možností je tedy využít přímo přístup k databázi. Lze používat standardní nástroje pro práci s PostgreSQL.



timestamp	area	sensor	type	value
2019-04-08 01:31:43.985137	ac_ven_pokojik_vstup	ds18b20	temperature	7.88
2019-04-08 01:31:43.985137	ac_ven_pokojik_vyparnik	ds18b20	temperature	8.56
2019-04-08 01:31:43.985137	ac_ven_pokojik_vystup	ds18b20	temperature	9.38
2019-04-08 01:31:43.985137	bojler_01	ds18b20	temperature	48.81
2019-04-08 01:31:43.985137	bojler_02	ds18b20	temperature	52.69
2019-04-08 01:31:43.985137	bojler_03	ds18b20	temperature	48.88
2019-04-08 01:31:43.985137	bojler_04	ds18b20	temperature	49.25
2019-04-08 01:31:43.985137	bojler_05	ds18b20	temperature	41.06
2019-04-08 01:31:43.558372	chodba	bme280	humidity	54.37
2019-04-08 01:31:43.558372	chodba	bme280	pressure	95672.1
2019-04-08 01:31:43.558372	chodba	bme280	temperature	22.07
2019-04-08 01:31:43.985137	izolace_01	ds18b20	temperature	21.56
2019-04-08 01:31:43.985137	izolace_02	ds18b20	temperature	20.87
2019-04-08 01:31:43.985137	izolace_03	ds18b20	temperature	19.94
2019-04-08 01:31:43.985137	izolace_04	ds18b20	temperature	19
2019-04-08 01:31:43.985137	izolace_05	ds18b20	temperature	17.87
2019-04-08 01:31:43.985137	izolace_06	ds18b20	temperature	17.06
2019-04-08 01:31:43.985137	izolace_07	ds18b20	temperature	16.06
2019-04-08 01:31:43.985137	izolace_08	ds18b20	temperature	14.75
2019-04-08 01:31:43.985137	izolace_09	ds18b20	temperature	13.81
2019-04-08 01:31:43.985137	izolace_10	ds18b20	temperature	12.25
2019-04-08 01:31:43.427029	kamna_pokojik	gy906_ambient	temperature	24.51
2019-04-08 01:31:43.399434	kamna_pokojik	gy906_object	temperature	21.71
2019-04-08 01:31:43.479247	kamna_pokojik	max6675	temperature	24.5
2019-04-08 01:31:43.956051	koupelna	bme280	humidity	54.15
2019-04-08 01:31:43.956051	koupelna	bme280	pressure	95695.7
2019-04-08 01:31:43.956051	koupelna	bme280	temperature	23.8
2019-04-08 01:31:43.90774	kuchyn	bme280	humidity	50
2019-04-08 01:31:43.90774	kuchyn	bme280	pressure	95658.1
2019-04-08 01:31:43.90774	kuchyn	bme280	temperature	24.42
2019-04-08 01:31:43.764892	obyvak	bme280	humidity	45.08
2019-04-08 01:31:43.764892	obyvak	bme280	pressure	95697.2
2019-04-08 01:31:43.764892	obyvak	bme280	temperature	26.52
2019-04-08 01:31:43.669642	pokojik	bme280	humidity	44.3
2019-04-08 01:31:43.669642	pokojik	bme280	pressure	95609.3
2019-04-08 01:31:43.669642	pokojik	bme280	temperature	24.81
2019-04-08 01:31:43.985137	pokojik_kamna_drat	ds18b20	temperature	22
2019-04-08 01:31:43.985137	pokojik_kamna_pcb	ds18b20	temperature	24.06
2019-04-08 01:31:43.510649	puda	bme280	humidity	63.63
2019-04-08 01:31:43.510649	puda	bme280	pressure	95649.7
2019-04-08 01:31:43.510649	puda	bme280	temperature	11.95
2019-04-08 01:31:43.985137	reku_do_bytu	ds18b20	temperature	20.81
2019-04-08 01:31:43.985137	reku_ven	ds18b20	temperature	16.75
2019-04-08 01:31:43.813016	reku_za_klapkou	bme280	humidity	65.58
2019-04-08 01:31:43.813016	reku_za_klapkou	bme280	pressure	95698.3
2019-04-08 01:31:43.813016	reku_za_klapkou	bme280	temperature	12.15
2019-04-08 01:31:43.985137	reku_z_bytu	ds18b20	temperature	20.81

Obrázek B.4: Přístup k aktuálním údajům

Na obrázku B.5 jsou vykreslena starší data z testování izolace. Graf byl vykreslen pomocí skriptu, který je přiložen na konec této příručky (B.1). Tento skript ukazuje, jak jednoduše lze pomocí Pythonu přistupovat k databázi.



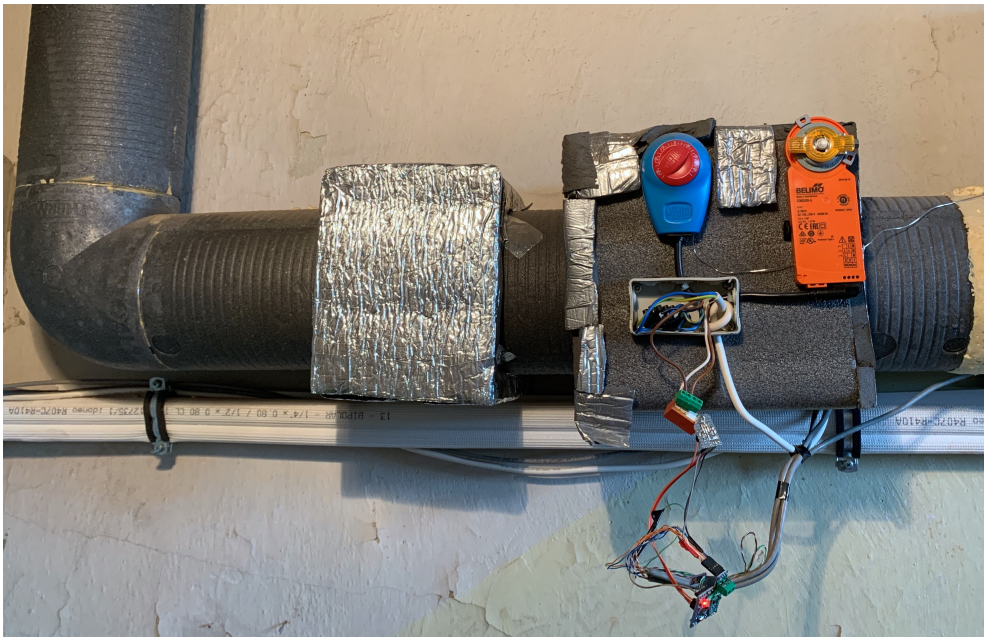
Obrázek B.5: Graf teplot – izolace

```
# formát času 'YYYY-MM-DD HH:MM:SS'
od = '2017-11-19 00:00:00'
do = '2017-11-28 00:00:00'

# seznam jmen čidel (ošklivý zápis pro zkrácení)
ids = [f'izolace_{x:02}' for x in range(1, 11)]
ids += ['puda', 'kuchyn']

import matplotlib.pyplot as plt
import psycopg2

conn = psycopg2.connect("...")
cur = conn.cursor()
dates = {}
values = {}

fmt = 'SELECT timestamp, value FROM data WHERE "timestamp" '
fmt += '<= \'{}\' AND "timestamp" >= \'{}\' AND area = \'{}\' '
fmt += 'AND type = \'temperature\';'

for x in ids: # dotaz na databázi
    cur.execute(fmt.format(do, od, x))

    dates[x] = list()
    values[x] = list()

    for row in cur.fetchall():
        dates[x].append(row[0])
        values[x].append(row[1])

plt.xticks(rotation = 45) # nastavení os
plt.xlabel('Datum')
plt.ylabel('Teplota')

# změna barev pro konkrétní čidla
rules = {'puda': 'g', 'kuchyn': 'b'}

for x in ids:
    plt.plot(dates[x], values[x], rules.get(x, 'r'))

plt.show()
```

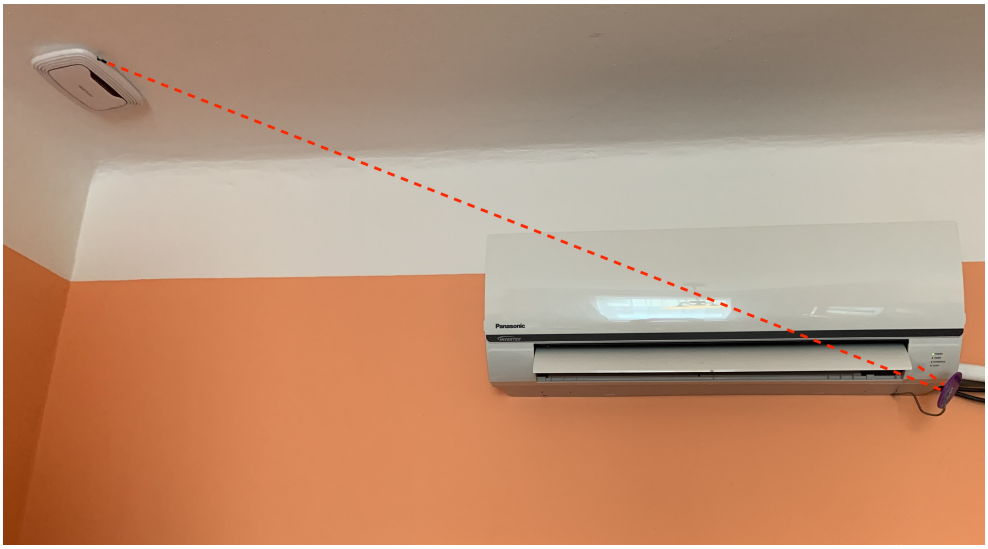
Kód B.1: Tisk grafů – izolace

Dodatek C

Fotodokumentace vybraných částí práce



Obrázek C.1: Vzduchotechnika s jednotkou – Zprava je zvenku přiváděn vzduch, který vstupuje do protimrazové klapky, dále do vzduchového filtru a následně odchází směrem vlevo do rekuperační jednotky. Na pravé části obrázku je k vidění jednotka a ovládání klapky (bez krytu). Jednotka je připojena k síti RS-485.



Obrázek C.2: Klimatizační jednotka „pokojík“ – V levé horní části (stará krabička od routeru) je umístěna jednotka s IR ovládáním. Aby se signál dostal od vysílače k přijímači muselo být použito zrcátko. Červenou přerušovanou čárou je naznačen směr IR vysílaní.

Obrázek C.3: Klimatizační jednotka „obývák“ – jednotka chytrého domu je schována v odsávání rekuperace. Pokud není zrovna klimatizace vypnutá je přímý výhled zastíněn – proto bylo použito zrcátko. Druhý otvor v krytu bude sloužit k ovládání televize.

Obrázek C.4: Rekuperační jednotka – S připojenými čidly chytrého domu.

Dodatek D

Obsah přiložené paměťové karty

Veřejný klíč pro kontrolu archivu dostupný z internetové adresy:
`https://keys.kyzek.cz/jakubkyzek.gpg`.

Otisk GPG klíče pro ověření:
`30DC B24C 839A 5131 129A  AFDC 5889 978F 5F3C 7BBE`

- `README.txt` ... stručný popis práce s archivem
- `BI-BAP_kyzekjak.tar.gz.sig` ... GPG podpis archivu
- `BI-BAP_kyzekjak.tar.gz` ... archiv s prací
  - `README.txt` ... popis obsahu archivu
  - `BP_Kyzek_Jakub_2019.pdf` ... text bakalářské práce včetně zadání
  - `tex` ... zdrojový kód pro text práce
  - `bin` ... zkompilované soubory
    - `bootloader.hex` ... zavaděč v IHEX formátu
    - `firmware.hex` ... firmware v IHEX formátu
  - `etc` ... ostatní soubory
  - `src` ... zdrojové kódy
    - `arduino_core` ... upravené Arduino core
    - `bootloader` ... zavaděč
      - `scripts` ... scripty pro nahrávání firmware
    - `firmware` ... firmware pro jednotky
    - `smarthomeserver` ... server
    - `smarthomeflask` ... webové rozhraní