Česká zemědělská univerzita v Praze

Technická fakulta

Automatizace řízení teploty a vlhkosti vzduchu v datovém rozvaděči

Bakalářská práce

Vedoucí práce: doc. Ing. Stanislava Papežová, CSc.

Autor práce: Petr Šlapák

PRAHA 2020

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Technická fakulta

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Petr Šlapák

Zemědělské inženýrství
Inženýrství údržby

Název práce

Automatizace řízení teploty a vlhkosti vzduchu v datovém rozvaděči

Název anglicky

Automation of temperature and humidity control in a data rack

---

Cíle práce

Cílem práce je seznámit se s principy senzorů pro měření teploty a vlhkosti, dále navrhnout a realizovat funkční model řídicí jednotky pro stabilizaci teploty a vlhkosti v datovém rozvaděči. Systém bude pracovat autonomně s možností nastavení požadovaných hodnot uživatelským rozhraním počítačové aplikace.

Metodika

1. Seznamte se se základními principy senzorů pro měření teploty, vlhkosti a jejich vlastnostmi.

2. S ohledem na zadané parametry systému zvolte vhodný typ řídící jednotky, senzorů a periferních obvodů včetně přizpůsobení rozhraní.

3. Zařízení realizujte a vybavte programem pro průběžný sběr dat, jejich zpracování, vyhodnocování a re-prezentaci.

4. Vytvořte uživatelské rozhraní pro nastavení vstupních parametrů zařízení.

5. Funkčnost zařízení ověřte.

6. Vyhodnoťte výsledky měření.

Doporučený rozsah práce

30 – 40 stran

Klíčová slova

teplota, vlhkost, automatizace, regulace, měření

Doporučené zdroje informací

BALÁTĚ, Jaroslav. Automatické řízení. 2., přeprac. vyd. Praha: BEN, 2004. ISBN 978-80-7300-148-3.

KREIDL, Marcel. Měření teploty: senzory a měřící obvody. Praha: BEN – technická literatura, 2005. Senzory neelektrických veličin. ISBN 80-7300-145-4.

MARTINEK, Radislav. Senzory v průmyslové praxi. Praha: BEN – technická literatura, 2004. ISBN 80-7300-114-4.

PETZOLD, Charles. Programování Microsoft Windows Forms v jazyce C#: [vytváříme uživatelské rozhraní aplikací]. Brno: Computer Press, 2006. ISBN 80-251-1058-3.

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

Předběžný termín obhajoby

2019/2020 LS – TF

Vedoucí práce

doc. Ing. Stanislava Papežová, CSc.

Garantující pracoviště

Katedra elektrotechniky a automatizace

Elektronicky schváleno dne 29. 1. 2019

Ing. Miloslav Linda, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 15. 2. 2019

doc. Ing. Jiří Mašek, Ph.D.

Děkan

V Praze dne 09. 04. 2020

Čestné prohlášení

Prohlašuji, že jsem bakalářskou práci na téma *Automatizace řízení teploty a vlhkosti vzduchu v datovém rozvaděči* vypracoval samostatně a použil jen pramenů, které cituji a uvádím v seznamu použitých zdrojů. Jsem si vědom, že odevzdáním bakalářské práce souhlasím s jejím zveřejněním dle zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů, ve znění pozdějších předpisů, a to i bez ohledu na výsledek její obhajoby. Jsem si vědom, že moje bakalářská práce bude uložena v elektronické podobě v univerzitní databázi a bude veřejně přístupná k nahlédnutí. Jsem si vědom že, na moji bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů, ve znění pozdějších předpisů, především ustanovení § 35 odst. 3 tohoto zákona, tj. o užití tohoto díla.

V Praze dne 9.4.2020 _____

Petr Šlapák

Poděkování

Rád bych touto cestou poděkoval doc. Ing. Stanislavě Papežové, CSc. za odborné vedení bakalářské práce, její vstřícnost a cenné rady a podněty, jež vedly k její dokončení.

Abstrakt: Tato bakalářská práce se zabývá návrhem a realizací zařízení na udržování teploty a vlhkosti vzduchu uvnitř datového rozvaděče v požadovaném rozsahu mezních hodnot. Práce je rozdělena na teoretická východiska a praktickou část práce. Část teoretických východisek se věnuje problematice měření neelektrických veličin, především pak teploty a vlhkosti vzduchu. Následuje ucelený přehled modelů mikrokontroleru Arduino, včetně vybraných senzorů a rozšíření. Poslední podkapitola teoretických východisek poskytuje úvod do programovacích jazyků a programovacích prostředí. Praktická část je věnována výběru jednotlivých komponent pro realizaci zařízení a tvorbě programu pro jeho obsluhu. Závěr práce je zaměřen na úpravu komponent pro jejich použití v praxi a návrh konstrukčního uspořádání zařízení.

Klíčová slova: teplota, vlhkost, automatizace, Arduino, C#, 3D tisk

Automation of temperature and humidity control in the data rack

Summary: This bachelor thesis deals with the design and implementation of the device for maintaining temperature and humidity inside the data rack within the required range of target values. The thesis is divided into theoretical background and practical part. Part of the theoretical background is devoted to the measurement of non-electrical quantities, especially temperature and humidity. Followed by a comprehensive overview of Arduino microcontroller models, including selected sensors and extensions. The last subchapter of theoretical background provides an introduction to programming languages and programming environments. The practical part is devoted to the selection of individual components for the implementation of the device and the creation of a program for its operation. The conclusion of the thesis is focused on the modification of components for their use in practice and design of the structural arrangement of the device.

Key words: temperature, humidity, automation, Arduino, C#, 3D print

Obsah

Seznam obrázků

Seznam tabulek

Seznam použitých zkratek

ADC	Analog-Digital Convertor, analogově digitální převodník
ARM	Advanced RISC Machine, architektura procesorů
ARP	Address Resolution Protocol, protokol k získání linkové adresy
AVR	RISC mikroprocesory firmy Atmel
CD	Compact Disc, kompaktní disk
CSV	Comma-Separated Values, souborový formát hodnot oddělených čárkami
DHCP	Dynamic Host Configuration Protocol, protokol k přidělování síťových nastavení
DLL	Dynamic-link library, dynamicky linkovaná knihovna
EEPROM	Electrically Erasable Programmable Read-Only Memory, elektronicky vymazatelná paměť pouze pro čtení
FTP	Foil shielded Twisted Pair, fólií stíněný kroucený pár
HTML	Hypertext Markup Language, hypertextový značkovací jazyk
I^2C	Inter-Integrated Circuit, počítačová sériová sběrnice
ICMP	Internet Control Message Protocol, protokol pro odesílání služebních informací
IDE	Integrated Development Environment, vývojové prostředí
IEEE	Institute of Electrical and Electronics Engineers, Institut pro elektrotechnické a elektronické inženýrství
IGMP	Internet Group Management Protocol, protokol obsluhy multicastu v protokolu IP
IP	Internet Protocol, internetový protokol
IPv4	Internet Protocol version 4, čtvrtá verze internetového protokolu
LED	Light-Emitting Diode, svítivá dioda
MAC	Media Access Control, identifikace síťového zařízení

MISO	Master In Slave Out, master vstup sběrnice SPI
MOSI	Master Out Slave In, master výstup sběrnice SPI
NC	Normally Closed, v klidovém stavu sepnutý výstup relé
NO	Normally Open, v klidovém stavu otevřený výstup relé
PoE	Power over Ethernet, napájení po Ethernetovém kabelu
PPPoE	Point-to-Point Protocol over Ethernet, protokol komunikace v Ethernetu
PWM	Pulse Width Modulation, pulzní šířková modulace
RGB	Red Green Blue, aditivní způsob míchání barev červená – zelená – modrá
RH	Relative Humidity, relativní vlhkost
RISC	Reduced Instruction Set Computer, označení procesorů s redukovanou instrukční sadou
RJ-45	Konektor kabelu kroucených párů
RS-232	Sériový port
SCK	Serial Clock, časování sběrnice I^2C
SCL	Synchronous Clock, časování sběrnice SPI
SDA	Synchronous Data, synchronní data sběrnice I^2C
SMD	Surface Mount Device, součástky pro povrchovou montáž
SPI	Serial Peripheral Interface, sériové periferní rozhraní
SRAM	Static Random Access Memory, statická paměť náhodného přístupu
SS	Slave Select, výběr podřízeného zařízení sběrnice SPI
TCP	Transmission Control Protocol, protokol transportní vrstvy se zárukou doručení
UDP	User Datagram Protocol, protokol transportní vrstvy bez záruky doručení
USB	Universal Serial Bus, univerzální sériová sběrnice
WiFi	Wireless Fidelity, obecné standardů bezdrátové komunikace IEEE 802.11

1 Úvod

Uvnitř datového rozvaděče se soustředí zpravidla síťové prvky. Tyto prvky mohou být aktivní či pasivní, přičemž aktivní prvek bude vždy během svého provozu emitovat teplo do okolí, které v případě uzavřeného datového rozvaděče nemůže odcházet pryč. Vnitřní teplota tak začne růst, a i při použití průmyslových zařízení, jež mají rozšířené provozní parametry, není možné zajistit bezproblémový chod v prostředí vysokých teplot.

Datové rozvaděče jsou však ve smyslu použití univerzální a je možné je využít nejen pro umístění síťových prvků. V tomto kontextu se jedná o datové rozvaděče speciálního použití, které používá např. zadávací firma této práce. Konkrétně jde v tomto případě především o umístění tiskárny uvnitř datového rozvaděče. Společně s tiskárnou je samozřejmě v rozvaděči umístěn papír, který do sebe absorbuje vzdušnou vlhkost. To komplikuje tisk a tento stav tedy není žádoucí. Mimo jiné bývají tyto rozvaděče umístěny venku, v nevytápěných prostorech, a jsou tak vystaveny klimatickým podmínkám odpovídajícím mírnému klimatickému pásu. Znamená to tedy, že teploty okolí rozvaděče klesají i pod bod mrazu. Při dlouhodobém vystavení těmto podmínkám by bez další kompenzace stavu teplota uvnitř datového rozvaděče rovněž klesla. Tento stav je opět z hlediska provozovaných zařízení nežádoucí.

Zadávající firma má své datové rozvaděče vybaveny systémy topení a ventilace. Tyto rozvaděče jsou ale v bezobslužném provozu a při jejich počtu by nebylo ruční spouštění těchto kompenzačních zařízení možné. Obsluha v tomto ohledu zprostředkovává pouze uvedení do provozu a posléze pravidelnou údržbu.

Tato bakalářská práce představuje návrh a realizaci řešení tohoto konkrétního problému, definovaného zadávající firmou. Zařízení, jež je výstupem této bakalářské práce, je dostatečně univerzální pro použití v jiných sférách, bylo vyvíjeno především pro zajištění požadovaných klimatických podmínek v rozvaděči s ohledem na technologie, které firma používá.

2 Cíl práce

Cílem této práce je návrh a realizace zařízení, jež bude udržovat klimatické podmínky uvnitř uzavřeného datového rozvaděče v požadovaném rozsahu. Požadované hodnoty teploty a vlhkosti určuje obsluha s ohledem na obsah datového rozvaděče a kritické provozní hodnoty zařízení v něm obsažených.

Dalším cílem práce je i vytvoření aplikačního rozhraní pro nastavení těchto hodnot obsluhou. Nutný je výběr mikrokontroleru, jímž bude tato regulace realizována, a zároveň i komponent pro komunikaci s aplikací a senzorů pro zjišťování stavů.

Zařízení je vyvíjeno pro zadávající firmu pro přímé použití v praxi. Je tedy přihlíženo na požadavky firmy. Pro optimální návrh je rovněž zohledněno prostředí, kam bude zařízení umístěno. Dílčím cílem je tedy i konstrukční řešení umístění zařízení v prostoru.

3 Metodika

Na začátku práce je rozebrána problematika měření neelektrických veličin, zejména teploty a vlhkosti. Pozornost je věnována i jednotlivým měřícím principům senzorů, parametrům senzorů a nejistotám měření. Dále je analyzován mikrokontroler Arduino, včetně jednotlivých typů/modelů a je proveden rozbor možností rozšíření tohoto mikrokontroleru vybranými senzory a moduly. Práce dále obsahuje stručný úvod do vývojových prostředí a programovacích jazyků.

Praktická část se zabývá vytvořením zařízení, jež má na základě nastavených hodnot autonomně udržovat klimatické podmínky uvnitř datového rozvaděče. Pozornost čtenáře je i směřována na využití vývojových prostředí a programovacích jazyků pro vytvoření aplikace pro obsluhu takového zařízení. V poslední části práce je řešen návrh konstrukčního uložení zařízení v praxi a připojení jednotlivých komponent.

4 Teoretická východiska

Pro zajištění bezproblémového chodu zařízení uvnitř datového rozvaděče je důležité udržovat hodnoty teploty a vlhkosti vzduchu v provozních hodnotách provozovaných zařízení. K tomu je nutné znát jak požadované hodnoty, tak aktuální hodnoty těchto veličin. Teplota i vlhkost jsou veličiny neelektrické. K jejich měření je potřeba použít vhodné senzory a data ze senzorů vyhodnotit např. pomocí mikrokontroleru. Nastavení limitů provozních hodnot bude zajištěno komunikací obsluhy s řídicí jednotkou přes komunikační rozhraní.

4.1 Měření neelektrických veličin

Měření neelektrických veličin vyžaduje shodné použití přístrojové techniky, jako měření veličin elektrických. Snímání těchto veličin dovolují senzory, které musí být impedančně přizpůsobené druhu měření a měřené veličině. Senzory převádějí neelektrické veličiny na veličiny elektrické na základě:

- **Principu fyzikálního převodu** – senzor převádí měřenou veličinu na elektrickou na základě její fyzikální změny – např. tenzometr, který v důsledku působící síly mění svůj elektrický odpor,
- **Polovodičových jevů** – nejčastěji se jedná o křemíkové senzory, které využívají jevů v polovodičích – např. změna odporu vrstvy na základě působení mechanického napětí nebo teploty.

Dále senzory rozlišujeme podle míry integrace převodníků:

- **Hybridní senzory** – jedná se o kombinaci převodníku veličiny na elektrický signál a elektronický obvod pro jeho zpracování, umístěných do jednoho pouzdra integrovaného obvodu,
- **Integrovaný senzor** – jedná se o vyšší integraci hybridního senzoru, kdy převodník i elektronický obvod je umístěn na jednom čipu,
- **Inteligentní snímač** – jedná se o sdružení integrovaného senzoru s mikroprocesorem a elektronickými obvody pro zpracování signálu. Pomocí tohoto snímače je možné korigovat vlastnosti senzoru na základě charakteristik uložených v mikroprocesoru z jeho kalibrace. [1]

U automatického měření můžeme měřit jak analogové, tak digitální hodnoty. Analogové hodnoty musí být pro další zpracování mikroprocesorem převedeny na digitální pomocí ADC. Převod analogových hodnot má však časovou prodlevu, proto je nutné stanovit časové intervaly měření. Tento interval musí být větší, než je minimální čas potřebný pro takový převod. V tomto kontextu se jedná o vzorkování, potažmo vzorkovací frekvenci, či periodu. Vzorkovací frekvence má značný vliv na stabilitu diskrétního regulačního obvodu a jeho vlastnosti. Nemůže být volena nahodile. [2], [3]

4.1.1 Parametry senzorů

Každý senzor libovolné veličiny je charakterizován určitými hodnotami jednotlivých parametrů. Pro každý senzor se tyto parametry liší, a na tomto základě jsou senzory pro jednotlivá použití různě vhodné. Výsledná volba vždy závisí na konkrétním řešení. Mezi hlavní parametry patří:

- **Přenosová funkce** – vyjadřuje závislost výstupní měřené veličiny na působící veličině. Obvykle je reprezentována formou grafu. U kalibrovaných senzorů se jedná o certifikovanou kalibrační křivku.
- **Rozlišovací schopnost** – představuje schopnost reprezentovat změnou elektrického signálu změnu fyzikální veličiny.
- **Rozsah** – určuje rozmezí změny fyzikálního vstupu, který je možné převést na elektrický výstup.
- **Nelinearita** – představuje míru odchylky přenosové funkce od lineárního výstupu.
- **Hystereze** – některé senzory po odeznění účinku na vstupu nenavrátí hodnotu výstupu na původní hodnotu. Hodnotu této výchylky nazýváme hysterezí.
- **Přesnost** – udává maximální odchylku naměřené hodnoty od skutečné hodnoty. [4], [5]

4.1.2 Teplota

Teplotu je možné definovat jako množství tepelné energie uvnitř tělesa, či systému. Jedná se o stavovou veličinu. Mnohdy se chybně zaměňuje s pojmem teplo, přičemž teplo je formou energie, jež souvisí s pohybem částic soustavy. Na rozdíl od teploty se v případě tepla nejedná o stavovou veličinu. Její hodnota je mimo aktuální stav systému, či tělesa, závislá zároveň na stavech předešlých. Teplotu není možné měřit přímo. Měří se nepřímo

pomocí teplotní závislosti jiných fyzikálních veličin, jako je například elektrická vodivost. [4], [5]

4.1.2.1 **Měření teploty**

Změna elektrické vodivosti materiálu v závislosti na změně teploty patří mezi nejběžnější principy senzorů teploty. Využívají jí např. termistory – z anglického názvu thermally sensitive resistor.

Termistor je polovodičový prvek vyroben z feroelektrických keramických materiálů. Jeho povrch je zapouzdřen vrstvou keramiky. Touto technologií je možné vyrobit termistory nejrůznějších tvarů. Mezi používané se řadí termistory ve tvaru disku, desky, válečku, či kapky. Výhody termistoru jsou především malé rozměry a velká teplotní rozlišovací schopnost.

Přenosová funkce senzoru má tvar:

$$R_T = A * e^{\frac{B}{T}}, \tag{4.1}$$

kde

A	je	konstanta (závisí na materiálu a tvaru daného typu termistoru),
B		teplotní konstanta termistoru, určená materiálem a
T		termodynamická teplota [K].

Podle teplotní konstanty termistoru rozděluje termistory na pozistory s kladnou teplotní konstantou a negastory se zápornou teplotní konstantou. [4]

Dalšími hojně používanými senzory teploty jsou senzory založené na principu teplotní závislosti napětí polovodičového PN přechodu v propustném směru. Tyto senzory jsou nazývány monolitické PN senzory teploty a mají široký teplotní rozsah – od -55 °C do +125 °C. Senzory jsou typicky realizovány formou diody, či tzv. tranzistorové diody – tranzistor s propojením mezi bází a kolektorem. Přenosová charakteristika je nelineární. Pro diodu má tvar:

$$U_D = m * \frac{k*T}{e} * \ln\left(\frac{I_D}{I_S} + 1\right), \tag{4.2}$$

a pro tranzistor:

$$U_{BE} = \frac{k*T}{e} * \ln\frac{I_C}{I_s} \ , \tag{4.3}$$

kde

U_D je napětí na PN přechodu diody v propustném směru,
U_{BE} napětí na PN přechodu tranzistorové diody v propustném směru,
I_s saturační proud PN přechodu v závěrném směru,
I_D saturační proud PN přechodu diody v propustném směru,
I_C saturační proud PN přechodu tranzistorové diody v propustném směru,
m rekombinační koeficient polovodiče (m ϵ ⟨1;2⟩),
k Boltzmannova konstanta k = 1,38*10^{-23} J*K^{-1},
T termodynamická teplota [K] a
e elementární náboj e = 1,602*10^{-19} C. [4], [5]

Pracovních principů senzorů teploty existuje nepřeberné množství, ale pro potřeby této práce byly zvoleny tyto dva základní principy. V provozu patří mezi nejpoužívanější, a drtivá většina senzorů vyráběných pro Arduino je založena právě na těchto principech.

4.1.3 Vlhkost

Vlhkost je definována jako množství vodní páry obsažené ve vzduchu, či jiném plynu. Nejdůležitější pojmy spjaté s vlhkostí jsou:

- **Absolutní vlhkost** – objem vodní páry obsažený v daném objemu vzduchu, či jiného plynu,
- **Relativní vlhkost (RH**) – poměr vlhkosti plynu vzhledem k saturovanému stavu při stejné teplotě a tlaku,
- **Rosný bod** – hodnoty teploty a tlaku, při kterém plyn začíná kondenzovat do kapalného stavu. [5]

4.1.3.1 **Měření vlhkosti**

Dominantní postavení na trhu se senzory vlhkosti zastávají kapacitní RH senzory. Jsou hojně využívané v průmyslu i jiných oblastech. Také jsou jediným typem senzorů, které dokážou měřit vlhkost v celém rozsahu 0 – 100 % RH, přičemž si zachovávají svou přesnost

měření i při nízkých hodnotách RH blížícím se 0 %. Jejich teplotní závislost je zanedbatelná, a tak jsou vhodné pro měření vlhkosti v širokém rozsahu teplot bez nutnosti kompenzace. Kapacita je s relativní vlhkostí plynu spřažena pomocí permitivity:

$$\varepsilon_r = 1 + \frac{211}{T} * \left(P + \frac{48 * P_s}{T} * H\right) * 10^{-6}, \tag{4.4}$$

kde

ε_r je permitivita,
T termodynamická teplota,
P tlak měřeného vzduchu,
P_s tlak saturovaného vzduchu při teplotě T a
H relativní vlhkost. [6], [7]

Tyto senzory jsou zpravidla ve tvaru rovinného deskového kondenzátoru, takže kapacita je určena vzorcem:

$$C = \varepsilon_0 * \varepsilon_r * \frac{S}{l}, \tag{4.5}$$

kde

C je kapacita,
ε_r permitivita,
ε_0 permitivita vakua,
S plocha elektrod a
l vzájemná vzdálenost elektrod. [8]

4.1.4 Nejistota měření

Nejistotou měření se vyjadřují nepřesnosti, vzniklé při měření v důsledku relativních chyb senzorů, měřících přístrojů a měřících metod. Nejistoty jsou rozlišovány:

- **Standartní typu A**

  Pro výpočet této nejistoty je nutné provést alespoň deset měření a spočtením aritmetického průměru tak odhadnout skutečnou hodnotu X. Samotná nejistota je pak vypočítána jako směrodatná odchylka k odhadu:

$$u_A(x) = \sigma(\bar{X}) = \sqrt{D(\overline{X})} = \sqrt{\frac{\sum_{i=1}^{n}(X_i - \bar{X})^2}{n(n-1)}}, \tag{4.6}$$

kde $\bar{X}$ je aritmetický průměr naměřených hodnot a $D(\overline{X})$ je rozptyl aritmetického průměru.

- **Standartní typu B**

  Vyjadřuje všechny další zdroje nejistot, které nejsou obsažené v nejistotě typu A. Jsou určeny všemi informacemi o daném měření – parametry měřících přístrojů, tolerancí senzorů a součástek, volbou měřící metody, zaokrouhlování, aj. [4]

4.2 Arduino

Arduino je název mikrokontroleru založeného na platformě AVR. Projekt vznikl v roce 2005 v italském městě Ivrea, kdy prof. Massimo Banzi společně s Davidem Cuartiellem sháněli podobnou technologickou učební pomůcku pro výuku studentů designu. [9], [10]

Jedná se o open-source projekt a původní Arduino bylo dodáváno v sadě, kdy si ho nejdříve student musel poskládat sám. Většina desek dnešní generace Arduino je postavena na Atmel AVR s redukovanou instrukční sadou (RISC) a obsahují 6 – 16 analogových vstupů, 14 – 54 digitálních vstupů/výstupů, ze kterých 6 – 14 může být použito jako PWM kanál. S mikroprocesorem je možné komunikovat po sériové lince, SPI a I^2C sběrnici. Další komunikační rozhraní je možné zprostředkovat pomocí rozšiřujících modulů, nebo tzv. shieldů. [9], [11]

Arduino existuje v několika odstupňovaných variantách. Jednotlivé varianty se liší především počtem digitálních a analogových vstupů/výstupů, počtem sběrnic, použitým mikroprocesorem, velikostí paměti a frekvencí krystalu. Na všech těchto parametrech bude záviset výběr použité vývojové desky tak, aby bylo možné připojit všechny senzory, výpočetní výkon byl dostačující, a aby paměť dostačovala chodu programu.

4.2.1 Arduino UNO

Z aktuálně používaných verzí Arduino je UNO nejběžněji používané. Bylo vydané již v roce 2010 a má kompatibilní rozvržení pinů se svými předchůdci (Duemilanove a Diecimila). Je založeno na ATmega328P s 16MHz krystalem, 32kB flash pamětí, 2kB SRAM a 1kB EEPROM. Na desce je osazen druhý mikroprocesor ATmega8U2, který je naprogramován jako převodník USB na RS-232. [11], [10]

UNO tvoří ideální základní startovní vývojovou desku pro začínající programátory této platformy. Je pro něj vytvořeno mnoho kompatibilních rozšíření, návodů, knih a má

bezkonkurenčně největší komunitní základnu. [12] Vzhled této desky je vidět na obrázku 1 a její kompletní schéma vstupů a výstupů zobrazuje příloha 1.

Tato deska byla předvybrána pro realizaci projektu této práce. Pokud by mělo v průběhu praktické části dojít ke zjištění, že by deska nebyla dostačující, či naopak by měla být nadbytečně vybavena, pak bude zvolena jiná deska z rodiny Arduino.

Obrázek 1: Arduino UNO

Zdroj: store.arduino.cc

4.2.2 Arduino Mega 2560

Jestliže UNO je ideální univerzální platformou pro začátky a malé projekty, tak Mega 2560 je vývojová deska pro velké projekty. Jádrem Arduino Mega 2560 je mikroprocesor ATmega2560 podpořený 256kB flash pamětí, 8kB SRAM a 4kB EEPROM. Stejně jako UNO je Mega 2560 taktováno 16MHz krystalem. Deska je vybavena 54 digitálními vstupy/výstupy, z čehož 15 z nich může být použito pro PWM, a 16 analogovými vstupy. [11], [10]

Na obrázku 2 je vidět, že Mega 2560 svými rozměry převyšuje UNO téměř o polovinu, ale zůstává kompatibilní s většinou rozšíření vyvíjených pro UNO.

Obrázek 2: Arduino Mega 2560

Zdroj: store.arduino.cc

4.2.3 Další Arduino

Rodina vývojových desek Arduino je poměrně četná a častým jevem je i její rozšiřování. Na druhou stranu ale rovněž některé verze zanikají, přičemž je vždy zachována kompatibilita s jiným modelem. Jednotlivé desky mají nejen rozdílné tvary a rozměry, ale liší se především počtem vstupů, výstupů, počtem dostupných sběrnic, velikostí jednotlivých pamětí, či dokonce typem mikroprocesoru. Není pravidlem, že s rostoucí velikostí desky proporčně roste i její počet výstupů, či její výkon. V tabulce 1 je vidět, že např. Arduino Micro předčí Arduino UNO ve všech parametrech, a to při menších rozměrech. Každá deska má však své výhody a pro UNO je to např. velká škála již vytvořených rozšiřujících modulů, tzv. shieldů, kterým bude věnována pozornost v nadcházející kapitole.

Tabulka 1: Přehled modelů Arduino

Označení modelu	Nano	Micro	UNO	Leonardo	Mega 2560
Mikroprocesor	ATmega328	ATmega32u4	ATmega328P	ATmega32u4	ATmega2560
Flash [kB]	32	32	32	32	256
EEPROM [kB]	1	1	1	1	4
SRAM [kB]	2	2,5	2	2,5	8
Počet digitálních I/O	22	20	14	20	54
Počet PWM kanálů	6	7	6	7	15
Počet analogových výstupů	0	0	0	0	0
Počet analogových vstupů	8	12	6	12	16
Frekvence časovače [MHz]	16	16	16	16	16
Rozměry [mm]	45 x 18	48 x 18	68,6 x 53,4	68,6 × 53.3	101,5.2 x 53,3

Zdroj: zpracováno podle [11]

Mimo modely obsažené v tabulce 1 obsahuje rodina Arduino i vývojové desky založené na bázi 32 bitových ARM procesorů. Jejich přehled je interpretován tabulkou 2. [11]

Tabulka 2: Přehled modelů Arduino s ARM architekturou

Označení modelu	Zero	Due	Portenta H7
Procesor	ATSAMD21G18	AT91SAM3X8E	STM32H747XI
Flash [kB]	256	512	2048
EEPROM [kB]	0	0	0
SRAM [kB]	32	96	864
Počet digitálních I/O	20	54	22
Počet PWM kanálů	10	12	7
Počet analogových výstupů	1	2	2
Počet analogových vstupů	6	12	3
Frekvence časovače [MHz]	48	84	480
Rozměry [mm]	68 x 53	101,52 x 53,3	doposud neuvedeno

Zdroj: zpracováno podle [11]

Model Portenta H7 je nejnovějším modelem rodiny Arduino. Jeho vydání bylo původně plánováno na konec března roku 2020. Výrobce ale z důvodu celosvětové pandemie COVID-19 v době psaní této práce vydání odložil a některá data nebyla ještě dostupná. Portenta H7 však bude vybaven WiFi, Bluetooth a je ho dokonce možné pomocí USB-C

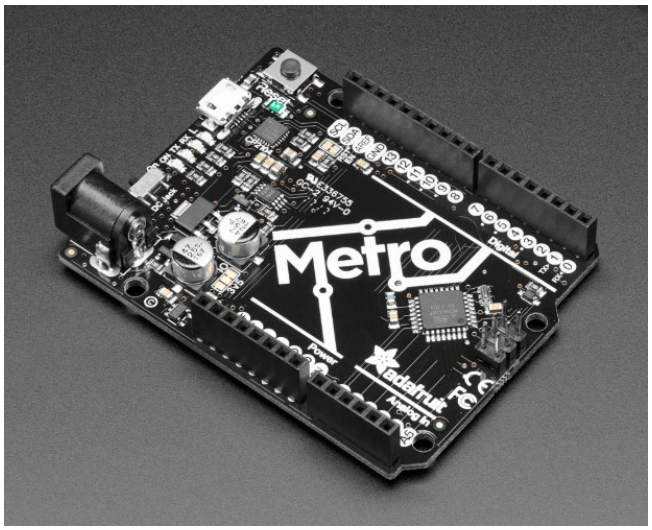
připojit k monitoru. Byl vyvíjen pod hlavičkou Arduino Pro, jež má být předurčena pro vývoj aplikací pro průmysl 4.0. [11]

4.2.4 Klony Arduino

Díky tomu, že je Arduino open-source projekt, tak jsou veškerá jeho výrobní data veřejně dostupná. Na základě těchto dat si může své Arduino vyrobit úplně každý a existuje i několik výrobců, kteří vyrábějí „neoriginální“ Arduino desky. Tyto desky jsou označovány jako klony. Klony bývají zpravidla plně kompatibilní s původním Arduino, bez ohledu na úpravy designu provedených výrobcem. Jediným chráněným prvkem je obchodní značka Arduino, která nemůže být s takovým klonem spojována, pokud nedostane výrobce povolení toto označení používat. [9], [13]

Příkladem takového klonu je Adafruit METRO 328, jež je plně kompatibilní s Arduino UNO. Jak je vidět na obrázku 3, tak hlavním rozdílem od Arduino UNO je použití mikroprocesoru v SMD pouzdře a micro USB konektoru.

Obrázek 3: Klon Adafruit METRO 328



Zdroj: adafruit.com

4.2.5 Shieldy

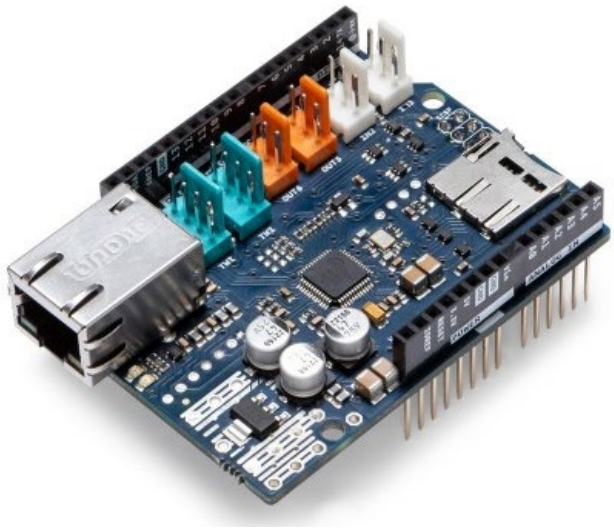
Slovem shield se označují desky plošných spojů, jež svým tvarem a vývody pasují na desku Arduino a plně, či částečně, ji překryjí. Shieldy se nasouvají na podélné konektory. Jejich spojení zajišťuje jak mechanickou montáž, tak elektrické propojení. Shieldy většinou kopírují stejné podélné konektory i na své desce plošných spojů, a tím umožňují nasazení

dalšího shieldu. Stohovatelnost těchto shieldů je omezena počtem pinů, jež jednotlivé shieldy používají. Shieldy nemohou nikdy rozšířit základní počet vstupů, či výstupů původní desky. [15]

4.2.5.1 **Ethernet shield**

Ethernet shield slouží k připojení Arduino k internetu. Rozměr jeho desky je identický s Arduino UNO a umožňuje stohovat na sebe další shieldy. Na desce je obsažen čip Wiznet W5500, který podporuje protokoly jako TCP a UDP. Podporuje až 8 souběžných navázaných spojení. Je možné jej ještě rozšířit o modul PoE (Power over Ethernet) a desku Arduino napájet skrze síťový kabel vedoucí do portu RJ45. Jak je vidět na obrázku 4, tak je dále Ethernet shield vybaven slotem pro micro-SD kartu. Komunikace s W5500 i SD kartou probíhá přes stejnou sběrnici, a tudíž nejdou používat zároveň. Pokud je nutné pro chod programu použití obou, tak je nutné mezi nimi přepínat. Pro přepínání jsou použity dva výstupy, které proto není možné použít k jinému účelu. [11]

Obrázek 4: Ethernet shield



Zdroj: store.arduino.cc

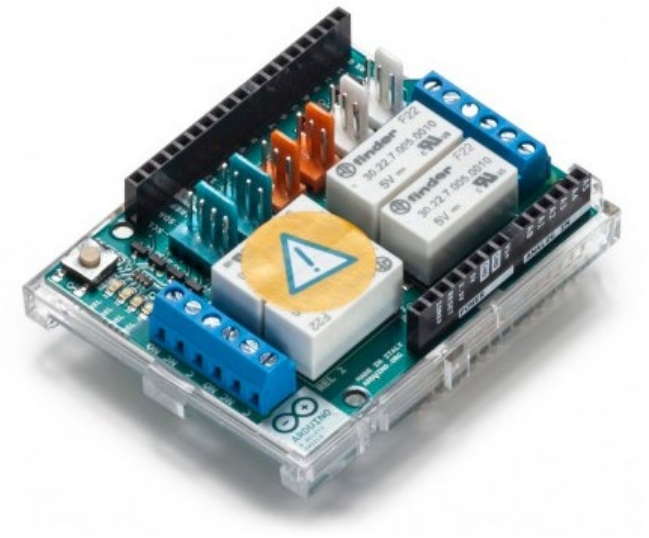
4.2.5.2 **Relay shield**

Relay shield slouží k ovládání obvodů, které překračují svým provozním proudem či napětím dovolené hodnoty, které deska Arduino dovoluje. Na obrázku 5 je možné vidět, že relay shield obsahuje 4 relé s přepínacím kontaktem – přepínání mezi NO a NC. Relé jsou dimenzována na výstupní napětí 30 V při proudu 2 A. Celkově je tedy možné spínat výkon 60 W každým relé. Pro zvětšení možného spínaného limitu je možné jednotlivá relé spojit

paralelně, a tím se zvýší spínatelný proud na 4 A. Ovládání těchto relé je realizováno pomocí pinů 4, 7, 8 a 12. Tyto piny tedy není možné použít pro jinou funkci. [11]

Na pinu 12 se nachází MISO sběrnice SPI. Relay shield tedy není možné použít společně s Ethernet shieldem.

Obrázek 5: Relay shield



Zdroj: store.arduino.cc

4.2.6 Senzory a moduly

Shieldy jsou komplexní elektronické desky a obsahují ve většině případů více než jednu dílčí komponentu. Přestože jsou stohovatelné, a tím minimalizují výsledné rozměry celé sestavy, tak vůči jednotlivým komponentám nesou nevýhodu svého rozměru a nutnosti být v bezprostředním kontaktu se samotnou deskou Arduino. Pokud pro zamýšlenou aplikaci postačí jednotlivá komponenta, tak může být výhodnější použití modulu, či senzoru, namísto shieldu. Samozřejmostí je, že podle druhu aplikace použitých senzorů a modulů je možné pro konkrétní řešení vyvinout speciální shield.

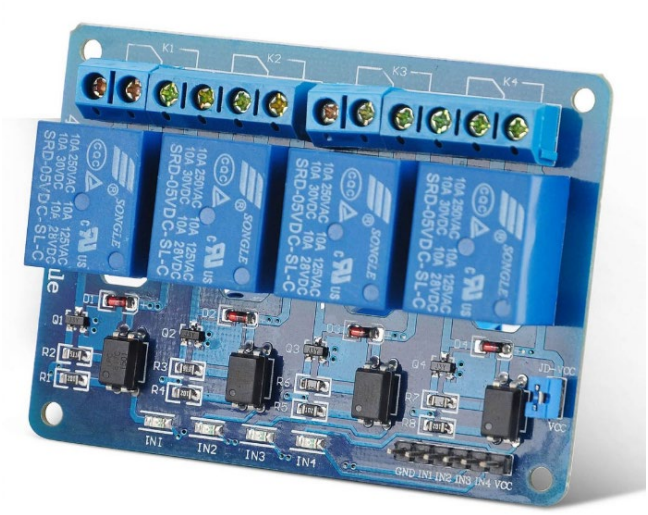
4.2.6.1 Relay modul

Existuje mnoho variant modulů s relé. Ať už v ohledu provozního napětí, tak počtu osazených relé. Těch může být na modulu různý počet, od jednoho po osm. Ve speciálních případech i více. Ovládací napětí je dimenzováno ve většině případů na 5 V. Vstup relé se zpravidla odděluje od ovládacího napětí pomocí optočlenu, a ke každému kanálu je přiřazena

jedna LED pro indikaci stavu sepnutí relé. Relé modulu 2PH63083A, jež je vyobrazen na obrázku 6, jsou dimenzována na výstupní napětí 250 V při 10 A. [14]

Hlavními rozdíly mezi relay shielem a relay moulem je, že relay modul není přímo spjat s Arduino deskou, a tak je možné prostorově oddělit ovládané napětí od ovládací elektroniky. Relé jsou dimenzována na větší proud i napětí, a jsou oddělena optočlenem. Významnou výhodou však je, že není předem určeno, jaké piny musí na desce Arduino používat, a tak je možné použít kterýchkoliv volných. [14], [11]

Obrázek 6: Relay modul

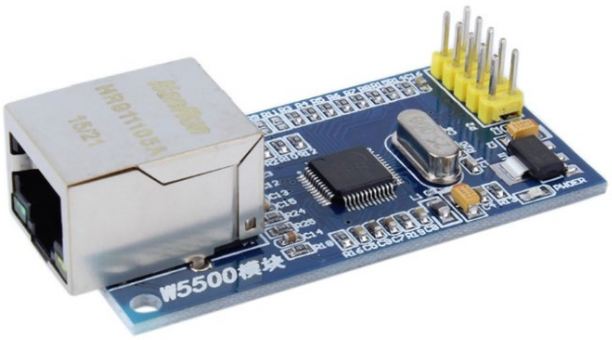


Zdroj: arduino-shop.cz

4.2.6.2 **Ethernet modul**

Ethernet modulů pro vývojové desky existuje nesčetné množství. Jako příklad je zde uveden modul W5500. Ten podporuje TCP/IP protokoly: TCP, UDP, ICMP, IPv4, ARP, IGMP a PPPoE. Pracuje na logické úrovni 3,3V, ale obsahuje i převodník napětí na 5 V. Komunikuje přes standardně používanou sběrnici SPI. Na obrázku 7 je vidět, že je tvořen samostatnou deskou a na jeho desce oproti Ethernet shieldu není slot pro SD kartu. [14]

Obrázek 7: Ethernet modul W5500



Zdroj: arduino-shop.cz

4.2.6.3 **Senzory teploty/vlhkosti**

Nejdůležitějšími senzory pro aplikaci této práce jsou senzory teploty a senzory vlhkosti. Tyto senzory existují jako jednotlivé senzory pro měření teploty a senzory pro měření vlhkosti, či dokonce kombinované senzory teploty a vlhkosti. Kvůli kompaktnosti zařízení vyvíjeného v rámci této práce, bude věnována pozornost pouze senzorům kombinovaným. Mezi tyto senzory patří především DHT11, DHT22, HTU21D a AM2320. Tyto senzory fungují na principech zmíněných v teoretických východiskách této práce. Obsahují vlastní mikroprocesor, a s Arduino deskou komunikují přes sběrnici. DHT11, DHT22 a AM2320 používají 1-Wire sběrnici a HTU21D používá sběrnici I^2C. [14]

4.3 Vývojové prostředí / programovací jazyky

Pro vývoj programových prostředků je zapotřebí znalost programovacího jazyka a jeho použití k sepsání obslužného kódu. Proces vývoje takového obslužného kódu se běžně odehrává ve vývojovém prostředí. Vzhledem k řešení multiplatformního projektu, jež je cílem této práce, není možná realizace pomocí jednoho programovacího jazyka, a tedy použití jednoho vývojového prostředí.

4.3.1 Programovací jazyk C#

C# je objektově orientovaný programovací jazyk vyšší úrovně, jehož syntaxe vychází z jazyků C a C++. Tímto jazykem jsou napsány jednotlivé knihovny DLL (.dll), jež dohromady tvoří rámec .NET Framework. Tento rámec je pravidelně doplňován o další knihovny a aktuálně existuje verze 4.8. [16]

Pro programování v jazyce C# je možné používat mnoho nejrůznějších nástrojů. Základem je textový editor, kompilátor, interpret a debugger. Všechny tyto komponenty sdružené dohromady tvoří IDE (Integrated Development Environment). Nejznámější IDE je Microsoft Visual Studio. Jedná se o robustní vývojové prostředí, jež programátorovi umožňuje, jak kód vytvářet, kompilovat i ladit. Používání Visual Studia rozhodně není nutností, ale obsahuje vše potřebné jak pro začínající, tak pro profesionální programátory. [16], [17]

Microsoft Visual Studio je pro veřejnost dostupné zdarma jako Visual Studio Community. Jednotlivé verze software korespondují s verzemi .NET Frameworku. Pro používání knihoven nejnovějšího frameworku 4.8 je tedy nutné použít nejnovější verzi Visual Studio 2019. IDE mimo potřebné komponenty pro programátora obsahuje i velice užitečný našeptávač, a v reálném čase ověřuje syntaxi. Programátorovi tak značně usnadňuje práci. [16], [18]

4.3.2 HTML

HTML je značkovací jazyk, s jehož pomocí se vytvářejí webové stránky. Vytváření takových webových stránek probíhá ve své podstatě tak, že se tyto stránky popisují. Jejich správnou interpretaci, potažmo jejich správné zobrazení, na základě tohoto popisu má na starosti webový prohlížeč. [19], [20]

Webových prohlížečů existuje mnoho a mohou se ve smyslu interpretace HTML lišit. Při používání některých pokročilých funkcí je možné, že webové stránky budou ve dvou webových prohlížečích vypadat odlišně, a bude je tedy potřeba optimalizovat pro konkrétní prohlížeč. [20]

K tvorbě kódu HTML, který reprezentuje tvořenou webovou stránku, je možné využívat libovolný textový editor. Existují samozřejmě i speciální IDE pro HTML, jež obsahují různé funkce pro zjednodušení a zkratky. Dále je možnost i používat nástroje v módu WYSIWYG (What you see is what you get), což v překladu znamená „Dostaneš, co vidíš“. Tímto způsobem je možné tvořit webové stránky bez znalosti HTML, neboť webové stránky se tvoří pomocí vkládání jednotlivých prvků stránky pomocí editoru, a výsledný HTML kód se vygeneruje až posléze. [20], [21]

4.3.3 Wiring

Wiring je hovorový název pro open-source platformu pro vývoj elektronických prototypů. Obecně se jím myslí programovací jazyk a IDE pro programování Arduino. Syntaxe tohoto samozvaného jazyka je potažmo samotný jazyk C a C++. Při programování Arduino se vytváří programy, kterým se říká sketch. Na obrázku 8 je vidět jednoduchý program, kde je možné si povšimnout, že tyto sketch se v základu člení na dvě základní metody – setup a loop. [9], [22]

Obrázek 8: Ukázkový program z Arduino IDE (blikání LED)

```
1  int LED = 10;                 //LED připojena na pin 10
2  void setup()
3  {
4     pinMode (LED, OUTPUT);     //nastavení pinu jako výstup
5  }
6  void loop()
7  {
8     digitalWrite(LED, HIGH);   //rozsvícení LED
9     delay (1000);              //zdržení 1000 ms = 1 s
10    digitalWrite(LED, LOW);    //zhasnutí LED
11    delay (1000);              //zdržení 1000 ms = 1 s
12 }                             //konec programu -> přeskok na řádek 6
```

Zdroj: vlastní zpracování

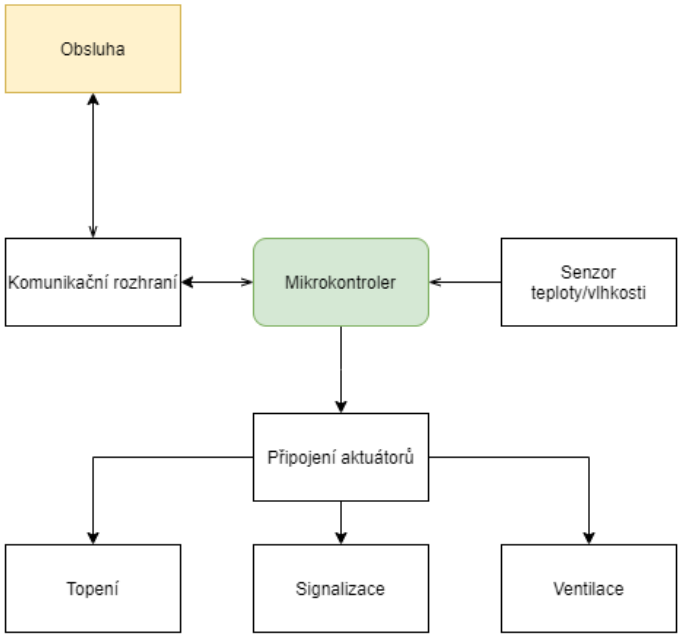
Ve své podstatě metoda setup proběhne na začátku hned po spuštění Arduino, či po provedení resetu, a metoda loop se poté provádí stále dokola.

5 Praktická část práce

Zařízení je navrženo k využití ve firmě pro vzdálený monitoring a případnou regulaci klimatických podmínek uvnitř uzavřeného prostoru datového rozvaděče při překročení mezních hodnot. Tento prostor obsahuje elektronické komponenty citlivé na klimatické podmínky. Především pak vysokou teplotu a vysokou vlhkost vzduchu.

Volba jednotlivých komponent zařízení byla konzultována se zástupci firmy a byla uzpůsobována potřebám a požadavkům firmy. Výchozím pro návrh bylo blokové schéma na obrázku 9.

Obrázek 9: Blokové schéma zařízení



Zdroj: vlastní zpracování

5.1 Volba použitých komponent

Již na počátku projektu bylo rozhodnuto o používání mikrokontroleru Arduino. Jaká konkrétní verze bude potřeba se odvíjí od komplexnosti řešení. Hlavní ovlivňující parametry jsou rozměry, kompatibilita s volenými senzory, velikost vnitřní paměti a výpočetní výkon. Požadavky na přídavné senzory a aktuátory byly ze strany firmy následující:

- Nízká cena,
- Adekvátní přesnost,

- Snadná vyměnitelnost,
- Nízká poruchovost,
- Malé rozměry,
- Dlouhodobá podpora zařízení,
- Velká dostupnost na trhu,
- Bezpečnost zařízení.

Na základě těchto požadavků byly voleny senzory a moduly určené pro vývojové sady.

5.1.1 Senzor teploty a vlhkosti

Na základě požadavků firmy na snadnou vyměnitelnost byl vybírán senzor těchto veličin jako kombinovaný. Z dostupných senzorů byly vybrány senzory DHT11, DHT22, HTU21D a AM2320. Z hodnot senzorů byla vytvořena tabulka 3, která byla předložena firmě pro udělení rozhodnutí.

Tabulka 3: Přehled vybraných senzorů teploty a vlhkosti

Název senzoru	**HTU21D**	**DHT11**	**DTH22**	**AM2320**
Rozsah měření teploty [°C]	-40 – 125	0 – 50	-40 – 125	-40 – 80
Rozsah měření vlhkosti [%]	0 – 100	20 – 90	0 – 100	0 – 99,9
Přesnost měření teploty [°C]	± 0,3	± 2	± 0,5	± 0,5
Přesnost měření vlhkosti [%]	± 2	± 5	± 2	± 3
Rozměry [mm]	13 x 10	23 x 12 x 5	25 x 15 x 8	21 x 13 x 8
Orientační cena bez DPH [Kč]	141	53	160	74

Zdroj: Zpracováno podle [14]

Firmou byl na základě předložené tabulky zvolen senzor HTU21D. Tento senzor komunikuje přes rozhraní I^2C a potřebuje pro svůj provoz napájení 3,3 – 5 V.

5.1.2 Akční členy

Datový rozvaděč, kde má být zařízení umístěno, je vybaven ventilátory a topným tělesem, které by firma chtěla pro regulaci využít. Firma tím může ušetřit náklady na pořizování nových komponent a na práci technika při instalaci zařízení. Jelikož je jejich provozní napětí 230 V, není je možné ovládat pomocí Arduino napřímo, ale je nutné toto provozní napětí oddělit. Aby byla dodržena bezpečnost zařízení, bude toto napětí od Arduino

odděleno pomocí relé. Relay shield pro tento účel není možné použít, neboť jeho relé jsou dimenzována na 30 V a navíc by muselo Arduino být v bezprostřední blízkosti s napětím 230 V. Vzdálenost by v tomto případě mohla představovat nebezpečí nejen v případě nutné výměny. Po komunikaci s firmou bylo tedy domluveno použití relay modulu. Důležité pak bylo určit počet relé, který bude pro ovládání všech komponent potřeba. Pro ovládání ventilátoru a topného tělesa je zapotřebí dvou relé.

Dodatečným požadavkem firmy byla jednoduchá a přímá reprezentace stavu regulace klimatu v datovém rozvaděči. Jako efektivní, a zároveň jednoduché řešení, bylo zvoleno umístění RGB LED pásku do dveří datového rozvaděče. Pokud budou hodnoty klimatických podmínek v nastavených mezích, budou LED pásky rozsvíceny zeleně, a pokud dojde k vybočení z nastavených mezí, LED pásky se rozsvítí červeně. K tomuto ovládání je zapotřebí dalšího relé. Celkem je tedy zapotřebí tří relé. Modul se třemi relé se standardně nevyrábí. S ohledem na požadavek dostupnosti na trhu přichází v úvahu použití současně modulu s jedním a dvěma relé. Aby bylo možné další rozšíření funkcionality zařízení v budoucnu, tak bude pro ovládání regulačních prvků a signalizaci použit čtyřkanálový relay modul. Volné relé bude tak tvořit možnou rezervu.

5.1.3 Komunikační rozhraní

Arduino disponuje řadou komunikačních rozhraní a o další je možné jej rozšířit pomocí modulů, či shieldů. Firma má za požadavek sbírat data ze zařízení a ukládat je pro další analýzu. Data budou analyzována pomocí počítače, a je tedy vhodné sběr dat provádět pomocí počítače, na němž bude probíhat analýza. Vhodným připojením je RS-232, či protokol Ethernet.

Po jednání s firmou byl jako komunikační rozhraní zvolen Ethernet shield, neboť není jednoznačné, jaká bude v praxi reálná vzdálenost mezi zařízením a počítačem pro sběr a analýzu dat. Mimo jiné bude v datovém rozvaděči s výhodou použito napájení Power over Ethernet, jež Ethernet shield umožňuje.

5.1.4 Schéma zapojení

Jednotlivými komponenty pro sestavení zařízení jsou Arduino UNO, Ethernet shield, čtyřkanálový Relay modul a kombinovaný senzor HTU21D. Ethernet shield a HTU21D používají ke komunikaci sběrnice, jež předurčují, jak mají být s Arduino deskou spojeny.

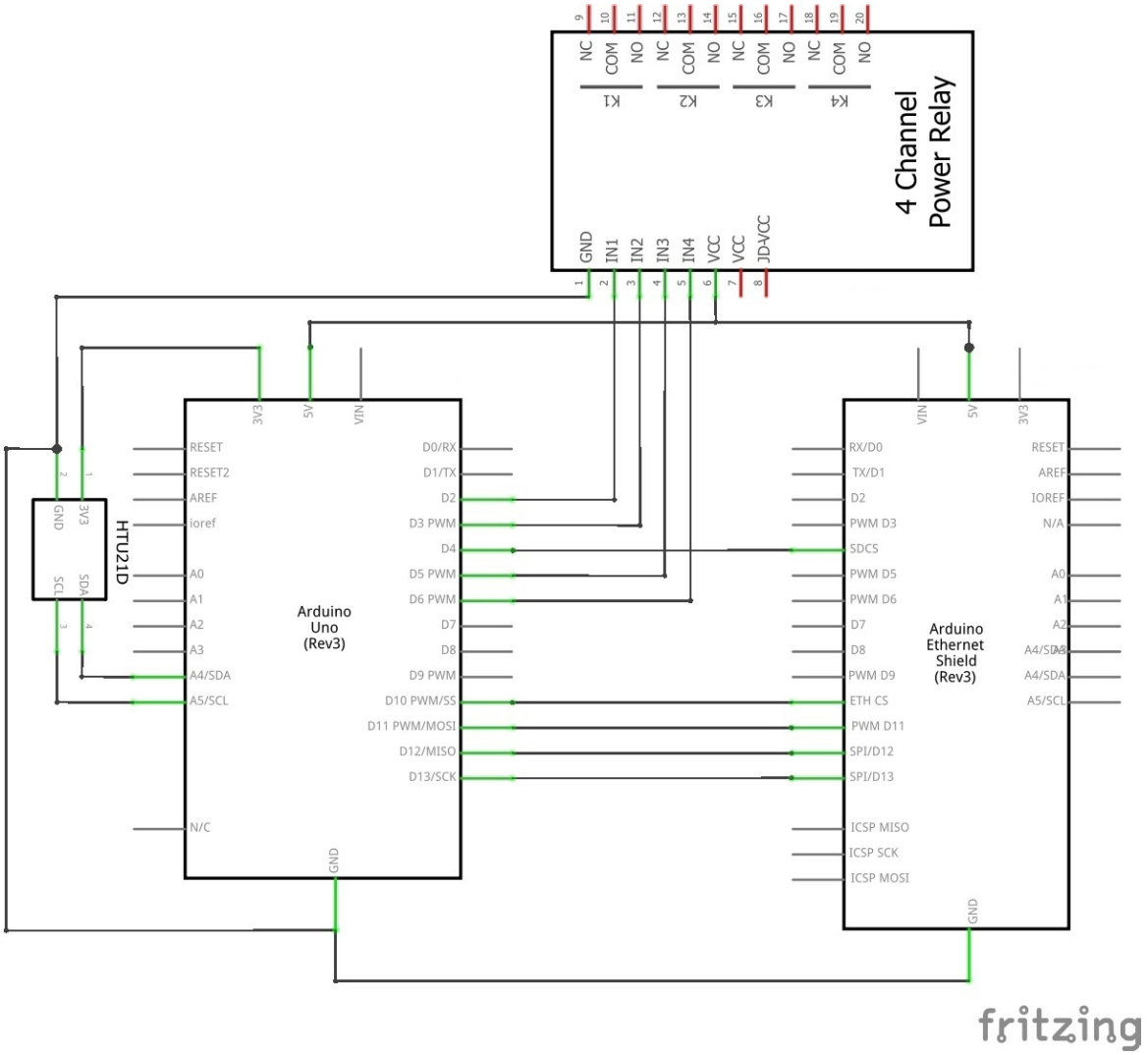
Ethernet shield využívá sběrnici SPI, jež ke svému fungování potřebuje SS, MOSI, MISO a SCK. Ty se na Arduino nachází na pinech 10, 11, 12 a 13. Jak bylo již zmíněno v teoretické části práce, tak Ethernet modul obsahuje čip W5500 pro síťovou komunikaci a slot pro SD kartu. Obojí komunikuje přes SPI sběrnici a je nutné mezi těmito funkcionalitami přepínat. V rámci tohoto projektu nebude slot pro SD kartu využíván, ale přesto není možné použít pin 4, který právě funguje jako tento přepínač. Ethernet shield využívá napájení 5 V z desky Arduino, pokud není osazen modulem pro Power over Ethernet. Pro účely vývoje není tento modul osazen, ale v provozu se počítá s jeho využitím.

Senzor HTU21D pro svou komunikaci potřebuje SDA a SCL sběrnice I^2C. Ty se na Arduino UNO nachází na pinech A4 a A5. Napájení 3,3 V je opět realizováno přímo z desky Arduino.

Relay modul nekomunikuje přes sběrnici. Pro připojení k Adruino je možné použít libovolných digitálních výstupů, jež zůstaly volné. K připojení bude použito pinů 2, 3, 5 a 6. Přestože je nutné použít pouze tří ze čtyř relé, tak budou připojeny všechny čtyři, pro případ budoucího využití. K relay modulu je ještě nutné připojit napájení. Napájení je možné realizovat přes Arduino, nebo externě. Ovládání jednoho relé vyžaduje přibližně 40 mA. V praxi může nastat situace, kdy budou sepnuta všechna tři relé najednou, tj. 120 mA. Pokud bude započítána rezerva ve formě posledního volného relé, tak 160 mA. Maximální proud, jež je možný z Arduino odebírat, je 200 mA [11]. Ethernet shield bude v praxi napájen přímo z PoE modulu a jediným dalším zařízením napájeným z Arduino bude senzor HTU21D. Odběr tohoto senzoru je 0,045 mA. [14] Relay modul je tedy možné napájet přímo z desky Arduino.

Celé toto zapojení tvoří výslednou formu zařízení a jeho schéma, vytvořené v programu Fritzing, je vyobrazeno na obrázku 10. Zařízení je pak pomocí Ethernet patch kabelu připojeno do firemní sítě s podporou napájení přes síť a jednotlivé kompenzační a signalizační prvky jsou připojené prostřednictvím relé. Pro topení bude použit relé výstup 1, pro ventilátor relé výstup 2 a pro přepínání barvy LED pásku bude použit relé výstup 3. Čtvrté relé zůstane volné jako rezerva.

Obrázek 10: Schéma zapojení zařízení



Zdroj: vlastní zpracování

5.2 Komunikace s aplikací

Pro komunikaci s aplikací byl zvolen Ethernet shield, tudíž komunikace s aplikací bude probíhat pomocí protokolů TCP/IP. Ve vývojovém prostředí Arduino je připravena knihovna pro připojení k internetu pomocí Ethernet shieldu. Knihovna Ethernet.h se musí vložit do programu, čímž se zvýší jeho velikost po kompilaci. Velikost programu je jedním z parametrů, které ovlivní, jaký model Arduino bude muset být ve výsledku zvolen.

5.2.1 Konfigurace síťového rozhraní

Každé zařízení v internetu musí mít jedinečný označující prvek. Tímto prvkem je MAC adresa. Ta má běžně délku 48 bitů a vyjadřuje se v hexadecimálním tvaru. Skládá se

z první poloviny, která značí kód výrobce a druhé poloviny, jež si výrobce určí sám. Prefix adresy výrobce přiděluje IEEE (Institute of Electrical and Electronics Engineers). [23]

Vzhledem k tomu, že autor práce není v seznamu výrobců IEEE, tak je potřeba MAC adresu zvolit. Aby při náhodné volbě adresy nedošlo ke konfliktu adres v síti, byla zvolena adresa z již vyřazené síťové karty D8:9E:F3:3A:FC:2C. Díky své jedinečnosti je adresa ideální pro použití jako sériové číslo zařízení.

MAC adresa představuje fyzickou adresu, která je v rámci síťového rozhraní TCP/IP protokolu jedinečná. Pro správné fungování příslušného zařízení v síti je zapotřebí dalších informací. Těmito údaji jsou IP adresy, adresa výchozí brány a maska sítě. [23], [24] Tyto údaje je možné získat z DHCP serveru, ale je možné, že zařízení bude umístěno v síti, kde DHCP není podporován. Jak je vidět na obrázku 11, je pro tyto případy v kódu zařízení nastavená konfigurace, která se použije jako výchozí, pokud není DHCP dostupný.

Obrázek 11: Nastavení síťového rozhraní Arduino

```
20 byte mac[] = { 0xD8, 0x9E, 0xF3, 0x3A, 0xFC, 0x2C };  // MAC adresa / sériové číslo zařízení
21 byte ip[] = { 192, 168, 0, 88 };                      // výchozí IP adresa zařízení
22 byte gateway[] = { 192, 168, 0, 1 };                  // výchozí brána
23 byte mask[] = { 255, 255, 255, 0 };                   // maska sítě
24
25 void setup()
26  {
27      loadFromEEPROM ();
28
29      if (Ethernet.begin (mac) == 1) // příkaz vrátí 1, pokud je DHCP k dispozici
30      {
31          Ethernet.begin (mac); //Ethernet se nastaví dle DHCP, pokud je k dispozici
32      }
33      else
34      {
35          Ethernet.begin(mac, ip, gateway, mask); //pokud DHCP není k dispozici, použijí se výchozí hodnoty
36      }
```

Zdroj: vlastní zpracování

5.2.2 Odesílání a příjem dat

Odesílání dat je realizováno pomocí protokolů TCP/IP. Odesílaná data tvoří řetězec znaků ve formátu:

&spodníLimitTeploty&horníLimitTeploty&spodníLimitVlhkosti&horníLimitVlhkosti@

Pomocí algoritmu, na obrázku 12, je řetězec rozčleněn na jednotlivá data a uložen do proměnných, se kterými program pracuje. Algoritmus prochází jednotlivé znaky přijatého řetězce a pomocí pomocné proměnné **index** určuje dění. V podstatě čeká, než se objeví symbol „&“, který je v řetězci použit jako prefix každé hodnoty. Symbol tedy značí, že

dalším znakem bude nějaká hodnota. Jednotlivé hodnoty mají striktně určené pořadí, se kterým také algoritmus počítá. Jakmile se v průběhu chodu algoritmu objeví „&“, tak inkrementuje hodnotu proměnné index a provádí následující krok. Jednotlivé kroky jsou odstupňovány použitím switch – case v následujícím pořadí:

0 čekání na symbol „&“,
1 načítání dat do pomocného řetězce **str**, dokud nepřijde další prefix,
2 uložení dat z pomocného řetězce **str** do proměnné **tempDownLimit** a přepsání pomocné proměnné **str,**
3 načítání dat do pomocného řetězce **str**, dokud nepřijde další prefix,
4 uložení dat z pomocného řetězce **str** do proměnné **tempUpLimit** a přepsání pomocné proměnné **str,**
5 načítání dat do pomocného řetězce **str**, dokud nepřijde další prefix,
6 uložení dat z pomocného řetězce **str** do proměnné **humdDownLimit** a přepsání pomocné proměnné **str,**
7 načítání dat do pomocného řetězce **str**, dokud nepřijde další prefix a
8 uložení dat z pomocného řetězce **str** do proměnné **humdUpLimit.**

Celý program je součástí této práce jako příloha 2.

Obrázek 12: Algoritmus členění řetězce

```
void parseData(String str)
  {
      //vzor přijatých dat: &15&60&30&70*@
      String parsed = "";
      int index = 0;
      int len = str.length();
      for (int i = 0; i < len; i++)
      {
          if (str[i] != '@') //konec řetězce
          {
              if (str[i] == '&' ) //oddělující znak
              {
                  index++;
              }
              else
              {
                  switch (index)
                  {
                      case 1:
                          parsed += str[i];
                          break;
                      case 2:
                          //mezikrok po prvním oddělujícím znaku - spodní limit teploty
                          tempDownLimit = parsed.toInt();
                          parsed = str[i];
                          index++;
                          break;
                      case 3:
                          parsed += str[i];
                          break;
                      case 4:
                          //mezikrok po druhém oddělujícím znaku - horní limit teploty
                          tempUpLimit = parsed.toInt();
                          parsed = str[i];
                          index++;
                          break;
                      case 5:
                          parsed += str[i];
                          break;
                      case 6:
                          //mezikrok po třetím oddělujícím znaku - spodní limit vlhkosti
                          humdDownLimit = parsed.toInt();
                          parsed = str[i];
                          index++;
                          break;
                      case 7:
                          parsed += str[i];
                          break;
                  }
              }
          }
      }

      //konec algoritmu. Krok po posledním oddělujícím znaku - horní limit vlhkosti
      humdUpLimit  = parsed.toInt();
  }
```

Zdroj: vlastní zpracování

Naměřené veličiny jsou přijaty ve formátu obdobného řetězce, jako odeslaného. Konkrétně pak ve formátu:

&naměřenáTeplota&naměřenáVlhkost&macAdresa@

K rozdělení přijatého řetězce v aplikaci uživatelského rozhraní je použit obdobný algoritmus. Rovněž je řetězec dělen dle oddělovače ve tvaru „**&**“ a ukládán do proměnných, se kterými dál program pracuje. Celý algoritmus je obsahem kompletního kódu obsaženého v příloze 3 této práce.

5.2.3 Regulace

Program zařízení Arduino ve smyčce načítá aktuální hodnoty teploty a vlhkosti ze senzoru a porovnává je s nastavenými limity. Pokud vyhodnotí, že hodnota je mimo nastavený limit, sepne relé příslušného kompenzačního prvku a zapne chybovou signalizaci ve formě LED pásku. Kompenzačními prvky, které má zařízení k dispozici, jsou topné těleso a ventilátor. Díky tomu, že toleranční pole hodnot je interval, a nikoliv konkrétní hodnota, bylo zvoleno použití dvoustavové regulace pomocí relé. Kompenzační prvky jsou tedy používány na plný výkon.

Chybový stav, kdy klesne teplota pod spodní mez, je kompenzován sepnutím topného tělesa. Topné těleso začne emitovat teplo do okolí a vnitřní teplota začne stoupat. Stav, kdy teplota naopak stoupne nad povolenou teplotu, bude kompenzován spuštěním ventilátoru, který bude odsávat ohřátý vzduch z rozvaděče a vyfukovat jej do vnějšího prostředí.

Prostředky pro kompenzaci stavu vysoké vlhkosti zůstávají topné těleso a ventilátor. Pro kompenzaci tohoto stavu bude použita jejich kombinace. Zvýšením vnitřní teploty začne vzduch uvnitř datového rozvaděče zvětšovat svůj objem dle vzorce:

$$\frac{V_1}{V_2} = \frac{T_1}{T_2} \quad \Rightarrow \quad V_2 = V_1 * \frac{T_2}{T_1}, \tag{5.1}$$

kde

V_2 je objem ohřátého vzduchu při teplotě T_2,

V_1 původní objem vzduchu při teplotě T_1,

T_2 teplota ohřátého vzduchu a

T_1 původní teplota vzduchu. [25]

Tato změna objemu vzduchu při zachování stejného objemu vodní páry zapříčiní snížení relativní vlhkosti vzduchu. Absolutní vlhkost zůstává stejná. Spuštěný ventilátor zajistí cirkulaci vzduchu a tím výměnu ohřátého vzduchu za chladnější. Stav, kdy hodnota relativní vlhkosti klesne pod stanovenou hranici, není dle rozhodnutí firmy potřeba programově ošetřit. Zařízení umístěná v datovém rozvaděči jsou bez obtíží schopná pracovat při velmi nízkých hodnotách vlhkosti, tj. 5 % – 10 % RH.

5.3 Uživatelské rozhraní

Pro uživatelské rozhraní byla v rámci této práce vytvořena aplikace Windows form, naprogramovaná v jazyce C#. S vývojovou deskou Arduino komunikuje pomocí protokolů TCP/IP, a s její pomocí je možné ovlivňovat aktuální konfiguraci zařízení. Mimo jiné vhodným způsobem reprezentuje naměřená data, údaje o zařízení a naměřená data ukládá do souboru pro jejich další analýzu.

Uchovávaná data tvoří CSV soubor, který je možné dále editovat např. v Microsoft Excel, či importovat do jiného software pro analýzu dat. Jako název souboru je vždy uvedeno datum měření, a jako data se uchovávají aktuální čas, naměřená teplota, naměřená vlhkost a sériové číslo měřícího zařízení. Toto měření se ukládá každé dvě minuty.

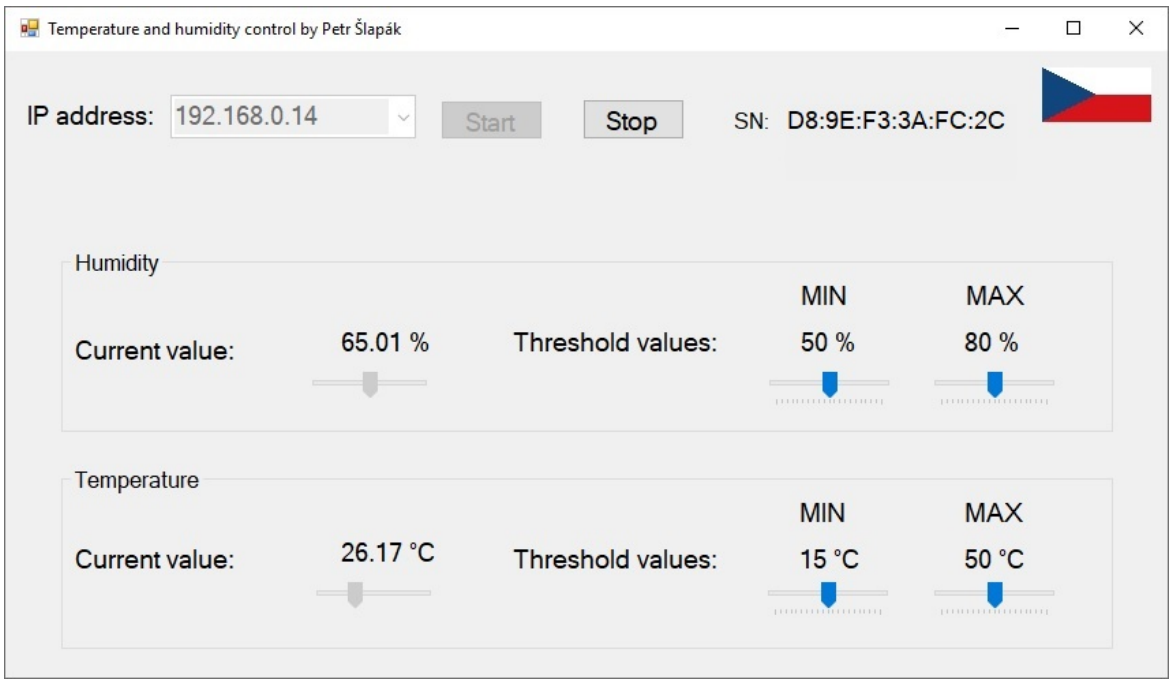
5.3.1 Vzhled aplikace

Po spuštění aplikace uživatelského rozhraní je možné zadat IP adresu sledovaného zařízení a stisknout tlačítko Start. Pokud je na zadané IP adrese dostupné zařízení ke komunikaci, tak dojde k navázání komunikace. Před navázáním spojení je ještě možné přednastavit limity teploty a vlhkosti. IP adresu je možné zadat ručně, nebo vybrat z rozbalovacího pole. V tomto poli je k dispozici výchozí adresa, 192.168.0.88, a dále poté veškeré adresy, na kterých došlo k úspěšnému navázání spojení. Pokud je ale v síti aktivován DHCP, tak je velice pravděpodobné, že zařízení bude mít jinou adresu než výchozí. Tuto adresu je nejprve nutné zjistit. Zařízení tuto adresu komunikuje přes sériový port. Dále je možné zajistit statickou IP adresu pomocí rezervace na DHCP serveru dle MAC adresy zařízení.

Jak je možné vidět na obrázku 13, vzhled aplikace je možné rozdělit na tři jednotlivé bloky. První slouží k základnímu ovládání spojení se zařízením. Obsahuje pole, kam je nutné zadat adresu, či ji vybrat z menu, tlačítka Start a Stop pro ovládání spojení, dále pak pole, kde se zobrazí sériové číslo připojeného zařízení a tlačítko pro přepnutí jazykové mutace.

Druhý a třetí blok je možné rozdělit na dvě části, přičemž první polovina slouží k reprezentaci naměřených hodnot a druhá polovina k nastavování mezí. Druhý blok odpovídá vlhkosti a třetí teplotě.

Obrázek 13: Vzhled aplikace



Zdroj: vlastní zpracování

V druhém a třetím bloku se v části reprezentace naměřených hodnot nachází textové pole, kde je číselně vypsána aktuálně změřená hodnota příslušné veličiny. Pod tímto polem se nachází posuvník, který zobrazuje, kde se hodnota nachází v nastaveném tolerančním poli. Pokud je aktuální hodnota mimo stanovenou hranici, zobrazí se namísto posuvníku chybová hláška, že hodnota je mimo požadované hranice.

Druhá polovina bloku pak slouží k nastavení hraničních hodnot. Pro teplotu je možné volit spodní hranici v rozsahu 5 – 25 °C, horní pak 40 – 60 °C. Pro vlhkost je spodní hranice volena z rozsahu 40 – 60 % a horní 70 – 90 %. Nastavení těchto hodnot je závislé na volbě obsluhy, která má za úkol vyhodnotit, jaká zařízení se v datovém rozvaděči nacházejí a před jakým klimatem je nutné je chránit.

5.3.2 Jazykové mutace

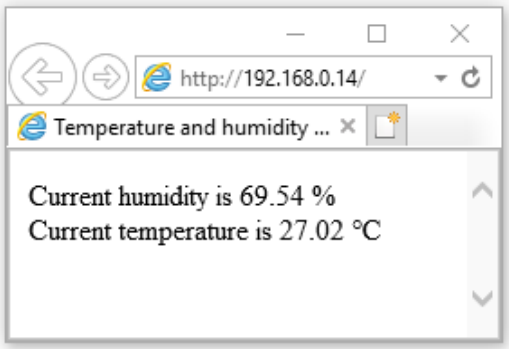
Vzhledem k multinárodnímu rozložení personálu firmy bylo ještě nutné pracovat s jazykovými mutacemi rozhraní aplikace. Pro jednoduchost bylo stanoveno, že aplikace

musí být textována v češtině a angličtině. Jako primární jazyk aplikace byla zvolena angličtina. Tu je poté možné přepnout do češtiny tlačítkem ve tvaru české vlajky v pravém horním rohu aplikace. Posléze umožňuje aplikace opětovné přepnutí do jazyka anglického.

5.3.3 Webové rozhraní

Data jsou ještě zobrazována na jednoduché webové stránce pro přístup k datům bez nutnosti navázání TCP/IP spojení. Webová stránka je vytvořena pomocí HTML. Její vzhled je možné vidět na obrázku 14. Obsahuje pouze data ze senzorů bez možnosti editovat jakákoliv nastavení. Nastavení je nutné provádět pomocí aplikace uživatelského rozhraní. Webová stránka se automaticky obnovuje každých 5 vteřin.

Obrázek 14: Webové rozhraní zařízení



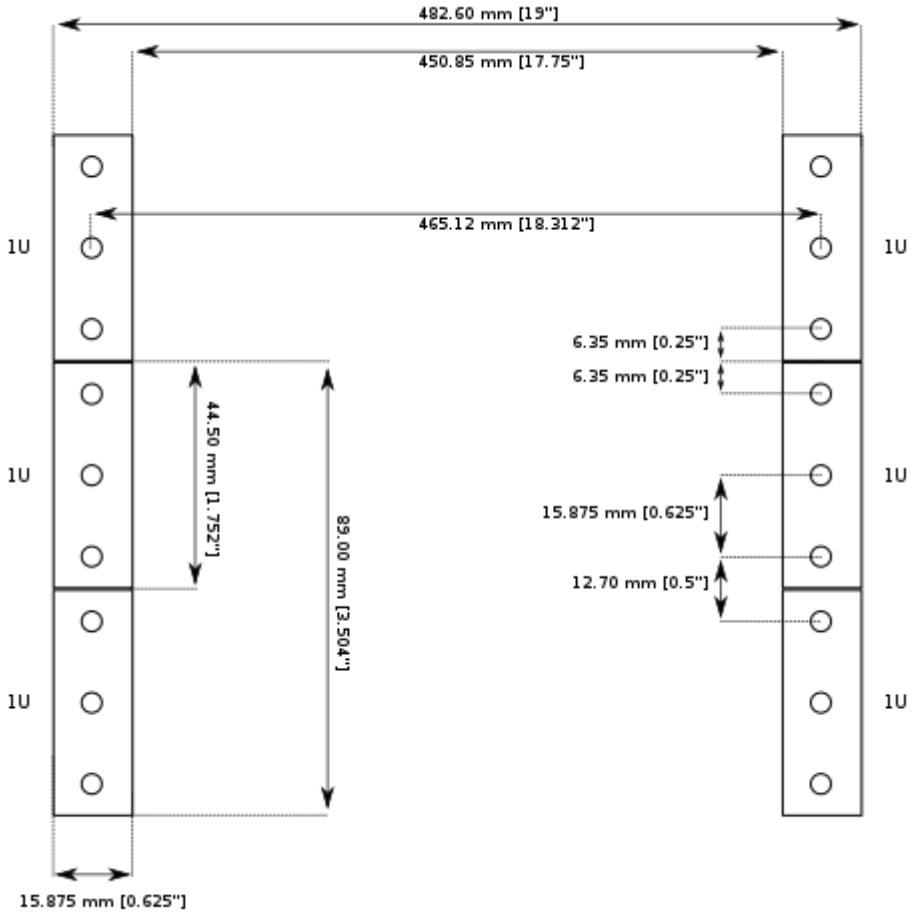
Zdroj: vlastní zpracování

Webové rozhraní je pro jednoduchost napsáno pouze v jedné jazykové mutaci. Slouží spíše pro rychlý náhled z jiného zařízení než z obslužného počítače, kde probíhá sběr dat. Je zároveň multiplatformní a záleží pouze na tom, zda se zařízení nachází ve stejné síti jako přístroj, ze kterého je webové rozhraní otevíráno.

5.4 Kompletace a osazení zařízení

Výsledné zařízení bude umístěno v datovém rozvaděči, takže se nabízí konstrukčně ho upevnit k samotné konstrukci datového rozvaděče. Konstrukce datového rozvaděče obsahuje ližiny, jež mají standardizované rozměry dle obrázku 15. Ližiny se člení na jednotlivé bloky, takzvané 1U (popř. násobky 2U, 4U). Rozteč 1U činí 44,5 mm. Sériově se nevyrábí žádný držák, jež by byl pro Arduino vhodný. Ideálním řešením se nabízí takový držák vyrobit pomocí technologie 3D tisku. Pro snazší zapojení na místě je také vhodné pro zapojení vytvořit Arduino shield.

Obrázek 15: Rozměry lišt datového rozvaděče

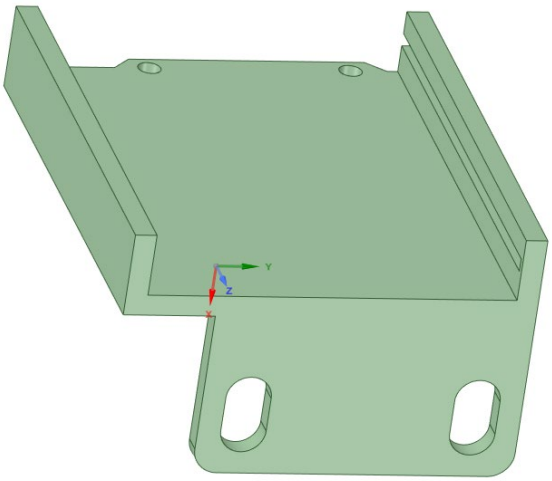


Zdroj: wikimedia.org

5.4.1 Návrh držáku

Základním stavebním kamenem celého zařízení je deska Arduino UNO. Na ní jsou až dále stohovány jednotlivé shieldy. Vhodným řešením je vytvoření držáku, do kterého se umístí samotné Arduino, na které bude uchycen zbytek zařízení. Šířka i výška Arduino UNO převyšuje rozměr 44,5 mm, jež odpovídá rozteči 1U, takže mu bude muset být vyhrazen prostor nejméně 2U. Aby nebylo blokováno v rozvaděči více místa, než je třeba, nebude držák osově souměrný. V držáku budou vytvořeny drážky, kam se celá deska Arduino vsune a na ní se poskládá samotné zařízení. Navrhovaný vzhled vytvořený programem DesignSpark Mechanical je vidět na obrázku 16.

Obrázek 16: Držák Arduino UNO do datového rozvaděče

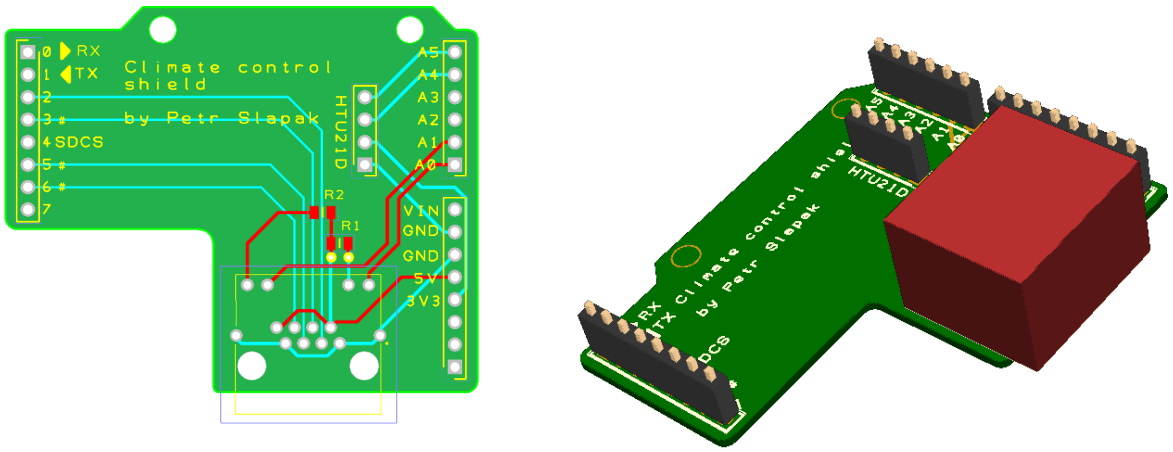


Zdroj: vlastní zpracování

5.4.2 Sestavení zařízení

Arduino UNO a Ethernet shield jsou navzájem v bezprostřední blízkosti, neboť jsou stohovány na sebe. Senzor HTU21D může být umístěn prakticky kdekoliv v datovém rozvaděči. Relay modul bude umístěn v bloku napájení rozvaděče, aby do něj mohlo být zapojeno napájení ventilace a topení. Nebude tedy v bezprostřední blízkosti se zařízením, ale je jej nutno elektricky připojit. Připojení bude realizováno pomocí UTP kabelu. Tyto kabely mají relativně malý elektrický odpor – řádově desítky ohmů na kilometr – a tak jsou pro tento způsob využití vhodné. Pro zjednodušení připojení bude vytvořen vlastní shield, díky kterému bude realizováno připojení senzoru HTU21D a relay modulu pomocí konektoru RJ45. Schéma a 3D vizualizace speciálního shieldu jsou vidět na obrázku 17. Shield byl vytvořen v programu DesignSpark PCB 8.1.

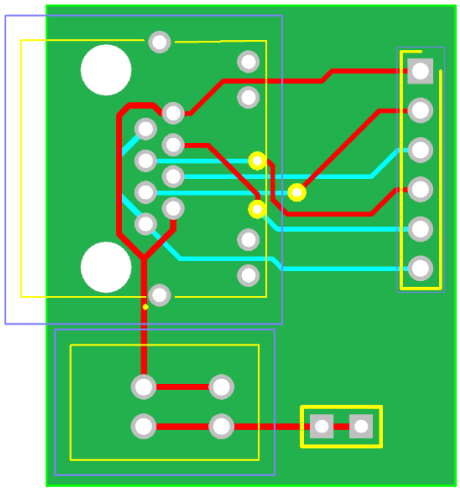
Obrázek 17: Schéma a 3D vizualizace shieldu pro připojení modulů k Arduino UNO



Zdroj: vlastní zpracování

Pro relay modul bude vytvořena redukce z konektoru RJ45 ve formě malé desky plošných spojů Bude tvořena konektorem RJ45 a řadou otvorů, kam se připájí jednotlivé piny relay modulu. Dále budou na desce umístěny dvě svorky od firmy Wago, kam bude možné připojit externí napájení pro případ, že napájení z Arduino nebude dostačovat. Při použití externího napájení se musí napájení přivést na pin JD-VCC. Pro realizaci tohoto propojení bude na desce připraven výstupní pin. Schéma redukční desky je možné vidět na obrázku 18. Konstrukčně bude relay modul uložen na DIN liště společně s dalšími ovládacími prvky napájení. Vzhledem k tomu, že se jedná o komponentu využívanou širokou veřejností, existuje již několik variant držáků na DIN lištu. Pro tento účel tedy nebude třeba vytvářet nový, ale použije se již vytvořený model z www.thingiverse.com.

Obrázek 18: Shield redukční desky pro relay modul



Zdroj: vlastní zpracování

Pro výrobu desek plošných spojů nevlastní firma dostatečné technologie. Aby byla realizace zařízení možná, budou jednotlivé díly vyrobeny externě. Výrobu zajistí čínská firma PCBWay, se kterou firma spolupracuje na několika projektech. Tisk 3D modelů si firma zajistí na vlastních 3D tiskárnách. Všechna schémata a 3D model jsou na CD, jež je přílohou 4 této práce.

6 Výsledky a diskuse

Výsledkem této bakalářské práce je návrh a realizace zařízení pro udržování požadovaných klimatických podmínek uvnitř datového rozvaděče. Zařízení je schopno pracovat autonomně na základě obsluhou stanovených mezí těchto klimatických podmínek. Obsluha ovládá zařízení pomocí aplikace, která je rovněž obsahem této práce. Na obslužném počítači se navíc ukládají naměřená data ve formě CSV souboru, aby je bylo možné dále analyzovat. Zařízení slouží také jako jednoduchý webový server a je možné přistupovat na jeho webovou stránku pro zjištění aktuálních hodnot teploty a vlhkosti uvnitř datového rozvaděče.

Pro zařízení byl vytvořen speciální shield pro praktické připojení zvolených komponent. Pro snadnou montáž byl vytvořen držák pomocí 3D tisku. Model tohoto držáku je umístěn na CD nosiči, jež je součástí této bakalářské práce.

Zařízení bylo testováno v provozu, se vzdáleností 7 m mezi samotným zařízením a relay modulem. Zařízení bylo s relay modulem propojeno FTP kabelem kategorie 5E od firmy Solarix. Tento kabel se prokázal jako vyhovující pro tuto aplikaci. U relay modulu bylo testováno nejmenší napětí, při kterém jsou schopné sepnou relé. Tato relé jsou dimenzována na 5 V, ale testováním bylo prokázáno, že jsou schopna spolehlivě spínat již při 3,3 V. U nižších napětí nebylo sepnutí pod zátěží spolehlivé. V praxi však takový pokles napětí není předpokládán. Nastane-li, tak je možné použít externí napájení relay modulu. Na testované vzdálenosti 7 m nebylo použití externího napájení zapotřebí.

3D tiskárnou vytvořený držák se prokázal být užitečným pro fixní umístnění do datového rozvaděče. Jeho nevýhodou je, že Arduino UNO, a tím i samotný držák, má větší rozměr, než 1U a musí tak zabírat 2U prostoru. Díky tomu se však stává stále dobře přístupným i při plně obsazeném rozvaděči.

7 Závěr

Bakalářská práce je zaměřena na návrh a realizaci zařízení pro řízení teploty a vlhkosti vzduchu v datovém rozvaděči. Na základě teoretických východisek byl vybrán mikrokontroler, vhodný senzor pro měření teploty a vlhkosti, komunikační rozhraní a modul pro připojení aktuátorů. V praktické části je proveden návrh programové obsluhy zařízení a obslužné aplikace. Dále je řešeno umístění zařízení a připojení jednotlivých komponent.

Obslužný program v řídící jednotce byl vyvinut v prostředí Arduino IDE verze 1.8.9 na vývojové desce Arduino UNO. Celý program je součástí přílohy. Program ve smyčce vyhodnocuje aktuální hodnoty teploty a vlhkosti snímané senzorem a porovnává je s nastavenými limity. Limity zařízení obdrží od obsluhy přes komunikační rozhraní, nebo se použijí výchozí hodnoty přímo nastavené v programu. Zařízení s obslužnou aplikací komunikuje pomocí protokolu TCP/IP na portu 2211. Limitní hodnoty přijaté touto cestou si řídící jednotka průběžně ukládá do paměti EEPROM, kde je schopna data uchovat i v případě výpadku napájení. Po spuštění opět načte poslední nastavené limity.

Aplikace obsluhy je vyvinuta v prostředí Microsoft Visual Studio Community 2019 verze 16.4.5. Obsluze umožňuje navázat spojení s řídící jednotkou zařízení pro řízení teploty a vlhkosti vzduchu v datovém rozvaděči a nastavit patřičné limity teploty a vlhkosti vzduchu. Interpretuje naměřené hodnoty a vytváří z nich log v paměti počítače. Aplikace si pamatuje poslední realizovaná připojení a nabídne obsluze výběr z IP adres těchto předchozích připojení. Aplikace rovněž obsahuje dvě jazykové mutace, českou a anglickou.

Program řídící jednotky má výslednou velikost 24 246 Bytů, globální proměnné zabírají 1 135 Bytů dynamické paměti a 913 Bytů zůstává pro lokální proměnné. Je zapotřebí sběrnice SPI, I^2C a čtyř digitálních výstupů. Arduino UNO pro tyto požadavky dostačuje. Hardwarové nároky aplikace jsou minimální. Aplikace potřebuje přibližně 18 MB operační paměti a log celého dne má velikost 28 kB.

Pro použití v praxi byl též vytvořen držák za použití 3D tisku, díky kterému je možné celé zařízení pro řízení teploty a vlhkosti vzduchu v datovém rozvaděči umístit do datového rozvaděče. Dále byl vytvořen speciální shield, díky kterému lze snadno připojit senzor teploty a vlhkosti pouhým zasunutím do patice. Připojení modulu s relé je díky shieldu realizováno pomocí připojení Ethernet patch kabelu.

Po dokončení vývoje byl firmou zaveden testovací provoz v jednom z provozovaných rozvaděčů. Zařízení je plně funkční a splňuje všechny požadavky firmy.

8 Seznam použitých zdrojů

[1] BERAN, Vlastimil, GIRG, Josef, TŮMOVÁ, Olga. *Měření neelektrických veličin*. Plzeň: Západočeská univerzita, 1994. ISBN 80-7082-158-2.

[2] BALÁTĚ, Jaroslav. *Automatické řízení*. Praha: BEN – technická literatura, 2003. ISBN 80-7300-020-2.

[3] KESTER, Walt. *Data conversion handbook*. Boston: Newnes. ISBN 0-7506-7841-0.

[4] KREIDL, Marcel. *Měření teploty: senzory a měřící obvody*. Praha: BEN – technická literatura, 2005. ISBN 80-7300-145-4.

[5] WILSON, Jon S. *Sensor technology handbook*. Boston: Elsevier. ISBN 0-7506-7729-5.

[6] FRADEN, Jacob. *Handbook of modern sensors: physics, designs, and applications*. New York: Springer. ISBN 03-870-0750-4.

[7] MARTINEK, Radislav. Senzory v průmyslové praxi. Praha: BEN - technická literatura, 2004. ISBN 80-7300-114-4.

[8] POKORNÝ, Karel. *Elektrotechnika I*. Praha: Naroma, 2003. ISBN 80-213-1023-5.

[9] EVANS, Martin, NOBLE Joshua, HOCHENBAUM, Jordan. *Arduino in action*. Shelter Island : Manning Publications Co., 2013. ISBN 978-1-617290-24-4.

[10] VODA, Zbyšek. Průvodce světem Arduina. Bučovice: Martin Stříž, 2015. ISBN 978-80-87106-90-7.

[11] *Arduino - Home* [online]. Arduino: © 2020 [cit. 6.3.2020]. Dostupné z: https://www.arduino.cc/

[12] KURNIAWAN, Agus. *Arduino programming with .NET and sketch*. New York: Apress, [2017]. ISBN 978-1-4842-2658-2.

[13] DUKISH, Bob. *Coding the arduino*. New York, NY: Springer Science+Business Media, 2018. ISBN 978-1-4842-3509-6.

[14] *Arduino-shop.cz, VELKOOBCHOD, MALOOBCHOD S ARDUINEM* [online]. ECLIPSERA s.r.o. [cit. 12.3.2020]. Dostupné z: https://arduino-shop.cz/

[15] WHEAT, Dale. *Arduino internals*. Berkeley, CA: Apress, 2011. ISBN 978-1-4302-3883-6.

[16] PETZOLD, Charles. *Programování Microsoft Windows Forms v jazyce C#: [vytváříme uživatelské rozhraní aplikací]*. Brno: Computer Press, 2006. ISBN 80-251-1058-3.

[17] TROELSEN, Andrew W. *Pro C# 2010 and the .NET 4 platform*. New York, NY: Apress. ISBN 978-1-4302-2549-2.

[18] OLSSON, Mikael. *C# 7 Quick Syntax Reference: A Pocket Guide to the Language, APIs, and Library*. New York, NY: Springer Science+Business Media, 2018. ISBN 978-1-4842-3816-5.

[19] PÍSEK, Slavoj. *HTML: začínáme programovat. 3., aktualiz. vyd*. [i.e.] 1. vyd. Praha: Grada, 2010. ISBN 978-80-247-3117-9.

[20] PÍSEK, Slavoj. *HTML: tvorba jednoduchých internetových stránek*. Praha: Grada, 2001. ISBN 80-247-0094-8.

[21] REFSNES, Hege. *Learn HTML and CSS with w3schools*. Hoboken, NJ: Wiley. ISBN 978-0-470-61195-1.

[22] RAMON Manoel C. *Intel® Galileo and Intel® Galileo Gen 2*. Berkeley, CA: Apress, 2014. ISBN 978-1-4302-6839-0.

[23] PUŽMANOVÁ, Rita. *TCP/IP v kostce*. České Budějovice: Kopp, 2004. ISBN 80-7232-236-2.

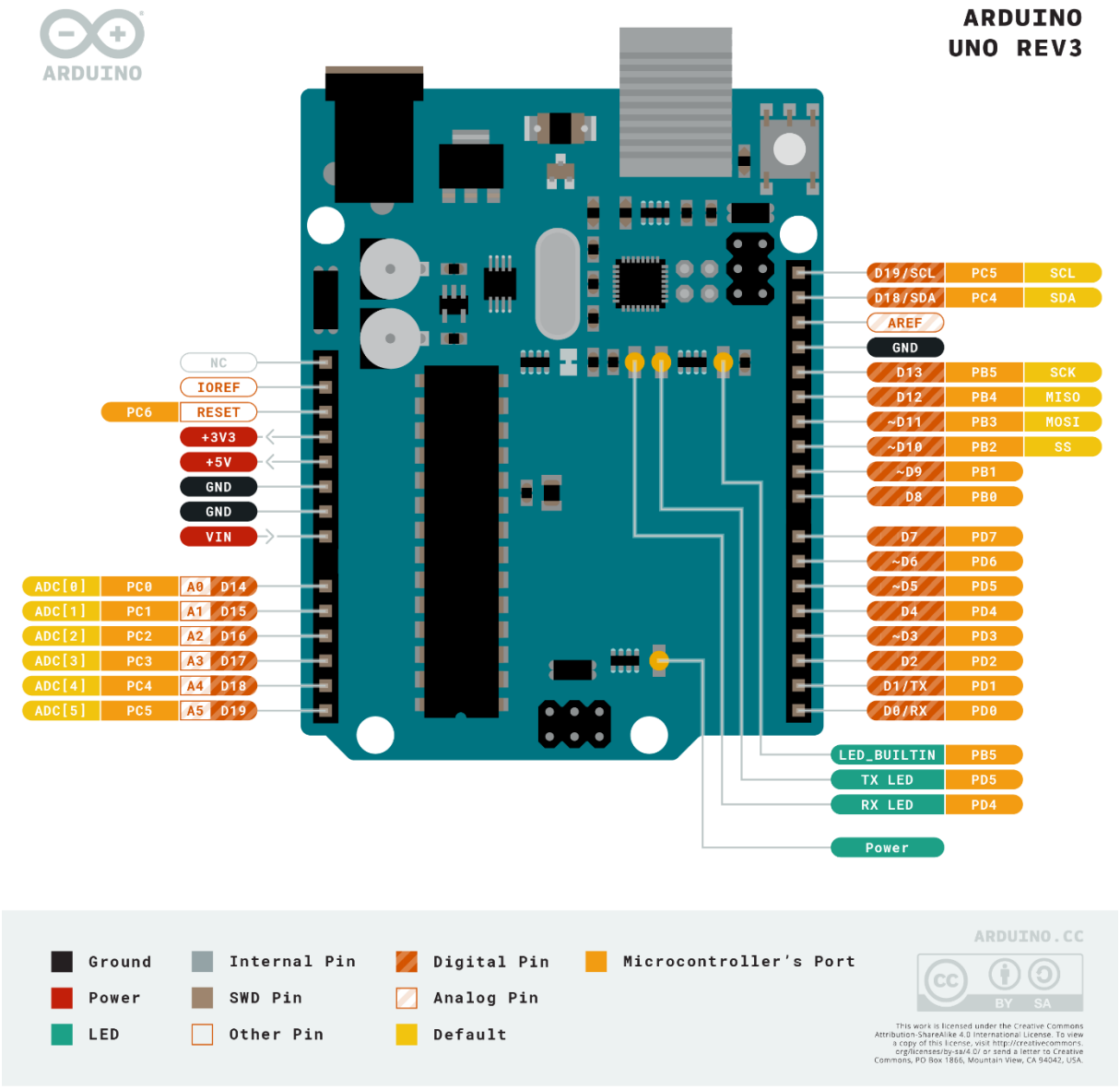
[24] ŠMRHA, Pavel, RUDOLF, Vladimír. *Internetworking pomocí TCP/IP*. České Budějovice: Kopp, 1994. ISBN 80-85828-09-x.

[25] NEUBERGER, Pavel, ADAMOVSKÝ, Daniel, ADAMOVSKÝ, Radomír. *Termomechanika*. Praha: Česká zemědělská univerzita v Praze, 2007. ISBN 978-80-213-1634-8.

9 Přílohy

Seznam příloh

Příloha 1: Arduino UNO Rev3 pinout



Zdroj: store.arduino.cc

Příloha 2: Obslužný program řídící jednotky

```
001 #include <Wire.h>
002 #include <SHT2x.h>
003 #include <Ethernet.h>
004 #include <EEPROM.h>
005
006 EthernetServer serverHTML( 80);
007 EthernetServer serverDotNet(2211);
008 EthernetClient clientHTML;
009 EthernetClient clientDotNet;
010
011 String dataToSend;
012 int tempUpLimit;
013 int tempDownLimit;
014 int humdUpLimit;
015 int humdDownLimit;
016 bool loweringHumd = false;
017 bool loweringTemp = false;
018 bool risingTemp = false;
019
020 byte mac[] = { 0xD8, 0x9E, 0xF3, 0x3A, 0xFC, 0x2C };  // MAC adresa / sériové číslo
zařízení
021 byte ip[] = { 192, 168, 0, 88 };                      // výchozí IP adresa zařízení
022 byte gateway[] = { 192, 168, 0, 1 };                  // výchozí brána
023 byte mask[] = { 255, 255, 255, 0 };                  // maska sítě
024
025 void setup()
026   {
027       loadFromEEPROM ();
028
029       if (Ethernet.begin (mac) == 1) // příkaz vrátí 1, pokud je DHCP k dispozici
030       {
031           Ethernet.begin (mac); //Ethernet se nastaví dle DHCP, pokud je k dispozici
032       }
033       else
034       {
035           Ethernet.begin(mac, ip, gateway, mask); //pokud DHCP není k dispozici, použijí
se výchozí hodnoty
036       }
037
038       Wire.begin(); //otevření portu pro komunikaci se senzorem
039       Serial.begin(9600); //sériový port pro debugging
040       serverHTML.begin(); //otevření portu 80
041       serverDotNet.begin(); //otevření portu 2211
042       Serial.print("server is at ");
043       Serial.println (Ethernet.localIP()); //výpis pro debug - užitečné v případě DHCP
044
045
046       pinMode(2, OUTPUT);
047       pinMode(3, OUTPUT);
048       pinMode(5, OUTPUT);
049       pinMode(6, OUTPUT);
050       digitalWrite(2, HIGH); //topení
051       digitalWrite(3, HIGH); //ventilátor
052       digitalWrite(5, HIGH); //LED signalizace
053       digitalWrite(6, HIGH); //rezerva
054
055   }
056
057   void loop()
058   {
059       //výpis přes sériový port je čistě pro účely ladění
060       Serial.print("Humidity: ");
061       Serial.println(SHT2x.GetHumidity());
062       Serial.print("Temperature(C): ");
063       Serial.println(SHT2x.GetTemperature());
064       Serial.println();
065       regulate ();
066       HTMLpage();
067       appComunication ();
068       saveToEEPROM ();
069       delay(1000);
070   }
071
072 void parseData(String str)
```

```
073   {
074       //vzor přijatých dat: &15&60&30&70*@
075       String parsed = "";
076       int index = 0;
077       int len = str.length();
078       for (int i = 0; i < len; i++)
079       {
080           if (str[i] != '@') //konec řetězce
081           {
082               if (str[i] == '&' ) //oddělující znak
083               {
084                   index++;
085               }
086               else
087               {
088                   switch (index)
089                   {
090                       case 1:
091                           parsed += str[i];
092                           break;
093                       case 2:
094                           //mezikrok po prvním oddělujícím znaku - spodní limit teploty
095                           tempDownLimit = parsed.toInt();
096                           parsed = str[i];
097                           index++;
098                           break;
099                       case 3:
100                           parsed += str[i];
101                           break;
102                       case 4:
103                           //mezikrok po druhém oddělujícím znaku - horní limit teploty
104                           tempUpLimit = parsed.toInt();
105                           parsed = str[i];
106                           index++;
107                           break;
108                       case 5:
109                           parsed += str[i];
110                           break;
111                       case 6:
112                           //mezikrok po třetím oddělujícím znaku - spodní limit vlhkosti
113                           humdDownLimit = parsed.toInt();
114                           parsed = str[i];
115                           index++;
116                           break;
117                       case 7:
118                           parsed += str[i];
119                           break;
120                   }
121               }
122           }
123       }
124
125       //konec algoritmu. Krok po posledním oddělujícím znaku - horní limit vlhkosti
126       humdUpLimit  = parsed.toInt();
127   }
128 /*IO -
129  *
130  * Relay board
131  * 2 - Relay1 - topeni
132  * 3 - Relay2 - ventiltor
133  * 5 - Relay3 - green/red LED
134  * 6 - Relay4 - blue LED
135  *
136  *
137  * HTU210
138  * A4 - DA
139  * A5 - CL
140  *
141  *
142  * Ethernet shield
143  * 4 - SD CS
144  * 10- ETH CS
145  *
146  *
147  *
148  * Relay board - jumper ON + 5V power supply
149  * SHT21 - 3,3V power supply
```

```
150  *
151  */
152
153 void regulate ()
154   {
155       //nastavení ideálních hodnot - polovina regulovaného intervalu
156       int idealTemp = (tempUpLimit + tempDownLimit) / 2;
157       int idealHumd = (humdUpLimit + humdDownLimit) / 2;
158
159       if (SHT2x.GetHumidity() > humdUpLimit || loweringHumd) //stav vysoké vlhkosti
160       {
161           if (SHT2x.GetHumidity() > idealHumd)
162           {
163               //pokud je vlhkost větší, než ideální, tak se sepne ventilátor
164               loweringHumd = true;
165               digitalWrite(3, LOW);
166               if (SHT2x.GetTemperature() < tempUpLimit)
167               {
168                 //pokud zároveň teplota není vyšší, než stanovený maximální limit, sepne i
topení
169                 digitalWrite(2, LOW);
170               }
171           }
172           else
173           {
174               loweringHumd = false;
175               digitalWrite(3, HIGH);
176               if (!risingTemp)
177               {
178                   digitalWrite(2, HIGH);
179               }
180           }
181           if (SHT2x.GetHumidity() > humdUpLimit)
182           {
183               digitalWrite(5, LOW);//rozsvícení chybové signalizace
184           }
185           else
186           {
187               digitalWrite(5, HIGH);//zhasnutí chybové signalizace
188           }
189       }
190
191       if (SHT2x.GetTemperature() > tempUpLimit || loweringTemp) //stav vysoké teploty
192       {
193           if (SHT2x.GetTemperature() > idealTemp)
194           {
195               //pokud je teplota větší, než ideální, tak se sepne ventilátor
196               loweringTemp = true;
197               digitalWrite(3, LOW);
198           }
199           else
200           {
201               loweringTemp = false;
202               digitalWrite(3, HIGH);
203           }
204           if (SHT2x.GetTemperature() > tempUpLimit)
205           {
206               digitalWrite(5, LOW);//rozsvícení chybové signalizace
207           }
208           else
209           {
210               digitalWrite(5, HIGH);//zhasnutí chybové signalizace
211           }
212       }
213
214       if (SHT2x.GetTemperature() < tempDownLimit || risingTemp) //stav nízké teploty
215       {
216           if (SHT2x.GetTemperature() < idealTemp)
217           {
218               //pokud je teplota menší, než ideální, tak se sepne topení
219               risingTemp = true;
220               digitalWrite(2, LOW);
221           }
222           else
223           {
224               risingTemp = false;
225               digitalWrite(2, HIGH);
```

```
            }
            if (SHT2x.GetTemperature() < tempDownLimit)
            {
                digitalWrite(5, LOW);//rozsvícení chybové signalizace
            }
            else
            {
                digitalWrite(5, HIGH);//zhasnutí chybové signalizace
            }
        }
    }


void appComunication ()
    {
        clientDotNet = serverDotNet.available();
        dataToSend = "";
        String dataReceived = "";
        char c = 0x00;
        if(serverDotNet.available())
        {
            while (c != '@')
            {
              // příjem dat, dokun nenarazí na '@'
              c = clientDotNet.read();
              dataReceived += c;
            }
            parseData (dataReceived);
            Serial.print ("tempDownLimit: ");
            Serial.println (tempDownLimit);
            Serial.print ("tempUpLimit: ");
            Serial.println (tempUpLimit);
            Serial.print ("humdDownLimit: ");
            Serial.println (humdDownLimit);
            Serial.print ("humdUpLimit ");
            Serial.println (humdUpLimit);

            //vytvoření řetězce k odeslání
            dataToSend += "&";
            dataToSend += SHT2x.GetTemperature();
            dataToSend += "&";
            dataToSend += SHT2x.GetHumidity();
            dataToSend += "&";
            dataToSend += getMACstring(mac);
            dataToSend += "&";

            clientDotNet.print(dataToSend); // odeslání řetězce
            clientDotNet.stop(); //ukončení spojení
        }
    }
void HTMLpage()
    {

        clientHTML = serverHTML.available();
        if (clientHTML.available())
        {

          //vytvoření jednoduché webové stránky

          clientHTML.println("<!DOCTYPE html>");
          clientHTML.println("<title>Temperature and humidity control by Petr
Slapak</title>"); //název stránky
          clientHTML.println("<html>");
          clientHTML.println("<meta http-equiv=\"refresh\" content=\"5\">"); //automatický
refresh stránky po 5 vteřinách
          clientHTML.print("Current humidity is ");
          clientHTML.print(SHT2x.GetHumidity());
          clientHTML.print(" %");
          clientHTML.println("<br />");
          clientHTML.print("Current temperature is ");
          clientHTML.print(SHT2x.GetTemperature());
          clientHTML.print(" &#8451"); //ASCII kód '°C'
          clientHTML.println("<br />");
          clientHTML.println("</html>");
          clientHTML.stop();
        }
    }
```

```
301
302 String getMACstring (byte adr [])
303   {
304       //metoda pro standardizování tvaru MAC adresy
305       String hexString = "";
306       for (int i = 0; i < 6; i++)
307       {
308           hexString += String(adr[i], HEX);
309           if (i < 5)
310           {
311               hexString += ":";
312           }
313       }
314       hexString.toUpperCase ();
315       return hexString;
316  }
317 void saveToEEPROM()
318   {
319
320       //uložení hodnot a konkrétní místa v EEPROM paměti
321       EEPROM.write(0, tempUpLimit);
322       EEPROM.write(1, tempDownLimit);
323       EEPROM.write(2, humdUpLimit);
324       EEPROM.write(3, humdDownLimit);
325   }
326 void loadFromEEPROM()
327   {
328     //tato metoda ověří, zda načtené hodnoty z paměti EEPROM jsou ve stanovených limitech
a případně je načte. Pokud ne, tak jsou použity výchozí hodnoty
329     if((int) EEPROM.read(0) >= 40 && (int) EEPROM.read(0) <= 60 )
330     {
331       tempUpLimit = (int) EEPROM.read(0);
332     }
333     else
334     {
335       tempUpLimit = 50;
336     }
337     if((int) EEPROM.read(1) >= 5 && (int) EEPROM.read(1) <= 25 )
338     {
339       tempDownLimit = (int) EEPROM.read(1);
340     }
341     else
342     {
343       tempDownLimit = 15;
344     }
345     if((int) EEPROM.read(2) >= 70 && (int) EEPROM.read(2) <= 90 )
346     {
347       humdUpLimit = (int) EEPROM.read(2);
348     }
349     else
350     {
351       humdUpLimit = 80;
352     }
353     if((int) EEPROM.read(3) >= 40 && (int) EEPROM.read(3) <= 60 )
354     {
355       humdDownLimit = (int) EEPROM.read(3);
356     }
357     else
358     {
359       humdDownLimit = 50;
360     }
361   }
```

Zdroj: vlastní zpracování

Příloha 3: Zdrojový kód aplikace obsluhy

```
001 using System;
002 using System.IO;
003 using System.Data;
004 using System.Drawing;
005 using System.Linq;
006 using System.Net;
007 using System.Net.Sockets;
008 using System.Windows.Forms;
009 using System.Net.NetworkInformation;
010 
011 
012 namespace bcToBe
013 {
014     public partial class Form1 : Form
015     {
016         //globální proměnné
017         public int currTemp = 0;
018         public string realTemp = "";
019         public int currHumd = 0;
020         public string realHumd = "";
021         public string adressToConnect = "";
022 
023         public System.Windows.Forms.Timer timer1 = new System.Windows.Forms.Timer();
//časovač pro připojení
024         public System.Windows.Forms.Timer timer2 = new System.Windows.Forms.Timer();
//časovač pro ukládání dat
025 
026         public bool czech = false;
027 
028         public Form1()
029         {
030             InitializeComponent();
031         }
032 
033 
034         private void Form1_Load(object sender, EventArgs e)
035         {
036             //nastavení hranic trackbarů
037             VarSetup();
038 
039             currTemp = trcCurrTemp.Value = (trcCurrTemp.Minimum + trcCurrTemp.Maximum) /
2;
040             realTemp = Convert.ToString(currTemp);
041             lblCurrTemp.Text = realTemp;
042             lblCurrTemp.Text += " °C";
043 
044             currHumd = trcCurrHumd.Value = (trcCurrHumd.Minimum + trcCurrHumd.Maximum) /
2;
045             realHumd = Convert.ToString(currHumd);
046             lblCurrHumd.Text = realHumd;
047             lblCurrHumd.Text += " %";
048 
049             //nastavení angličtiny, jako výchozí hodnot
050             grBoHumd.Text = "Humidity";
051             grBoTemp.Text = "Temperature";
052             lblCurHum.Text = lblCurTemp.Text = "Current value:";
053             lblSetHum.Text = lblSetTemp.Text = "Threshold values:";
054             lblIPadress.Text = "IP address:";
055             lblErrHumd.Text = lblErrTemp.Text = "Current value is out of the regulated
range...";
056             lblErrAddrress.Text = "Invalid IP address!";
057             this.Text = "Temperature and humidity control by Petr Šlapák";
058             flag.Image = Properties.Resources.czech80x40;
059 
060             //načtení IP adres ze souboru
061             try {
062                 foreach (string line in
File.ReadLines(@"C:\Arduino\ClimateControl\listOfAddresses.ini"))
063                 {
064                     comBoIPadress.Items.Add(line);
065                 }
066             }
067             catch {
068                 //pokud soubor neexistuje, tak se vloží defaultní adresa
```

```
069                 comBoIPadress.Items.Add("192.168.0.88");
070             }
071 
072         }
073         private void VarSetup()
074         {
075             //nastavení limitů trackbarů
076             trcCurrHumd.Minimum = trcSetHumdDownLimit.Value;
077             trcCurrHumd.Maximum = trcSetHumdUpLimit.Value;
078             trcCurrTemp.Minimum = trcSetTempDownLimit.Value;
079             trcCurrTemp.Maximum = trcSetTempUpLimit.Value;
080         }
081         private void ParseData(string str)
082         {
083             //vzor řetězce: &22.65&40.75&DE:AD:BE:EF:FE:ED&
084             string parsed = "";
085             int index = 0;
086 
087             for (int i = 0; i < str.Length; i++)
088             {
089                 if (str[i] == '&') //inkrementace pomocného indexu při oddělovači
090                 {
091                     index++;
092                 }
093                 else
094                 {
095                     switch (index)
096                     {
097                         case 1:
098                             if (str[i] == '.') //použití mechanického zaokrouhlení
odříznutím
099                             {
100                                 currTemp = Convert.ToInt32(parsed);
101 
102                                 /*
103                                  *
104                                  * Tělo podmínky se provádí pouze pokud už je načtena
celočíselná část hodnoty
105                                  * Hodnoty za desetinnou čárkou budou ignorovány
106                                  * Způsobí to nepřesnost +1 -0 °C
107                                  * Nepřesnost se ale odrazí pouze v interpretaci trackbaru
108                                  *
109                                 */
110 
111                                 if (currTemp < trcCurrTemp.Minimum || currTemp >
trcCurrTemp.Maximum)
112                                 {
113                                     //v případě, že je hodnota mimo definovaný rozsah je
interpretováno aplikací:
114                                     trcCurrTemp.Visible = false; //odstraněním trackbaru
115                                     lblErrTemp.Visible = true; //zobrazením chybové hlášky
a
116                                     grBoTemp.BackColor = SystemColors.Info; //změnou barvy
groupbaru
117                                 }
118                                 else
119                                 {
120                                     trcCurrTemp.Value = currTemp; //pokud je hodnota v
normě, tak je reprezentována na trackbaru
121                                 }
122                             }
123                             parsed += str[i];
124                             break;
125 
126                         case 2:
127 
128                             /*
129                              *
130                              * Mezikrok mezi zak=odovanými hodnotami
131                              *
132                              */
133 
134                             lblCurrTemp.Text = parsed;
135                             realTemp = parsed;
136                             lblCurrTemp.Text += " °C";
137                             parsed = Char.ToString(str[i]);
138                             index++;
```

```
139                             break;
140
141                         case 3:
142                             if (str[i] == '.')
143                             {
144                                 currHumd = Convert.ToInt32(parsed);
145
146                                 /*
147                                  *
148                                  * Tělo podmínky se provádí pouze pokud už je načtena
celočíselná část hodnoty
149                                  * Hodnoty za desetinnou čárkou budou ignorovány
150                                  * Způsobí to nepřesnost +1 -0 °%
151                                  * Nepřesnost se ale odrazí pouze v interpretaci trackbaru
152                                  *
153                                 */
154
155                                 if (currHumd < trcCurrHumd.Minimum || currHumd >
trcCurrHumd.Maximum)
156                                 {
157                                     trcCurrHumd.Visible = false;
158                                     lblErrHumd.Visible = true;
159                                     grBoHumd.BackColor = SystemColors.Info;
160                                 }
161                                 else
162                                 {
163                                     trcCurrHumd.Value = currHumd;
164                                 }
165                             }
166                             parsed += str[i];
167                             break;
168
169                         case 4:
170
171                             /*
172                              *
173                              * Mezikrok mezi zak=odovanými hodnotami
174                              *
175                              */
176
177                             lblCurrHumd.Text = parsed;
178                             realHumd = parsed;
179                             lblCurrHumd.Text += " %";
180                             parsed = Char.ToString(str[i]);
181                             index++;
182                             break;
183
184                         case 5:
185
186                             parsed += str[i];
187                             break;
188
189                         case 6:
190
191                             lblServerMAC.Text = parsed;
192                             break;
193                     }
194                 }
195             }
196
197             lblServerMAC.Text = parsed; //Konec algoritmu. Uložení poslední hodnoya - MAC
adresa/sériové číslo zařízení
198
199         }
200         private void Timer1_Tick(object sender, EventArgs e)
201         {
202             /*
203              *
204              * Vytvoření řetězce k odeslání
205              *
206              * Spojení limitů hodnot, proložených oddělovačem "&"
207              *
208              * Vzor: &15&60&30&70@
209              *
210              */
211
212             string dataToSend = "&";
```

```
213             dataToSend += Convert.ToString(trcSetTempDownLimit.Value);
214             dataToSend += "&";
215             dataToSend += Convert.ToString(trcSetTempUpLimit.Value);
216             dataToSend += "&";
217             dataToSend += Convert.ToString(trcSetHumdDownLimit.Value);
218             dataToSend += "&";
219             dataToSend += Convert.ToString(trcSetHumdUpLimit.Value);
220             dataToSend += "@";
221 
222             /*
223              *
224              * V rámci navázání spojení dojde k odeslání zprávy a závoveň přijetí odpovědi
225              *
226              */
227 
228             ParseData(Connect(adressToConnect, dataToSend));
229             CheckLimits();
230 
231         }
232         private void Timer2_Tick(object sender, EventArgs e)
233         {
234             /*
235              *
236              * Časovač pro ukládání hodnot do logu
237              *
238              * Složení skrutkrury .csv - data oddělená čarkami
239              *
240              */
241 
242             DateTime thisDay = DateTime.Today;
243             string actDate = thisDay.Day + "." + thisDay.Month + "." + thisDay.Year;
244             thisDay = DateTime.Now;
245             string actTime = thisDay.Hour + ":" + thisDay.Minute + ":" + thisDay.Second;
246             string dataToStore = actTime + "," + realTemp + "," + realHumd + "," +
lblServerMAC.Text + "\r\n"; //enter na konci řádku
247 
248             string filePath = @"C:\Arduino\ClimateControl\Logs\" + actDate + ".csv";
//cesta k souboru je fixně definována
249 
250             if (!File.Exists(filePath))
251             {
252 
253                 // Pokud soubor neexistuje, tak je nutné vytvořit záhlaví
254 
255                 File.AppendAllText(filePath, "Date,Temperature,Humidity,SN" + "\r\n");
256             }
257 
258             //uložení dat do souboru
259             File.AppendAllText(filePath, dataToStore);
260         }
261 
262         private string Connect(string server, string message)
263         {
264 
265             /*
266              *
267              * Nejprve se ověří dostupnost adresy pomocí metody Ping
268              * Pokud vrátí hodnotu - na adrese je nějaké zařízení - aplikace se zkusí
připojit
269              *
270              */
271 
272             Ping ping = new Ping();
273             PingReply pingReply = ping.Send(adressToConnect);
274             string replyString = string.Empty;
275             if (pingReply.Status == IPStatus.Success)
276             {
277                 try
278                 {
279                     /*
280                      *
281                      * Deklarace TCP klienta a streamu
282                      *
283                      * Po připojení je možné otevřít stream a posílat/přijímat data
284                      *
285                      */
286                     TcpClient client = new TcpClient();
```

```
287                     NetworkStream stream;
288
289
290                     Int32 port = 2211;//2211
291                     if (!comBoIPadress.Items.Contains(adressToConnect))
292                     {
293                         //pokud došlo k úspěšnému připojení, tak se adresa uloží do
souboru pro další použití
294                         comBoIPadress.Items.Add(adressToConnect);
295
File.WriteAllLines(@"C:\Arduino\ClimateControl\listOfAddresses.ini",
comBoIPadress.Items.Cast<string>());
296                     }
297
298                   /*
299                    *
300                    * Vytvoření TCP klienta. Musí mu odpovídat server na druhé straně pod
příslušným portem, jinak vyhodí chybu
301                    *
302                    */
303                     client.Connect(server, port);
304                     stream = client.GetStream();
305
306                     //překlad zprávy do Bytového pole v ASCII
307                     Byte[] data = System.Text.Encoding.ASCII.GetBytes(message);
308
309                     stream.Write(data, 0, data.Length);
310
311                     data = new Byte[256]; // zahodí zakédovanou zprávu a připraví se na
příjem dat
312
313
314                     Int32 bytes = stream.Read(data, 0, data.Length); // načtení příchozí
zprávy
315                     replyString = System.Text.Encoding.ASCII.GetString(data, 0, bytes); //
překlad zprávy z ACSII
316
317                     // Uzavření spojení
318                     stream.Close();
319                     client.Close();
320
321                 }
322                 catch
323                 {
324                     /*
325                      *
326                      * Pokud spojení selže, tak se reprezentováno jako chyba spojení
327                      *
328                      */
329
330                     comBoIPadress.Enabled = true;
331                     comBoIPadress.BackColor = SystemColors.Info;
332                     btnStart.Enabled = true;
333                     btnStop.Enabled = false;
334                     lblErrAddrress.Visible = true;
335                     timer1.Stop();
336                     timer2.Stop();
337                 }
338             } else
339             {
340
341                 /*
342                 *
343                 * Pokud Ping selže, tak se reprezentováno jako chyba spojení
344                 *
345                 */
346
347                 comBoIPadress.Enabled = true;
348                 comBoIPadress.BackColor = SystemColors.Info;
349                 btnStart.Enabled = true;
350                 btnStop.Enabled = false;
351                 lblErrAddrress.Visible = true;
352                 timer1.Stop();
353                 timer2.Stop();
354             }
355
356             return replyString;
```

```
357
358         }
359         private bool IsAddressOK(string adr)
360         {
361
362             /*
363              *
364              * Ověření, zda napsaný text odpovídá IP adrese
365              *
366              * Adresa se zkusí parsovat dle standartů IPv4
367              *
368              */
369
370             if (IPAddress.TryParse(adr, out IPAddress address))
371             {
372                 switch (address.AddressFamily)
373                 {
374                     case System.Net.Sockets.AddressFamily.InterNetwork: //případ, kdy je
zadaná adresa v rozsahu stejné sítě
375                         return true;
376                     default:
377                         return false;
378                 }
379             }
380             else return false;
381         }
382         private void BtnStart Click(object sender, EventArgs e)
383         {
384
385             /*
386              *
387              * Stisk tlačítka pro připojení
388              *
389              * Provede se test, zda je adresa dostupná a případně se spustí časovače
390              *
391              * Časovač pro připojení a časovač pro ukládání naměřených hodnot
392              *
393              */
394             lblErrAddrress.Visible = false;
395             adressToConnect = comBoIPadress.Text;
396             if (IsAddressOK(adressToConnect))
397             {
398                 comBoIPadress.BackColor = SystemColors.Window;
399                 btnStart.Enabled = false;
400                 btnStop.Enabled = true;
401                 comBoIPadress.Enabled = false;
402                 timer1.Interval = (5 * 1000); // 5 sec
403                 timer1.Tick += new EventHandler(Timer1_Tick);
404                 timer1.Start();
405
406                 timer2.Interval = (2 * 60 * 1000); // 2 min
407                 timer2.Tick += new EventHandler(Timer2_Tick);
408                 timer2.Start();
409             }
410             else
411             {
412
413                 //pokud není připojení možné, tak je reprezontováno jako chyba
414
415                 comBoIPadress.Enabled = true;
416                 comBoIPadress.BackColor = SystemColors.Info;
417                 lblErrAddrress.Visible = true;
418
419             }
420         }
421
422         private void TrcSetHumdDownLimit_Scroll(object sender, EventArgs e)
423         {
424
425             //nastavení hodnot na základě tahu posuvníkem
426
427             lblSetHumdDownLimit.Text = Convert.ToString(trcSetHumdDownLimit.Value);
428             lblSetHumdDownLimit.Text += " %";
429             CheckLimits();
430             VarSetup();
431         }
432         private void CheckLimits()
```

```
433         {
434
435             //ověření, zda jsou naměřené hodnoty ve stanovených mezích
436
437
438             if (trcSetHumdDownLimit.Value > currHumd || trcSetHumdUpLimit.Value <
currHumd)
439             {
440                 trcCurrHumd.Visible = false;
441                 lblErrHumd.Visible = true;
442                 grBoHumd.BackColor = SystemColors.Info;
443             }
444             else
445             {
446                 trcCurrHumd.Visible = true;
447                 lblErrHumd.Visible = false;
448                 grBoHumd.BackColor = SystemColors.Control;
449             }
450
451             if (trcSetTempDownLimit.Value > currTemp || trcSetTempUpLimit.Value <
currTemp)
452             {
453                 trcCurrTemp.Visible = false;
454                 lblErrTemp.Visible = true;
455                 grBoTemp.BackColor = SystemColors.Info;
456             }
457             else
458             {
459                 trcCurrTemp.Visible = true;
460                 lblErrTemp.Visible = false;
461                 grBoTemp.BackColor = SystemColors.Control;
462             }
463         }
464
465             private void TrcSetHumdUpLimit_Scroll(object sender, EventArgs e)
466         {
467
468             //nastavení hodnot na základě tahu posuvníkem
469
470             lblSetHumdUpLimit.Text = Convert.ToString(trcSetHumdUpLimit.Value);
471             lblSetHumdUpLimit.Text += " %";
472             CheckLimits();
473             VarSetup();
474         }
475
476         private void TctSetTempDownLimit_Scroll(object sender, EventArgs e)
477         {
478
479             //nastavení hodnot na základě tahu posuvníkem
480
481             lblSetTempDownLimit.Text = Convert.ToString(trcSetTempDownLimit.Value);
482             lblSetTempDownLimit.Text += " °C";
483             CheckLimits();
484             VarSetup();
485         }
486
487         private void TrcSetTempUpLimit_Scroll(object sender, EventArgs e)
488         {
489
490             //nastavení hodnot na základě tahu posuvníkem
491
492             lblSetTempUpLimit.Text = Convert.ToString(trcSetTempUpLimit.Value);
493             lblSetTempUpLimit.Text += " °C";
494             CheckLimits();
495             VarSetup();
496         }
497
498         private void BtnStop_Click(object sender, EventArgs e)
499         {
500             /*
501              *
502              * Tlačítko pro ukončení spojení
503              *
504              * Návrat viditelnosti tlačítek do výchozího stavu
505              * Zastavení časovačů
506              * Vyprázdnění pomocných proměnných
507              * Uvedení posuvníků do výchozího stavu
```

```
508             *
509             */
510
511             btnStart.Enabled = true;
512             btnStop.Enabled = false;
513             comBoIPadress.Enabled = true;
514             timer1.Stop();
515             timer2.Stop();
516
517             lblServerMAC.Text = "";
518             currTemp = 0;
519             realTemp = "";
520             currHumd = 0;
521             realHumd = "";
522             adressToConnect = "";
523
524             VarSetup();
525
526             currTemp = trcCurrTemp.Value = (trcCurrTemp.Minimum + trcCurrTemp.Maximum) /
2;
527             realTemp = Convert.ToString(currTemp);
528             lblCurrTemp.Text = realTemp;
529             lblCurrTemp.Text += " °C";
530
531             currHumd = trcCurrHumd.Value = (trcCurrHumd.Minimum + trcCurrHumd.Maximum) /
2;
532             realHumd = Convert.ToString(currHumd);
533             lblCurrHumd.Text = realHumd;
534             lblCurrHumd.Text += " %";
535         }
536
537         private void Flag_Click(object sender, EventArgs e)
538         {
539
540             /*
541              *
542              * Překlady jednotlivých mutací
543              *
544              * Vytvořena česká a anglická jazyková mutace
545              *
546              */
547
548             czech = !czech;
549             if (czech)
550             {
551                 grBoHumd.Text = "Vlhkost";
552                 grBoTemp.Text = "Teplota";
553                 lblCurHum.Text = lblCurTemp.Text = "Aktuální hodnota:";
554                 lblSetHum.Text = lblSetTemp.Text = "Hraniční hodnoty:";
555                 lblIPadress.Text = "IP adresa:";
556                 lblErrHumd.Text = lblErrTemp.Text = "Aktuální hodnota je mimo regulovaný
rozsah...";
557                 this.Text = "Ovládání teploty a vlhkosti od Petra Šlapáka";
558                 lblErrAddrress.Text = "Neplatná IP adresa!";
559                 flag.Image = Properties.Resources.english80x40;
560
561             }
562             else
563             {
564                 grBoHumd.Text = "Humidity";
565                 grBoTemp.Text = "Temperature";
566                 lblCurHum.Text = lblCurTemp.Text = "Current value:";
567                 lblSetHum.Text = lblSetTemp.Text = "Threshold values:";
568                 lblIPadress.Text = "IP address:";
569                 lblErrHumd.Text = lblErrTemp.Text = "Current value is out of the regulated
range...";
570                 this.Text = "Temperature and humidity control by Petr Šlapák";
571                 lblErrAddrress.Text = "Invalid IP address!";
572                 flag.Image = Properties.Resources.czech80x40;
573
574              }
575          }
576
577         private void ComBoIPadress_Enter(object sender, EventArgs e)
578         {
579             //odstranění chybové hlašky, při pokusu o přepis adresy
580
```

```
581             lblErrAddrress.Visible = false;
582             comBoIPadress.BackColor = SystemColors.Window;
583 
584         }
585     }
586 }
```

Zdroj: vlastní zpracování