UNIVERZITA PARDUBICE

FAKULTA ELEKTROTECHNIKY A INFORMATIKY

BAKALÁŘSKÁ PRÁCE

2020 Miloš Janíček

Univerzita Pardubice

Fakulta elektrotechniky a informatiky

Vzdálená správa terárií

Bakalářská práce

2020 Miloš Janíček

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: 2019/2020

ZADÁNÍ BAKALÁŘSKÉ PRÁCE
(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Miloš Janíček**
Osobní číslo:	**I17048**
Studijní program:	**B2612 Elektrotechnika a informatika**
Studijní obor:	**Komunikační a mikroprocesorová technika**
Téma práce:	**Vzdálená správa terárií**
Zadávající katedra:	**Katedra elektrotechniky**

Zásady pro vypracování

Cílem práce je navrhnout systém, který by umožňoval vzdálenou kontrolu a základní řízení prostředí terárií, případně jiných systémů pro chov zvířat. Práce bude obsahovat rešerši možností měření základních veličin terárií a způsobů jejich ovlivňování (řízení), dále bude obsahovat rozbor možností vzdáleného přístupu ať už v rámci lokální sítě, tak mimo ni. Praktická část bude obsahovat návrh, konstrukci a oživení jednoduchého modulu pro řízení a monitoring teária umožňující vzdálenou správu alespoň po lokální síti.

Rozsah pracovní zprávy: **30-60**
Rozsah grafických prací:
Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

[1] VÁŇA, V. Mikrokontroléry ATMEL AVR: popis procesoru a instrukční soubor. Praha: BEN technická literatura, 2003.336 s. ISBN 978-80-7300-083-0.
[2] VÁŇA, V. Mikrokontroléry ATMEL AVR: programování v jazyce C. Praha: BEN technická literatura, 2003. 216 s. ISBN 978-80-7300-102-0.
[3] VLACH, J. Řízení a vizualizace technologických procesů. Praha: BEN technická literatura, 2002. 160 s. ISBN 978-80-86056-66-X.
[4] BRTNÍK, B. Základní elektronické obvody. Praha:BEN technická literatura, 2011. 156s. ISBN 978-80-7300-408-8
[5] RIPKA, P.; TIPEK, A. Master Book of Sensors. Praha : BEN, 2003. ISBN 0-12-752184

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

Datum zadání bakalářské práce: **15. listopadu 2019**
Termín odevzdání bakalářské práce: **7. května 2020**

Univerzita Pardubice Česká republika
L.S.

Ing. Zdeněk Němec, Ph.D.
děkan

Ing. Jan Pidanič, Ph.D.
vedoucí katedry

V Pardubicích dne 17. prosince 2019

Prohlašuji:

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

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

Beru na vědomí, že v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, a směrnicí Univerzity Pardubice č. 7/2019 Pravidla pro odevzdávání, zveřejňování a formální úpravu závěrečných prací, ve znění pozdějších dodatků, bude práce zveřejněna prostřednictvím Digitální knihovny Univerzity Pardubice.

V Pardubicích dne 15. 5. 2020

Miloš Janíček

PODĚKOVÁNÍ

Tímto bych rád poděkoval panu Ing. Rozsívalovi za vedení bakalářské práce a za znalosti, které mi byly předány. Dále bych rád poděkoval všem učitelům, kteří mě po dobu mého studia vzdělávali a předávali jejich zkušenosti a znalosti.

ANOTACE

Práce se zabývá propojením počítačového serveru a mikrokontroleru za účel řízení terária. Řídit terárium lze poté lokálně, pomocí přímého přístupu na server, ale také vzdáleně skrze veřejně dostupné internetové stránky. Terárium poté obsahuje velmi často používané prvky, jako je teploměr, vlhkoměr, žárovku a topnou podložku. Tyto prvky jsou poté podle účelu obsluhovány mikrokontrolerem.

KLÍČOVÁ SLOVA

terárium, vzdálená správa, regulace teploty a světla, měření teploty a vlhkosti

TITLE

Remote terrarium management.

ANNOTATION

The work deals with the connection of a computer server and a microcontroller for the purpose of terrarium control. The terrarium can then be managed locally, using direct access to the server, but also remotely through a publicly available website. The terrarium then contains very frequently used elements, such as a thermometer, hygrometer, light bulb and heating pad. These elements are then operated by a microcontroller according to their purpose.

KEYWORDS

terrarium, remote controll, heat and light regulation, temperature and humidity measurement

OBSAH

PODĚKOVÁNÍ 6

ANOTACE 7

KLÍČOVÁ SLOVA 7

TITLE 7

ANNOTATION 7

KEYWORDS 7

OBSAH 8

SEZNAM ILUSTRACÍ 10

SEZNAM TABULEK 10

SEZNAM ZKRATEK A ZNAČEK 11

TERMINOLOGIE 12

Internet věcí (IoT) 12

ÚVOD 13

1. Rozbor problematiky 14

2. Sledované veličiny v teráriu 15

2.1. Teplota 15

2.2. Vlhkost 15

3. Řízená zařízení 15

3.1. Tepelné zdroje 15

3.2. Světelné zdroje 16

4. Zpracování dat a přímé řízení terária 17

4.1. Kombinované řízení 17

4.2. Jednotka přímého řízení - DCU 17

5. Centrální systém 18

5.1. Hardware 19

5.2. Operační systém 19

5.3. Aplikace 19
6. Výsledný návrh řešení 20
7. Kompletace řešení 21
7.1. Komunikační rozhraní DCU a Centrálního serveru 21
7.2. DCU terária a senzory 22
7.3. DCU terária a řízené zdroje 23
7.4. Centrální systém a autonomní řízení 24
7.4.1. Příprava hardwaru 24
7.4.2. Instalace OS 24
7.4.3. Instalace WampServeru 24
7.4.4. Modifikace konfigurace pro přístup z LAN 24
7.4.5. Centrální systém – webové stránky 26
7.4.6. Datové balíčky a komunikační rozhraní s DCU 27
7.4.7. Autonomní řízení 30
7.5. Přístup operátora do systému 31
7.6. Vzdálené řízení centrálního systému 32
8. Shrnutí řešení a demonstrativní ukázka 34
ZÁVĚR 37
POUŽITÁ LITERATURA 38
Bibliografie 38
PŘÍLOHY 39
PŘÍLOHA A – Testovací kód pro ověření funkčnosti Ethernetového modulu 39
PŘÍLOHA B – httpd-vhosts.conf 1.část 41
PŘÍLOHA C – httpd-vhosts.conf 2.část 42
PŘÍLOHA D – Kód obnovující IP adresy v ARP tabulce 43
PŘÍLOHA E – Ukázka části struktury Entity Terária 44

SEZNAM ILUSTRACÍ

Obrázek 1 - diagram návrhu řešení 20
Obrázek 2 - Originální záznam v httpd.conf 25
Obrázek 3 - přidané řádky do httpd.conf 25
Obrázek 4 - část metody zasílající dotaz DCU o data 28
Obrázek 5 - část funkce těžící data z balíčku 29
Obrázek 6 - obnova dat z terárií 30
Obrázek 7 - metoda spouštějící funkci kontroly na terárium 31
Obrázek 8 - Demonstrativní terárium obsahující data ze senzorů 32
Obrázek 9 - Příprava dat k odeslání 33
Obrázek 10 - seznam úloh CRONu 34
Obrázek 11 - řídící obvod zátěže 35

SEZNAM TABULEK

Tabulka 1 - logické stavy v řízení 17

SEZNAM ZKRATEK A ZNAČEK

IoT – internet věcí

DCU – jednotka přímého řízení

IDE – vývojové prostředí

LAN – lokální síť

TERMINOLOGIE

Internet věcí (IoT)

Internet věcí je věčně opakované téma. Většinou je spojováno s Průmyslem 4.0, ve kterém dochází k automatizacím. Hlavní myšlenkou Internet of Things (dále jen IoT) je vzájemná komunikace zařízení. Dále by zařízení měla být schopna si s dalšími zařízeními vyměňovat různá data, ale také v rámci internetu fungovat samostatně. Tato síť zařízení má velikou perspektivu uplatnění. Aktuálně je již tato síť postupně aplikovaná. Jako předním developerským projektem jsou takzvaná „Chytrá města“. Tento projekt má jako jeden z cílů zefektivnit dopravní situace v centrech měst. Mohlo by to být použito pro řízení semaforových časovačů, které by mohli komunikovat s kamerovým systémem, který by identifikoval počet aut. Jako další výhoda vzniká v automatizaci výrobních procesů v průmyslu, kde bude docházet k přesunu repetitivních činností z lidských zdrojů na mechanické. Toto může výrobcům pomoci hlavně z několika důvodů. Klesají režijní náklady (mzdy), ale hlavně stroje neznají každodenní únavu z fyzické činnosti. Dochází u nich však k opotřebení a díky tomu je nutná údržba.

ÚVOD

Dnešní doba umožňuje lidem si pořizovat stále širší okruh zvířat jako domácí mazlíčky. Mezi tato zvířata s rozšiřující se oblibou spadají i terarijní exotická zvířat. Ta však k životu potřebují specifická prostředí s určitou charakteristickou teplotou, dobou a typem světelného záření, rozmanitostí i vlhkostí vzduchu. Zde však již vyvstává otázka, jak dané veličiny kontrolovat a řídit? Na trhu nalezneme veliké množství senzorů, tepelných lamp či podložek, časových spínačů apod. Avšak, vše je řízeno manuálně a zakládáno na znalostech chovatele. Pro běžnou domácnost, kde je terarijní zvíře pouze mazlíčkem a počet terárií je velmi nízký, není manuální obsluha až tak náročná. Problém však nastává u chovatelů, kteří mají terária v řádech desítek. Zde již celková obsluha je časově i fyzicky náročná.

Pokrok techniky nám již pomalu a jistě přináší technologii IoT, kterou již používají některé „chytré domácnost“. Tímto se nám tedy rozšiřují možnosti aplikace jednotlivých čidel pro větší rozlohy (např. dům) a řízení jednotlivých technických produktů (např. dálkově řízené osvětlení). Tato technologie nám tedy umožňuje sestrojit různá řídící centra pro větší prostory (domy, firemní podniky apod.).

Tato práce se bude zabývat možným řešením vzdáleného řízení terárií a jeho výslednou aplikací. Jelikož každý terarijní druh zvířete, vyžaduje specifická prostředí, bude tato práce bez specifického zaměření na druh zvířete pro její širší uplatnění, avšak výsledné praktické využití uvedené v této práci bude pro druh gekončíka, Gekončík Noční. Jedná se totiž o nenáročnou ještěrku na chov, díky čemuž je velmi často chována jako terarijní zvíře.

1. Rozbor problematiky

Pokud rozebereme problematiku vzdáleného řízení terárií na menší segmenty, všimneme si několika podobností s již využívanými principy IoT pro domácnosti.

Prvním určením zjistíme, že je potřebné sledovat několik parametrů prostředí terária. Mezi ta patří teplota a vlhkost vzduchu. Na tyto fyzikální veličiny existuje nepřeberné množství senzorů, lišící se cenou, místem uplatnění, napájecím napětím, rozsahem měřené hodnoty.

Jako další segment lze uvést zpracovávání dat ze senzorů a následné odeslání do řídícího systému. Tento segment se do značné části odvíjí od použitých senzorů pro měření veličin a má na výběr senzorů vliv. Proto jsem se rozhodl využít pro vývoj řešení Arduino Nano, z důvodu snadné manipulace, modulárnosti a nízké pořizovací ceny.

Komunikace mezi jednotkou zpracování dat a řídícím systémem je dalším dílem skládačky.

Velkým segmentem je následně řídící systém jako takový. Tento systém by měl být částečně autonomní, ale také připraven pro manuální obsluhu. Autonomní část systému by měla zamezovat kritickým chybám ve vývoji sledovaných veličin. Toto autonomní opatření by mělo zabránit případném úmrtí zvířete z důvodů nevyhovujících podmínek prostředí. Manuální řízení by mělo poté sloužit k úpravě sledovaných veličin na základě specifických podmínek (např. zazimování zvířete).

Za poslední segment pak lze označit komunikaci mezi chovatelem a řídícím systémem. Zde se nabízí několik možností a jejich řešení.

2. Sledované veličiny v terárii

Pro různé druhy terarijních zvířat lze sledovat různé veličiny. Pro druh, který byl vybrán pro příklad této práce, jsou nejdůležitějšími veličinami teplota a vlhkost ovzduší. Tyto 2 veličiny jsou těmi nejdůležitějšími pro většinu exotických druhů, chované v teráriích.

2.1. Teplota

Teplotní faktor je pro mnoho terarijních tím nejdůležitějším faktorem. Jelikož se Česká Republika nachází v mírném podnebném pásu, tak zde jsou životní podmínky pro exotická zvířata nevyhovující. Primární příčinou je teplotní deficit, který zde většina exotických živočišných druhů má. Gekončík noční obvykle obývá pouštní oblasti v Afghánistánu, Indii, Íránu a Iráku. [1] Z tohoto důvodu, je potřeba terária, v nichž je tento Gekončík chován, dostatečně vyhřívat pro udržení optimální teploty pro život Gekončíka nočního.

Ke sledování této veličiny lze na trhu elektroniky nalézt široké spektrum senzorů. Většina z těchto senzorů je použitelných pro sledování v potřebném rozsahu. Jako primární parametr výběru je tedy zvolena cena a snadnost použití.

2.2. Vlhkost

Další velmi významnou veličinou životního prostředí v teráriích je vlhkost. Vlhkost ovzduší lze měřit buď jako absolutní nebo relativní. Absolutní vlhkost vyjadřuje skutečné množství vodních par zastoupených ve vzduchu. Relativní vlhkost udává procentuální zastoupení vůči maximálnímu možnému nasycení vodních par. Tato veličina je výrazně závislá na teplotě. S rostoucí teplotou klesá schopnost vzduchu absorbovat vodní páry.

3. Řízená zařízení

Mezi nejpoužívanější zařízení ovlivňující životní prostředí pro chov terarijních zvířat patří tepelný zdroj (topná podložka, topný kámen) a světelný zdroj (zářivka nebo tepelná žárovka). Žárovky vytváří ideální lokální teplejší místa v terárii, které využívají zvířata k ohřátí. Čistě tepelné zdroje se využívají především pro udržení teploty v terárii v noci nebo ve studenější místnosti.

3.1. Tepelné zdroje

Jako tepelné zdroje zde uvažuji pouze čistě tepelné zdroje, nikterak kombinované. Tyto tepelné zdroje fungují na stejném principu. Jedná se o odporový prvek (většinou vodič ve tvaru desky) s regulátorem pro použití v síťovém napětí. Tento zdroj je napájen síťovým napětí 230 V/50 Hz. Na odporovém prvku dochází k výkonové „ztrátě“, která je přímo

úměrná odporu prvku a proudu protékajícím skrze odporový prvek. Ztrátový výkon je ve formě tepla a z prvku uniká do okolí, čímž dochází k ohřevu podloží (desky se pro větší účinnost zasypávají např. do písku). V mnou vybraném řešení by se dalo uvažovat nad řízení pomocí relé či dvojice tranzistorů. Jelikož je tranzistor pro sinusové napětí z půli vodivý (v propustném režimu), bylo by nutno vytvořit dvojici. Na trhu nalezneme velké množství rozměrů i výkonů tepelných zdrojů, řádově jednotky až desítky Watt-ů. Pokud budeme uvažovat např. 45 W tepelnou podložku, můžeme si za pomocí vztahu (1) spočítat proud protékající topnou podložkou.

$$P = U_{ef} * I_{ef} * cos\varphi \tag{1}$$

Jelikož se v obvodu nenachází nelineární prvek, lze považovat účiník jako roven jedné. Dostaneme se ke vztahu.

$$P = U_{ef} * I_{ef} \tag{2}$$

Tento vztah je stejný jako pro stejnosměrný proud pouze s rozdílem, že zde počítáme efektivní hodnoty napětí i proudu. Po dosazení hodnot dostaneme, že efektivní hodnota proudu je zhruba *196 mA*. Pokud tuto hodnotu dosadíme do vzorce (3), dostaneme maximální proud protékající obvodem.

$$\sqrt{2} * I_{ef} = I_{max} \tag{3}$$

Po dosazení efektivní hodnoty proudu, obdržíme *277,2 mA*. Tento proud není nikterak vysoký, proto by bylo možné použít dvojici tranzistorů i relé. Rozhodující však je napětí. Běžně používané tranzistory nelze pro napětí *325 V* použít. Z tohoto důvodu a také z důvodu menšího namáhání relé, jsem se rozhodl jako řídící prvky použít relé. Relé je nezávislé na polaritě napětí. Funguje pouze jako spínač.

3.2. Světelné zdroje

Mezi světelné zdroje lze uvést bodové žárovky, pro vytvoření lokálního osvětlení a výhřevného místa a zářivky pro vytvoření celkového osvětlení (simulace dne).

V případě použití žárovek a průzkumu jejich zastoupení na trhu lze nalézt od 10 W až po 200 W. Vyšší výkony žárovek jsou již velmi individuální. Pro demonstrativní terárium jsem vybral žárovku o výkonu *50 W*. Pokud tedy použijeme vztahy (1) a (3) a dosadíme za ní

hodnotu výkonu žárovky, zjistíme maximální proud. Tento proud je zhruba *307 mA*. Pro řízení světelných a tepelných zdrojů jsem tedy vybral relé typu SRD-05VDC-SL-C. Toto relé je řízeno napětím *5 V* a proudem *70 mA*. Přípustný maximální proud tohoto relé je tedy více než dostatečný pro použití v této práci.

4. Zpracování dat a přímé řízení terária

V této části bych rád probral základní myšlenku řízení veličin životního prostředí v teráriu. Prvotní myšlenkou bylo hybridní řízení. Manuální v kombinaci s automatickým. Zde vyvstával však problém s již zvoleným relé, jak takovéto relé řídit. Jako první se naskytla možnost, hierarchického rozložení přednosti. Tímto by bylo manuální řízení nadřazeno autonomnímu a pokud by manuální spínač byl vypnutý, nebylo by možné daný prvek autonomně řídit (nebo řídit vzdáleně). Zde by však vznikal prostor pro chybu člověka, kde by mohlo dojít k opomenutí zapnutí a tím by nebylo možné např. dostatečně vytápět terárium a mohlo by dojít k úhynu chovu.

4.1. Kombinované řízení

Jako řešení jsem tedy vymyslel kombinované softwarové řízení v nezávislosti jednotlivých řízení. Toto kombinované řízení bude spočívat v paralelním zapojení relé a manuálního vypínače. Pokud bude operátor chtít nechat řízení na autonomním řízení či vzdálené správě, bude manuální vypínač stále v rozepnutém stavu. Pokud by došlo k výpadku systému, bude tímto zapojením zachována možnost manuálně zapnout či vypnout daný teplený či světelný zdroj. Toto zapojení lze pak charakterizovat pomocí logické funkce OR.

Logický stav vypínače	Logický stav relé	Zdroj
0	0	vypnuto
0	1	zapnuto
1	0	zapnuto
1	1	zapnuto

Tabulka 1 - logické stavy v řízení

4.2. Jednotka přímého řízení - DCU

Jako vývojovou řídící jednotku jsem se rozhodl použít Arduino Nano. Tento programovatelný kit jsem vybral pro jeho modulárnost, snadnou programovatelnost pomocí Arduino IDE a

nízkou pořizovací cenu. Výstupní proud na jednotlivém pinu je *40 mA*. Pro správnou funkčnost relé je však potřeba *70 mA*. Proto pro výsledné řešení bude využito tranzistorů NPN, které budou spínacími prvky pro jednotlivá relé. Tímto bude dodržena myšlenka řízení skrze relé a také dostatečné napájení pro relé. Tranzistor bude spínací prvek mezi napájením a pinem na relé.

Jako informační prvky o stavu prostředí budu používat jednoduché senzory teploty a vlhkosti. Z těchto prvků bude docházet ke čtení dat a překlad do SI jednotek, tyto informace budou poté odeslány na server, kde dojde k dalšímu zpracování. V případě příjmu dat, dochází k přijmu pouze řídících pokynů, podle nich jsou pak nastaveny výstupní piny.

Pro spojení mezi centrálním systémem a DCU je možné použít 3 existující technologie. Wi-fi, Bluethoot a Ethernet. Všechny tyto technologie by byly použitelné pro přenos dat. U bezdrátových technologií by nebyl problém s kabeláží k síťové jednotce (např. switch). Avšak bezdrátová zařízení jsou komplexnější na softwarová řešení. Proto jsem se rozhodl využít technologii Ethernet a propojení přes UTP kabely. Tímto řešením lze využít technologii PoE, která bude sloužit jako napájení pro DCU. Pokud vezmeme celkové proudové zatížení všech dílů, dostaneme se odhadem maximálně k *0,5 A*. Zde největší zátěží budou relé. V případě jejich zapnutí (velmi častá situace), bude celkový proud tekoucí těmito relé *210 mA*. Arduino Nano spotřebovává při zátěži *20 mA*. Pro případné další napájení senzorů je zde velký prostor. Jelikož však vodič v páru v UTP kabelu je malého průřezu, nelze zde použít větší proudy. V případě běžně dostupného UTP kabelu o průřezu vodiče 0,45 mm, je maximální přípustný proud zhruba *0,457 mA*. [2] Tento proud je však nad uvažovanou teoretickou hodnotou. Avšak pro použití kroucené dvoulinky pro přenos informací pomocí Ethernetu stačí 2 páry. Z tohoto hlediska pak lze tedy využít zbývající 2 páry pro přenos elektrické energie. Pro každou polaritu bude tak použit jeden pár vodičů, což zdvojnásobuje maximální přípustný proud tekoucí vodiči.

5. Centrální systém

Jako centrální systém lze použít samostatný server ve formě mikropočítače (např. Raspberry Pi) nebo i složitější techniku ve formě klasického serveru/počítače. Pro zvolení vhodného řešení jsem stanovil jako stěžejní následující faktory. Pořizovací cena, náročnost na údržbu, náročnost rozšíření a drobných úprav.

	Raspberry Pi 4	Server podobný PC	Starý PC, úpravený
Pořizovací cena	Nízká – zhruba 2 tis.	Vysoká – od 9 tis.	Nízká až nulová
Náročnost údržby	Vyšší – kompaktní zařízení	Nízká – často velmi dostupný	Nízká – často velmi dostupný
Náročnost rozšíření a drobných úprav	Vyšší – způsobeno kompaktností	Nízká – lze snadno hardware upravit	Nízká – hardware je starší a dostupný

Tabulka č. 2 – porovnání serverových řešení

Na základě tabulky *Tabulka č. 1* jsem se rozhodl pro využití starého počítače. Je to především z důvodu snadné upravitelnosti hardwaru, nízkých pořizovacích nákladů (v mém případě nulových) a snadnému rozšíření pro další účely.

5.1. Hardware

Jako základní hardware využiji starý notebook. Přestala mu fungovat obrazovka. Proto mu byla odebrána. Jelikož je nyní nepoužívaný, není prioritou dlouhodobá funkčnost. Pro finální a širší využití by bylo vhodnější využít kvalitnější sestavu.

5.2. Operační systém

Aktuálně se nabízí dvě hlavní možnosti. Využití OS Windows server nebo Linux. Použitím Linuxu lze získat větší kontrolu nad systémem, avšak ne všechny aplikace by bylo možné použít. Windows server redukuje přímou pravomoc nad systémem, avšak je zde využitelnost více aplikací a podpora C# skriptů. Proto jsem se rozhodl pro aktuální řešení využít Windows server se studentskou licencí.

5.3. Aplikace

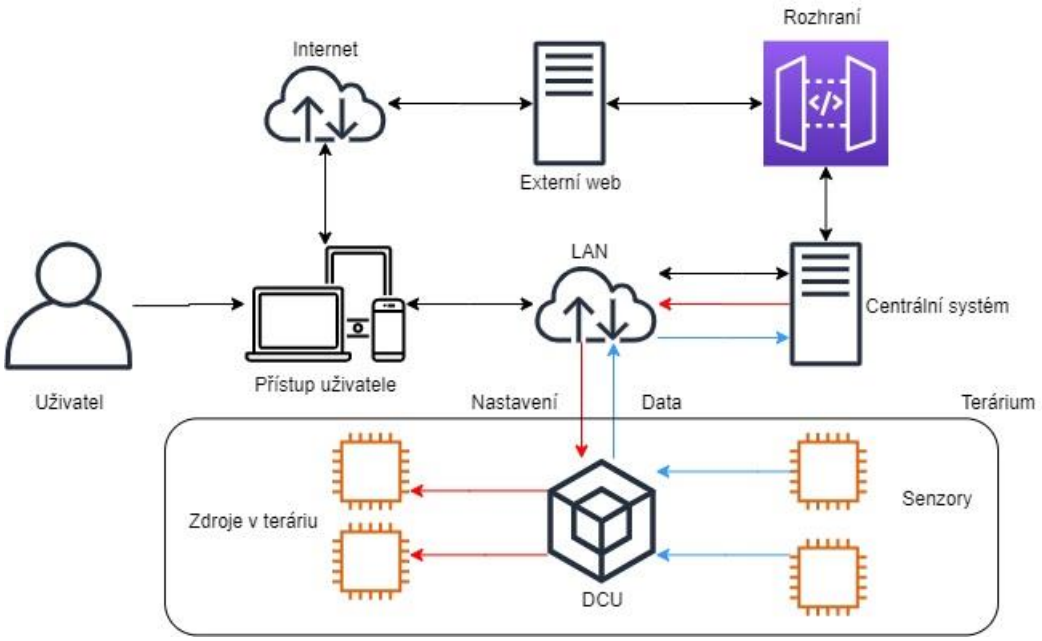
Jako hlavní aplikaci pro zařízení databáze a základní uživatelský přístup a správu, jsem se rozhodl využít WampServer. Toto řešení je volně dostupné, má snadné použití a velkou možnost úpravy parametrů virtuálního serveru. Dalším důvodem je osobní zkušenost a praxe v používání toto řešení.

GUI bude řešeno ve formě lokálních webových stránek. Veškerá data budou ukládána do databáze a data budou uživateli/chovateli dostupná přes zaheslované webové stránky s kontrolním rozhraním pro terárium. S tímto řešením mám vlastní několikaleté zkušenosti. Další výhodou tohoto řešení, je snadná dostupnost v lokální sítí. V případě výpadku připojení LAN do internetu, je toto řešení stále snadno dostupné. V dnešní době používá smartphone veliké procento populace. Případně se v domácnostech nachází PC/notebook. Další předností

je snadná propojitelnost s internetem. Zde lze využít VPN, ne však všechny VPN jsou dostupné na smartphone. Proto se jako další řešení naskytuje použit free-hosting a vytvoření klientských webových stránek. Toto řešení by mohlo být pojato ve stylu API. Lze pak doprogramovat do lokálního serveru propojení s webovými stránkami, které jsou dostupné na internetu. Toto řešení má pak výhodu v bezpečnosti na kterou je kladen důraz. Jelikož se bude jednat o obousměrnou komunikaci, avšak s omezením stahování dat do lokálního centrálního serveru. Toto bude řešeno prostou výměnou dat mezi databázemi. Jelikož se jedná o velmi malou databázi, lze využít freehostingu který je hojně dostupný i v české lokalizaci internetu.

Hlavním komunikačním prvkem mezi řídící jednotkou a centrální serverem, bude zajištěno C# scripty, které budou zajišťovat výměnu dat a řídících informací.

6. Výsledný návrh řešení



Obrázek 1 - diagram návrhu řešení

Výsledný návrh je tedy tvořen ze senzorů a zdrojů (tepla, světla) připojených napřímo do jednotky přímého řízení. Tato jednotka je pak přes Ethernetový modul připojena do LAN. Do stejné LAN je poté zapojen i centrální systém, který zajišťuje obsluhu všech jednotek v síti a také zpracovává dotazy od uživatele skrze webové stránky. Uživatel může přistupovat na tyto webové stránky pouze z lokální sítě. V případě, že přistupuje skrze internet na externí web, je zde API, které obsluhuje centrální systém. Toto API zajišťuje potřebnou výměnu dat. Skrze lokální webové stránky je uživatel pak schopen provádět správu všech komponent na centrálním systému (databáze terárií, jednotek, senzorů jednotek atd.) a také správu samotných

terárií (tedy řízení zdrojů v teráriích). Pomocí externích webových stránek, je uživatel oprávněn pouze řídit terária. Toto je hlavně z hlediska bezpečnosti i logiky práce s tímto systémem (terárium lze připravit pouze v místě, kde se nachází další jednotky, tedy LAN). Centrální systém poté obsahuje autonomní procesy, řízené časovými intervaly, které mohou (pokud je řízení zapnuto), hlídat teplotu v teráriu a tedy zabránit například úhynu zvířete, v případě že dojde k velké teplotní výchylce oproti jeho přirozenému prostředí.

7. Kompletace řešení

Jako první krok k řešení jsem se rozhodl zprovoznit nejvíce komplexní část celé Bakalářské práce. Tou je komunikace a předávání dat mezi Centrálním serverem a DCU. Tato část bude vyvíjena na testovacích podkladech a datech. Místo serveru bude pro snazší vývoj použit klasický notebook na kterém bude vyvíjen i kód pro Arduino. Jako testovací data, budou použita náhodná čísla, simulující data ze senzorů.

Dalším krokem následně bude zprovoznění komunikace mezi DCU a senzory. V této fázi bude vyvíjen kód, který bude zajišťovat získání informací ze senzorů a jejích odeslání do Centrálního systému. Také zde bude vyvíjen kód, který při obdržení řídících informací z Centrálního systému, nastaví korektně výstupní řídící data, která budou spravovat jednotlivá zařízení v teráriu.

V předposlední fázi bude pak vyvíjen centrální systém. Tato fáze bude zahrnovat korektní instalaci Windows server, vytvoření databáze pro sběr dat, sestavení grafického rozhraní pro vhodnou prezentaci dat pro uživatele ve formě webových stránek. Následně bude navrhnuto a sestrojeno jednoduché API, které bude fungovat pro jednosměrnou komunikaci s externím řídícím systémem. Tento externí systém bude ve formě dalších webových stránek, které již však budou na internetovém hostingu, čímž se zajistí dostupnost z internetu a tím mimo lokální síť. Pro bezpečnost a snadnost implementace jsem zvolil jednosměrnou komunikaci z lokálního centrální systému (iniciátor komunikace) do externího systému. V externím systému bude navrhnuto a sestrojeno API, které bude zajišťovat přenos dat. Veškerá komunikace mezi centrálním a externím systémem, bude zabezpečena přes výměnu hashů jako klíčů pro komunikaci. Integrita dat bude následně validována na straně centrálního systému.

7.1. Komunikační rozhraní DCU a Centrálního serveru

Při vývoji této fáze jsem jako první musel řešit mikrokontroler a jeho nahrávání kódu. Jelikož jsem se rozhodl na vývoj rozhodl využít Arduino Nano, je tato část ulehčena. Jako IDE jsem

tedy použil Arduino Software, který je uzpůsoben pro programování mikrokontrolerů Arduino nebo jejich kopií. Přes sériovou komunikaci pomocí USB jsem pak snadno nahrával jednotlivé kódy pro ověřování funkčností. Jako první jsem otestoval spojení mezi PC a Arduinem. Toto bylo nejsnáze řešeno pomocí jakéhokoli kódu. Protože v případě že by nedošlo ke správnému nahrání kódu do mikrokontroleru, hlásilo by IDE chybu.

Dalším segmentem v této fázi bylo zapojení Ethernet modulu. Tento modu je obsluhován Ethernetovým řadičem W5500. Po korektním zapojení na Arduino Nano, bylo nutné otestovat správnou funkčnost připojení do LAN. Využil jsem pro to testovací kód (PŘÍLOHA A – Testovací kód pro ověření funkčnosti Ethernetového modulu) poskytovaný prodejcem. Pro použití v mnou využívané LAN, bylo nutné upravit IP adresu aby patřila do sítě 192.168.0.1/24. Tento kód však není úplným testem. K ověření funkčnosti je potřeba vytvořit HTTP dotaz na IP adresu Arduina s modulem. Je to z důvodu, že je v kódu server hostován na portu 80. Tento port je obvykle používán pro HTTP dotazy na síti. Vytvořením HTTP dotazu lze velmi snadno dosáhnout otevřením jakéhokoli webového prohlížeče (např. Google Chrome) a zadáním IP adresu modulu (v mém případě 192.168.0.50) do URL adresy v prohlížeči. Po ověření dostupnosti modulu z počítače ve stejné síti, jsem zjistil následující. Ethernet modul není poškozený a pracuje správně, router daný modul korektně zařadil do interní ARP tabulky a modul je dostupný z jakéhokoliv zařízení v síti, ve které se Arduino s modulem nachází.

Komunikace mezi DCU a centrálním systémem je v této fázi tedy dokončena. Jediným segmentem který aktuálně nelze otestovat je datový balíček, který bude vyměňován mezi DCU a centrálním systémem. Jedná se o 2 balíčky. Jeden balíček bude obsahovat informace o kterou DCU (jednoznačný identifikátor) a data ze senzorů teploty a vlhkoměru. Dalším datovým balíčkem jsou řídící signály. V tomto balíčku bude přenášena informace, které zátěžové zařízení má být vypnuto či zapnuto. Překlad mezi pojmenováním zařízení a výstupním pinem bude provádět až DCU. Přesnou definici struktury těchto datových balíčků lze vytvořit až při konstrukci centrálního systému, při které lze přesně definovat komunikaci.

7.2. DCU terária a senzory

V této fázi bylo hlavním cílem vytvořit správnou komunikaci mezi DCU a senzory a korektní čtení dat ze senzorů. Jelikož jako vývojový kit je použito Arduino Nano, pořídil jsem k němu jeden samotný termistor, vhodný i pro měření teploty vody a kombinovaný senzor pro měření vzdušné teploty a vlhkosti. Jednoduchý termistor typu NTC neobsahující žádné další

komponenty. Kombinovaný senzor již obsahuje vlastní A/D převodník, který převádí hodnoty teploty a vlhkosti na digitální signál.

Termistor má velmi závislý odpor na teplotě, avšak se dle podmínek použití rozlišují 2 typy, NTC a PTC. Jelikož měřená teplota je v rozsahu běžných teplotních podmínek na Zemi, rozhodl jsem se použít termistor typu NTC. Odpor termistoru NTC s rostoucí teplotou klesá, odtud též zkratka tohoto typu (*Negative Temperature Constant*). Dalším charakteristickým jevem tohoto termistoru je jeho velká citlivost v teplotním rozsahu blízkém běžným teplotám v mírném podnebním pásu. Pro čtení hodnot jsem využil kód a zapojení, která jsou dostupná na stránkách prodejce. [3]

Kombinovaný senzor DHT11 obsahuje NTC termistor a komponentu, která má závislý odpor na vlhkosti. Jelikož se jedná o levnější a dostupnější verzi, má tento senzor menší snímaný rozsah 0°C – 50°C a 20% - 90% RH. Přesnost těchto hodnot je též nižší ± 2 °C pro teplotu a 5% RH pro vlhkost. Přesnost je v takto velkém intervalu z důvodu, že senzor obsahuje vnitřní 8-mi bitový A/D převodník. Prodejce poskytuje i pro tento senzor kód i doporučené zapojení, proto jsem jej využil a pouze modifikoval proměnné. [4]

Hlavním důvodem zvolení termistoru a senzoru DHT11 byla pořizovací cena a rozsah teplot ve kterých měří. Primárním cílem této práce je možnost použití mezi chovateli. Jelikož tito chovatelé mají často více než jedno terárium, je pořizovací cena důležitým faktorem pro implementaci zařízení.

7.3. DCU terária a řízené zdroje

Pro řízení zdrojů jsem jako první navrhnul použít relé, kvůli vysoké zátěži a střídavému napětí. Avšak relé má spínací proud *70 mA*, které vývojový kit Arduino Nano není schopno na výstupním digitálním pinu poskytnout. Z tohoto důvodu jsem se rozhodl pro řízení relé využít tranzistor. Tranzistor je napájen Arduinem skrze napěťový pin, řízen je pomocí datového pinu a výstup z tranzistoru je poté řídícím signálem relé. Tento způsob by bylo možné použít, ale není podchycena zpětná vazba z relé do Arduina. Proto jsem se ve výsledku rozhodl pro využití kombinace optočlen a relé na řízení zdrojů. Jelikož tepelné a světelné zdroje jsou vždy napájeny síťovým napětím *AC 230V* a z pohledu DCU je irelevantní jejich účel, lze k nimi přistupovat stejně a využít jednotné zapojení pro všechny výstupy.

7.4. Centrální systém a autonomní řízení

Centrální systém se skládá z hardwarové části, samostatné PC řízeno vzdáleně, a softwarové části, Windows server 2016 v kombinaci s lokálním server hostující databázi a webové stránky jako rozhraní vhodné pro uživatele.

7.4.1. Příprava hardwaru

Prvotním krokem byla příprava dostupného PC pro serverové účely. Hlavním požadavkem byl nepřetržitý provoz a dostatečná výpočetní kapacita. Z těchto důvodů jsem do hardwaru přidal další komponenty a to navýšením počtu RAM a HDD. Jedno HDD slouží aktuálně čistě pro operační systém a zbylé 4 HDD jsou zapojeny virtuálně jako RAID 5 a to z důvodu stáří HDD, aby se předešlo ztrátě dat v případě, že se jeden z disků fyzicky poškodí.

7.4.2. Instalace OS

Následovala instalace operačního systému, kde bylo na výběr větší množství OS Windows. Z důvodu staršího hardwaru v PC jsem se rozhodl využít starší verze OS Windows. Jako první jsem tedy nainstaloval Windows server 2008, který má oficiálně neomezenou licenci. Lze tedy tento sytém provozovat po neomezenou dobu. Po instalaci jsem však narazil na problém a to, že tento systém již není podporován a tím pádem s ním nelze řádně pracovat. Dalším vybraným systém byla tedy již novější verze a to Windows 2016 Hyper-V. Tato verze sice má neomezenou licenci, avšak nemá GUI. Pro kontrolu komunikačních scriptů, webového rozhraní a komunikaci mezi webovým rozhraním dostupném skrze Internet a centrálním systémem, se tento systém nehodil. Jako finální systém jsem tedy vybral Windows 2016. Tato verze má omezenou neplacenou licenci na 180 dní. Pro vývojové účely tato lhůta však postačí.

7.4.3. Instalace WampServeru

Po instalaci OS následovalo vytvoření lokálního webového serveru. Pro tento účel jsem zvolil WampServer. Jedná se o multifunkční server spojující PHP, databázi i Apache server. Tato kombinace tedy umožňuje pomocí jedné aplikace hostovat skoro plnohodnotné webové stránky pro lokální působení. Po úpravě několika konfiguračních souborů, lze webové stránky poskytovat do celé LAN.

7.4.4. Modifikace konfigurace pro přístup z LAN

Jelikož aplikace WampServer obsahuje nastavení pouze pro lokální stroj, přesněji adresu localhost, která se skrývá za vnitřní IP adresou počítače (127.0.0.1), je nutné upravit konfigurační soubory Apache serveru, aby byl přístupný LAN.

Jako první je nutné přidat TCP port, na kterém bude Apache server naslouchat. To lze provést upravením souboru *httpd.conf* umístěného ve složce „*Instalační cesta“\wamp64\bin\apache\apache2.4.37\conf*. „Instalační cesta“ je jako výchozí Jednotka C:\ avšak lze ji libovolně měnit. Po editaci lze vyhledat text: „#Listen“. Nalezneme 3 řádky, které mohou vypadat takto (liší se verzí aplikace):

```
#Listen 12.34.56.78:80
Listen 0.0.0.0:80
Listen [::0]:80
```

Obrázek 2 - Originální záznam v httpd.conf

Za tyto řádky je nutno přidat další 2 řádky.

```
Listen 0.0.0.0:8081
Listen [::0]:8081
```

Obrázek 3 - přidané řádky do httpd.conf

Tímto nastavíme Appache server tak, že bude naslouchat na portu 8081. Tento port můžeme zvolit jakýkoliv v rozsahu 1-65535, avšak je nutné se ujistit, zda již daný port není využíván jinou službou. Např. port 80 je běžně používán pro HTTP. Pokud bychom daný port chtěli používat pro naší službu celosvětově napříč internetem, bylo by nutné se zaregistrovat a vyřídit oficiální přiřazení organizací ICANN, která tyto záležitosti vyřizuje. Avšak pro lokální použití lze využít tento port bez nutnosti oficiálního přiřazení.

Dalším krokem v nastavování je úprava v souboru *httpd-vhosts.conf*. Zde pro Symfony bylo nutné upravit již existující záznam (PŘÍLOHA B – httpd-vhosts.conf 1.část a PŘÍLOHA C – httpd-vhosts.conf 2.část). Za tento upravený záznam přidáme podobný kód, který je však upravený. Obsahuje nyní port *:8081*, který jsme definovali v předchozím kroku. Další úpravou je přidáním *Require ip 192.168.0*, čím jsme omezili přístup na server pouze z podsítě 192.168.0.xxx. Jedná se o jeden ze zabezpečovacích prvků. Při vstupu z jiné adresy, je přistup zablokován a jako odpověď uživateli je zobrazena stránka „Forbidden“ se stavovým kódem 403 (Přístup zamítnut).

Posledním krokem pro zprovoznění přístupu na server z lokální sítě je nutné zvolený port přidat do firewallu počítače. Tento port totiž nebyl softwarově vyžádán a přidán do firewallu, je tedy nutné jej přidat manuálně. Na OS Windows lze toto učinit pouze s administrátorskými právy uživatele a to tak, že otevřeme „Ovládací panely“. Následně vybereme sekci „Firewall v programu Windows Defender“ a v levém menu vybereme možnost „Upřesnit nastavení“.

Tímto se nám zobrazí dialogové okno. V levém menu pak vybereme položku „Příchozí pravidla“, jelikož potřebujeme námi zvolený port otevřít tak, aby ostatní zařízení se mohla připojit k centrálnímu serveru. Jedná se tedy o příchozí spojení. Na pravé straně zvolíme „Nové pravidlo…“, čímž otevřeme další dialogové okno. Zvolíme možnost „Port“, klikneme na „Další >“. V této sekci vybereme možnost „TCP“ (již předvybráno) a „Konkrétní místní porty“ s kolonkou napravo od této volby. Do této kolonky pak vložíme námi zvolený port (např. 8081, který jsem zvolil). V další sekci „Akce“ zvolíme „Povolit připojení“. V sekci „Profil“ zvolíme všechny volby. Jelikož se server bude nacházet vždy pouze v lokální síti. V poslední kroku přidáme jakýkoliv název. Jedná se o název daného pravidla ve firewallu pro snazší identifikaci. Po dokončení je nám námi zvolený port otevřen. Pro správnou funkčnost je však nutné ještě restartovat WampServer. Po restartu jsou již webové stránky alias centrální systém dostupný v LAN síti. Jelikož však nebývá v domácnostech běžně dostupný DNS není tedy možné založit lokální doménu pro centrální systém. Lze tedy pouze přistupovat na webové stránky skrze adresu *192.168.0.102:8081*. Pro pevné počítače, lze udělat konfigurace, které by simulovali činnost DNS. Avšak v případě mobilního telefonu tohoto nelze snadno dostáhnout.

7.4.5. Centrální systém – webové stránky

Jako základ centrálního systému jsem použil framework Symfony. Tento framework je na vzestupu jak z pohledu popularity, tak z pohledu údržby a rozšiřování funkčností. Prvním krokem pro vytvoření webových stránek bylo zprovoznění Symfony frameworku. Tento proces a postup je detailně popsán v dokumentaci frameworku [5]. Dokumentace je psaná výhradně v Anglickém jazyce. Pro snazší vytvoření Symfony projektu, lze využít Composer. Jedná se o software, který umožní snadnější instalaci na Windows. Po vytvoření projektu následovalo správné spojení Symfony a WampServeru, jenž je lokálním serverem zajišťující dostupnost a funkčnost webových stránek. Symfony projekt využívá již vlastní strukturu složek, která není pro WampServer nativní. Bylo tedy nutné upravit nastavení Appache (server) aby pro danou doménu vyhledával ve specifickém adresáři, kde se nachází soubor *index.php*. Tento soubor je inicializován vždy při vstupu na jakoukoliv webovou stránku, proto bylo nutné upravit konfigurační soubor, kde lze definovat virtuální hostované weby. Zde se definovala cesta do adresáře obsahující *index.php*. Po této úpravě a následném ověření funkčnosti, následovala již tvorba centrálního systému.

Symfony umožňuje pracovat se strukturou databáze již pomocí definice objektů (tzv. Entit - PŘÍLOHA E – Ukázka části struktury Entity Terária) v kódu. Toto zajišťují takzvané

„migrace“. Po vytvoření Entity (např. Terrarium) a následné definici jednotlivých vlastností, lze pomocí nástroje skrze příkazový řádek, vytvořit novou „migraci“. Po vygenerování této migrace, lze dalším nástrojem v příkazovém řádku tuto „migraci“ spustit. Tím poté dojde ke spuštění několika SQL příkazů do databáze, které upraví strukturu tabulky (např. terrarium). Již samotné vytváření Entit je ulehčeno dalším nástrojem skrze příkazový řádek.

Po vytvoření potřebných Entit a spuštění migrací bylo možné pokračovat ve tvorbě grafického rozhraní (GUI) a komunikačního rozhraní pro komunikaci s moduly terária. Předtím jsem se však rozhodl přidat bezpečnostní prvek. Nejsnazším řešení je vytvoření přihlašovacího formuláře. I pro tento případ Symfony obsahuje nástroj, který po správném zadání informací pro aktuální projekt, sám automaticky vytvoří všechny soubory a provede potřebné modifikace v již existujících. Také vytvoří přihlašovací formulář podle předdefinované šablony. Nástroj pro toto vygenerování se spouští pomocí konzole (stejně jako ostatní nástroje ulehčující práci se Symfony) a to příkazem „php bin/console make:auth“. Tento příkaz je součástí oficiální dokumentace, kde lze nalézt přesné znění příkazu, ale také i postupu na kompletní vyhotovení zabezpečení skrze tento formulář.

7.4.6. Datové balíčky a komunikační rozhraní s DCU

Při tvorbě GUI jsem narazil na myšlenku, že by nebylo ani potřeba tvořit speciální externí komunikační rozhraní za účelem komunikace s DCU. Přivedla mě na to myšlenka, ve které jsem se zaobíral fungováním DCU. Pro Arduino Ethernetový modul, lze totiž vytvořil jednoduchý server, který vyčkává na připojení. Tohoto jsem také využil již při testování komunikace a přenosu informací pomocí Ethernetového připojení. Z tohoto důvodu jsem se rozhodl, nevyužívat funkce napsané v jiném programovacím jazyce a vyzkoušet přímé propojení centrálního systému a DCU. Pro správnou komunikaci bylo však nutné definovat si styl komunikace mezi DCU a centrálním systém. V jaké formě bude dostávat centrální systém data a v jaké formě bude odesílat příkazy centrální systém DCU? Jelikož komunikace probíhá přes HTTP, je potřeba využít textového řetězce pro strukturalizaci dat. DCU přijímá příkaz prostřednictvím adresy, bylo vhodné příkazy pro DCU vložit jako součást adresy.

Proto jsem pro kontrolní balíček dat navrhl spojení několika částí v jeden odkaz. Jako výsledek tohoto spojení vznikne např. „192.168.0.101?lon“. Vždy se jedná o řízení jediného zdroje. Toto spojení se skládá z:

- „192.168.0.101“ – IP adresa DCU
- „?“ – oddělovač IP adresy a dat

- „l“ – označení typu zdroje („l“ – světelný, „h“ – tepelný¨)
- „on“ – stav do kterého má být zdroj přepnut („on“ – zapnuto, „off“- vypnuto)

Tímto lze řídit specifické DCU a jejich specifické zdroje.

Dalším krokem, bylo nutné definovat datový balíček obsahující informace ze senzorů DCU. Zde bylo velkou výhodou, že data lze předávat jako víceřádkový textový řetězec. Proto jsem se rozhodl využít podobné metody jako je CSV (comma separated values).

Předání dat ze senzoru je inicializováno centrálním systémem, obdobně jako při odesílání příkazu do DCU a to přístupem na adresu „192.168.0.101?data“. Opět je na počátku IP adresa následována „?“, jako separátorem, avšak je tato adresa zakončena sufixem „data“, který definuje, že se má jednat o odeslání dat ze senzorů do centrálního systému.

```
$modules = $this->em->getRepository( entityName: Module::class)->findAll();
foreach ($modules as $module)
{
    $curl = curl_init( url: $module->getIp().'?data');
    curl_setopt($curl,  option: CURLOPT_RETURNTRANSFER,  value: true);
    curl_setopt($curl,  option: CURLOPT_HEADER,  value: 0);
    curl_setopt($curl,  option: CURLOPT_TIMEOUT,  value: 5);
    $data = curl_exec($curl);
    curl_close($curl);
    if($data === false)
        continue;
    $data = explode( delimiter: ';',$data);
```

Obrázek 4 - část metody zasílající dotaz DCU o data

Datový balíček s informacemi ze senzorů vždy začíná prvním řádkem MAC adresou, která je jednoznačně definována v dané síti pro DCU. Následující řádky již obsahují jednotlivé senzory (1 řádek = 1 senzor). Řádek dat senzoru začíná písmenem T (Temperature) nebo H (Humidity), pro určení typu senzoru a číslo senzoru v DCU. Následuje oddělovač v tomto případě pomlčka „-„. Další informace je výrobní typ senzoru např. „NTC. Jako další separátor jsem použil dvojtečku „:“ následovanou číselnou hodnotou, získanou ze senzoru. Výsledný řádek může pak vypadat následovně „T2-NTC:27,65“
Celková struktura datového balíčku může pak vypadat takto:
„MAC:11-22-33-44-55-01
T1-DHT11:26
H1-DHT11:87
T2-NTC:27,65“

Způsob předávání informací má jedno slabší místo v integritě dat. Tím je zaručení stejných výrobních typů senzorů v DCU a v centrálním systému. Tuto integritu musí zajišťovat operátor (chovatel). Výhodou aktuální verze DCU (verze „mini“) má pak v jednoznačnosti zdrojů. Obsahuje jediný zdroj tepla a jediný zdroj světla. Při tvorbě větší DCU obsluhující např. 3 zdroje (bodové světlo, podsvícení terária led pásky a tepelnou podložku). Bylo by nutné modifikovat příkazovou část v adrese a to rozšířením o index zdroje. Příkazová část adresy by pak vypadala např. takto „?l2on“.

```
foreach ($data as $line)
{
    switch (true){
        case strpos($line, needle: 'MAC') !== false:
            $mac = explode( delimiter: ':',$line)[1];
            $checked = $this->em->getRepository( entityName: Module::class)->findOneBy(array('mac' => $mac));
            if($checked !== null && ($checked->getId() === $module->getId()))
                $stopUpdate = false;
            break;
        case preg_match( pattern: '/T\d?-/',$line):
            $sensorData = $this->parseSensorLine($line);
            $sensor = $this->em->getRepository( entityName: Sensor::class)->findOneBy(array(
                'name' => $sensorData['type'], 'idSensorType' => 1, 'idModule' => $module->getId()
            ));
            if($sensor !== false)
            {
                $sensor->setValue($sensorData['val']);
                $this->em->flush($sensor);
            }

            break;
```

Obrázek 5 - část funkce těžící data z balíčku

Pro chovatele, který je technicky znalý webových aplikací a programování, by bylo možné, aby si jednotlivé IP adresy v DCU a v centrálním systému spravoval sám. Avšak pro běžného chovatele, toto není možné. Bylo tedy nutné místo statických IP adres DCU využít možnosti DHCP. Jedná se tedy o dynamické přidělování IP adres. DCU obsahující Ethernetový modul, je na toto připraven a stačí při zahájení Ethernetového spojení nedefinovat IP adresu. Tímto dojde k inicializaci komunikace s DHCP a následným dotazem na přidělení IP adresy.
Z pohledu centrálního systému však nastává komplikace. IP adresy u jednotlivých modulů (DCU) jsou pevně dané v databázi. Bylo tedy nutné vytvořit nástroj pro alokaci správných IP adres k jednotlivým modulům. Jako řešení jsem tedy navrhnul využít možnosti systémových nástrojů OS Windows a ARP tabulku. ARP tabulky obsahuje IP adresy a MAC adresy jednotlivých zařízení, se kterými byla navázána v minulosti komunikace. Vytvořil jsem tedy v centrálním systému nástroj, který v prvním kroku promaže záznamy v ARP tabulce serveru. (PŘÍLOHA D – Kód obnovující IP adresy v ARP tabulce). Následně pak pomocí příkazu ping, otestuje všechny IP adresy v DHCP rozsahu. V případě, že se na dané IP adrese nachází nějaké zařízení, je jeho MAC a IP adresa zařazena do ARP tabulky. Po dokončení tohoto

kroku, následuje převzetí ARP tabulky ve formě textového řetězce. Ten je následně zpracováván po řádcích a je separován na IP adresu a MAC adresu. Ke zpracování jsou předávány pouze však ty řádky ARP tabulky, které končí slovem „dynamic“, jelikož se jedná o dynamicky přiřazené IP adresy DCHP serverem/routerem. V případě, že je MAC adresa nalezena v databázi, je k ní přiřazena nová IP adresa. Tímto je pak obnova IP adres dokončena a lze již správně navazovat komunikaci mezi centrálním systémem a DCU.

7.4.7. Autonomní řízení

Autonomní řízení jsem vytvářel až jako jednu z posledních částí. Je to z důvodu, že bylo nutné nejdříve správně vyhotovit a otestovat předchozí části. Kompletní strukturu všech Entit a jejich tabulek, komunikaci mezi centrálním systémem a DCU. Hlavní výhoda vyhotovení až na konci je, že lze využít již existující kódů a tím lze předejít novým možným chybám při realizaci duplicitních kódů.

První metodou autonomního řízení je obnova dat ze senzorů jednotlivých DCU. Tato metoda využije již existující metody pro obnovu dat ze senzorů a řádně je přiřadí k daným senzorům. Pro správně fungování je však nutné tuto metodu využívat opakovaně, což je hlavním úkolem Autonomního systému. Opakované spouštění pak zajišťuje tzv. CRON, který lze jako jeden balíček připojit k Symfony. Časování CRONu obecně nedovoluje spouštět metody častěji než jedna minuta. Proto jsem jako interval opakování nastavil nejmenší možný interval a to jednu minutu.

```
public function renewDataFromTerras()
{
    $modulInterface = new ModulInterface($this->em);
    $modulInterface->getDataFromTerra();
}
```

Obrázek 6 - obnova dat z terárií

Další metoda autonomního systému je správa terárií. Tato metoda je spouštěna každých 5 minut pro zajištění dostatečné péče o zvíře. Před zahájením řízení metoda ověří nastavení v systému, pokud je autonomní řízení terárií zapnuto, pokračuje metoda v procesu, jinak je ukončena. Metoda následně získá všechna terária z databáze, ke kterým je přidělena DCU. Následně ověří teplotní rozdíly v teráriu vůči nastaveným limitním teplotám u terária. Pokud je zjištěno, že je v teráriu větší teplota než maximální přípustná, je tepelný zdroj vypnut. V případě, že je teplota v teráriu nižší než minimální přípustná, je tepelný zdroj zapnut. Tento

přístup zajistí, že teplota v teráriu nebude mimo vhodný rozsah po delší dobu. Tímto tak systém sám zabrání situaci, kdy nevhodná teplota v teráriu může ublížit zvířeti a v extrémní situaci úhynu zvířete.

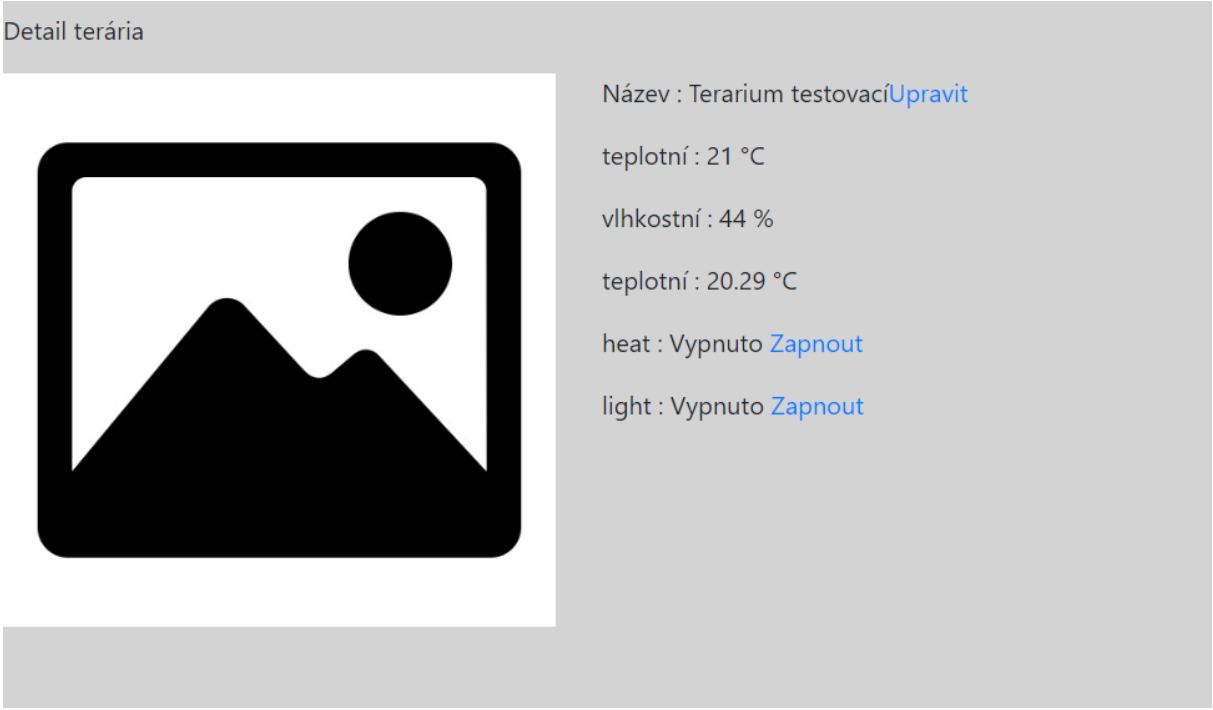
```
public function checkAutonomousStateAndSet()
{
    $system = $this->em->getRepository( entityName: System::class)->find(1);
    if (!$system->getAutonomousState())
        return false;
    $terras = $this->em->createQueryBuilder()
        ->from( from: Terrarium::class, alias: 't')
        ->where( predicates: 't.idModule IS NOT NULL')
        ->getQuery()
        ->getResult();
    $moduleManager = new ModuleManager($this->em);
    foreach ($terras as $terra)
    {
        $moduleManager->checkTempAndManage($terra);
    }
    return true;
}
```

Obrázek 7 - metoda spouštějící funkci kontroly na teráriu

Jako poslední metodu autonomního řízení, jsem pro pohodlí chovatele přidal metodu, kdy systém sám zaktualizuje IP adresy DCU. Toto je zajištěno pomocí funkce popsané v kapitole *6.4.6.* spojené s automatickým spouštěním pomocí CRONu. Jedná se o déle pracující funkci a pro nekorektní přiřazení IP adres je nutné odpojit z LAN a následně připojit DCU zpět do sítě nebo vypršením platnosti přiřazení IP adresy DCU. K těmto událostem však nedochází velmi často. Proto jsem nastavil interval, kdy má dojít k obnově IP adres na 1x denně o půlnoci. Tímto je tak omezena možnost, kdy by tato funkce omezila fungování systému pro chovatele.

7.5. Přístup operátora do systému

Při tvorbě GUI jsem jako hlavní prioritu zvolil bezpečnost. Jako první je tedy pro přístup do centrálního systému nutné se přihlásit. Heslo je v databázi uloženo zašifrované. Dalším hlavním prvkem GUI byla jednoduchost a přehlednost a širší možnosti zásahu do dat. Proto operátor může přidávat dynamicky terária, moduly i senzory. Pro reálnější použití je zde také přidána možnost správy zvířat. Záznamy o zvířatech lze pak snadno přiřazovat k jednotlivým teráriím, ve kterých se fyzicky nachází.



Obrázek 8 - Demonstrativní terárium obsahující data ze senzorů

7.6. Vzdálené řízení centrálního systému

Webové stránky pro přístup z internetu slouží pouze a výhradně pro zobrazení stavů v terárii a možnému řízení jednotlivých zdrojů. Z toho důvodu jsou webové stránky dostupné skrze hosting, který je poskytován zdarma a to společností Seznam.cz. Pro zachování základní bezpečnosti je zde jako první vyžádané přihlášení. To lze dokončit pouze pomocí velmi dlouhého textového řetězce. Pokud by došlo k prolomení této ochrany, je zde vytvořeno větší bezpečností opatření, které vychází již z návrhu řešení, které jsem použil.

Jelikož nelze zajistit komunikaci mezi centrálním systém a externím webe obousměrně inicializovanou z jakékoliv strany, bylo nutné veškerou komunikační logiku přenosu naprogramovat na straně centrálního systému.

První metodou tohoto rozhraní je příprava dat k odeslání. Protože externí hosting nemá v bezplatném tarifu databázi, použil jsem tedy jako úložiště dat JSON soubory. Tyto soubory mají velkou výhodu ve strukturalizaci a snadném zpracování pomocí PHP. První metoda tedy vezme data všech terárií a pomocí nástroje Symfony, připraví výstupní data ve formátu, který lze zvolit. Pro použití v této práci tedy formát JSON. Jelikož však jsou u terárií přiřazené též obrázky, bylo vhodné přenášet i tyto soubory. Avšak jejich změny nejsou tak časté a pro omezení datových toků jsem se rozhodl separovat předávání dat a obrázků do oddělených metod. Každá metoda může být pak spouštěna v různý čas.

```php
public function transferData()
{
    $this->compareToExtern();
    $this->transferTerras();
    $this->sendToExtern();
}

private function transferTerras()
{
    $terras = $this->em->getRepository( entityName: Terrarium::class)->findAll();
    $data = $this->serializer->serialize($terras,
        format: 'json',
        array('groups' => array(
            'API_DATA'
        ),
            AbstractNormalizer::IGNORED_ATTRIBUTES => array(
                '__initializer__',
                '__cloner__',
                '__isInitialized__'
            )));
    $this->createTmpJSON( name: 'terras', $data);
}
```

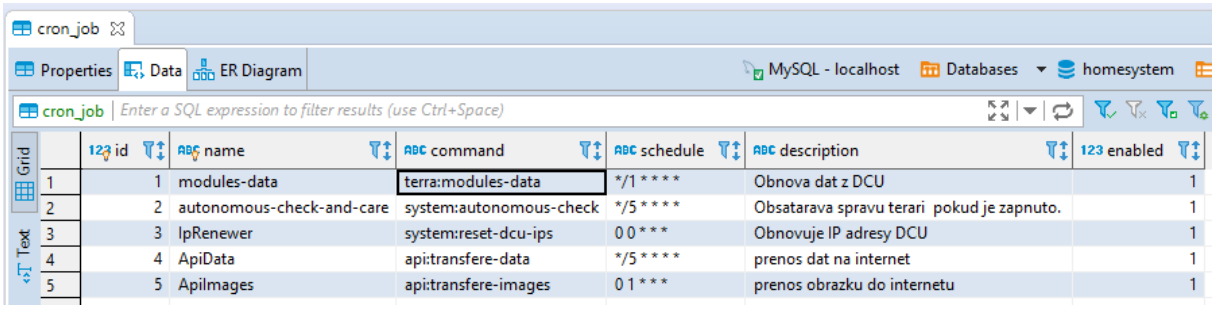
Obrázek 9 - Příprava dat k odeslání

Následně je vytvořen dočasný soubor, který obsahuje JSON data. Tímto pak lze snadno oddělit zpracování a přípravu dat o jejich odesílání na externí server. O toto odeslání se stará následující metoda, která naváže spojení s externím serverem a následně do správného adresáře umístí předpřipravené soubory s daty.

Jako další krok bylo nutné vyřešit řízení terárií z externího serveru. Jelikož se jedná o dvoustavovou veličinu (vypnuto/zapnuto) nebylo nutné řešit přednostní právo řízení zda lokální nebo vzdálené. Mohla totiž nastat pouze situace, kdy obě tato řízení změnili stav a to do identického stavu. V jiném případě ke kolizi nedochází.

Pro správnost fungování těchto metod bylo vyřešit jejich opakované spouštění. Toto jsem vyřešil tak, že jsem použil stejné metody jako u autonomní správy, tedy skrze nástroj zvaný CRON.

Pro metodu obsluhující přenos informací o řízení a dat, jsem nastavil spouštění opakované každých 5 minut. Metoda zajišťující přenos obrázků na externí server je pak spouště jedenkrát denně v 01:00. Tímto opět tedy nedochází ke zbytečnému vytěžování serveru přes den.



	id	name	command	schedule	description	enabled
1	1	modules-data	terra:modules-data	*/1 * * * *	Obnova dat z DCU	1
2	2	autonomous-check-and-care	system:autonomous-check	*/5 * * * *	Obsatarava spravu terari pokud je zapnuto.	1
3	3	IpRenewer	system:reset-dcu-ips	0 0 * * *	Obnovuje IP adresy DCU	1
4	4	ApiData	api:transfere-data	*/5 * * * *	prenos dat na internet	1
5	5	ApiImages	api:transfere-images	0 1 * * *	prenos obrazku do internetu	1

Obrázek 10 - seznam úloh CRONu

8. Shrnutí řešení a demonstrativní ukázka

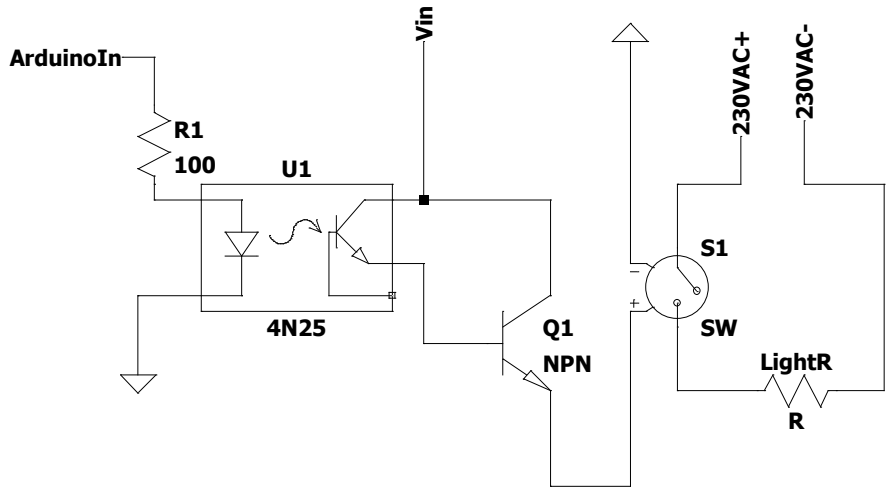
Při vývoji řešení jsem při zpracovávání dílčích části dělal kontroly funkčnosti. V aktuální fázi jsou všechny dílčí části funkční, avšak hlavním testem řešení je celková kompletace a následné otestování řešení.

Prvním krokem při kompletaci bylo nasazení DCU a připojených senzorů do terária, které bylo za účelem demonstrace řešení, vytvořeno nízkonákladově. V aktuálním stavu jej tedy nelze použít jako chovné terárium a to z důvodu chybějící podestýlky, nezadělaných větracích děr a neošetřeného povrchu stěn terária lakem. Toto jsou však jen a pouze detailní úpravy, které lze kdykoliv bez toho aniž by ovlivnili práci DCU.

Vybraný senzor NTC byl dodán již s metr dlouhou dvoulinkou. Zasazení tohoto senzoru tedy nebylo obtížné. Avšak senzor DHT11 je dodáván bez kabeláže. Bylo tedy nutné sehnat tří žilový kabel. Použil jsem tří žílový starý napájecí kabel k tiskárně, který již nebyl používán.

Jelikož vlastněný Arduino Ethernetový modul nepodporoval PoE, musel jsem pro simulaci využít externí napájení přímo u terária. Toto bylo nahrazeno starším počítačovým spínaným zdrojem. Po zapojení DCU na napájení, připojení senzorů a zapojení kroucené dvoulinky konektorem RJ45 k DCU, následovalo připojení demonstrativního zdroje (pro časovou efektivnost jsem vybral pro demonstraci světelný zdroj). Zde však při zapojení nastal první zádrhel, který zkomplikoval kompletaci. Při testování spínání zdroje na dálku skrze přímý odkaz (nikoli přes centrální systém), kde byla přímo řízena LED, nenastali komplikace. Avšak při zapojení optočlenu, který spíná relé, docházelo ke stavu, kdy i při přepnutí výstupního pinu Arduina do stavu logické 1, relé nebylo sepnuto. První domněnka byla, že je nedostatečný proud skrze fototranzistor tekoucí do relé (spínací proud relé je 70 mA). Proto jsem se rozhodl zkusit použít NPN tranzistor, který je řízený optočlenem. Optočlen jsem použil 4N35, jelikož jsem jej měl dostupný. Jelikož však program LTSpice tento model optočlenu nemá, použil jsem pro následující schéma 4N25, který je svým zapojením

identický. Schéma je pouze názorné.



Obrázek 11 - řídící obvod zátěže

Toto zapojení se již osvědčilo pro spínání zátěže bez ovlivnění funkčnosti Arduina (relé při spínání vytváří napěťové špičky).

Při vyřešení spínání relé a následném testu opakovaného přístupu, docházelo k restartu Ethernet modulu a již tedy nebylo možné aby probíhala komunikace mezi DCU a centrálním systémem. Toto jsem vyřešil hot-fixem, kde při zachycení Arduina do stavu, kdy již není přidělena IP adresa, je Arduino uspáno na 2 vteřiny pro řádnou inicializaci modulu a následně je opětovně zahájeno připojení do Ethernetu.

Následně jsem zapnul centrální systém, který již obsahoval informace o daném teráriu, avšak IP adresa modulu nebyla známá. Spustil jsem vyhledávání IP adres modulů v síti. Daný proces trval zhruba 3-4 minuty, kdy byl centrální systém pro operátora nedostupný. Volba časování obnovy IP adres CRONem byla tedy správná (mimo dobu vytížení operátorem). Po dokončení vyhledávání IP adres, byla IP adresa přiřazena a bylo tedy možné otestovat komunikaci mezi DCU a centrálním systémem. Testování proběhlo úspěšně a to tedy tak, že probíhala obnova dat ze senzorů na straně centrálního systému a při odeslání požadavku na zhasnutí či rozsvícení žárovky, došlo ke správnému sepnutí/vypnutí relé a zdroj byl v ten daný okamžik zapnut či vypnut.

Jako poslední jsem otestoval přenos dat z centrálního systému do externího webu. Zde již nenastali žádné komplikace a vše fungovalo, dle návrhu.

Poslední zbývající část bylo spuštění CRONu. Zde však nastala situace, která zneplatnila některá rozhodnutí. Přesněji volba operačního systému serveru. Vybral jsem Windows Server z důvodu, že podporuje spouštění C# metod, které měli zajišťovat komunikaci bez dodatečného softwaru. Od tohoto jsem v průběhu realizace upustil, jelikož jsem navrhl snazší řešení. Avšak CRON vyžaduje pro automatické spouštění rozšíření pcntl pro PHP. Toto však není pro Windows OS dostupné a pro správnou funkčnost by bylo nutné přeinstalovat operační systém serveru, což by následně výrazně ztížilo přenos centrálního systému z vývojového notebooku na server.

ZÁVĚR

Na počátku této práce bylo nutné nastavit způsob jakým budu vypracovávat tuto práci. Rozhodl jsem se využít možností, které vývoj této práce značně ulehčují. Těmi jsou Arduino deska a Symfony framework. Arduino deska mi značně ulehčila práci při kompletaci a vývoji DCU, softwaru řídící DCU a vzájemné spojení DCU, senzorů a relé, která řídí zdroje tepla či světla v teráriu. Symfony framework je v celkovém rozsahu velice komplexní a objemný nástroj, který značně ulehčuje práci s některými náležitostmi týkajících se webových stránek (např. integrita struktury tabulek v databázi a objektů v kódu). Avšak pro malé projekt je tento nástroj příliš přehnaný a sám o sobě spíše náročnost na hardwarové prostředky zbytečně navyšuje. Pro vypracování této práce je však Symfony vhodnou volbou. Urychluje tvorbu některých kódů pomocí nástrojů a snadno lze přidávat další funkčnosti v průběhu realizace práce. V průběhu realizace a kompletace některých částí práce, jsem dostával různé nápady na úpravy, které bylo možné zrealizovat a tím zjednodušit některé aspekty práce (např. komunikace mezi centrálním systémem a DCU). V závěru lze tedy říci, že provedená realizace, je použitelná v praxi pokud by centrální systém byl umístěn na server fungující s operačním systémem Linux. Také je řešení použitelné primárně jako soukromé projekty. Pro zkomercializování by bylo nutné unifikovat některé metody zajišťující komunikaci mezi centrálním systémem a DCU, za účelem jednotné komunikace pro různé typy modulů (DCU).

POUŽITÁ LITERATURA

Bibliografie

[1] Regents of the University of Michigan. *Animal Diversity Web* [online]. University Michigan Museum of Zoology: University Michigan, 2014 [cit. 2020-02-26]. Dostupné z: https://animaldiversity.org/accounts/Eublepharis_macularius/

[2] Wire Gauge and Current Limits Including Skin Depth and Strength. *PowerStream* [online]. Orem Utah: PowerStream Technology, 2020 [cit. 2020-03-03]. Dostupné z: https://www.powerstream.com/Wire_Size.htm

[3] NTC termistor - datasheet. *Arduino-shop* [online]. Havlíčkův Brod: ECLIPSERA s.r.o, 2020 [cit. 2020-04-18]. Dostupné z: https://arduino-shop.cz/docs/produkty/0/115/1488979094.pdf

[4] DHT11 senzor - datasheet. *Arduino-shop* [online]. Havlíčkův Brod: ECLIPSERA s.r.o, 2020 [cit. 2020-04-18]. Dostupné z: https://arduino-shop.cz/docs/produkty/0/133/1500635986.pdf

[5] Symfony installing. *Symfony* [online]. Symfony: SensioLab, 2020 [cit. 2020-05-08]. Dostupné z: https://symfony.com/doc/current/setup.html

[6] Arduino Ethernet modul W5500. *Arduino návody* [online]. Havlíčkův Brod: ECLIPSERA s.r.o, 2020 [cit. 2020-04-18]. Dostupné z: https://navody.arduino-shop.cz/navody-k-produktum/arduino-ethernet-modul-w5500.html

PŘÍLOHY

PŘÍLOHA A – Testovací kód pro ověření funkčnosti Ethernetového modulu

```
// LAN modul W5500 - web server

// připojení potřebných knihoven
#include <SPI.h>
#include <Ethernet2.h>
// nastavení MAC adresy
byte mac[] = {
  0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED
};
// nastavení IP adresy - musí být přidělitelná DHCP serverem,
// tedy ve správném rozsahu, případně přidělena routerem ručně
IPAddress ip(192, 168, 0, 50);
// inicializace serveru na portu 80
EthernetServer server(80);

void setup() {
  // inicializace komunikace po sériové lince rychlostí 9600 baud
  Serial.begin(9600);
  // zapnutí komunikace s Ethernet Shieldem
  Ethernet.begin(mac, ip);
  server.begin();
  // výpis informace o nastavené IP adrese
  Serial.print("Server je na IP adrese: ");
  Serial.println(Ethernet.localIP());
}

void loop() {
  // načtení připojených klientů
  EthernetClient klient = server.available();
  // pokud se připojí nějaký klient, provedeme následující
  if (klient) {
    Serial.println("Novy klient:");
    // http request končí prázdným řádkem
    boolean aktualniRadkaPrazdna = true;
    // pokud je klient připojen a k dispozici,
    // vytiskneme mu dostupná data
    while (klient.connected()) {
      if (klient.available()) {
        // načtení a vytištění informace o klientovi
        char c = klient.read();
        Serial.write(c);
        // pokud se dostaneme na konec řádku a následující je prázdný,
        // byl ukončen request a my můžeme poslat odezvu
        if (c == '\n' && aktualniRadkaPrazdna) {
          // nejprve pošleme standardní http odezvu
          klient.println("HTTP/1.1 200 OK");
          klient.println("Content-Type: text/html");
          klient.println("Connection: close");
          // nastavení automatické obnovy stránky po 5 vteřinách
          klient.println("Refresh: 5");
```

```
            klient.println();
            klient.println("");
            klient.println("");
            // místo pro tištění vlastních údajů
            klient.print("Cas od spusteni: ");
            klient.print(millis() / 1000);
            klient.print(" vterin.<br />");
            // ukázka výpisu dat ze všech analogových vstupů
            for (int analogPin = 0; analogPin < 6; analogPin++) {
              int analogData = analogRead(analogPin);
              klient.print("Analogovy vstup A");
              klient.print(analogPin);
              klient.print(": ");
              klient.print(analogData);
              // příkaz "<br />" funguje jako krok na další řádek
              klient.println("<br />");
            }
            klient.println("");
            break;
          }
          // začátek nové řádky
          if (c == '\n') {
            aktualniRadkaPrazdna = true;
            // detekce znaku na nové řádce
          } else if (c != '\r') {
            aktualniRadkaPrazdna = false;
          }
        }
      }
      // pauza pro prohlížeč, aby stihl zpracovat všechny data
      delay(1);
      // uzavření spojení
      klient.stop();
      Serial.println("Klient odpojen.");
      Serial.println("---------------------");
    }
  }
```

Zdroj: https://navody.arduino-shop.cz/ [6]

PŘÍLOHA B – httpd-vhosts.conf 1.část

Kód v souboru httpd-vhosts.conf, první část.

```
<VirtualHost *:80>
  ServerName localhost
  ServerAlias localhost
  DocumentRoot "${INSTALL_DIR}/homesystem/public"
  DirectoryIndex /index.php
  <Directory "${INSTALL_DIR}/homesystem/public">
      AllowOverride None
      Order Allow,Deny
      Allow from All

      FallbackResource /index.php
  </Directory>
  <Directory /var/www/project/public/bundles>
     FallbackResource disabled
  </Directory>
</VirtualHost>
```

PŘÍLOHA C – httpd-vhosts.conf 2.část

Kód v souboru httpd-vhosts.conf, druhá část.

```
<VirtualHost *:8081>
  ServerName mjhomesystem.cz
  ServerAlias mjhomesystem.cz
  DocumentRoot "${INSTALL_DIR}/homesystem/public"
  DirectoryIndex /index.php
  <Directory "${INSTALL_DIR}/homesystem/public">
        AllowOverride None
        Order Allow,Deny
        Allow from All
       Require ip 192.168.0

      FallbackResource /index.php
  </Directory>
  <Directory /var/www/project/public/bundles>
     FallbackResource disabled
  </Directory>
</VirtualHost>
```

PŘÍLOHA D – Kód obnovující IP adresy v ARP tabulce

```
public function renewModuleIps()
{
    $this->setEmptyIps();
    set_time_limit( seconds: 0);
    $arpReset = new Process(array(
        'arp', '-d'
    ));
    $arpReset->run();
    $ipRangeFrom = '192.168.0.';
    $ipRange = 51;
    for ($i = 0;$i < $ipRange;$i++)
    {
        $lookedIp = $ipRangeFrom.($i+100);
        $process = new Process(array('ping','-n','1',$lookedIp));
        $process->run();
        if(!$process->isSuccessful()){
            continue;
        }
    }
    $arpCheck = new Process(array(
        'arp', '-a'
    ));
    $arpCheck->run();
    if(!$arpCheck->isSuccessful()){
        throw new ProcessFailedException($arpCheck);
        return;
    }
    $output = $arpCheck->getOutput();
    set_time_limit( seconds: 120);
    $this->updateIps($output);
}
```

PŘÍLOHA E – Ukázka části struktury Entity Terária

```php
/**
 * @ORM\Entity(repositoryClass="App\Repository\TerrariumRepository")
 */
class Terrarium
{
    /**
     * @ORM\Id()
     * @ORM\GeneratedValue()
     * @ORM\Column(type="integer")
     */
    private $id;

    /**
     * @ORM\Column(type="string", length=255, nullable=true)
     */
    private $pictureUrl;

    /**
     * @ORM\Column(type="string", length=255)
     */
    private $name;

    /**
     * @ORM\OneToOne(targetEntity="App\Entity\Module", cascade={"persist", "remove"})
     * @ORM\JoinColumn(nullable=true)
     */
    private $idModule;

    /**
     * @ORM\Column(type="boolean",options={"default":false})
     */
    private $automaticControll;
```