Česká zemědělská univerzita v Praze

Technická fakulta

Katedra elektrotechniky a automatizace

Diplomová práce

Regulační systém na platformě Node-RED

Albert Suchopár

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Technická fakulta

ZADÁNÍ DIPLOMOVÉ PRÁCE

Bc. Albert Suchopár

Informační a řídicí technika v agropotravinářském komplexu

Název práce

Regulační systém na platformě Node-RED

Název anglicky

Regulation system on the Node-RED platform

Cíle práce

Cílem práce je vytvoření inteligentního regulačního systému založeného na platformě Node-RED. V práci budou využity nody pro čtení dat o počasí, komunikaci v síti, komunikaci s domácím asistentem a regulaci. Systém bude umožňovat regulovat teplotu v prostoru na základě predikce vývoje a předpovědí teploty v místnosti a mimo ni. Zařízení bude připojeno k asistentu domácí automatizace a bude možné uživatele informovat pomocí SMS či emailem.

Metodika

Prostudování hardwarových a softwarových možností řešení. Navržení několika variant provedení úlohy. Výběr nejvhodnější varianty s kritickým hodnocením návrhu. Specifikace funkcí modelu podle cílů práce.

Doporučený rozsah práce

60stran, bez příloh

Klíčová slova

Arduino, detektor, WiFi, smart, automatizace

Doporučené zdroje informací

DENNIS, Andrew K. *Raspberry Pi home automation with Arduino : automate your home with a set of exciting projects for the Raspberry Pi!.* Birmingham: Packt Publishing, 2013. ISBN 978-1-78439-920-7.

GOODWIN, Steven. *Smart home automation with Linux and raspberry Pi.* New York: Apress, 2013. ISBN 978-1-4302-5887-2.

Learn IoT Programming Using Node-RED, Begin to Code Full Stack IoT Apps and Edge Devices with Raspberry Pi, NodeJS, and Grafana. BPB Publications, 2022. ISBN 9391392385.

MORRISS, S. Brian. *Automated manufacturing systems : actuators, controls, sensors, and robotics.* New York: Glencoe, 1995. ISBN 0028023315.

Předběžný termín obhajoby

2023/2024 LS – TF

Vedoucí práce

doc. Ing. Miloslav Linda, Ph.D.

Garantující pracoviště

Katedra elektrotechniky a automatizace

Elektronicky schváleno dne 17. 1. 2023

doc. Ing. Monika Hromasová, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 8. 3. 2023

doc. Ing. Jiří Mašek, Ph.D.

Děkan

V Praze dne 14. 9. 2023

Čestné prohlášení

Prohlašuji, že svou diplomovou práci " Regulační systém na platformě Node-RED" jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu použitých zdrojů na konci práce. Jako autor uvedené bakalářské práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne: ....

Albert Suchopár

Poděkování

Děkuji svému vedoucímu práce, doc. Ing. Miloslavu Lindovi, Ph.D., za jeho ochotu a možnost pracovat pod jeho vedením. Rovněž bych chtěl poděkovat své rodině a blízkým za podporu, trpělivost a motivaci při tvorbě této práce.

Regulační systém na platformě Node-RED

Abstrakt

Diplomová práce se zabývá návrhem a vývojem systému pro regulaci teploty využívajícího komponenty internetu věcí (IoT). Klíčové technologie zahrnují platformu Node-RED pro vytváření aplikací, mikrokontrolery ESP8266 pro sběr dat a programovatelné logické automaty (PLC) pro akční řízení.

Práce se zaměřuje na integraci těchto komponent do koherentního systému schopného sběru a analýzy dat. Diplomová práce dokumentuje stanovení cílů a jejich odůvodnění, proces výběru hardwaru, softwarového vývoje, a nakonec evaluaci systému. Výsledkem je prototyp regulačního systému, který představuje možnosti v oblasti aplikace IoT technologií pro domácí automatizaci.

Klíčová slova: Arduino, detektor, Wifi, smart, automatizace

Regulation system on the Node-RED platform

Abstract

The thesis deals with the design and development of a temperature control system utilizing Internet of Things (IoT) components. Key technologies include the Node-RED platform for application creation, ESP8266 microcontrollers for data collection, and programmable logic controllers (PLC) for action control.

The work focuses on integrating these components into a coherent system capable of data collection and analysis. The thesis documents the setting of objectives and their justification, the process of hardware selection, software development, and finally, the system evaluation. The result is a prototype of a control system, which demonstrates possibilities in the field of applying IoT technologies for home automation.

Keywords: Arduino, detector, Wifi, smart, automation

Obsah

Seznam obrázků

Seznam tabulek

Seznam příloh

Seznam zkratek

IoT – Internet věcí

PLC – Programovatelný logický automat

AP – Přístupový bod

IEEE – Institut pro elektrické a elektronické inženýrství

Wi-Fi – Bezdrátová síť s vysokým přenosem dat

VPN – Virtuální privátní síť

WPA – zabezpečení bezdrátových sítí

IDE – Integrovaný vývojový prostředí

DC – Stejnosměrný proud

AC – Střídavý proud

SSR – Polovodičové relé

VCC – Společný kladný napěťový zdroj

GND – Zemní kontakt

ROM – Paměť pouze pro čtení

HTTP – Hypertextový přenosový protokol

MQTT – Protokol pro přenos zpráv

IP – Internetový protokol

PWM – Pulsní šířková modulace

DAC – Digitálně-analogový převodník

ADC – Analogově-digitální převodník

I2C – Multimasterová počítačová sériová sběrnice

GPG – Multiplatformní software pro asymetrickou i symetrickou kryptografii

TCP – Protokolem transportní vrstvy

RTU – Jednotka vzdáleného terminálu

ASCII – Americký standardní kód pro výměnu informací

RAM – Paměť s náhodným přístupem

USB – Univerzální sériová sběrnice

GPIO – Vstupně-výstupní piny s obecným účelem

API – Rozhraní pro programování aplikací

SPI – Sériové periferní rozhraní

UART – Univerzální asynchronní přijímač/vysílač

1 Úvod

Ve světě, kde internet věcí (IoT) nabývá na významu jak v domácnostech, tak v průmyslu, přichází platforma Node-RED s relativně jednoduchým, avšak výkonným způsobem propojení různých zařízení a služeb. Tato diplomová práce se zaměřuje na vytvoření teoretických základů pro následné praktické využití Node-RED pro vývoj prototypu regulačního systému. Přístupnost a flexibilita Node-RED umožňuje snadno integrovat teplotní senzory a další IoT komponenty do uceleného systému. Z tohoto důvodu se práce věnuje jak hardwarovým, tak zejména softwarovým řešením s cílem vytvořit systém, který bude nejen spolehlivý, ale také snadno adaptovatelný a kompatibilní s potencionálními rozšířeními.

Pro demonstraci a ověření regulace byl zvolen prostor s omezenou izolací, aby vnější podmínky měly významný vliv na vnitřní teplotu, což vyžaduje adekvátní reakci regulačního systému. Jako zdroj tepla bylo vybráno olejové topení, u kterého se bude modulovat výkon, což umožňuje částečnou simulaci reálných podmínek obvyklého objektu s centrálním vytápěním na bázi vodního oběhu.

Cílem této diplomové práce je nejen představit praktické aplikace Node-RED v kontextu měření a regulace v budovách, ale také ukázat, jak může být využita k řešení aktuálních výzev v tomto oboru. Práce si tedy klade i za cíl nabídnout inovativní řešení ve srovnání s tradičními přístupy, které je nejen technicky proveditelné, ale také ekonomicky a ekologicky výhodné.

2 Cíl práce

Hlavním cílem této diplomové práce je navrhnout a implementovat inteligentní regulační systém pro domácí automatizaci na bázi platformy Node-RED. Tento prototyp regulačního systému má za úkol ukázat, jak lze Node-RED a širší koncept internetu věcí využít pro regulaci v domácnosti, přičemž klade důraz na několik podstatných aspektů:

- Vytvoření uživatelského rozhraní pro řízení a monitoring systému.
- Zajištění kompatibility s existujícími tradičními regulačními systémy, s cílem rozšířit jejich stávající funkce a začlenit je do ekosystému IoT.
- Možnost regulačního systému predikovat vývoje venkovní teploty a případně i vnitřní teploty místnosti.
- Vyvinout funkční úlohu, která bude reagovat na současné potřeby v oblasti měření a regulace.

3 Metodika práce

Metodika práce zahrnuje nejprve analyzování vlastností platformy Node-RED a relevantních technologií a trendů v oblasti domácí automatizace, což umožní následné vymezení podstatných požadavků pro prvotní představu specifikací při návrhu regulačního systému. Následovat poté bude etapa výběru a zdůvodnění problémů, na které se bude řešení soustředit. Zde se identifikují specifické potřeby a výzvy, kterým má regulační systém čelit.

Další bude fáze návrhu, kde se bude formulovat potenciální řešení a provede se výběr vhodných hardwarových a softwarových komponentů. Implementace systému pak bude zahrnovat analýzu vlastností vybraných komponentů pro jejich následnou instalaci a konfiguraci.

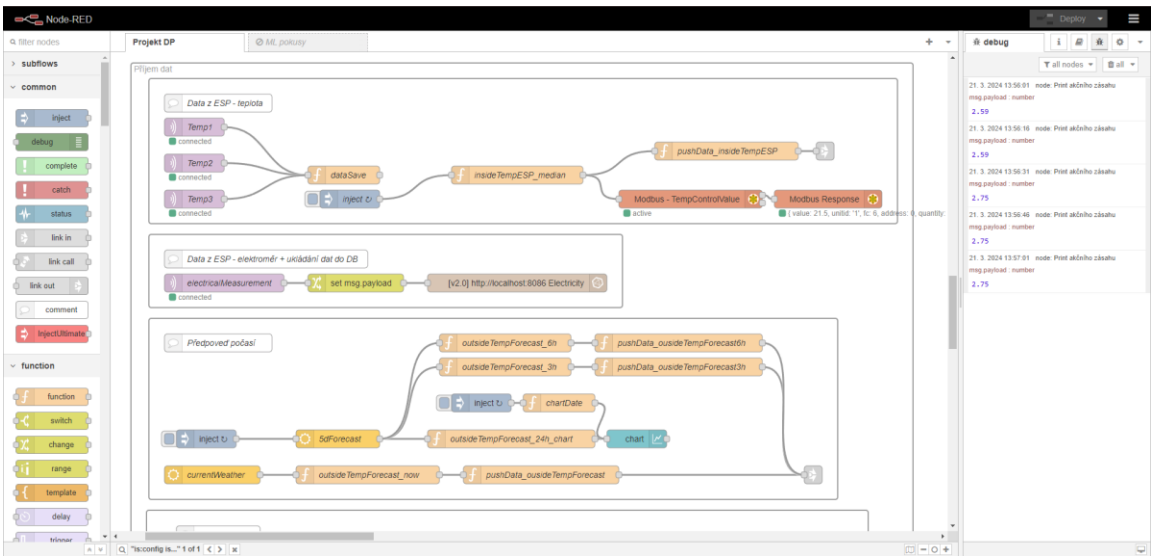
Poslední fáze realizace regulačního systému bude zahrnovat propojení všech zařízení skrze platformu Node-RED, zajištění spolehlivé a bezpečné komunikace, vytvoření logiky pro regulaci a vývoj uživatelského rozhraní. Po dokončení celého regulačního systému bude následovat jeho praktické ověření a stručné hodnocení funkčnosti.

4 Přehled řešené problematiky

V první části této kapitoly se práce soustředí na představení nástroje Node-RED, jeho klíčových vlastností a možností nasazení v praxi. Zvláštní pozornost je věnována také alternativním platformám a komparativní analýze jejich vlastností. Dále kapitola definuje specifické problémy a požadavky na regulační systém, které tato práce má za cíl řešit. Dále se představí několik potenciálních řešení zmiňovaných problémů a provede se výběr toho nejvhodnějšího pro další realizaci, která je zpracována v následující kapitole.

4.1 Node-RED

Node-RED je flexibilní programovací nástroj sloužící k propojení různých zařízení, rozhraní API a online služeb, a který se uplatňuje zejména v oblasti internetu věcí (IoT). Samotné jméno nástroje Node-RED odkazuje na to, že tento nástroj pracuje s takzvanými uzly, z angličtiny nodes. Jeden tento uzel představuje službu v jazyce JavaScript se specifickým účelem: přijímat data, provádět nad nimi dané operace a následně je předávat dále. Spojením několika uzlů vznikne síť s tekoucími daty, což je důvod, proč se tento typ programování označuje jako Flow-based programování. Pro práci s uzly a tvorbu komplexních aplikací, Node-RED poskytuje intuitivní prostředí. viz. Obr. 1. ve formě spustitelného editoru v prohlížeči.[1]



Obr. 1 Ukázka rozhraní Node-RED a Flow-based programování

Co se týče dosavadní historie Node-RED, byl představen v roce 2013 Nickem O'Learym a Davem Conway-Jonesem Emerging odboru Emerging Technology Services společnosti IBM. V září téhož roku byl Node-RED uvolněn jako open-source projekt, což umožnilo jeho rychlý rozvoj a rozšíření. V roce 2016 se Node-RED stal jedním ze zakládajících projektů JS Foundation, která se roku 2019 spojila s Node.js, což vedlo k vytvoření OpenJS Foundation. Tato nadace podporuje růst a udržitelnost open-source projektů v oblasti webu a JavaScriptu.[1, 2]

4.1.1 Klíčové vlastnosti Node-RED

Node-RED je efektivní platforma pro typy vývoje aplikací, zejména v oblasti IoT. Je však podstatné zvážit její omezení pro její konkrétní použití. Proto se v následující části práce shrne podstatné vlastnosti platformy Node-RED a její omezení.

Flow-based programování

Jak již bylo zmíněno, Node-RED využívá programování založené na toku dat, kde každý uzel představuje buď službu nebo zařízení. Uživatelé vytvářejí aplikace spojením specifických uzlů, což umožňuje vizualizovat a editovat celý proces zpracování dat od začátku až do konce. Tento přístup usnadňuje pochopení a úpravy datového toku. Například, některé uzly mohou představovat senzory, jiné uzly mohou zajišťovat rozhodovací logiku, posílat upozornění nebo spouštět další uzly na základě dostupných dat.[1, 3]

Node-RED je navržen s důrazem na jednoduchost a uživatelskou přívětivost, což je v některých případech na úkor výkonu, proto pro tvorbu složitějších nebo výkonově náročnějších aplikací může být Node-RED méně vhodný ve srovnání s tradičními programovacími přístupy, které umožňují více kontrolních a optimalizačních možností. [3, 4]

Open-source dostupnost

Platforma Node-RED je open-source, což znamená, že její zdrojový kód je volně dostupný a licence je přístupná bez omezení. Díky tomu je možné Node-RED provozovat zdarma, což přispělo k vytvoření rozsáhlé komunity uživatelů a vývojářů. Tato komunita se významně podílí na rozvoji platformy prostřednictvím tvorby vlastních knihoven uzlů,

dokumentace a vzorových příkladů, což zvyšuje její flexibilitu a rozšiřitelnost. Vývojářům a výrobcům též umožňuje sdílet vlastní uzly vyvinuté pro specifické účely nebo pro integraci s novými zařízeními a službami.[1, 3]

Nicméně jako každá open-source platforma, i Node-RED může představovat bezpečnostní rizika. Zejména pokud nejsou udržovány některá bezpečnostní opatření, jako je autentizace, autorizace a šifrování. Krom toho při využívání neoficiálních knihoven a uzlů je nutné postupovat s opatrností, neboť mohou obsahovat bezpečnostní slabiny. Rovněž při použití uzlů a knihoven vytvořených komunitou může vzniknout problém s kompatibilitou a udržitelností, zvláště pokud tyto zdroje nejsou dostatečně pravidelně aktualizovány.[1, 3]

Podpora externích služeb a protokolů

Node-RED nabízí rozsáhlou sadu uzlů pro integraci s širokou škálou API a protokolů. V základní sadě uzlů lze nalézt možnosti přijmu i odeslání dat přes HTTP, WebSocket, MQTT, TCP a sériovou komunikaci. Tato základní sada uzlů lze však dále rozšířit, jak již bylo zmíněno, pomocí komunitních knihoven, které poskytují podporu i pro průmyslové protokoly jako Modbus či M-Bus. Díky této rozšiřitelnosti je Node-RED schopen získávat a předávat data téměř z libovolného zdroje, včetně webových služeb, databází, sociálních sítí a IoT zařízení.[3]

Multiplatformní podpora

Node-RED, založený na Node.js, využívající jazyk JavaScript, se vyznačuje svou multiplatformní flexibilitou. Tato vlastnost umožňuje jeho spouštění na různých operačních systémech, včetně Windows, macOS a Linuxu. Díky této univerzálnosti je Node-RED vhodný pro nasazení na různých typech hardwaru, jako jsou Raspberry Pi, BeagleBone, nebo dokonce zařízení s Androidem. Kromě svého využití na koncových zařízeních, Node-RED lze rovněž provozovat, jak je uvedeno na oficiálních stránkách, v cloudovém prostředí, konkrétně na Amazon Web Services nebo Microsoft Azure.[1]

4.1.2 Porovnání Node-RED s alternativními řešeními

V následující části práce jsou stručně popsány a porovnány s Node-RED dvě různé softwarové alternativy pro možnou realizaci regulace, spolu s jednou službou založenou na Node-RED. Ačkoliv na trhu existuje mnoho alternativních řešení, pro účely této analýzy byla vybrána specifická řešení, aby byly zřetelně patrné rozdíly ve vlastnostech Node-RED oproti konkurenčním platformám. Tato práce úmyslně vynechává cloudová řešení, jako jsou AWS IoT Core od Amazonu nebo Azure IoT Hub od Microsoftu, kvůli jejich technické komplexitě a potřebě širších znalostí a porozumění daným platformám.

Je vhodné poznamenat, že ačkoli budou v této práci následující produkty diskutovány jako alternativy k Node-RED, v praxi se často využívá kombinace těchto nástrojů, protože každý nástroj má své konkrétní přednosti a use-casy, není neobvyklé, že v rámci jednoho projektu může být využito i vícero platforem.

Prvním softwarem je Home Assistant, který se specializuje na využití v domácnostech a nabízí pokročilé možnosti pro domácí automatizaci. Dalším zástupcem je IFTTT, který vyniká svým intuitivním a jednoduchým způsobem automatizování úkonů a posledním nástrojem je nadstavba platformy Node-RED nazývaný FlowFuse.

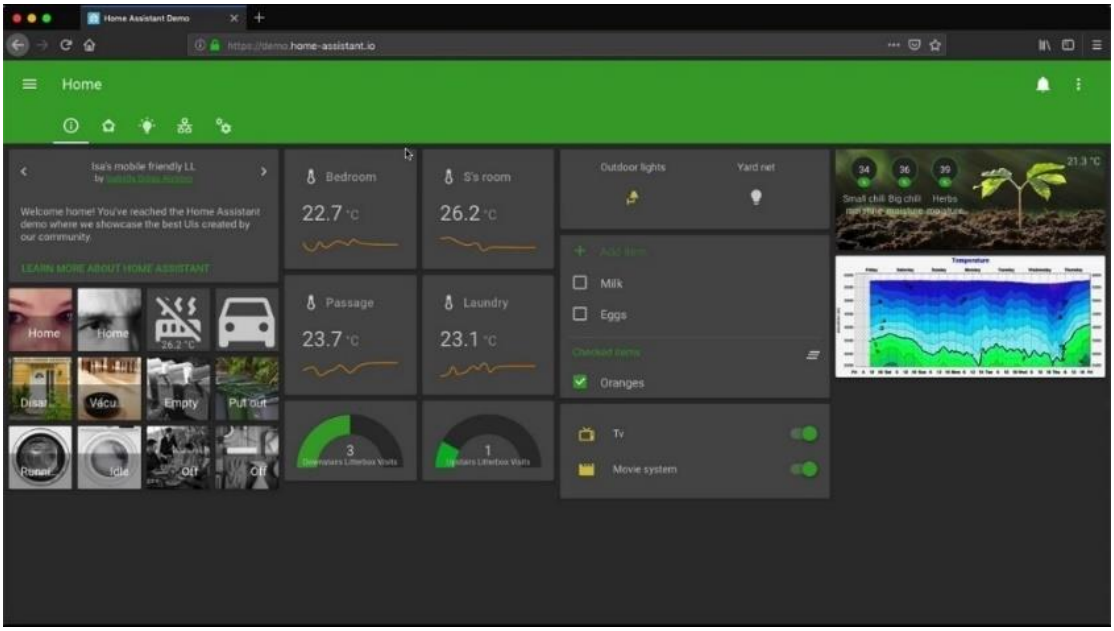
Home Assistant

Home Assistant je open-source platforma zaměřená na uživatelsky přívětivou integraci a automatizaci chytrých zařízení v domácnosti, včetně osvětlení, termostatů a kamer. Home Assistant nabízí informační panely, viz. Obr. 2, které ve srovnání s dashboardem Node-RED poskytují mnohem širší možnosti pro jejich úpravy a vizualizaci různorodého obsahu.[5]

Ve srovnání s Node-RED, Home Assistant nabízí srovnatelnou podporu pro externí protokoly, API a služby. Avšak manipulace s daty v Home Assistant probíhá pomocí vlastních skriptů. To je částečně důvod, proč se Home Assistant často používá ve spojení s Node-RED, který zpracovává externí data, a následně upravené informace posílá do Home Assistant pro vizualizaci dat.[6]

Stejně jako Node-RED, i Home Assistant disponuje díky své otevřené licenci širokou podporou komunitních doplňků a lze jej nasadit na různých zařízeních. Je často využíván i na NAS zařízeních jako domácí multimediální centrum.[6]

Lze konstatovat, že zatímco Home Assistant je ideální pro uživatele hledající specializované řešení pro domácí automatizaci, Node-RED nabízí větší flexibilitu a širší možnosti v obecné automatizaci a integraci různých systémů a služeb.



Obr. 2 Příklad informativních panelů Home Assistant [7]

IFTTT

If This Then That (z angl. Jestli tohle, tak tamto) je populární webová služba pro automatizaci, která se zaměřuje na propojení již existujících webových služeb, aplikací a zařízení tvořením jednoduchých podmínek, známých jako "applets". Ačkoliv IFTTT nabízí základní plán zdarma, za pokročilé funkce a větší počet appletů si služba účtuje poplatek.[8]

Ve srovnání s Node-RED, IFTTT nemá tak rozsáhlé možnosti pro komplexní automatizaci, ale to, co nabízí, poskytuje velmi intuitivním způsobem, včetně možnosti ovládání přes mobilní aplikaci. Například, umožňuje vytvořit applet, který propojí ekosystém chytrých termostatů a hlasového asistenta s podmínkou: Když dojde k hlasovému příkazu, tak se spustí termostat v určité místnosti. Tato platforma umožňuje

automatizovat jednoduché úkony napříč různorodými službami a ekosystémy zařízení a spolupracuje s více než 600 různými webovými aplikacemi a službami, včetně technologických gigantů jako Google, Amazon, Facebook a Dropbox.[8]

Pokud jde o využití IFTTT pro vytváření komplexních regulačních systémů, není to obecně doporučováno, jelikož IFTTT neposkytuje pokročilé nástroje pro regulaci. Nicméně, ve spolupráci s Node-RED, může být IFTTT užitečný pro integraci s externími webovými aplikacemi a službami, které Node-RED nemusí přímo podporovat.

FlowFuse na platformě Node-RED

V kontextu alternativ k Node-RED je vhodné zmínit produkt vyvinutý tvůrci Node-RED, známý jako FlowFuse. Tento produkt byl vyvinut, aby řešil specifická omezení Node-RED, zejména v oblastech bezpečnosti a nasazení v podnikovém prostředí. FlowFuse staví na základech Node-RED a rozšiřuje je o klíčové vlastnosti, jako je hosting s údržbou na vlastních serverech, což zvyšuje bezpečnost užívání. Kromě toho byla přidána funkce kontroly verzování projektu a centralizace managementu instancí, umožňující přístup ke všem verzím projektů z jednoho místa. Navíc FlowFuse nabízí vlastní certifikované knihovny uzlů, u kterých zaručuje jejich kvalitu, bezpečnost a dlouhodobou podporu. Tyto kroky zvyšují celkovou robustnost původního Node-RED a rozšiřují jeho využití v podnikovém prostředí, ve kterém dosud Node-RED nenašel takové uplatnění.[7]

4.2 Analýza a specifikace požadavků

Tato část práce se zaměřuje na definici konkrétních problému spojených s regulací tepla a následné specifikace funkčních a nefunkčních požadavků na systém. Bude zde vysvětleno, proč byly tyto požadavky vybrány a jak přispívají k celkovému cíli projektu.

4.2.1 Definice problému

Jedním ze základních problémů, kterým se práce věnuje, je řízení teplotního komfortu v prostorách s omezenou schopností tepelné izolace. Pro praktickou demonstraci byl vybrán zahradní domek, který je typickým příkladem stavby s nízkými izolačními vlastnostmi, a proto je zde teplota venkovního prostředí významným faktorem ovlivňujícím vnitřní teplotu místnosti.

Aby byly v diplomové práci demonstrovány možnosti kompatibility regulačního systému, začleněn je také programovatelný logický automat (PLC), konkrétně model AMiNi4DW2 od společnosti Amit. Toto PLC zodpovídá za řízení olejového radiátoru Eurom RAD 2000, který reguluje topný výkon skrze kontrolu přívodu elektřiny. Další výzvou v rámci této práce bude tedy i implementace a zprovoznění komunikace mezi platformou Node-RED a AMiNi4DW2.

S účinností od 1.1.2023 byl schválen Zákon č. 424/2022 Sb., který nařizuje od 1.1.2024 odečítat měřiče tepla a teplé vody 1x měsíčně a naměřené hodnoty předávat konečným spotřebitelům. Dílčím cílem této diplomové práce je zaměřit se na aktuální problematiku v oblasti měření a regulace. Z tohoto důvodu práce definuje potřebu periodického odečtu energie, aby vyhověla požadavkům zmíněné vyhlášky. Pro tento účel je využit jednofázový elektroměr s funkcí dálkového odečtu pomocí S0 impulsů. S0 impulsy představují standardizovaný výstup, který je běžně využíván jak u elektroměrů, tak i u dalších přístrojů měřící spotřebu energií. Z toho důvodu by mělo být možné aplikovat navrhované řešení nejen na elektroměry, ale i na vodoměry, měřiče tepla a kalorimetry. [9]

4.2.2 Specifikace funkčních a nefunkčních požadavků

Funkční požadavky jsou popsány následovně:

- **Data o počasí:** Systém bude schopen přijímat meteorologická data v reálném čase.
- **Komunikace v síti:** Zařízení musí být schopna vzájemné komunikace, preferovaně na bezdrátové technologie.
- **Uživatelské rozhraní:** Systém bude poskytovat interface pro nastavení a monitoring regulace, včetně odesílání notifikací pomocí SMS či emailů. Dovolí-li to možnosti, bude integrován do domácího hlasového asistenta.
- **Měření spotřeby energie:** Systém bude schopen zaznamenávat spotřebu elektřiny pro pravidelné měsíční odečty elektroměrů.
- **Prediktivní regulace:** Systém bude predikovat teplotní změny pro optimální nastavení regulace.

4.2.3 Specifikace nefunkčních požadavků

Nefunkční požadavky na regulační systém jsou definovány takto:

- **Bezpečnost:** Cílem je zajistit, aby byl celý systém provozován převážně v lokální síti, přičemž přístup k internetu bude povolen pouze pro ověřené a bezpečné služby. To zahrnuje implementaci vhodných bezpečnostních protokolů a šifrování komunikace pro minimalizování veškerých rizik.
- **Robustnost:** Systém musí být dostatečně robustní a spolehlivý pro nepřetržitý dlouhodobý provoz. V případě výpadku energie bude systém schopen automatické obnovy bez zásahu uživatele. Je také kritické zajistit efektivní monitorování a kontrolu komunikace mezi zařízeními, aby byl potvrzen jejich chod. V případě ztráty komunikace nebo jiných technických problémů musí systém přejít do bezpečnostního režimu, který zajistí základní funkčnost a omezení rizik až do obnovení plného provozu.
- **Škálovatelnost a rozšiřitelnost:** Systém musí být navržen tak, aby umožňoval snadné přidávání nových zařízení a poskytoval flexibilitu pro rozšiřování funkcí nebo integraci nových služeb a aplikací.

4.3 Návrh řešení

V následující kapitole budou představeny různé přístupy k realizaci tepelné regulace v místnosti. Návrhy jsou koncipovány tak, aby co možná nejlépe řešily definované problémy práce a dodržely veškeré funkční i nefunkční požadavky.

Následně proběhne výběr nejvhodnější varianty s jejím kritickým ohodnocením. Posledním krokem v této kapitole bude návrh vybraného řešení a popis veškerých komponentů, které bude toto řešení vyžadovat.

4.3.1 Návrh a výběr řešení

Na základě již definovaných požadavků na regulační systém, je možné formulovat předběžnou koncepci návrhu. Podstatným prvkem je samotný výběr platformy pro nasazení Node-RED. Dále je nutné vybrat protokol pro komunikaci mezi Node-RED a PLC, stejně jako mezi Node-RED a periferními zařízeními. V závislosti na volbě protokolu bude potřeba vybrat i vhodná periferní zařízení a případně i další komponenty. Je také důležité zvážit způsob ukládání dat a tvorba uživatelského rozhraní.

Platforma pro Node-RED

V případě nasazení Node-RED lze uvažovat o dvou hlavních alternativách: nasazení na vzdáleném hostovaném serveru (cloudovém prostředí) nebo na lokálním zařízení. [10]

V případě cloudového nasazení Node-RED je nutné pro splnění požadavků zabezpečit komunikační kanály mezi místní sítě a Node-RED, pravděpodobně prostřednictvím VPN bridge. Tato metoda vyžaduje zařízení, které funguje jako VPN gate a zajišťuje bezpečný přenos dat, což vede ke zvýšené složitosti systému kvůli požadavkům na konfiguraci VPN.

Aby byly splněny bezpečnostní požadavky pro regulační systém a zároveň se minimalizovaly technické a finanční nároky, je preferováno nasazení Node-RED na hardware umístěný v lokální síti. Jak již bylo zmíněno, Node-RED lze implementovat na kompaktních modulárních počítačích jako Raspberry Pi nebo BeagleBone, které jsou poměrně cenově dostupné a vyšší modely poskytují i potřebný dostatečný výkon pro chod všech potřebných služeb.

S ohledem na jeho rozsáhlé použití v oblasti IoT a širokou dostupnost komunitních zdrojů a návodů, byl pro tento projekt zvolen hardware Raspberry Pi. Tato volba byla motivována nejen širokou podporou v komunitě oproti BeagleBone, ale také vyšší úrovní adaptability a spolehlivosti výrobku.

Protokol komunikace Node-RED s PLC

Kompaktní PLC AMiNi4DW podporuje několik komunikačních rozhraní s různými protokoly. Konkrétně umožňuje sběrnicovou komunikaci prostřednictvím RS232 a RS485 nebo síťovou komunikaci prostřednictvím ethernetu. V případě sběrnicové komunikaci jsou podporovány protokoly Modbus RTU, Modbus ASCII a specifický protokol ARION, vyvinutý společností AMIT pro výhradně komunikaci mezi jejími zařízeními. Přes ethernet je zařízení schopné komunikovat pomocí protokolů Modbus TCP a telemetrie IEC 60870-5-104. [11]

Vzhledem k rozšířenosti a flexibilitě je pro komunikaci mezi Node-RED a AMiNi4DW doporučován protokol Modbus TCP přes ethernet. Tato volba je výhodná nejen kvůli použití ethernetu, který je běžným standardem v mnoha aplikacích, eliminujícím potřebu dodatečné infrastruktury, ale také kvůli rozšíření protokolu Modbus TCP, což usnadňuje integraci díky dostupnosti odpovídajících knihoven v Node-RED.

*Protokol komunikace Node-RED s periférie*mi

Co se týče komunikace mezi periferiemi a Node-RED, existuje mnoho dostupných možností. Avšak v mnoha projektech se upřednostňuje protokol MQTT. Tento protokol, zaměřený na machine-to-machine (M2M) komunikaci, je vhodný pro aplikace v oblasti internetu věcí**,** což jej činí ideální volbou pro použití v rámci této práce.

Jednou z klíčových charakteristik MQTT je potřeba brokeru, což je serverová služba, která koordinuje komunikaci mezi zařízeními, zajišťuje jejich autentizaci a umožňuje filtrování zpráv. V kontextu tohoto projektu bylo již rozhodnuto využít lokální hardware pro nasazení systémových komponent. V důsledku tohoto rozhodnutí lze MQTT broker též implementovat na zmíněném zařízení. Vhodnou volbou implementace je Mosquitto, open-source MQTT broker od Eclipse Foundation, který je není výpočetně příliš náročný a současně splňuje veškeré požadavky projektu. [11]

Periferní zařízení

Pro výběr periferních zařízení je klíčové zohlednit podporu komunikace prostřednictvím protokolu MQTT, ideálně přes TCP/IP. Výhodou je přístup k těmto zařízením prostřednictvím Wi-Fi, což vzhledem k její všeobecné dostupnosti usnadňuje implementaci.

Dále je důležité zvážit funkčnost těchto zařízení. V kontextu této práce budou zařízení sloužit k měření teploty v prostoru a vyčítání dat z elektroměrů. I když na trhu existují hotové produkty splňující tyto požadavky, může být jejich kompatibilita s vybranými systémy omezená a často jsou finančně náročnější ve srovnání s dostupnými mikrokontrolery.

Energeticky nenáročné mikrokontrolery, jako jsou ESP32 a ESP8266, nabízejí možnost programování pro specifické účely, jako je měření teploty nebo odečty hodnot z elektroměrů. Tyto zařízení rovněž obsahují integrovaný Wi-Fi modul s frekvencí 2,4 GHz, díky čemuž se hojně užívají pro aplikace v IoT. Obecně v porovnání s hotovými produkty mají tyto mikrokontrolery tendenci k nižší robustnosti, z důsledku toho jsou vyžadována preventivní opatření pro zajištění spolehlivosti a odolnosti vůči poruchám. Přesto, díky jejich nízkým nákladům a vysokému stupni přizpůsobitelnosti, představují mikrokontrolery vhodnou volbu pro nasazení v rámci tohoto projektu.

Ukládání dat a uživatelské rozhraní

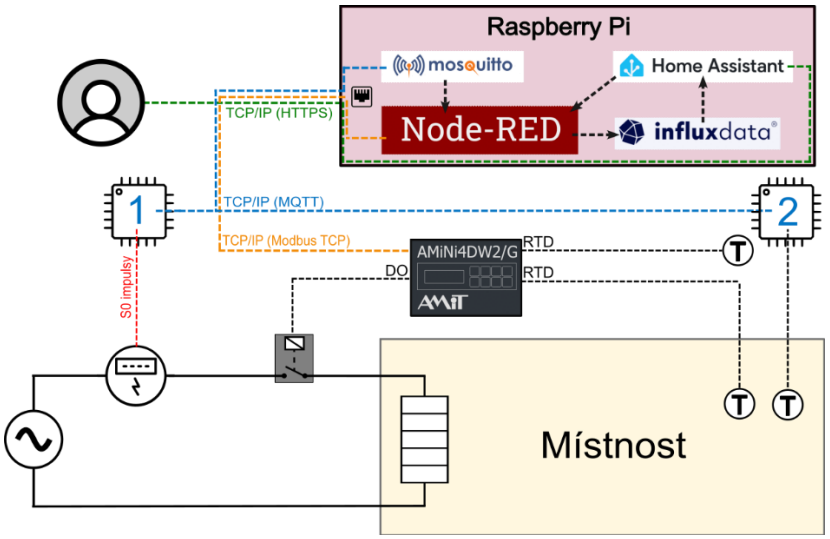
Node-RED, s jeho vysokou úrovní flexibility, umožňuje integraci s širokou škálou databázových systémů, včetně běžně používaných platforem jako MySQL, Oracle nebo PostgreSQL. V oblasti IoT se však zvláště osvědčila open-source platforma InfluxDB od společnosti InfluxData, která je speciálně navržena pro sběr, ukládání, zpracování a vizualizaci dat z měření. Tato platforma je pro potřeby tohoto projektu vhodnou volbou.[12]

Pokud jde o uživatelské rozhraní, nelze se spolehnout pouze na standardní UI funkce InfluxDB, jelikož podporují pouze vizualizaci dat bez možnosti interaktivního ovládání systému. Pro regulaci domácího prostředí lze zvažovat použití dashboardu Node-RED nebo Home Assistant. V rámci tohoto projektu byla zvolena platforma Home

Assistant, která nabízí rozsáhlejší možnosti přizpůsobení. Navíc Home Assistant podporuje integraci multimediálního centra a snadnější integraci hlasových asistentů.

4.3.2 Popis vybraného řešení

Při zohlednění zvolených řešení, konečný návrh systému odpovídá schématu prezentovanému na Obr. 3. V tomto návrhu je platforma Node-RED implementována na lokálním zařízení, mikropočítači Raspberry Pi. Na stejném zařízení je také nasazen MQTT broker Mosquitto, který zprostředkovává přenos dat mezi mikrokontrolery a Node-RED. Pro ukládání dat byla vybrána platforma InfluxDB od společnosti InfluxData, která slouží jako zdroj dat pro Home Assistant. Home Assistant v tomto kontextu funguje jako uživatelské rozhraní a rozšiřuje spektrum možností interakce s uživatelem.



Obr. 3 Schéma vybraného návrhu technologie

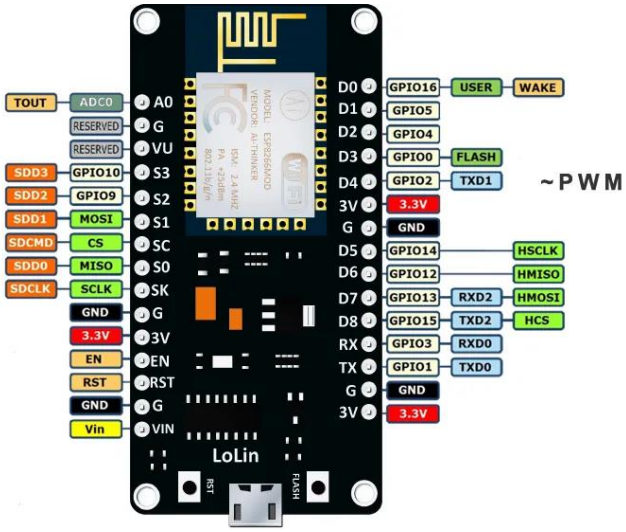
Co se týče periferních zařízení, mikrokontroler 1 je určen pro čtení impulzů z elektroměru, zatímco mikrokontroler 2 je nasazen pro měření teploty v místnosti. Obě tyto zařízení budou připojena bezdrátově přes Wi-Fi k lokální síti, ve které se nachází i Raspberry Pi se všemi službami. V této síti se nachází i PLC AMiNi4DW2, jež řídí stykač přívodu energie k topnému systému. K PLC jsou připojena také dvě teplotní čidla, měřící teplotu v místnosti a venkovní teplotu, jejichž údaje slouží pro srovnání s daty z mikrokontroleru 2 a získanými meteorologickými daty z Node-RED.

V další části se diplomová práce podrobně věnuje vybranému hardwarovému řešení, včetně jeho konektivity, použitého vývojového prostředí a dostupných periferních zařízení.

4.3.2.1 Vývojová desky ESP8266 Lua Nodemcu v3

ESP8266 je mikrokontroler navržený a vyrobený společností Espressif Systems, známý pro svou cenovou dostupnost a schopnost Wi-Fi připojení. Tento mikrokontroler poskytuje samostatné řešení pro Wi-Fi komunikaci, což umožňuje jeho využití v různých síťových aplikacích. Mikrokontrolery ESP8266 však nejsou přímo určeny pro okamžité použití v běžných aplikacích; proto jsou často adaptovány výrobci třetích stran, kteří je zakomponovávají do obvodů svých desek plošných spojů, které pak slouží jako vývojové desky. Tyto modulární desky pak nabízejí možnost připojení přes standardní GPIO piny. Cena těchto desek se liší podle velikosti, počtu GPIO pinů a dalších funkčních prvků.

ESP8266 Lua Nodemcu v3 je vývojová deska, která obsahuje mikrokontroler ESP8266, a umožňuje snadnou integraci a vývoj vlastního hardwaru díky flexibilnímu pinoutu. Detailní rozložení pinů této vývojové desky je patrné z Obr. 4. Kromě GPIO pinů, deska také nabízí výstupy pro napájení a podporuje UART komunikaci skrze USB konektor, který je zároveň využíván pro programování a nahrávání zdrojového kódu.[13]



Obr. 4 Schéma vývojové desky ESP8266 Lua Nodemcu v3 [13]

ESP8266 Lua Nodemcu v3 má vlastní napěťový regulátor a lze jej kromě USB napájet i pinem *Vin*. Na desce lze najít kromě Wi-Fi čipu i ESP8266EX i 32bitový procesor Tensilica L106. FLASH paměť vývojové desky umožňuje nahrání programů o velikosti do 4 MB. Další parametry této vývojové desky jsou podrobně specifikovány v následující Tab. 1.[13]

Tab. 1 Parametry ESP8266 Lua Nodemcu v3 [14]

Vstupní napětí	DC 4,5 ~ 9 V, lze napájet přes USB
Proudový odběr	70 mA ~ 200 mA; (Standy: <200 uA)
Výstupní napětí pro periférie	DC 3,3 V a 5 V (pokud je napájen z USB)
Paměť FLASH	4 MB
GPIO	11 × digitálních I/O pinů 1 × AI multiplexovány GPIO funkce PWM, I2C, UART, SPI, one-wire
Wi-Fi	Standard 802.11 b/g/n, 2,4 GHz, podpora zabezpečení WPA / WPA2; podpora až 5 klientských TCP/IP spojeních
Pracovní teplota (neoficiální)	-40–85 °C

Wi-Fi

Co se týče čipu ESP8266EX a jeho schopnosti Wi-Fi přenosu, Wi-Fi je technologie bezdrátového přenosu dat mezi zařízeními nebo pro připojení k internetu. Termín Wi-Fi je zkratkou pro Wireless Fidelity. Tato technologie využívá pro přenos dat rádiové vlny s frekvencemi 2,4 a 5 GHz v oblasti ISM pásma, což jsou bezlicenční pásma umožňující volné vysílání. Frekvence 5 GHz umožňuje přenos většího množství dat, ESP8266 však používá frekvenci 2,4 GHz, která je energeticky méně náročná. Wi-Fi standardy jsou definovány organizací IEEE, která určuje všechny protokoly, jež mikrokontroler podporuje, tj. 802.11 b/g/n. Každý z těchto standardů se liší rychlostí přenosu dat, dosahem signálu a dalšími parametry.[15]

Pro spojení mikrokontroleru se sítí je nutné další zařízení, které plní funkci vysílače/přijímače a umožňuje převod signálů Wi-Fi na ethernet. Tyto zařízení, známá jako Access Pointy (AP), jsou běžným prvkem pro vytvoření síťové infrastruktury. AP obvykle vysílá Wi-Fi signál, ke kterému se mohou připojit klientská zařízení, jako je například mikrokontroler ESP8266. Vysílání Wi-Fi signálu přináší specifická bezpečnostní rizika související s potenciálními zranitelnostmi sítě. Za účelem zajištění bezpečného přenosu dat se proto implementují různé verze Wi-Fi Protected Access (WPA) protokolů.[15]

Konektivita

Aby se zajistilo správné sestavení hardwarových komponent, je nezbytné porozumět detailním funkcím GPIO pinů. Piny označené jako S0 až S3, SK a SC jsou určeny pro spojení s vnitřní pamětí vývojové desky. Klíčové piny pro účely tohoto projektu, D0 až D8, fungují jako digitální vstupy a výstupy, kromě toho, že podporují pulsní šířkovou modulaci (PWM) a přenos dat po 1-wire, nabízí i další možnosti využití. Detailní přehled funkcí těchto pinů poskytuje následující tabulka.[16]

Tab. 2 GPIO ESP8266 Lua Nodemcu v3 [16]

Popis na desce	GPIO	Dodatečné funkce	Komentář
Analogové piny			
A0	ADC0	Vstup	10bitový ADC
Digitální vstupy a výstupy			
D0	GPIO16	Vstup/Vystup; Wake up; PWM	Probuzení zařízení
D1	GPIO5	Vstup/Vystup; SCL; PWM; 1-Wire	I2C
D2	GPIO4	Vstup/Vystup; SDA; PWM; 1-Wire	I2C
D3	GPIO0	FLASH tlačítko	Pro bootování
D4	GPIO2		Pro bootování
D5	GPIO14	Vstup/Vystup; SCLK; PWM; 1-Wire	SPI
D6	GPIO12	Vstup/Vystup; MISO; PWM; 1-Wire	SPI
D7	GPIO13	Vstup/Vystup; MOSI; PWM; 1-Wire	SPI
D8	GPIO15	Vstup/Vystup; CS	SPI
RX	GPIO3	Vstup; log 1 při spuštění	UART
TX	GPIO1	Výstup; log 1 při spuštění	UART

Pulzně šířková modulace (PWM) je na vývojových deskách využívána jako technika, která umožňuje simulaci analogového výstupu prostřednictvím digitálních výstupních pinů. Princip PWM spočívá v rychlém spínání výstupu mezi stavem logické jedničky (log 1) a logické nuly (log 0). Poměr mezi dobou, po kterou je výstup ve stavu logické jedničky, a celkovou periodou signálu určuje efektivní analogovou hodnotu výstupu, kterou zařízení interpretují jako kontinuální úroveň výkonu. Tento poměr, nazývaný též střída či z angličtiny duty cycle, je vyjádřen v procentech.[17]

1-Wire představuje komunikační protokol, který umožňuje jednoduchou sériovou komunikaci pomocí jediného vodiče (a vodiče pro zem) mezi mikrokontrolerem a jedním nebo více periferními zařízeními. Zařízení komunikující přes 1-Wire mohou být napájena

přímo z datového vodiče, což je proces známý jako parazitické napájení. Unikátní 64bitový kód každého 1-Wire zařízení umožňuje jeho jednoznačnou identifikaci v rámci systému. Tento protokol podporuje komunikaci na vzdálenosti řádově jednotek metrů.[18]

V předchozí tabulce si lze všimnout zkratky I2C, což označuje sériový komunikační protokol určený pro komunikaci na krátkou vzdálenost mezi integrovanými obvody. I2C využívá dva signální vodiče: SDA (Serial Data Line) pro přenos dat a SCL (Serial Clock Line) pro synchronizaci přenosu dat. Všechna zařízení připojená k I2C sběrnici mají svou jedinečnou adresu, přičemž komunikační architektura je založena na master/slave principu, kde master (řídící zařízení) iniciuje komunikaci s jedním či více slave zařízeními.[19]

Další komunikační protokol uvedený v Tab. 2 je SPI (Serial Peripheral Interface). Na rozdíl od I2C, SPI využívá čtyři vodiče, což umožňuje vyšší přenosovou rychlost. Vodič MISO (Master In Slave Out) umožňuje přenos dat od periferních zařízení k mikrokontroleru, zatímco MOSI (Master Out Slave In) slouží k odesílání dat od mikrokontroleru k periferním zařízením. SCK (Serial Clock) zajišťuje synchronizaci datového přenosu. Vodič CS (Chip Select) umožňuje master zařízení vybrat konkrétní slave zařízení pro komunikaci, což eliminuje potřebu adresování zařízení jako je v případě případě I2C.[20]

Posledním protokolem uvedeným v Tab. 2 je UART (Universal Asynchronous Receiver/Transmitter), který se liší tím, že poskytuje asynchronní sériovou komunikaci pro point-to-point spojení. Používá dva vodiče: TX (Transmit) pro odesílání dat a RX (Receive) pro příjem dat Jednou z vlastností UART je jeho schopnost full duplexní komunikace, což znamená, že umožňuje současné odesílání a přijímání dat bez vzájemného rušení.[21]

Vývojové prostředí

K programování ESP8266 Lua Nodemcu v3 lze využít řadu vývojových prostředí jako je například ESP-IDF, PlatformIO nebo Arduino IDE. Arduino IDE je častá volba, kvůli jeho velkému množství uživatelských návodu a přívětivému grafickému prostředí pro tvorbu, nahrání a odladění programů. Jeho editor nabízí například takové funkce jako

barevné zvýraznění syntaxe, automatické doplňování kódu či možnost tvorby projektu z vícero souborů.[14, 22]

Arduino IDE též podporuje širokou škálu vývojových desek a mikrokontrolerů, což umožňuje v rámci jednoho projektu kombinovat hardware od různých výrobců. Výhodou vývojového prostředí Arduino IDE je též jeho možnost stažení zdarma a jeho kompatibilita s mnohými operačními systémy jako je Windows, macOS či Linux.[22]

Samotný program pro ESP8266 Lua Nodemcu v3 lze psát programovacími jazyky Lua, C či C++. V případě vývojového prostředí Arduino IDE se použije modifikovaná verze jazyka C. [16, 22]

Periferie

Za účelem měření teploty je potřeba připojení periferního zařízení k vývojové desce, pro tento účel byl vybrán digitální teploměr DS18B20. Tento senzor nabízí relativně jednoduchou integraci prostřednictvím 1-Wire sběrnice, což umožňuje připojení více senzorů k jedinému datovému vodiči.

Obr. 5 UMW DS18B20 Digitální čidlo teploty TO-92 a jeho pinout [23, 24]

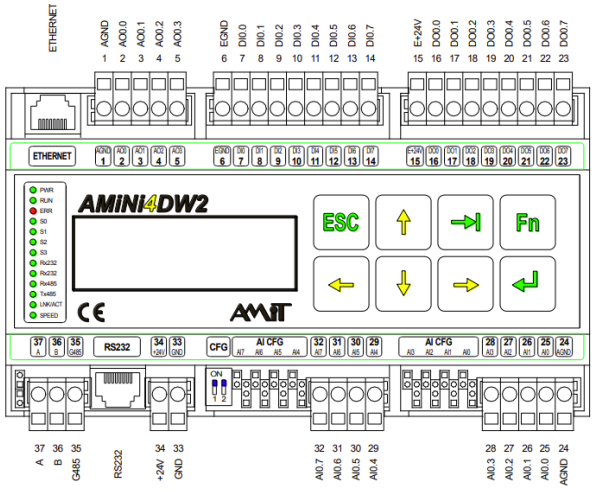
V rámci tohoto projektu bylo preferováno použití pouzdra TO-92, jehož konfigurace pinů je zobrazena na Obr. 4: PIN 1 (GND) vodič pro zem, PIN 2 (DQ) má dvojí funkci pro vstupní a výstupní data, jakož i pro napájení skrze 1-Wire sběrnici, zatímco PIN 3 (VDD) je designován pro možnost samostatného napájení. Další podrobnosti a specifikace digitálního teploměru DS18B20 jsou uvedeny v Tab. 4.

Tab. 3 Parametry digitální teploměru DS18B20

Rozsah měření teploty	-55 °C až +125 °C
Přesnost	±0,45 °C v rozsahu -10 °C až +70 °C
Napájecí napětí	2,5 V až 5,5 V
Komunikační rozhraní	1-Wire
Programovatelné rozlišení	9 nebo 12 bitů
Výstupní formát teploty	Digitální, přes sériovou komunikaci
Doba převodu teploty	400 ms (při rozlišení na 12 bitů)

4.3.2.2 PLC Amit AMiNi4DW2

PLC neboli programovatelný logický automat, v současné době lze charakterizovat jako specializovaný průmyslový počítač určený primárně pro účely automatizace. Zjednodušeně řečeno, pro PLC je typické provádění programů v neustálých cyklech, kde nejprve dochází k odečítání hodnot ze vstupních portů, k nimž jsou připojeny různé typy senzorů. Následuje zpracování těchto dat v rámci uživatelsky definovaného programu a v závěru cyklu jsou zpracovaná data vysílána do výstupních portů, odkud jsou přenášena na akční členy, jako jsou motory, ventily či jiné akční členy. Tento proces se poté cyklicky opakuje.



Obr. 6 Rozložení terminálu a portů AMiNi4DW od společnosti Amit [25]

V kontextu PLC, model AMiNi4DW2 od společnosti Amit je charakterizován jako kompaktní PLC, které je vhodné pro malé až střední automatizační úlohy. Toto zařízení poskytuje řadu komunikačních možností a je kompatibilní s běžnými průmyslovými protokoly, jak je specifikováno v Tab. 4. Vzhledem k jeho konstrukci, pro rozšíření počtu

vstupů a výstupů je nutné připojit dodatečné rozšiřovací moduly. Jak je však zřejmé z Obr. 6, zařízení AMiNi4DW2 poskytuje dostatečné množství vstupů a výstupů pro potřeby této práce. [25]

Tab. 4 Parametry Amit AMiNi4DW2 [25]

Vstupní napětí	DC 19.2 ~ 28.8 V
Paměť FLASH	2 MB + 256 KB
Vstupy	8 × DI (24 V AC/DC) 8 × AI (0-5 V / 0-10 V / 0-20 mA / Ni1000 / Pt1000
Výstupy	8 × DO (24 V / 0.3 A DC) 4 × AO (0-10 V max. 10 mA)
Komunikační rozhraní	RS232 (RJ45) RS485 (Wago konektor) Ethernet (RJ45)
Komunikační protokoly	Modbus RTU/TCP/ASCII; ARION; IEC 60870-5-104
Pracovní teplota	0–50 °C

Modbus TCP

AMiNi4DW2 poskytuje širokou škálu komunikačních rozhraní a protokolů. Vzhledem k tomu, že v rámci projektu byl již předem zvolen Modbus TCP, bude se následující část práce věnovat výhradně tomuto protokolu.

Modbus TCP, komunikační protokol založený na master-slave architektuře, představuje rozšíření tradičního sériového standardu Modbus RTU pro využití v TCP/IP sítích. Tento protokol je široce využíván v průmyslových aplikacích pro účely sběru dat z čidel, dálkového monitorování a řízení zařízení, jakož i pro síťovou komunikaci mezi HMI (Human Machine Interface) panely a řídícími systémy.[26]

Na rozdíl od sériové verze Modbusu, Modbus TCP zahrnuje MBAP (Modbus Application Protocol Header) hlavičku pro každou zprávu v rámci TCP/IP, která obsahuje identifikaci transakce, identifikaci protokolu, délku zprávy a identifikaci jednotky pro efektivní síťovou komunikaci. Maximální délka zprávy může dosáhnout 260 bajtů (včetně MBAP hlavičky). Přesto si protokol zachovává standardní funkční kódy Modbusu určené pro čtení a zápis dat do specifických registrů.[26]

Ke každému registru se přistupuje konkrétním funkčním kódem, který definuje operaci a datový typ hodnoty s kterou se pracuje. Každý registr má rozsah adres

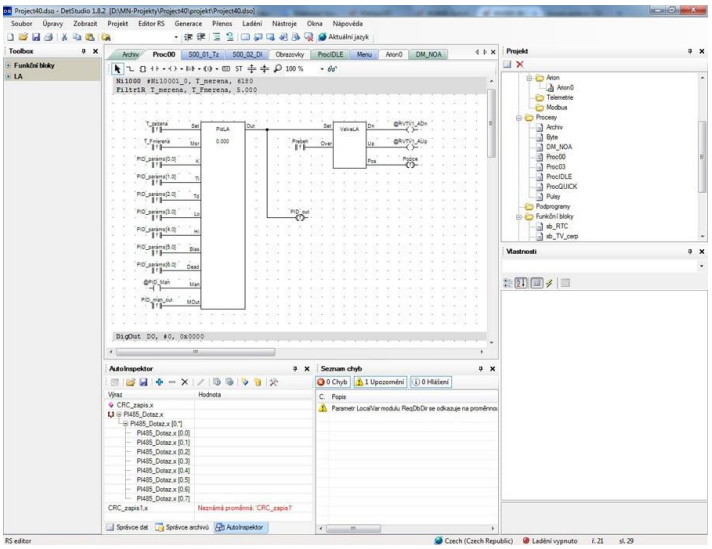
od 0 do 65 535, kdy množství uložených hodnot je ovlivněno jejich datovým typem. Přehled běžně používaných funkčních kódů je popsán v Tab. 5.[26, 27]

Tab. 5 Běžně používané funkční kódy komunikačního protokolu Modbus [27]

Funkční kód	Jméno funkce	Funkce
01	Coils	čtení a zápis on/off stavů zařízení
02	Discrete Inputs	čtení on/off stavů zařízení
03	Multiple Holding Registers	čtení a zápis matic numerických hodnot z více adres
04	Input Registers	čtení numerické hodnoty
05	Single_Coil	zápis on/off stavů zařízení
06	Single Holding Register	zápis numerické hodnoty
15	Multiple Coils	zápis on/off stavů zařízení do více adres
16	Multiple Holding Registers	zápis více numerických hodnot do více adres

Vývojové prostředí

Pro tvorbu uživatelských aplikací na všech standardních řídích systému společnosti Amit se využívá programovací prostředí DetStudio. Podle standardu IEC 61131-3 pro programování PLC nabízí kromě strukturovaného programování i tvorbu logiky pomocí leader diagramů, nebo funkčních blokových diagramů. Všechny tyto způsoby programování lze kombinovat. Samotné Součástí DetStudio je i grafický editor pro tvorbu ovládacích prvků na terminech zařízení. [28]



Obr. 7 Funkční blokové diagramy ve vývojovém prostředí DetStudio [28]

4.3.2.3 Mikropočítač Raspberry Pi 4 Model B

Raspberry Pi je série kompaktních jednodeskových počítačů vyvinutých Raspberry Pi Foundation s primárním cílem podpořit výuku informatiky ve školách. Během let se Raspberry Pi rozšířilo a získalo popularitu nejen v edukační sféře, ale také mezi amatérskými nadšenci a v oblasti průmyslové automatizace.

Obr. 8 Raspberry Pi 4 Model B [29]

Pro tento projekt byl z široké škály modelů Raspberry Pi zvolen, v období realizace této práce, jeden z nejvýkonnějších, konkrétně Raspberry Pi 4 Model B, jež se nachází na Obr. 6. Oproti svým předchůdcům nabízí výkonnější čtyřjádrový procesor s taktem 1,5 GHz a možností až 8 GB RAM. Pro tento projekt bude využita pouze 4GB verze, která by měla být dostačující pro všechny potřebné služby.[29]

Tab. 6 Parametry Raspberry Pi 4 Model B [29]

Vstupní napětí	DC 5 V (přes USB nebo GPIO header)
Procesor	1.5 GHz quad-core ARM Cortex-A72 (64btiový)
Paměť RAM	4 GB (LPDDR4)
Úložiště	Mikro SD karta
Konektivita	2,4GHz a 5GHz IEEE 802.11.b/g/n/ac Wi-Fi Bluetooth 5.0 (BLE) Gigabit Ethernet (1000 Mbit/s) 2 × USB 2.0 konektor 2 × USB 3.0 konektor
GPIO	standardní GPIO header se 40 piny
Pracovní teplota	0 –85 °C

Raspberry Pi 4 Model B nabízí rozsáhlé možnosti konektivity, včetně Wi-Fi a Ethernet připojení, Bluetooth 5.0 a celkem pěti USB portů. Dále je vybaven 40pinovým GPIO konektorem, který umožňuje i připojení rozšiřovacích modulů a periférií.

Pro ukládání operačního systému a dat se používá microSD karta. V rámci operačních systémů je možné využít různé odlehčené linuxové distribuce, pro tento projekt ale bude specificky použit oficiální operační systém Raspberry Pi OS (verze bullseye). Podrobnější specifikace, podstatné pro realizaci projektu, jsou uvedeny v Tabulce 3.

5 Praktická část práce

V rámci této kapitoly se práce nejprve zaměřuje na implementaci vybraného řešení. To zahrnuje popis vybraného hardwaru, což umožní detailně se věnovat fyzickému aspektu projektu, především konfiguraci a instalaci zařízení. Po zprovoznění zařízení následuje fáze implementace softwaru a programování.

5.1 Implementace vybraného řešení

Pro zlepšení přehlednosti při popisu implementace řešení, je tato část kapitoly rozdělena nikoliv na fáze implementace, což je běžně používaný přístup, ale na jednotlivé zařízení, u nichž se bude řešit konkrétní implementace. Ačkoli tento přístup neodpovídá časové posloupnosti, byl přesto zvolen, protože vzhledem k rozdílnosti zařízení v tomto projektu je potenciálně matoucí provádět společné implementace nad všemi zařízeními současně v jedné podkapitole.

Současně podstatnou roli v rámci této kapitoly zastává Raspberry Pi, který s platformou Node-RED, datově propojuje veškerá zařízení. Proto se nabízí, aby tato kapitola vyvrcholila právě implementací Raspberry Pi, během které dojde k syntéze a integraci celého systému.

5.1.1 Vývojová deska ESP8266 Lua Nodemcu v3

V návrhu řešení je předpokládáno, že ESP8266 bude sloužit k měření teploty v místnosti a k odečtům z elektroměru. Ačkoliv je technicky možné obě tyto úlohy realizovat s použitím jediné vývojové desky, pro lepší simulaci reálných podmínek se předpokládá umístění elektroměru mimo měřenou místnost. Z toho důvodu se požijí dvě vývojové desky**,** které budou mít rozdílné zapojení. Co se týče samotného programu, ten se bude již z části shodovat, protože obě vývojové desky budou využívat podobný algoritmus pro připojení na Wi-Fi a odesílání dat z měření.

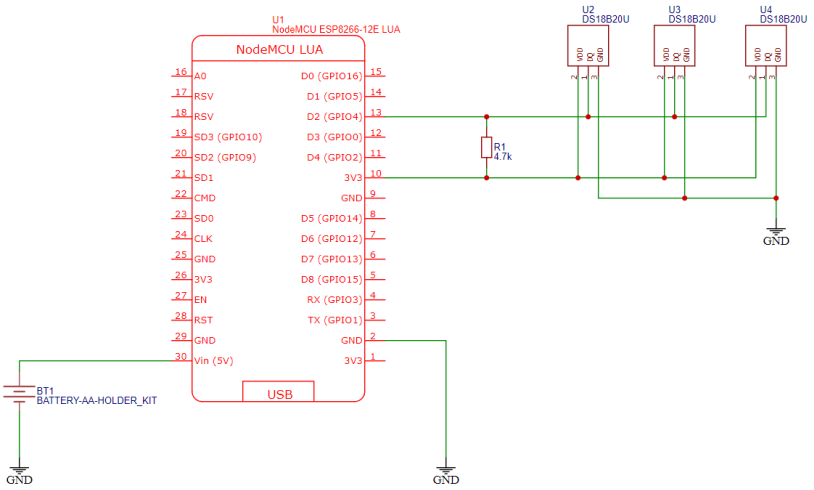
5.1.1.1 Zapojení a konfigurace

Měření teploty v místnosti

Pro měření teploty budou použity digitální teplotní senzory DALLAS – DS18B20, které umožňují komunikaci přes 1-Wire a jejich řazení do série. Volba nasazení více

senzorů byla učiněna především z důvodu jejich dostupnosti za výhodnou cenu, díky čemuž lze získávat spolehlivějších měřících výsledků bez podstatného navyšování nákladů. Fotografie reálného zapojení vývojové desky lze nalézt v Příloha *16*.

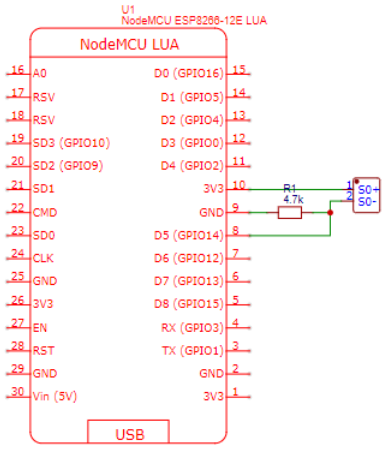
Jak je na Obr. 9, ukázáno, napájení prostřednictvím baterie je také zvažováno. Vývojová deska ESP8266 Lua Nodemcu v3 je vybavena vlastním napěťovým regulátorem, což usnadňuje připojení externího napájení. Nicméně vysílání dat na frekvenci 2,4 GHz prostřednictvím Wi-Fi je energeticky náročnější, a proto se očekává kratší životnost baterie v závislosti na pravidelnosti vysílání. V případě příliš nízké životnosti baterie, lze napájet vývojovou desku přes USB.



Obr. 9 Schéma zapojení teplotních senzorů a baterie k ESP8266

Čtení spotřeby z elektroměru

Čtení spotřeby elektřiny z elektroměrů bude realizováno pomocí S0 impulsů. Vzhledem k tomu, že S0 výstupy z měřičů jsou typicky bezpotenciálové, je nezbytné je externě napájet přes vhodně zvolený externí rezistor, jak je na Obr. 10. Pro účely měření je dále nutné v programu aktivovat funkci pull-up rezistoru pro odpovídající pin. Pull-up rezistor je komponenta, která udržuje úroveň napětí pinu na logické 1 v případě, že není aplikován externí signál. Ideálně by též mělo dojít ke galvanickému oddělení elektroměru a vývojové desky, například použitím optoizolátoru, aby byla zajištěna ochrana a izolace obou systémů.



Obr. 10 Schéma zapojení S0 výstupu z elektroměru do ESP8266

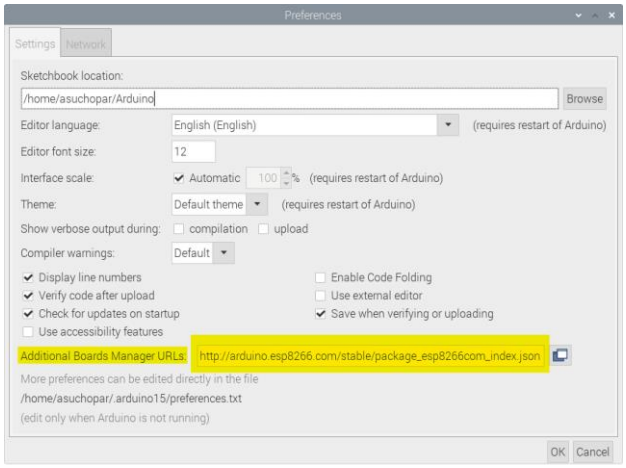
Tato vývojová deska může být připojena k externímu zdroji napájení, podobně jako v předchozím případě. Avšak v kontextu praktického nasazení, kde by se zařízení typicky nacházelo v rozvodné skříni společně s elektroměrem s přístupem k napájení, se předpokládá, že použití baterie jako zdroje napájení nebude nutné.

5.1.1.2 Implementace softwaru

V první řade je nezbytné pro programování vývojové desky ESP8266 nainstalovat vývojové prostředí Arduino IDE. Následně se popíší společné části programu pro obě desky, zahrnující připojení k místní síti Wi-Fi a k serveru MQTT. Poté se rozeberou specifické části algoritmu, které se věnují činnostem určeným pro jednotlivé vývojové desky.

Instalace Arduino IDE

Instalační balíček Arduino IDE je dostupný na oficiálních stránkách Arduino (https://www.arduino.cc/en/software). Po úspěšném dokončení instalačního procesu a spuštění Arduino IDE je možné přistoupit k vývoji programu. Předtím, než je program možné nahrát do vývojové desky, je potřeba zvolit typ cílové desky. ESP8266 není v základním výběru desek v Arduino IDE, proto je vyžadováno přidání knihoven pro tuto desku. V menu pod kartou „Soubory“ ve volbě „Vlastnosti“ je třeba zadat URL pro stažení knihoven pro ESP8266: http://arduino.esp8266.com/stable/package_esp8266com_index.json. Po zadání této URL v „Manažéru desek“, dostupném v sekci „Nástroje“, viz. Obr. 11, lze instalovat knihovnu pro ESP8266.



Obr. 11 Okno vlastností v Arduino IDE – přidání rozšiřujících knihoven

Po přidání knihovny je již možné v sekci „Nástroje“ pod „Desky“ vyhledat vývojovou desku NodeMCU 1.0 (ESP-12E Module), kompatibilní s projektem. Pro nahrání programu stačí použít standardní nastavení desky, je nutné zvolit ještě příslušný USB COM port pro připojení vývojové desky.

Užitečným nástrojem pro ladění je sériový monitor, umístěný v pravém horním rohu uživatelského rozhraní, který umožňuje vyčítat výstupy z funkce *print*() v programu. Pro ESP8266 Lua Nodemcu v3 je charakteristická komunikační rychlost 9600 bps.

Připojení ESP8266 k Wi-Fi a MQTT

Pro realizaci programu pro připojení k místní síti Wi-Fi a k MQTT serveru jsou klíčové dvě knihovny.Knihovna ESP8266WiFi, umožňuje Wi-Fi komunikaci a TCP spojení se zařízeními pro příjem a odesílání dat z místní sítě nebo internetu.

Druhá knihovna PubSubClient, umožnuje tvorbu MQTT klienta, který umožňuje odesílání a přijímání MQTT zpráv ze serveru. Tato knihovna podporuje nejnovější verzi protokolu MQTT 3.1.1 a je konfigurovatelná pro kompatibilitu se staršími verzemi. [30]

```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
```

Následující úryvky kódu prezentované v této části byly modifikovány s cílem zvýšit jejich srozumitelnost a přehlednost prezentace funkcí.

Tato konkrétní část skriptu zajišťuje inicializaci klienta pro vyhledání Wi-Fi sítě a následně se pokouší připojit k této síti s využitím přihlašovacích údajů. Tyto kroky jsou

implementovány ve funkci *setup_wifi(),* která průběžně vypisuje informace o stavu připojení prostřednictvím sériové komunikace.

```
// Prihlasovaci udaje na WiFi
const char* ssid = "Turris";
const char* password = "********";

void setup() {
  setup_wifi();//spusti se pri spusteni ESP
}

// Initializeace espClient (zmen nazev pro vicero ESP)
WiFiClient espClient2;
// Pripojeni ESP k wifi
void setup_wifi() {
  delay(10);
  // Pokus o pripojeni
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.println("."); //ceka se na pripojeni
  }

  Serial.println("");
  Serial.println("WiFi connected - ESP IP address: ");
  Serial.println(WiFi.localIP());
}
```

V případě komunikace s MQTT serverem lze přijímat či odesílat data. Konkrétně v tomto projektu se však do periferií neodesílají žádná data, tím pádem bude řešena převážně problematika odesílání zpráv z ESP do MQTT serveru.

Následující algoritmus po inicializaci proměnných pro připojení k MQTT serveru a nastavení jeho adresy a portu, ve funkci *reconnect()* cyklicky pokračuje v pokusu o připojení klienta, dokud není spojení úspěšně navázáno. Podobně jako v předchozím příkladu jsou informace o stavu spojení vypisovány.

```
// Prihlasovaci udaje na MQTT (NULL jestli nejsou potreba)
const char* MQTT_username = "asuchopar";
const char* MQTT_password = "**********";

const char* mqtt_server = "192.168.*.*"; //ip MQTT serveru
char message = "Zprava k odeslani"; //toto se posle v ramci topicu

void setup() {
  //Nastaveni pripojeni k MQTT serveru a jeho portu
  client.setServer(mqtt_server, 1883);
  //Funkce se zavola vzdy kdyz pridje nova zprava (pro prijem zprav)
  client.setCallback(callback);
}
```

```
// Pokus o pripojeni k MQTT serveru
void reconnect() {
  // Loop dokud se nepripoji
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    // pokus o pripojeni
     if (client.connect("ESP8266_Client", MQTT_username, MQTT_password)) {
      Serial.println("connected");
      // odebirat nebo odeslat topic
      client.publish("topic/send", message);
      //client.subscribe("topic/rcv");prijem topicu se zpravou ze serveru
    } else { //Vypis pri neuspesnem pripojeni
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      delay(5000);
    }
  }
}
```

Při tvorbě vláken zpráv prostřednictvím MQTT se předpokládá, že každé zařízení a zdroj zpráv na něm disponují unikátním „topicem“, na který se mohou po odeslání zprávy na MQTT server dotazovat ostatní klienti.

Měření teploty

V případě vyčítání hodnot z digitálních čidel teploty, se využijí funkce knihoven *OneWire* pro zprovoznění komunikace po sběrnici a knihovny *DallasTemperature* pro čtení dat ze senzorů.

```
#include <OneWire.h>
#include <DallasTemperature.h>
```

Pro přístup k hodnotám je nejprve zapotřebí identifikovat senzor na sběrnici, což lze provést dvěma rozdílnými metodami. První metoda spočívá v definování senzoru podle jeho indexu přiděleného knihovnou *DallasTemperature*. Druhá metoda využívá funkce 1-wire sběrnice a umožňuje identifikaci senzoru podle jeho jedinečné 64-bitové adresy. V tomto projektu bude využita druhá metoda, a proto bude nutné použít následující skript, který vypíše množství zařízení na lince spolu s jejich adresou.

```
// sensor je zapojen na D2
#define ONE_WIRE_BUS 2
// Nastaveni instance oneWire pro komunikaci s jakymkoli zarizenim OneWire
OneWire oneWire(ONE_WIRE_BUS);

// Predani reference oneWire do Dallas Temperature.
DallasTemperature sensors(&oneWire);

// promenna pro drzeni adres zarizeni
DeviceAddress Thermometer;
//promenna pro pocet nalezenych zarizeni
int deviceCount = 0;
```

```
void setup(void)
{
  // Spusteni knihovny
  sensors.begin();
  // vyhledani zarizeni na sbernici
  Serial.println("Locating devices...");
  Serial.print("Found ");
  deviceCount = sensors.getDeviceCount(); //mnozsvi nalezenych zarizeni
  Serial.print(deviceCount, DEC);
  Serial.println(" device.");
  Serial.println("");
  Serial.println("Printing addresses...");
  //vypisovani vsech nalezenych zarizeni na sbernici
  for (int i = 0;  i < deviceCount;  i++)
  {
    Serial.print("Sensor ");
    Serial.print(i+1);
    Serial.print(" : ");
    sensors.getAddress(Thermometer, i);
    printAddress(Thermometer);
  }
}
```

V iteraci v rámci každého nalezeného zařízení se vyvolá funkce *printAddress*(), která vypíše adresu zařízení.

```
//funkce pro cteni a vypsani adres zarizeni
void printAddress(DeviceAddress deviceAddress) //cteni pole bajtu deviceAddress
{
  for (uint8_t i = 0; i < 8; i++) //itarace pro kazdy bajt v adrese
    Serial.print("0x"); // predpona

    //jestlize je hodnota mensi nez 16, prida se nula
    if (deviceAddress[i] < 0x10) Serial.print("0");
    Serial.print(deviceAddress[i], HEX);
    //pridani oddlovace mezi bajty
    if (i < 7) Serial.print(", ");
  }
  Serial.println("");
}
```

Po identifikaci adres zařízení na sběrnici je možné navrhnout algoritmus pro čtení dat z digitálních teplotních senzorů. Pro tyto účely se využívají funkce z již dříve použitých knihoven, a naměřené hodnoty jsou okamžitě přenášeny na MQTT server.

```
// Datovy vodic je pripojen k GPIO15
#define ONE_WIRE_BUS 4
// Nastaveni instance oneWire pro komunikaci se zarizenim OneWire
OneWire oneWire(ONE_WIRE_BUS);
// Predani reference oneWire cidelu teploty Dallas
DallasTemperature sensors(&oneWire);
// Adresy senzoru
DeviceAddress sensor1 = { 0x28, 0x81, 0x72, 0x20, 0xF, 0x0, 0x0, 0x2 };
DeviceAddress sensor2 = { 0x28, 0xE9, 0x59, 0x21, 0xF, 0x0, 0x0, 0x67 };
DeviceAddress sensor3= { 0x28, 0x43, 0x13, 0x21, 0xF, 0x0, 0x0, 0xA0 };
// Promenne pro teploty ze sensoru
float temp1 = 0;
float temp2 = 0;
float temp3 = 0;
```

Nejprve je nutná deklarace proměnných, nastavení adres senzorů, a definice pinu pro sběrnicovou komunikaci. Následně lze implementovat funkci, která se periodicky připojuje k místní Wi-Fi a k MQTT serveru, během jednoho cyklu vyčítá hodnoty senzorů na sběrnici, převádí je na formát vhodný pro odeslání a publikuje je prostřednictvím MQTT.

```
void loop() {

  // Pokud klient neni pripojen, znovu se pripoj
  if (!client.connected()) {
    reconnect();
  }
  // Pokud nebezi klientova smycka, pripoj klienta
  if(!client.loop())
    client.connect("ESP8266Client", MQTT_username, MQTT_password);

    Serial.print("Requesting temperatures...");
    // Zadani prikazu pro ziskani teplot
    sensors.requestTemperatures();
    Serial.println("DONE");

    // Teplota ve stupnich Celsia
    temp1 = sensors.getTempC(sensor1);
    // Prevod hodnoty teploty na retezec stringu pro sensor 1
    char tempString1[8];
    dtostrf(temp1, 1, 2, tempString1);
    Serial.print("Sensor 1(*C): ");
    Serial.println(tempString1);
    // Publikovani teploty sensoru 1 na MQTT tema
    client.publish("room/temp1", tempString1);

      }
```

Předchozí ukázka je omezena na odesílání dat z pouze jednoho senzoru, kompletní skript, pro vyčítání teploty a jeho odesílá nacházející se na modulu ESP8266 je součástí Příloha 1.

Čtení spotřeby z elektroměru

V programu pro čtení spotřeby z elektroměru se využívají pouze knihovny nezbytné pro připojení k Wi-Fi a komunikaci s MQTT serverem. Struktura kódu zůstává podobná té, která byla aplikována v předchozím případě, avšak s klíčovým rozdílem v implementaci funkce přerušení, jež zajišťuje vyšší spolehlivost počítání impulsů vygenerovaných elektroměrem.

```
// Pomocna promena pro casovac
long now = millis();
long lastMeasure = 0;

//promenne pro pocitani S0 pulsu
volatile unsigned long pulseCount = 0;
unsigned long lastPulseTime = 0;
const int pulsePin = D5;
void setup() {
  // Nastaví pin jako vstup s pull-up rezistorem
  pinMode(pulsePin, INPUT_PULLUP);
  // Nastaví preruseni na sestupnou hranu signalu
  attachInterrupt(digitalPinToInterrupt(pulsePin), handlePulse, FALLING);
}
```

Stejně jako v předchozím případě, i zde se nejprve deklarují proměnné, avšak při prvním spuštění se také definuje režim pull-up rezistoru pro vstupní pin, jak bylo řečeno v kapitole 5.1.1.1. Následně se nastaví přerušení programu a spustí funkce *handlePulse*(), která se vykoná, když signál na pinu přechází z log.1 do log.0.

Během přerušení je zásadní, aby se funkce vykonala co nejrychleji a co možná nejméně narušila běžný průběh programu. K tomu slouží atribut *IRAM_ATTR*, který zajišťuje umístění dané funkce do instrukční RAM (IRAM) místo do běžné flash paměti, což umožňuje rychlejší přístup k této funkci.

```
// Zpracovani pulsu
void IRAM_ATTR handlePulse() {
  unsigned long currentPulseTime = millis();
  // Debounce - ignoruje pulzy, které prichazi rychle za sebou
  if (currentPulseTime - lastPulseTime > 10) { // 10 ms debounce
    pulseCount++;
    lastPulseTime = currentPulseTime;
  }
}
```

Následující skript pak pravidelně každých 10 sekund čte počet pulsů a zasílá tyto informace na MQTT broker pomocí tématu „outside/pulses“. V případě, že mikrokontroler ztratí připojení k MQTT brokerovi, pokusí se znovu připojit. Při zasílání dat je přerušení dočasně zastaveno, aby byla zajištěna integrita sdílených dat během jejich čtení. Stejně jako v předchozím případě, kompletní skript lze dohledat v Příloha 2 této práce.

```
void loop() {

  if (!client.connected()) {
    reconnect();
  }
  if(!client.loop())
    client.connect("ESP8266Client2", MQTT_username, MQTT_password);
```

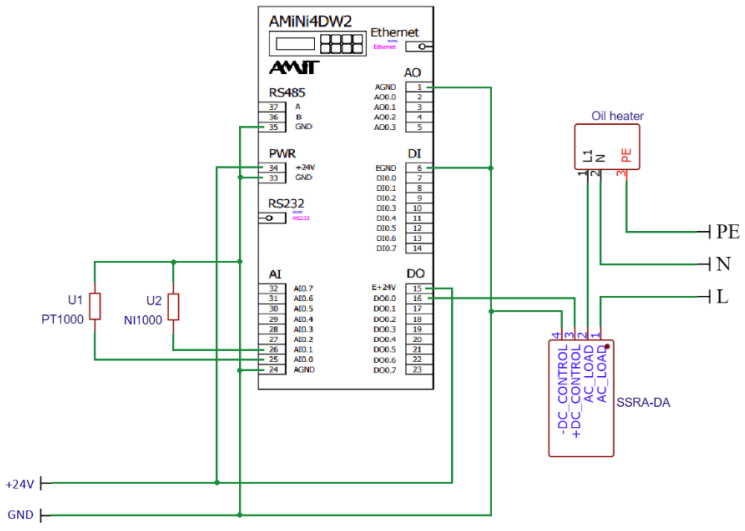
```
    static unsigned long lastPrintTime = 0;
  if (millis() - lastPrintTime > 10000) { // Kazdych 10 sekund vypise pocet pulsu
    noInterrupts(); // Zastavi preruseni pro pristup ke sdilene promenne
    Serial.print("Number of pulses: ");
    Serial.println(pulseCount);
    interrupts(); // Povoli preruseni
    char pulseString[8];
    dtostrf(pulseCount, 1, 0, pulseString);
    client.publish("outside/pulses", pulseString);
    lastPrintTime = millis();
  }
}
```

5.1.2 PLC Amit AMiNi4DW2

Hlavní funkcí PLC v projektu je řízení olejového topení v místnosti a dodatečné měření teploty pro porovnání s hodnotami z ESP8266 a meteorologickými hodnotami. Je důležité zdůraznit, že toto PLC je v projektu bráno jako již existující řešení, které se rozšiřuje o novou regulaci na bázi Node-RED. Proto bude jeho úloha diskutována spíše okrajově. Popis jeho konfigurace a způsobu regulace bude uveden, avšak hlavní důraz bude kladen na problematiku integrace s platformou Node-RED.

5.1.2.1 Zapojení a konfigurace

V následujícím Obr. 12 je zobrazeno schéma zapojení PLC s jeho vstupy a výstupy. Napájení PLC je zajištěno 24 VDC zdrojem. Zemní svorky AGND, EGND a GND jsou v rámci PLC propojeny, a doporučuje se jejich připojení k svorce PE, což je svorka ochranného uzemnění na rozváděči, za účelem zajištění správného uzemnění a ochrany zařízení. V Příloha *15* lze nalézt fotografii PLC Amit AMiNi4DW2 v provozu s fyzickým zapojením vycházejícího z následujícího obrázku.



Obr. 12 Schéma zapojení PLC Amit AMiNi4DW2

Měření vnitřní a venkovní teploty je zajištěno dvěma průmyslovými odporovými senzory teploty Pt1000 a Ni1000, které jsou připojeny ke svorce 25 a 26 analogového vstupu. Pro měření je však nutné na PLC přepnout svorky analogového vstupu do režimu RTD. Následně ve vývojovým prostředí DetStudio je možné využít funkční blok pro přepočet naměřeného odporu senzoru na stupně celsia.

Na svorce 16 pro digitální výstup je připojeno SSR (Solid State Relay), které je využito pro ovládání napájení olejového radiátoru. Vzhledem k tomu, že PLC ovládá radiátor pomocí PWM pulsů, bylo dáno přednost SSR před klasickým relé. Je nutné poznamenat, že aby bylo možné aktivovat SSR, je nezbytné zajistit dodání externího napájecího napětí, z toho důvodu ke svorce 15 je směrováno příslušné napájení.

5.1.2.2 Implementace softwaru

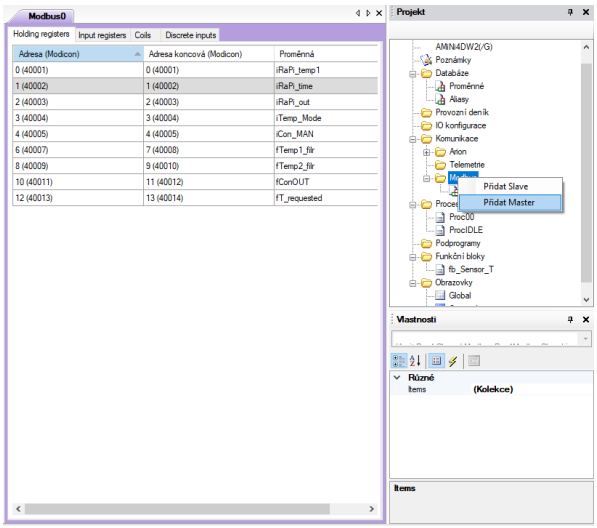
V rámci vývojového prostředí DetStudio byla vytvořena logika regulace prostřednictvím PID regulátoru, který určuje akční zásah na základě dat z připojených teplotních senzorů. Tento akční zásah je následně převeden na PWM signál na digitálním výstupu, jenž ovládá SSR pro regulaci olejového topení, čímž je PLC zařízení schopné samostatné funkce.

Toto řešení však nenaplňuje primární cíle této diplomové práce, proto je regulace v rámci PLC doplněna o výstupy z regulace realizované na platformě Node-RED, a činnost PLC se redukuje na reagování na příkazy z Node-RED, který bude disponovat všemi nezbytnými daty pro regulaci z vlastních zdrojů. Data z měření získaná pomocí PLC pak lze využít pro následné porovnání s měřenými daty z meteorologické předpovědi a z modulu ESP. Současně je možné program PLC rozšířit o časový čítač, jenž v případě dlouhodobé absence komunikace automaticky přepne systém zpět na původní regulaci zajišťovanou PLC.

Komunikace PLC s platformou Node-RED

Jak je uvedeno v návrhu projektu, komunikace mezi PLC a Node-RED bude realizována prostřednictvím protokolu ModBus. Aby se optimalizoval výpočetní výkon Raspberry Pi, na kterém Node-RED běží, ModBus master bude implementován na straně PLC. V prostředí DetStudio je tvorba ModBus serveru relativně jednoduchá. V navigaci

projektu, ve složce komunikace, lze pravým tlačítkem myši kliknout na podsložku Modbus a přidat Modbus Master, jak je demonstrováno na Obr. 13. Po jeho přidání lze přidávat, prostřednictvím tabulky, proměnné společně s adresami registrů, ke kterým pak lze přistupovat prostřednictvím ModBus Slave.



Obr. 13 Vývojové prostředí DetStudio – nasazení ModBus mastera

Hlídání komunikace

Kontrola komunikace se provádí prostřednictvím jednoduché logiky, kdy se periodicky porovnávají sekundy z aktuálního času z Node-RED. Pokud hodnoty zůstávají stejné po dobu delší než 5 minut, dojde ke změně stavu dané proměnné. V DetStudiu proměnné disponují specifickou vlastností, umožňující v rámci jedné proměnné, např datového typu integer, definovat 16 bitů pomocí tečky následované číslem bitu, což je využito v následujícím úryvku kódu.

```
// Kontrola komunikace
SyncMark 1, 30, 0, 0, 0, xSync_mark.0, NONE //Generuje každých 30 s puls
if xSync_mark.0
	//Porovnání nové a staré hodnoty
	If (iRaPi_time == iAmini_time)
		Let iError.0 = true //Zapnout TimerOn
		Let iError.2 = false // RS - reset = false
	else
		Let iError.0 = false //Vypnout TimerOn
		Let iError.1 = false //RS - set = false
		Let iError.2 = true //RS - reset = true
	endif
	Let iAmini_time = iRaPi_time //ukládá novou hodnotu z Node-RED
EndIf
	TimerOn iError.0, 300000, iError.1, NONE //Po 5 min vyhlásí poruchu
	RS iError.1, iError.2, iError_Comm.0 // iError_Comm.0 = porucha
```

Přepínaní mezi režimy regulace

Z Node-RED by mělo být možné řídit režim regulace. Je běžným standardem poskytovat kromě automatického ovládání také možnost manuálního zásahu nebo úplné deaktivace systému. Rovněž je žádoucí umožnit přepínání mezi regulací zajišťovanou PLC a Node-RED. Tato variabilita ovládání je řešena v následujícím úseku kódu s využitím funkce Switch. Výstup z této funkce je poté směrován do funkce PWM, která převádí hodnotu akčního zásahu na pulzní šířkovou modulaci pro digitální výstup DO_0.

```
// OFF/MAN/AUT/Node-RED řízení
Switch iTemp_Mode

	Case 0 //Vypnout
		Let fConOUT = 0
	EndCase

	Case 1 //MAN režim
		Limiter iCon_MAN, 0.000, 100.000
		Let fConOUT = iCon_MAN
	EndCase

	Case 2 //Regulace PLC - PID regulátor
		Limiter fCon_AUT, 0.000, 100.000
		Let fConOUT = fCon_AUT
	EndCase

	Case 3 //Regulace z Node-RED
		If iError_Comm.0
			Let fConOUT = fCon_AUT
		else
			Limiter iRaPi_temp1, 0.000, 100.000
			Let fConOUT = iRaPi_temp1
		Endif
	EndCase
EndSwitch
```

5.1.3 Mikropočítač Raspberry Pi 4 Model B

Stěžejní část projektu je realizovaná právě na zařízení Raspberry Pi 4 Model B. Toto zařízení nebude sloužit pouze k datovému propojení všech komponent, ale zajistí také služby pro ukládání a vizualizaci dat, a bude středobodem řízení a logiky celého regulačního systému.

Na rozdíl od předešlých implementací, u Raspberry Pi se nevyskytuje problém s fyzickým zapojením, stačí zajistit jeho napájení a připojení k místní Wi-Fi síti. Místo toho se však bude řešit instalace potřebných služeb nezbytných pro realizaci projektu. Po stručném přehledu instalací služeb, se kapitola zaměří na realizaci hlavních funkcí a cílů diplomové práce.

5.1.3.1 Instalace služeb

Jak už bylo uvedeno v návrhu řešení, Raspberry Pi bude provozováno na operačním systému Raspberry Pi OS ve verzi Bullseye, která je kompatibilní se všemi službami vyžadovanými pro tento projekt.

Před instalací jakéhokoli softwaru na hostitelský systém je doporučeno aktualizovat všechny dostupné balíčky na Raspberry Pi a případně aktualizovat i operační systém. Tento proces zajistí, že zařízení disponuje nejaktuálnějšími bezpečnostními opravami a stabilními verzemi softwaru, čímž se minimalizuje riziko potenciálních problémů během instalace nebo při užívání softwarů.

Pro aktualizaci seznamu dostupných balíčků a jejich instalaci lze spustit v linuxovém terminálu následující příkaz, který po získání oprávnění superuživatele (root) provede požadované operace:

```
sudo su -
apt-get update
apt-get upgrade
```

Následně po restartu zařízení lze pokračovat v instalaci jakéhokoli požadovaného softwaru.

Instalace a konfigurace platformy Node-RED

Jak je uvedeno na domovských stránkách Node-RED, k instalaci softwaru postačí pouhé spuštění následujícího skriptu v konzoli, který stáhne skript pro instalaci Node.js a Node-RED [31]:

```
bash <(curl -sL https://raw.githubusercontent.com/node-red/linux-installers/master/deb/update-nodejs-and-nodered)
```

Pro optimalizaci softwaru pro limitovanou paměť Raspberry Pi, je vhodné spustit následující script, který zajistí dřívější uvolnění alokované paměti pro procesy Node.js[31]:

```
node-red-pi --max-old-space-size=256
```

Též z hlediska spolehlivosti, při výpadku napájení, je žádané, aby se Node-RED automaticky spouštěl při startu Raspberry Pi. To zajistí následující skript:

```
systemctl enable nodered.service
```

Protože k webovému rozhraní Node-RED je možné přistupovat bez omezení z místní sítě, pro zvýšení bezpečnosti je nezbytné přijmout několik opatření. Prvním krokem je změna výchozího portu, aby se přístup k Node-RED realizoval jiným portem, než je standardní. Dále je důležité aktivovat autentifikaci, takže při pokusu o přístup k webovému rozhraní bude vyžadováno heslo. Vzhledem k tomu, že přes síť budou přenášena citlivá data, jako jsou hesla, je nutné přechod z nezabezpečeného protokolu HTTP na zabezpečený protokol HTTPS.

Veškeré tyto konfigurace Node-RED se provádí ve složce *~/.node-red*, v souboru *settings.js*, kde je třeba upravit následující konfigurační sekce:

```
uiPort: 1889,
adminAuth: {
    type: "credentials",
    users: [{
        username: "admin",
        password: "$$08$xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx",
        permissions: "*"
    }]
},
https: {
    key: require("fs").readFileSync('cesta/k/privkey.pem'),
    cert: require("fs").readFileSync('cesta/k/cert.pem')
```

V parametru *uiPort* lze vybrat port pro webovou vizualizaci. *Username* a *password* v parametru *adminAuth* slouží pro nastavení uživatelského jména a hesla, které se vkládá v hash formátu, pro autentizaci. V parametru *https* je třeba zadat cestu k privátnímu klíči a certifikátu, který lze vygenerovat například pomocí OpenSSL nebo získat od certifikační autority.

Instalace a konfigurace databázové služby InfluxDB

Při lokální instalaci InfluxDB nelze využít stejnou metodu jako v případě Node-RED, neboť nebyl dosud vytvořen instalační skript, a proto je nutné provést kompletně instalaci ručně. Prvním krokem je stáhnout GPG klíče pro ověření repositáře InfluxDB a ověřit jejich hash s očekávanou hodnotou. Pokud je vše v pořádku, převede se GPG klíč do binárního formátu a uloží do systému správy balíčků APT jako důvěryhodný klíč. To umožňuje systému ověřit pravost balíčků při jejich instalaci z InfluxData repositáře. Nakonec je

možné aktualizovat seznam balíčků a jejich verze tak, aby zahrnoval i InfluxDB repositář, a následně instalovat InfluxDB.

```
wget -q https://repos.influxdata.com/influxdata-archive_compat.key

echo'393e8779c89ac8d958f81f942f9ad7fb82a25e133faddaf92e15b16e6ac9ce4c influxdata-
archive_compat.key' | sha256sum -c
&&
cat influxdata-archive_compat.key | gpg --dearmor | tee
/etc/apt/trusted.gpg.d/influxdata-archive_compat.gpg > /dev/null

echo 'deb [signed-by=/etc/apt/trusted.gpg.d/influxdata-archive_compat.gpg]
https://repos.influxdata.com/debian stable main' | tee
/etc/apt/sources.list.d/influxdata.list

apt-get update && apt-get install influxdb2
```

Pro automatické spuštění InfluxDB při startu Raspberry Pi se použije stejný příkaz jako v případě Node-RED:

```
systemctl enable influxdb
```

Na rozdíl od Node-RED, webové rozhraní InfluxDB již z výchozího nastavení vyžaduje přihlášení, proto je nutné zajistit pouze změnu portu a přechod na zabezpečený protokol HTTPS. Stejně jako v případě Node-RED, je možné stáhnout TLS certifikáty a následně upravit konfigurační soubor config.toml v adresáři /etc/influxdb2:

```
[http]
  enabled = true
  bind-address = ":8086"
  tls-cert = "/etc/ssl/influxdb-selfsigned.crt"
  tls-key = "/etc/ssl/influxdb-selfsigned.key"
```

Kde parametr *bind-address* určuje port a parametry *tls-cert* a *tls-key* vyžadují cestu k certifikátu a k privátnímu klíči. Po restartování systému by se veškeré změny měly projevit.

Instalace a konfigurace MQTT serveru Mosquitto

Software Mosquitto je aplikace, která je dostupná ve výchozích repositářích Raspberry Pi, takže pro jeho instalaci stačí zadat následující příkaz, který zajistí kompletní instalaci:

```
apt-get install -y mosquitto
```

Služba Mosquitto je po instalaci automaticky nastavena na spouštění při startu systému. Aby byla dokončena celková konfigurace, je vhodné zajistit zabezpečení přístupu

k MQTT brokeru. Proto je potřeba zadat do konzole následující příkaz, který vytvoří pro Mosquitto soubor s uživatelskými jmény a hesly:

```
mosquitto_passwd -c /etc/mosquitto/passwd mqttuser
```

Následně je možné v konfiguračním souboru */etc/mosquitto/mosquitto.conf* vypnout anonymní přístup a specifikovat cestu k souboru s uživatelskými jmény a hesly:

```
allow_anonymous false
password_file /etc/mosquitto/passwd
```

Po restartu služby by se měly projevit veškeré změny.

Instalace a konfigurace platformy Home Assistant

Z důvodu nedostatečné výpočetní kapacity se po instalaci a provozování Home Assistant začaly objevovat nečekané záseky systému, které trvaly až několik desítek minut; řešením byl restart Raspberry Pi. Ačkoliv Raspberry Pi 4 Model B splňuje minimální požadavky pro provoz Home Assistant i s rezervou, v kombinaci s dalšími službami se systém pravděpodobně dostal na hranici svých možností. Bohužel, z dlouhodobého hlediska, jsou tyto stavy nepřijatelné, a proto se Home Assistant nebude implementovat do praktické části. Jako alternativní řešení bude využit Node-RED dashboard, který nebude mít tak významný dopad na výkon zařízení. Zároveň však nebude možné dosáhnout stejně rozsáhlých možností v oblasti vizualizace dat a řízení regulace, přičemž se zásadně zkomplikuje i zavedení hlasového asistenta, jak je uvedeno ve funkčních požadavkách na regulační systém.

5.1.4 Implementace softwaru

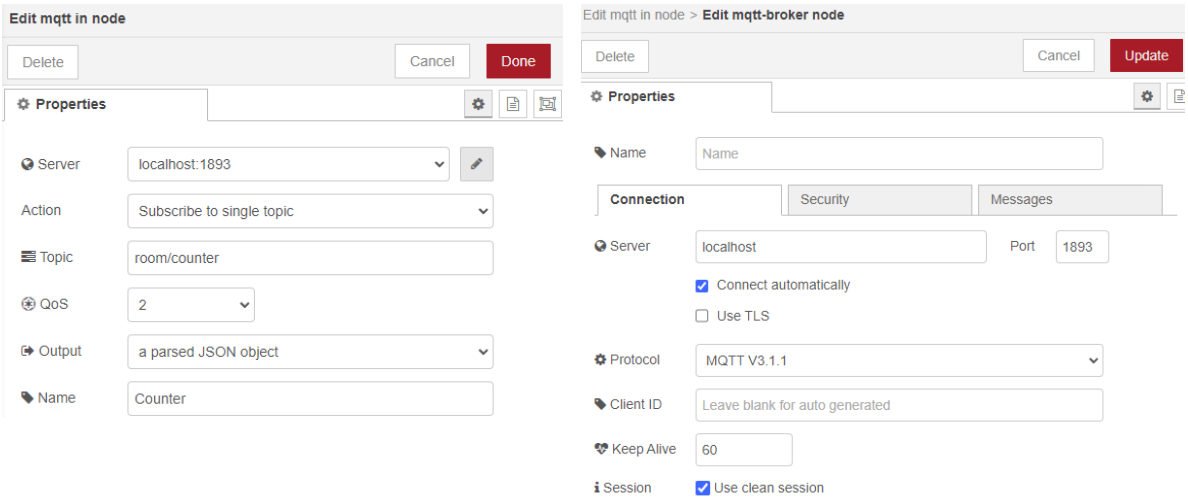
V této kapitole se bude práce zaobírat, až na drobné výjimky, výhradně tvorbou programu v prostředí Node-RED. Tato část bude představovat stěžejní část implementace řešení, neboť se nebude zabývat pouze logikou regulace a vizualizací dat, ale také bude řešeno propojení všech dosud zmíněných zařízení do jednoho funkčního celku. Pro každou z funkcionalit regulace, které byly definovány v cílech práce, bude vyčleněna podkapitola, ve které se bude zkoumat postup řešení a možné problémy při jejich realizaci podrobně.

K vývojovému prostředí Node-RED se přistupuje přes webové rozhraní, jehož adresa závisí na IP adrese zařízení, kde Node-RED běží, a na čísle portu umístěném za dvojtečkou. Standardně je nastaven port 1880. V rámci této práce se k Node-RED přistupuje na adrese 192.168.0.33:1885, což odpovídá adrese Raspberry Pi s Node-RED. Změna čísla portu byla provedena během instalace Node-RED. V případě aktivované autentifikace je nutné zadat přihlašovací údaje, aby bylo možné vstoupit do vývojového prostředí a začít s programováním.

5.1.4.1 Implementace modulů ESP8266 do Node-RED

V předešlé kapitole 5.1.1.2 byl popsán způsob připojení pomocí Wi-Fi a MQTT protokolu k MQTT brokeru, jehož instalace byla v předešlé kapitole 5.1.3.1. V této fázi se odesílají zprávy od senzorů do MQTT brokeru. Následně je tedy potřeba tyto zprávy z MQTT brokeru odebírat v Node-RED a zajistit spolehlivost komunikace.

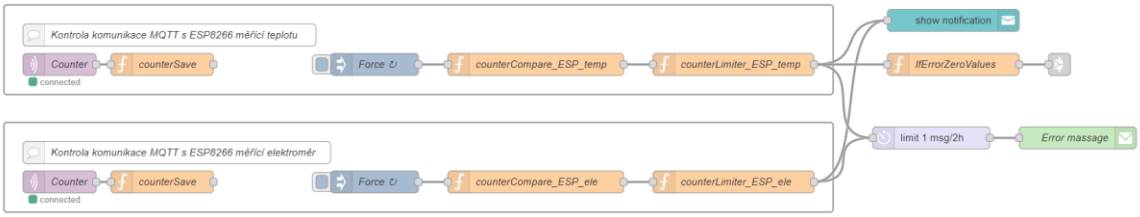
K odběru zpráv z MQTT serveru není nutná žádná dodatečná knihovna, jelikož Node-RED již podporuje práci s MQTT protokoly. Uzly nezbytné pro navázání spojení s MQTT brokerem jsou dostupné v sekci *network* v levé paletě uzlů. Hlavně se bude využívat uzel *MQTT in*, který umožňuje příjem zpráv z MQTT brokera. Po přetažení tohoto uzlu do pracovního prostoru a otevření jeho editoru lze upravit nastavení komunikace. Kliknutím na ikonu tužky, jak je zobrazeno na Obr. 14 , se otevře dialog pro nastavení parametrů pro MQTT brokera, který lze využít napříč všemi uzli pro komunikaci s MQTT brokerem.



Obr. 14 Nastavení uzlu MQTT in pro navázaní komunikace se MQTT brokerem

Do pole *Topic* se vepíše specifický MQTT topic, který definuje zprávy z modulů ESP8266, v tomto případě konkrétně z modulu určeného pro měření teploty s topicem: *„room/counter“*. Tento topic pravidelně odesílá čísla v rozmezí od 0 do 150, což slouží k monitorování spojení se zařízením. Jelikož po odeslání zprávy zůstává informace persistentní v MQTT brokeru až do přepisu novou zprávou, proto je nutné pravidelně měnit obsah zprávy pro ověření stavu komunikace.

V Node-RED se tudíž implementuje funkce, která uchovává obsah přijatých zpráv a porovnává nově příchozí zprávy s těmi uloženými. Pokud se zprávy shodují příliš dlouho, systém vygeneruje chybu, na kterou reaguje odesláním emailového upozornění na chybu, přepnutím řízení na PLC a nastavením hodnoty na 0 °C, aby bylo možné odlišit nesprávnou hodnotu při výpadku od validních dat v budoucí databázi.

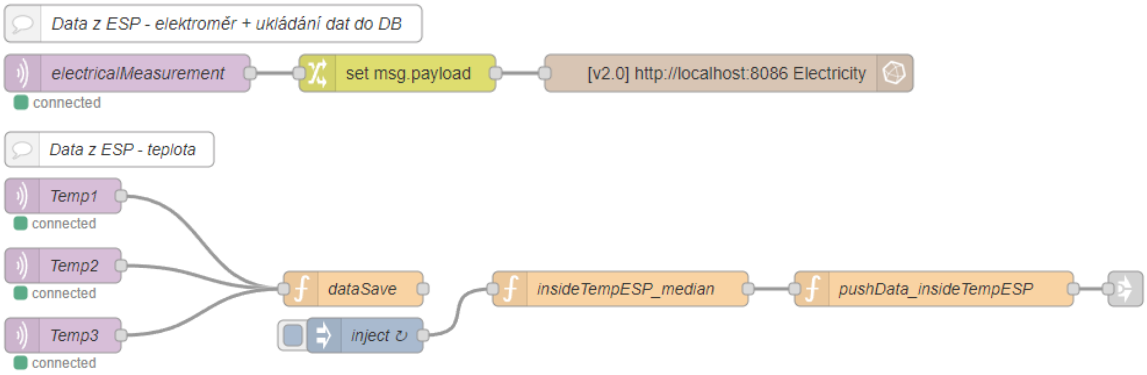


Obr. 15 Node-RED flow pro hlídání komunikace a odeslání chybového hlášení

Na Obr. 15 je zřejmé, že tok dat je přerušen za funkčním uzlem *counterSave*, což se na první pohled jeví jako rozpor s konceptem Node-RED. To je způsobeno využitím funkcí *flow.set()* a *flow.get()*. Ve funkčních uzlech je možné pomocí JavaScriptu definovat vlastní logiku zpracování, obvykle však neuchovávají data mezi jednotlivými zpracováními a spouštějí svůj algoritmus pouze při přijetí zprávy (payloadu). Nicméně, výše zmíněné funkce umožňují persistenci dat mezi zpracováními, přičemž tato data jsou dostupná v rámci celého projektu. Díky tomu je možné k nim přistupovat z libovolného uzlu, a to i bez přímého datového spojení.

Kromě funkčních bloků, jejichž obsah lze nalézt v Příloha 2 a Příloha 4 diplomové práce, na Obr. 15 je možné si též všimnout uzlu *Error message*, který odesílá emaily o výpadku komunikace. Uvedený uzel není standardně součástí Node-RED, pro jeho použití je třeba nainstalovat knihovnu *node-red-node-email*. Tuto knihovnu lze přidat prostřednictvím položky *Manage palette* v menu, které se nachází v pravém horním rohu

webového rozhraní. V rámci konfigurace tohoto uzlu se zadají přihlašovací údaje k emailovému účtu a parametry serveru, z něhož budou zprávy odesílány.



Obr. 16 Node-RED flow pro příjem dat z modulů ESP8266

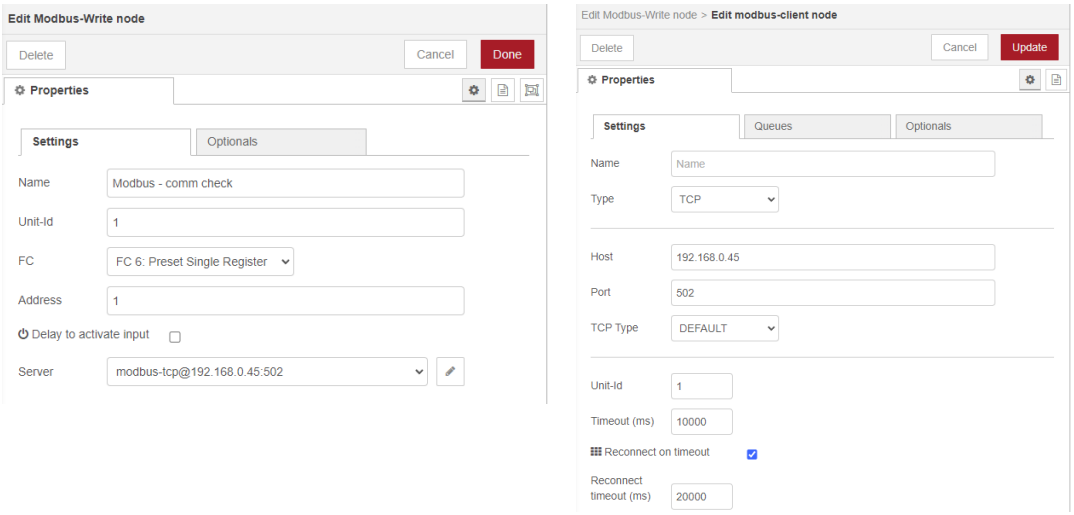
K příjmu dat z měření teploty a elektrické spotřeby, viz. Obr. 16, se použila stejná metoda jako v případě kontrolního čítače, jen se pozměnil topic. Při měření teploty, kdy jsou přijímaný tři hodnoty teploty, se pro určení reprezentativní hodnoty používá výpočet mediánu, viz. Příloha 6.

5.1.4.2 Implementace PLC do Node-RED

Pro komunikaci s PLC byl z kapitoly 5.1.2.2 již připraven ModBus TCP master. V následující fázi je proto potřeba vytvořit slave, který bude data odebírat od mastera. Uzly pro komunikaci pomocí protokolu Modbus TCP standardně nejsou součástí Node-RED, a proto je nutné doinstalovat knihovnu *node-red-contrib-modbus*.

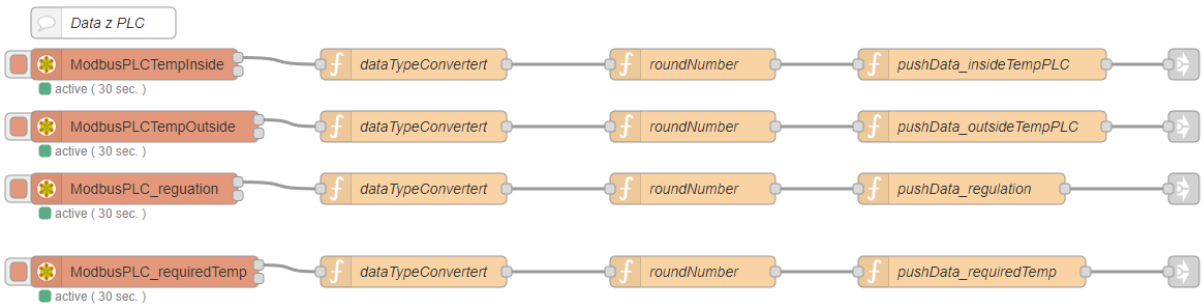
Tato knihovna rozšíří paletu uzlů o novou složku s rozsáhlým množstvím uzlů. V rámci tohoto projektu se však budou využívat primárně uzly *Modbus-Write* pro zápis do registrů, *Modbus-Read* pro čtení z registrů a *Modbus-Response* pro kontrolu komunikace.

Podobně jako u MQTT serveru, je třeba nastavit výchozí parametry ModBus TCP mastera, které se poté využijí ve všech uzlech pro ModBus komunikaci. Změny jsou následně prováděny pouze v adresách registrů, s nimiž se operuje. Na Obr. 17 je znázorněna konfigurace ModBus v projektu, kde v nastavení uzlu pro čtení se do pole *Address* zadá číslo registru definované v PLC, jak je uvedeno v tabulce registrů na Obr. 13. Je důležité upozornit, že někteří výrobci mohou používat v registrech určitý offset nebo specifickou paritu bitů, které je nutné zohlednit pro úspěšné navázání komunikace.



Obr. 17 Nastavení uzlu Modbus-Write a připojení k Modbus TCP Master

Při čtení a zápisu do registrů je klíčové vzít v úvahu datový typ hodnoty. Například, datový typ integer zaujímá 16 bitů, což odpovídá jednomu ModBus registru, zatímco long má rozsah 32 bitů, pro jehož zápis je potřeba využít dva registry. Vzhledem k tomu, že z PLC se vyčítá hodnota flow, která má rozsah 32 bitů. Proto je nutné udělat jistá opatření, kdy je zapotřebí spojit dva registry do jednoho, změnit paritu, a pro sjednocení hodnot zaokrouhlit na 2 desetinná čísla, jak je zobrazeno na datových tocích na Obr. 18. Současně v uzlu pro vyčítání je nutné nastavit v poli *Quantity*, že se vyčítá ze dvou registrů. Algoritmus funkčních uzlů lze nalézt v Příloha 7, Příloha 8 a Příloha 9.



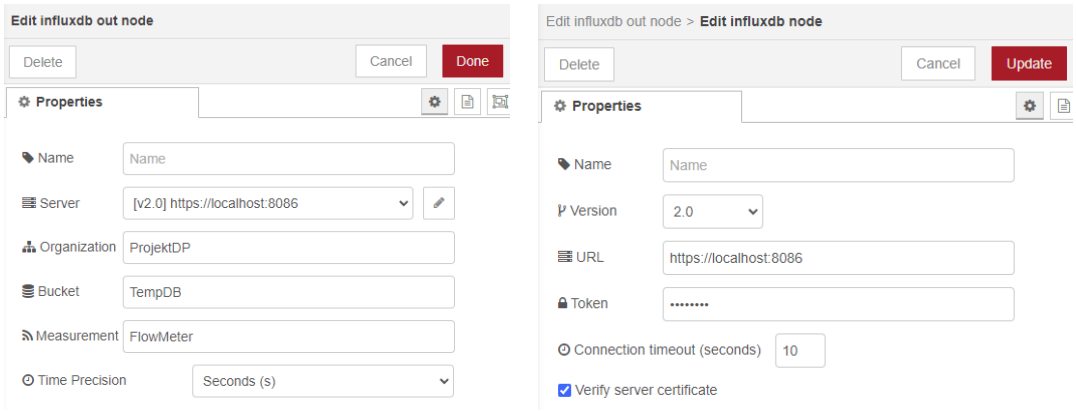
Obr. 18 Node-RED flow pro vyčítání dat z PLC

Podobně jako u komunikace MQTT se i při ModBus komunikaci dbá na spolehlivost přenosu dat. V tomto případě však není třeba využívat dříve vytvořenou logiku čítače. Místo toho je možné využít funkcionalitu uzlu *Modbus-Response*, která umožňuje detekci a oznámení chyby v případě ztráty spojení s PLC. Současně je možné automaticky odeslat informační e-mail a resetovat hodnoty na výchozí stav.

5.1.4.3 Implementace InfluxDB do Node-RED

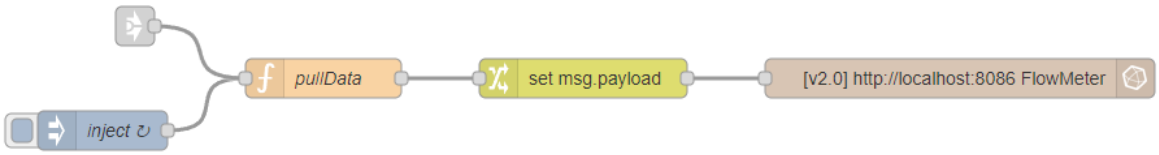
Po instalaci InfluxDB, která byla popsána v kapitole 5.1.3.1, je možné začít data zapisovat do databáze. Jako první krok je potřeba se registrovat a vytvořit novou organizaci prostřednictvím webového rozhraní InfluxDB. Přístup k tomuto rozhraní je podobný jako u Node-RED, avšak s rozdílným portem. Pro účely tohoto projektu se používá webová adresa: 192.168.0.33:8086. Dále je v sekci *Load Data* nezbytné přidat tzv. *Bucket*, což je skupina, do níž se budou odesílat data a kde se budou vytvářet tabulky dat.

Poté je třeba Node-RED rozšířit o knihovnu *node-red-contrib-influxdb*, která umožňuje komunikaci s InfluxDB. Tato knihovna do sekce *storage* v paletě uzlů přidává tři nové uzly. Pro potřeby projektu postačí uzly *InfluxDB in* a *InfluxDB out*, které slouží k zápisu a čtení z databáze. Podobně jako v předešlých případech, je nutné provést základní nastavení pro komunikaci se serverem, což je zobrazeno na Obr. 19.



Obr. 19 Nastavení uzlu influxdb out a konfigurace připojení k serveru

Při konfiguraci serveru je potřeba znát nejen jeho URL, ale také přístupový token, který lze vygenerovat ve webovém rozhraní InfluxDB, a to v sekci *API tokens* pod *Load Data*. Poté je v libovolném influxdb uzlu potřeba uvést název organizace, která byla stanovena během registrace, dále *Bucket* a nakonec *Measurement*, což představuje konkrétní tabulku s daty.



Obr. 20 Node-RED flow pro zápis dat do databáze InfluxDB

Zápis dat je znázorněn na datovém toku na Obr. 20, kdy funkční uzel *pullData, viz.* Příloha 10*,* vyčítá obsah zpráv dat v projektu, která se následně v uzlu *change* řadí do vhodného formátu pro ukládání do databáze. K jednotlivým částem zprávy jsou přiřazeny názvy, pod kterými se hodnoty ukládají do databáze, to vypadá následovně:

```
{
    "Inside temperature (ESP)" : msg.payload.insideTempESP,
    "Inside temperature (PLC)" : msg.payload.insideTempPLC,
    "Ouside Temperature (Forecast)": msg.payload.ousideTempForecast,
    "Ouside Temperature (PLC)": msg.payload.outsideTempPLC,
    "Control action": msg.payload.regulation,
    "Required temperature" : msg.payload.requiredTemp
}
```

Čtení dat z databáze, lze ilustrovat v části projektu, kdy je zapotřebí uložit data z databáze do souboru csv, za účelem učení ML modelu. V tomto případě na Obr. 21, je patrný funkční uzel *query,* v kterém se nachází skript s dotazem pro následující uzel InfluxDB na data v databázi.

Obr. 21 Node-RED flow export dat z databáze InfluxDB do souboru csv

Dotaz je formulován v jazyce InfluxQL, který je dotazovacím jazykem specificky navrženým pro InfluxDB a je v mnoha ohledech podobný SQL. Oproti SQL je navržen pro práci s časovými řadami, díky čemuž je možné se snadno dotázat na data v konkrétních časech, jak je demonstrováno v následujícím skriptu v uzlu *template*. Ten mění obsah payloadu na databázový dotaz, konkrétně na vypsání všech dat z tabulky *FlowMeter* v bucketu *TempDB* dne 12. ledna 2024 od 12:00 do 16:00 hodin. Funkční uzel *convertToCSV,* viz. Příloha 11, převede pak hodnoty do formátu vhodného pro čtení dat v tabulce.

```
from(bucket: "TempDB")
  |> range(start: {{2024-01-12T12:00:00Z}}, stop: {{2024-01-12T16:00:00Z}})
  |> filter(fn: (r) => r["_measurement"] == "FlowMeter")
  |> filter(fn: (r) => r["_field"] == "{{_field}}")
        |> yield(name: "mean")
```

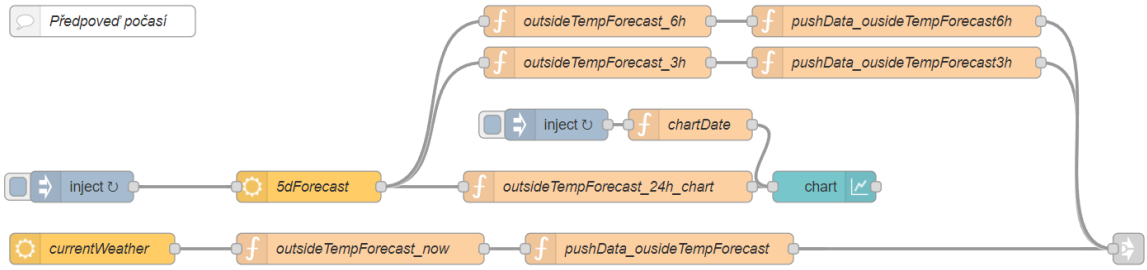
5.1.4.4 Implementace predikce teploty

Predikce venkovní teploty

Pro predikci venkovní teploty se využívají funkce knihovny *node-red-node-openweathermap*, která nabízí možnost odebírání meteorologických dat včetně

předpovědí teplot z portálu OpenWeather.org. Tento web, v rámci svého základního bezplatného tarifu, umožňuje aktualizovat informace o počasí až do 1 000 000 dotazů měsíčně, což pro potřeby tohoto projektu bohatě stačí.

Pro realizaci odběru meteorologických dat je nutné projít na webu OpenWeather.org procesem registrace a následně v sekci *My API keys* vygenerovat klíč, který se zadá do nově instalovaného uzlu. V rámci této knihovny byly přidány dva nové uzly, které se liší pouze ve způsobu aktivace: první vyžaduje vstupní signál pro odeslání dat, zatímco druhý automaticky posílá data při každé změně v předpovědi počasí.



Obr. 22 Node-RED flow pro odběr aktuální a budoucí teploty

Datový tok na Obr. 22 demonstruje využití obou typů uzlů. Kdy uzel *currentWeather* je konfigurován na odběr aktuální meteorologické situace, z které se využívá údaj o aktuální teplotě. Uzel *5dForecast* odebírá meteorologickou předpověď pro následujících 5 dní. Každá zpráva z těchto dvou uzlů je formovaná v jednom JavaScript objektu, z kterého je potřeba získat potřebná data. K tomu slouží funkční uzly nacházející se hned za blokem *openweathermap,* viz. Příloha 12.

Predikce vnitřní teploty v místnosti

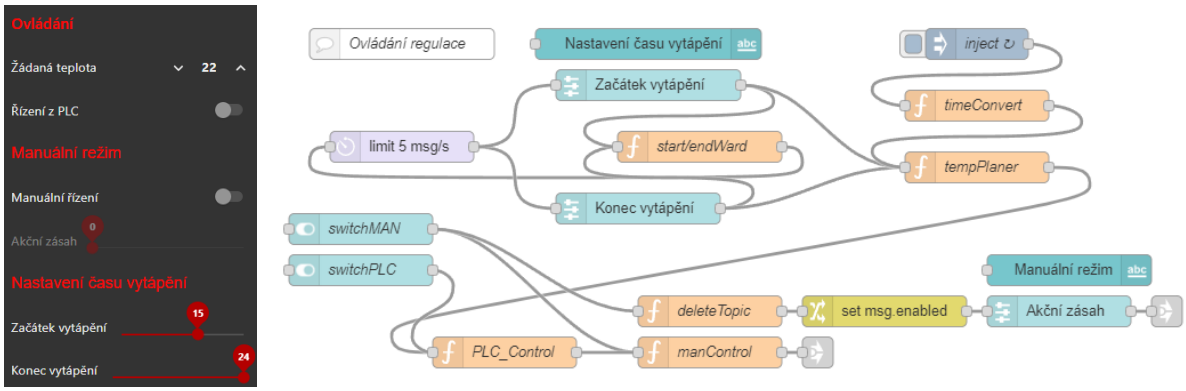
Pro předpověď teploty v místnosti se jeví jako vhodná volba využití různých variant modelů strojového učení (Machine Learning, ML). Ačkoliv v Node-RED knihovnách existují uzly pro integraci různých předtrénovaných ML modelů pro širokou škálu úkolů, pro specifické potřeby tohoto projektu se ukazují jako nevhodné. Důvodem je, že podmínky v dané místnosti jsou unikátní, což vyžaduje vytvoření a natrénování vlastního modelu. To si vyžaduje shromáždění dostatečného množství dat, aby bylo možné pokrýt ideálně všechny možné stavy prostředí a vytvořit několik tréninkových souborů pro trénování modelu.

Pro trénování a implementaci ML modelů byla z mnoha dostupných Node-RED knihoven vybrána *node-red-contrib-machine-learning-v2*. Tato knihovna nabízí rozsáhlé možnosti pro trénování modelů, což s sebou přináší vyšší komplexnost celého procesu a vyžaduje více času pro dokončení řešení. Ke dni psaní této práce model nedosahoval dostatečné kvality výsledků, aby mohl být prezentován jako spolehlivé řešení pro predikci vnitřní teploty v místnosti s následným použití v regulaci. Dosavadní postup lze nalézt v Příloha 13.

5.1.4.5 Implementace uživatelského rozhraní

Jak bylo zmíněno, pro vizualizaci nelze kvůli výpočetním omezením Raspberry Pi využít Home Assistant, a proto se práce obrací k použití knihovny Node-RED Dashboard. I když je tato knihovna často využívána, není součástí standardní instalace Node-RED, a je tedy nutné ji dodatečně instalovat.

Hlavním důvodem, který vedl k vyhýbání se použití Dashboardu, jsou omezené možnosti konfigurace jeho prvků. Ty často vyžadují rozšíření pomocí dodatečné logiky Node-RED. Příkladem může být srovnání ovládacího panelu pro regulaci a jeho datového toku na Obr. 23 , který zajišťuje funkční logiku. Skripty z tohoto toku dat lze nalézt v Příloha 14, přičemž výsledná podoba celého uživatelského rozhraní se nachází v Příloha **17**.

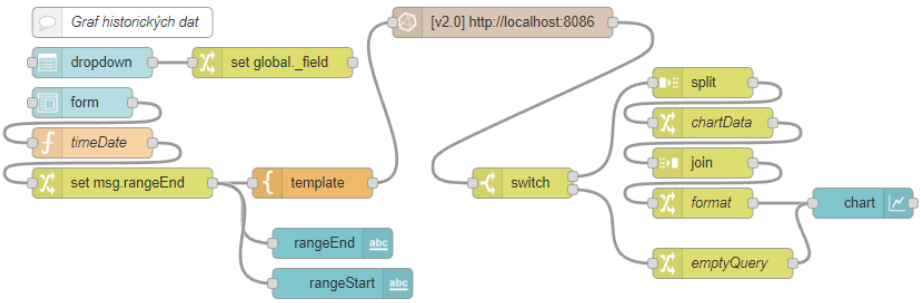


Obr. 23 Ovládací panel a Node-RED flow pro jeho logiku

Na předchozím obrázku je prezentována logika, která umožňuje nastavit jednoduchý časový plán pro spuštění a vypnutí vytápění v určitou hodinu dne s použitím dvou posuvníků. Zároveň je zde implementována kontrola, aby se nastavené časy spuštění a vypnutí navzájem nepřekrývaly. Další prvky umožňují volbu mezi řízením

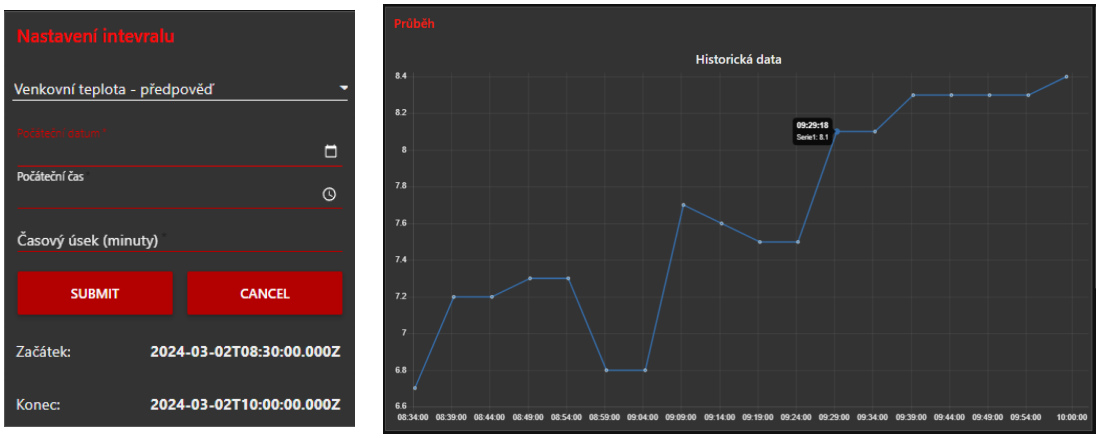
regulace prostřednictvím Node-RED nebo PLC a mezi automatickým a manuálním režimem. Podle zvoleného režimu se pak aktivuje posuvník pro manuální nastavení hodnoty akčního zásahu.

V uživatelském rozhraní jsou zahrnuty také různé grafy. Graf s poměrně komplexní logikou, viz Obr. 24, slouží k zobrazování historických dat získaných z databáze. Pro výběr specifického časového rozmezí a požadovaných dat je využit uzel *form*, který upravuje dotaz směřující do databáze. Data získaná z databáze jsou ve formě objektu, který je nutné přepracovat do formátu, který může uzel *chart* akceptovat a vizualizovat. K tomuto účelu slouží kombinace uzlů *split*, *chartData* a *format*. Pokud je vybráno časové rozmezí bez dostupných dat, což vede k vypsání prázdného objektu, uzel *switch* vybere datovou cestu skrze uzel *emptyQuery*, který informaci o prázdném objektu zobrazí v názvu grafu.



Obr. 24 Node-RED flow pro zobrazení hodnot z formuláře v grafu

Přestože databáze může obsahovat více hodnot v daném časovém okně, uzel *chart* není schopen zpracovat dva a více datových objektů obsahujících časové údaje pro osu x. Je tedy nutné provést dotaz na každou hodnotu zvlášť. Výsledkem zadaných hodnot ve formuláři je graf zobrazený na Obr. 25.



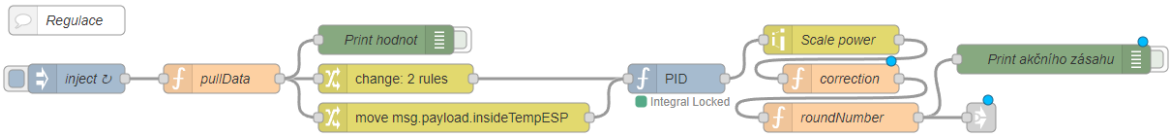
Obr. 25 Formulář pro výběr časového okna a graf historických dat z databáze

V rámci implementace hlasového asistenta nabízí dokumentace možnost integrace prostřednictvím knihovny *node-red-contrib-smartnora*, která umožňuje propojení Node-RED se serverem Smart NORA. Toto propojení umožňuje vytvoření virtuálních zařízení v ekosystému Google Home, jež lze ovládat pomocí Google Assistant.

Přestože integrace s Google Assistantem nabízí zajímavé možnosti pro ovládání regulace, její implementace je spojena s určitými komplikacemi. Google Assistant není v České republice plně podporován, a navíc se očekává, že v blízké budoucnosti bude služba převedena na platformu Gemini. Vzhledem k těmto okolnostem se projekt zaměřuje na cíle s vyšší prioritou. [32]

5.1.4.6 Implementace regulace

Původně bylo zamýšleno, že regulace bude zajištěna s využitím modelu strojového učení (ML), který by přijímal potřebná vstupní data a na základě nich generoval příslušné akční zásahy. Avšak kvůli potížím již s predikcí teploty je v současné době další využití ML modelu v regulaci vyřazeno z možností.



Obr. 26 Node-RED flow pro logiku regulace

Tradiční metoda regulace by obvykle zahrnovala použití PID regulátoru, který však nenabízí možnost předpovídání venkovní teploty. Z tohoto důvodu bude aplikována kombinace PID regulátoru a dodatečné funkce, která bude upravovat korekci výstupního akčního zásahu z PID regulátoru podle očekávaného vývoje venkovních teplot, jak je znázorněno na Obr. 26.

V případě výpočtu korekce akčního zásahu na základě predikovaného vývoje venkovní teploty byl použit funkční uzel *correction*. Tento uzel zajišťuje dynamickou regulaci teploty v místnosti tím, že kombinuje předpovědi venkovní teploty s aktuálním výstupem z PID regulátoru. Následující skript umístěný v tomto funkčním uzlu, začíná načtením aktuálních dat o teplotách z globální proměnné *data*. Vypočítává průměrnou budoucí venkovní teplotu, přičemž 3hodinové předpovědi přisuzuje dvojnásobnou váhu ve srovnání s 6hodinovými.

Rozdíl mezi aktuální venkovní teplotou a průměrnou budoucí venkovní teplotou je využit k výpočtu korekčního faktoru, který se pak aplikuje na akční zásah PID regulátoru. Ve skriptu je též definovaná funkce *calculateCorrectionImpact*, která snižuje dopad této korekce v závislosti na rozdílu mezi vnitřní teplotou a žádanou teplotou. Čím je vnitřní teplota blíže k žádané hodnotě, tím menší je korekční vliv. Výsledný korigovaný akční zásah je omezen na rozsah 0 až 100, což zajišťuje, že výstup z regulátoru zůstává v přijatelných mezích pro převedení na PWM.

```
//Globální proměnná
let data = flow.get("data") || {};
//Deklarace proměnných
let currentOutsideTemp = data.ousideTempForecast;
let futureOutsideTemp3h = data.ousideTempForecast3h;
let futureOutsideTemp6h = data.ousideTempForecast6h;
let insideTemp = data.insideTempESP;
let desiredTemp = data.requiredTemp;
let pidOutput = msg.payload
// Výpočet průměrné budoucí venkovní teploty
// Bližší předpověď má vyšší váhu
let averageFutureTemp = ((futureOutsideTemp3h*2) + futureOutsideTemp6h) / 3;
// Výpočet rozdílu mezi aktuální a průměrnou budoucí venkovní teplotou
let tempDifference = currentOutsideTemp - averageFutureTemp;
// Výpočet korekce
let offset = tempDifference * calculateCorrectionImpact(insideTemp, desiredTemp);
// Přidáme offset k výstupu PID
let adjustedOutput = Math.max(0, Math.min(100, (pidOutput + offset)));
// Funkce pro výpočet dopadu offsetu, snižuje dopad při blížení se k žádané
teplotě
function calculateCorrectionImpact(insideTemp, desiredTemp) {
    let tempGap = Math.abs(insideTemp - desiredTemp);
    // Například exponenciální zpomalení dopadu offsetu
    // Čím blíže je vnitřní teplota k žádané, tím menší bude vliv korekce
    return Math.exp(-tempGap);
}
// Nastavíme upravený akční zásah do payload
msg.payload = adjustedOutput;

return msg;
```

Při konfiguraci uzlu PID byl zvolen přístup manuální konfigurace PID složek. Důvodem proč se nepoužila žádná z mnoha metod pro nastavení PID složek regulátoru, je velké časové zpoždění systému a jeho integrační chování, kvůli dlouhodobým tepelným ztrátám. Z tohoto důvodu bylo upřednostněno manuální nastavení PID, kdy se počáteční hodnoty všech složek nastavily na nulový zásah a následně se postupně snižovalo proporcionální pásmo až do okamžiku, kdy systém začal oscilovat. Poté se postupně přidávali integrační a derivační složky do chvíle, kdy se dosáhlo žádaných výsledků.

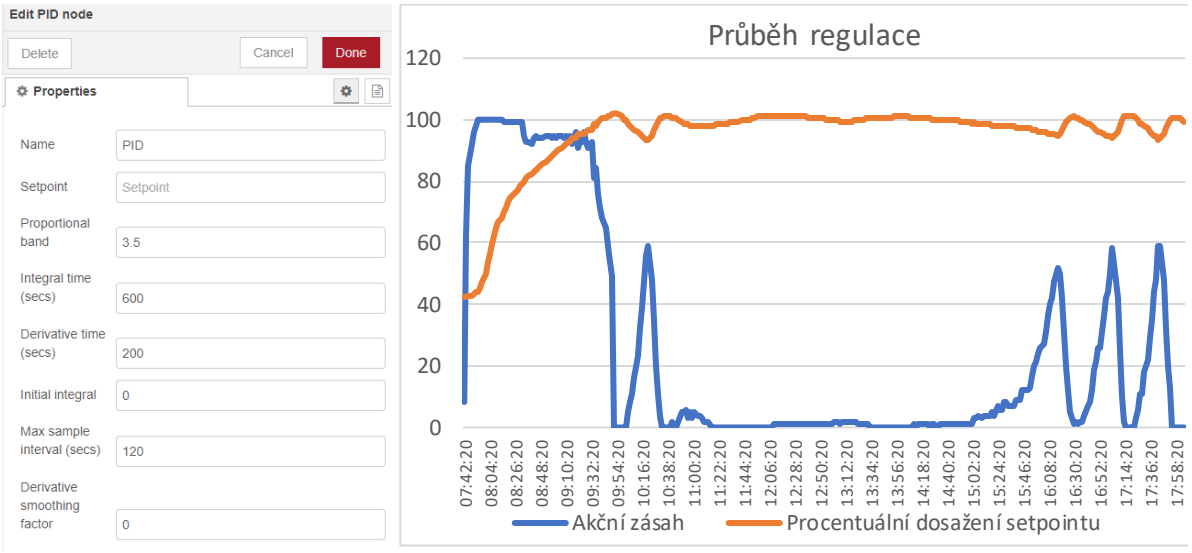
Na Obr. 27 je zobrazen konfigurační panel PID uzlu v Node-RED, jehož parametry byly aplikovány i v tomto projektu. Parametr *Proportional band* u PID regulátorů, které

používají proporcionální pásmo místo proporcionálního zesílení, určuje rozsah kolem cílové hodnoty (setpointu), v němž PID regulátor aktivně upravuje proces. Nižší hodnota tohoto pásma vede k silnější reakci na jakoukoliv odchylku od setpointu.

Parametr *Integral time* ovlivňuje integrační složku PID algoritmu, sloužící k odstranění stálé chyby mezi aktuální a žádanou hodnotou. Tento parametr definuje, s jakou rychlostí se integrační akce zvyšuje, aby kompenzovala přetrvávající odchylku.

Časová konstanta *Derivative time* pak určuje rychlost, s jakou bude derivační složka reagovat na změny v systému, a určuje vliv rychlosti změny chyby na regulaci.

Z dalších parametrů je podstatné zmínit už jen *Max sample interval* což je maximální interval mezi vzorky, který PID regulátor používá pro aktualizaci výpočtů. Delší interval může znamenat méně časté aktualizace a může způsobit pomalejší reakci na změny.



Obr. 27 Konfigurace PID uzlu v Node-RED a graf průběhu regulace

Na Obr. 27 je také vidět graf průběhu regulace bez vlivu předpovědi počasí. Modrá křivka označuje procentuální akční zásah PID regulátoru, zatímco oranžová křivka reprezentuje procentuálně dosažení žádané hodnoty v místnosti. Na grafu od 15. hodiny jsou zřetelné experimenty, kdy byl prostor místnosti úmyslně ochlazován za účelem posouzení odezvy regulačního systému na tento zásah.

6 Zhodnocení dosažených výsledků

Porovná-li se existující konečné řešení s cíli této práce, lze konstatovat, že hlavní cíl byl splněn. Hlavní logika regulačního systému je tvořena na platformě Node-RED, i když je součástí regulace i PLC, které je schopné převzít celou kontrolu nad regulací, v běžném provozu však reaguje pouze na povely z Node-RED. Nicméně, faktem, že tato funkce se v regulaci nachází, je plněn cíl zajistit kompatibilitu s již existujícím řídícím systémem a rozšířit jeho základní možnosti a případně implementovat jej do ekosystému internetu věcí.

Součástí regulace je i funkce získávání z internetových zdrojů aktuálních a prediktivních meteorologických informací, konkrétně se však pracuje pouze s teplotou, neboť ta je pro regulaci zásadní. Praktickou výhodou, kterou systém získávání informací o venkovní teplotě z externích zdrojů přináší, je řešení několika podstatných problémů spojených s tradičním instalováním venkovních čidel. Tyto problémy zahrnují například komplikace s vedením kabeláže mimo vnitřní prostory budovy, což obvykle vyžaduje rozsáhlejší elektroinstalatérské práce. Další výzvou je umístění senzorů tak, aby nebyly vystaveny přímému slunečnímu světlu, které může ovlivnit měřené hodnoty, což není vždy možné zajistit, zejména pokud není možné čidlo umístit na severní stranu budovy. Zavedením této funkcionality lze konstatovat, že jsou částečně reflektovány současné potřeby v oblasti měření a regulace.

K vyhodnocení přesnosti dat získaných z meteorologické modelu a měření prováděných teplotním čidlem PT1000, zapojeným do PLC, bylo využito analýzy více než 40 000 záznamů teploty, zaznamenaných v období mezi 18. lednem a 18. březnem 2024. Tato analýza, viz Tab. 7, využila různé statistické metody pro určení míry chyby měření. Podle koeficientu determinace indikují hodnoty měření z čidla Pt1000 a z meteorologických dat relativně nízkou úroveň shody. Meteorologická stanice, poskytující data, se nacházela přibližně 4,6 km od místa měření, což je s ohledem na měření teploty poměrně blízko. Po detailnější analýze dat bylo zjištěno, že hodnoty z čidla Pt1000 vykazují významné teplotní výkyvy, a to i přes to, že čidlo nebylo přímo vystaveno slunečnímu svitu. Tyto výkyvy mohly být způsobeny sáláním povrchu

zahradního domku, který akumuloval teplotu ze slunce, což by vysvětlovalo až několika stupňový rozdíl v naměřené hodnotě čidla Pt1000 a meteorologickými daty.

Tab. 7 Analýza shody měření teplot

Metoda	Venkovní teplota (Pt1000 z PLC / Meteorologická předpověď)	Vnitřní teplota (Ni1000 z PLC / DS18B20 z ESP8266)
Průměrná absolutní chyba	2,46 °C	0,95 °C
Střední kvadratická chyba	13,22	8,15
Koeficient determinace R^2	0,20	0,75

Co se týče srovnání měření vnitřní teploty z modulu ESP8266 a PLC, hodnoty mají znatelně výraznější shodu. Avšak byla očekávána ještě vyšší shoda, vzhledem k tomu, že všechna čidla byla umístěna pouhé jednotky centimetrů od sebe a při instalaci vykazovala téměř shodné hodnoty. Pravděpodobně mohly na patrné rozdíly v teplotách mít vliv vlastnosti obvodu a vodičů modulu ESP8266. U PLC systému by se očekávala nižší pravděpodobnost chyby, jelikož se jedná o robustnější řešení ve srovnání s vývojovými deskami.

Mezi cíle práce patří i zajistit, aby regulační systém mohl predikovat venkovní a vnitřní teplotu. Predikce venkovní teploty, získaná ze zdrojů meteorologických dat, je součástí regulace a umožňuje přizpůsobení akčních zásahů PID regulátoru. Pro predikci vnitřní teploty se ukázalo nezbytné využít modelu strojového učení (Machine learning, ML), který by byl vytrénován konkrétně na datech z regulovaného systému. Dosud se však pomocí knihoven z Node-RED nepodařilo docílit požadovaných výsledků predikce. V případě pokračujících problémů s predikcí v prostředí Node-RED se jako alternativa nabízí využití programovacího jazyka Python a jeho knihoven určeným pro trénování modelů, toto řešení se často využívá v oblasti strojového učení.

Při vývoji vizuálního prostředí pro řízení regulačního systému byla, navzdory původnímu plánu a problémům s výpočetní kapacitou zařízení, použita knihovna Node-RED Dashboard. Tato knihovna poskytuje základní nástroje pro vytváření ovládacích prvků, ale v některých ohledech má svá omezení. V důsledku toho navržené uživatelské rozhraní umožňuje pouze základní monitorování systémů, akorát dostatečné pro ladění

procesu regulace. Pro vytvoření plně funkčního ovládacího rozhraní, například pro domácí použití, by bylo vhodné začlenit časový plán umožňující nastavit žádanou teplotu samostatně pro každý den v týdnu v určitém čase. V Dashboardu by to bylo možné realizovat s pomocí rozsáhlé tabulky s pevně stanovenými časy a možností přiřadit žádané teploty, což by ale pro koncového uživatele nebylo příliš uživatelsky přívětivé.

Součástí hodnocení dosažených výsledků by bylo vhodné zohlednit i finanční náklady pro realizaci tohoto řešení. Nepočítaje PLC, elektroměr a síťové prvky, v tomto projektu byly využity výhradně cenově dostupné součástky, které oproti průmyslovým řešením, mají téměř zanedbatelnou hodnotu.

Tab. 8 Náklady na zařízení (bez vodičů a síťové infrastruktury)

Položka	Množství	Cena za kus	Cena
ESP8266 Lua NodeMcu V3	2	127 Kč	254 Kč
Raspberry Pi 4 Model B – 4GB RAM	1	2 790 Kč	2790 Kč
Digitální teplotní senzor – DS18B20	3	89 Kč	267 Kč
Relé SSR Solid State SSR-60DA 380V	1	184 Kč	184 Kč
Celková cena			3495 Kč

Pro ilustraci, pouhá funkce bezdrátového měření teploty, která je řešena v této práci, může být porovnána s průmyslovým řešením zastoupeným bezdrátovou technologií EnOcean fungující ve frekvenčním pásmu 868 MHz. Cena rádiového přijímačového modulu pro řídící stanice Wago se pohybuje okolo 19 000 Kč, zatímco cena solárních teplotních snímačů je přibližně 5 000 Kč. Tento výrazný rozdíl v nákladech na zařízení poukazuje na výraznou cenovou výhodu řešení prezentovaného v této práci, přesto však průmyslová řešení nabízejí mnohem vyšší úroveň robustnosti a obecně spolehlivosti. [33, 34, 35]

7 Závěr a doporučení

Výsledkem diplomové práce je funkční prototyp regulačního systému řízeného platformou Node-RED nasazený na zařízení Raspberry Pi 4 Model B. Součástí vývoje tohoto systému byla tvorba teoretických základů potřebných pro následný návrh a výběr vhodného řešení. Jednou z hlavních myšlenek autora, při realizaci této práce, byla snaha o využití ne zcela konvenčních metod pro tvorbu dostatečně robustního systému, který by alespoň částečně mohl konkurovat tradičním řídícím systémům jako jsou například programovatelné logické automaty (PLC).

Současně snahou autora bylo rozšířit omezené možnosti běžného PLC tím, že jej integroval do prostředí internetu věcí, čímž výrazně zvýšil jeho potenciál pro interakci s okolním světem. To je zřejmé například z možnosti, jak PLC může získávat meteorologická data díky internetu věcí. Avšak možnosti jsou daleko širší, systém může například sledovat polohu mobilního zařízení uživatele a podle toho upravovat parametry regulace v závislosti na jeho přítomnosti v domácnosti. S tím však souvisí i řada komplikací z hlediska bezpečnosti a spolehlivosti celého systému, což je typický problém pro aplikace v rámci internetu věcí. V reakci na tyto výzvy autor věnoval úsilí zajištění spolehlivé komunikace mezi zařízeními a bezpečnosti, i když to znamenalo méně času věnovaného na zlepšení uživatelského rozhraní.

Autor se rovněž zaměřil na reakci na nové legislativní požadavky zavedené zákonem č. 424/2022 Sb., podle kterého je od 1. ledna 2024 povinné provádět měsíční odečty měřičů tepla a teplé vody a předávat naměřené hodnoty konečným spotřebitelům. V rámci diplomové práce bylo proto implementováno zařízení schopné vyčítat S0 impulsy z elektroměru. Vzhledem k tomu, že S0 impulsy jsou uznávaným standardem, lze tento přístup aplikovat i na kalorimetry. Toto řešení umožňuje provádět dálkové měsíční odečty v objektech, kde dosud neexistuje specificky pro tyto účely vybudovaná infrastruktura. [9]

Realizace projektu též zdůraznila výzvy spojené s vývojem modelu strojového učení pro předpověď teploty v místnosti, což představuje výchozí bod pro budoucí práci. Přes snahu o inovativní řešení se v rámci projektu však dosud nepodařilo úspěšně vyvinout model strojového učení, který by s dostatečnou přesností předpovídal budoucí teplotu

v místnosti. Nicméně, využití strojového učení pro regulaci teploty v budovách nabízí obrovský potenciál. Je-li model strojového učení schopen zohlednit faktory jako vlhkost vzduchu, meteorologické předpovědi a aktuální ceny energií, může významně přispět k zvýšení efektivity vytváření komfortního prostředí a optimalizaci spotřeby energie.

8 Seznam použité literatury

[1] OPENJS FOUNDATION & CONTRIBUTORS. *About : Node-RED* [online]. [vid. 2023-11-16]. Dostupné z: https://nodered.org

[2] JAPÓN, Bernardo Ronquillo. *Learn IoT Programming Using Node-RED: Begin to Code Full Stack IoT Apps and Edge Devices with Raspberry Pi, NodeJS and Grafana* [online]. nedatováno. Dostupné z: www.bpbonline.com

[3] HAGINO, Taiji, Nick O'LEARY a an O'Reilly Media Company. SAFARI. *Practical Node-RED Programming*. nedatováno. ISBN 9781800201590.

[4] CLERISSI, Diego, Maurizio LEOTTA a Filippo RICCA. A set of empirically validated development guidelines for improving Node-RED flows comprehension. In: *ENASE 2020 - Proceedings of the 15th International Conference on Evaluation of Novel Approaches to Software Engineering* [online]. B.m.: SciTePress, 2020, s. 108–119. ISBN 9789897584213. Dostupné z: doi:10.5220/0009391101080119

[5] *Home Assistant* [online]. [vid. 2024-01-21]. Dostupné z: https://www.home-assistant.io/

[6] *Installation - Home Assistant* [online]. [vid. 2024-01-21]. Dostupné z: https://www.home-assistant.io/installation/

[7] *Dashboards - Home Assistant* [online]. [vid. 2024-01-21]. Dostupné z: https://www.home-assistant.io/dashboards/

[8] *Understanding the Meaning of IFTTT, Its Working, and Alternatives | Spiceworks - Spiceworks* [online]. [vid. 2024-01-21]. Dostupné z: https://www.spiceworks.com/tech/tech-general/articles/what-is-ifttt/

[9] *362/2021 Sb. Zákon, kterým se mění zákon č. 458/2000 Sb., o podmínkách podnikání a o výkonu státní správy v energ...* [online]. [vid. 2024-03-20]. Dostupné z: https://www.zakonyprolidi.cz/cs/2021-362

[10] *Getting Started : Node-RED* [online]. [vid. 2024-01-26]. Dostupné z: https://nodered.org/docs/getting-started/

[11] *Eclipse Mosquitto* [online]. [vid. 2024-01-28]. Dostupné z: https://mosquitto.org/

[12] *Get started with InfluxDB | InfluxDB OSS v2 Documentation* [online]. [vid. 2024-01-27]. Dostupné z: https://docs.influxdata.com/influxdb/v2/get-started/

[13] *IoT ESP8266 Lua NodeMcu V3 WIFI modul | LaskaKit* [online]. [vid. 2024-01-29]. Dostupné z: https://www.laskakit.cz/iot-esp8266-lua-nodemcu-v3-wifi-modul--tcp-ip/#productDiscussion

[14] NodeMCU ESP-12E ESP8266 WiFi Lua IoT CH340. nedatováno.

[15] *IEEE SA - The Evolution of Wi-Fi Technology and Standards* [online]. [vid. 2024-02-03]. Dostupné z: https://standards.ieee.org/beyond-standards/the-evolution-of-wi-fi-technology-and-standards/

[16] *NodeMCU V3 ESP8266 Pinout and Configuration - CyberBlogSpot* [online]. [vid. 2024-02-02]. Dostupné z: https://cyberblogspot.com/nodemcu-v3-esp8266-pinout-and-configuration/

[17] *Basics of PWM (Pulse Width Modulation) | Arduino Documentation* [online]. [vid. 2024-02-03]. Dostupné z: https://docs.arduino.cc/learn/microcontrollers/analog-output/

[18] *1-Wire Protocol | Arduino Documentation* [online]. [vid. 2024-02-03]. Dostupné z: https://docs.arduino.cc/learn/communication/one-wire/

[19] *Basics of the I2C Communication Protocol* [online]. [vid. 2024-02-02]. Dostupné z: https://www.circuitbasics.com/basics-of-the-i2c-communication-protocol/

[20] *Serial Peripheral Interface (SPI) - SparkFun Learn* [online]. [vid. 2024-02-02]. Dostupné z: https://learn.sparkfun.com/tutorials/serial-peripheral-interface-spi/all

[21] *Understanding UART | Rohde & Schwarz* [online]. [vid. 2024-02-02]. Dostupné z: https://www.rohde-schwarz.com/cz/products/test-and-measurement/essentials-test-equipment/digital-oscilloscopes/understanding-uart_254524.html

[22] DENNIS, Andrew K. *Raspberry Pi home automation with Arduino : automate your home with a set of exciting projects for the Raspberry Pi!* B.m.: Packt Pub, 2013. ISBN 9781849695862.

[23] PIN CONFIGURATIONS [online]. nedatováno [vid. 2024-02-03]. Dostupné z: www.umw-ic.com

[24] *UMW DS18B20 Digitální čidlo teploty TO-92 | LaskaKit* [online]. [vid. 2024-02-03]. Dostupné z: https://www.laskakit.cz/dallas-digitalni-cidlo-teploty-ds18b20--to-92/

[25] RO, spol. Compact control systems AMiNi4DW2 Compact control system with display [online]. nedatováno [vid. 2024-01-29]. Dostupné z: www.amit.cz/support

[26] *What is the Modbus Protocol & How Does It Work? - NI* [online]. [vid. 2024-02-04]. Dostupné z: https://www.ni.com/en/shop/seamlessly-connect-to-third-party-devices-and-supervisory-system/the-modbus-protocol-in-depth.html

[27] *Modbus Function Codes* [online]. [vid. 2024-02-04]. Dostupné z: https://ozeki.hu/p_5873-modbus-function-codes.html

[28] *DetStudio - vývojové prostředí - AMiT Automation* [online]. [vid. 2024-01-30]. Dostupné z: https://amitomation.cz/produkt/software/detstudio-navrhove-prostredi/

[29] *Raspberry Pi 4 Model B - 4GB RAM | dratek.cz* [online]. [vid. 2024-03-23]. Dostupné z: https://dratek.cz/arduino/74434-raspberry-pi-4-model-b-4gb-ram.html

[30] *PubSubClient - Arduino Reference* [online]. [vid. 2024-03-06]. Dostupné z: https://www.arduino.cc/reference/en/libraries/pubsubclient/

[31] *Running on Raspberry Pi : Node-RED* [online]. [vid. 2024-03-13]. Dostupné z: https://nodered.org/docs/getting-started/raspberrypi

[32] *Does Gemini completely replace Google Assistant?* [online]. [vid. 2024-03-18]. Dostupné z: https://9to5google.com/2024/02/27/google-assistant-gemini-replacement/

[33] *EnOcean Wireless Communication Protocol - Automation Community* [online]. [vid. 2024-03-23]. Dostupné z: https://automationcommunity.com/enocean-wireless-communication/

[34] *Bezdrátový snímač teploty SR65, do exteriéru - E-shop pro průmyslovou a domovní automatizaci REM-Technik s.r.o.* [online]. [vid. 2024-03-23]. Dostupné

z: https://www.rem-shop.cz/mereni-a-regulace/snimace-teploty/snimace-venkovni/bezdratovy-snimac-teploty-sr65-do-exterieru-973.html

[35] *Rádiový přijímač EnOcean WAGO 750-642* [online]. [vid. 2024-03-23]. Dostupné z: https://www.emas.cz/750-642

9 Přílohy

Příloha 1: Skript na modulu ESP8266 pro měření teploty a komunikace s Node-RED

```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
#include <OneWire.h>
#include <DallasTemperature.h>

// Prihlasovaci udaje na WiFi
const char* ssid = "Turris";
const char* password = "********";

// Prihlasovaci udaje na MQTT (NULL jestli nejsou potreba)
const char* MQTT_username = "asuchopar";
const char* MQTT_password = "********";
const char* mqtt_server = "192.168.***.***";

// Initializeace espClient (zmen nazev pro vicero ESP)
WiFiClient espClient;
PubSubClient client(espClient);

// Teplomery jsou pripojeny k GPIO15 (D4)
#define ONE_WIRE_BUS 4
// Nastavi instanci oneWire pro komunikaci s zarizenim OneWire
OneWire oneWire(ONE_WIRE_BUS);
// Preda referenci oneWire senzoru teploty Dallas
DallasTemperature sensors(&oneWire);
// Adresy senzoru
DeviceAddress sensor1 = { 0x28, 0x81, 0x72, 0x20, 0xF, 0x0, 0x0, 0x2 };
DeviceAddress sensor2 = { 0x28, 0xE9, 0x59, 0x21, 0xF, 0x0, 0x0, 0x67 };
DeviceAddress sensor3= { 0x28, 0x43, 0x13, 0x21, 0xF, 0x0, 0x0, 0xA0 };

//LED - GPIO 4 = D2 na desce ESP-12E NodeMCU (blikani z Node-RED)
const int LED = D1;

// Pomocne promenne pro casovace
long now = millis();
long lastMeasure = 0;

// Promenne pro teplotu
float temp1 = 0;
float temp2 = 0;
float temp3 = 0;

// Promena pro citac komunikace
int counter = 0;

// Pripoji ESP8266 k Wifi
void setup_wifi() {
  delay(10);
  // We start by connecting to a WiFi network
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.println("."); //ceka se na pripojeni
  }

  Serial.println("");
```

```
  Serial.println("WiFi connected - ESP IP address: ");
  Serial.println(WiFi.localIP());
}

// Tato funkce je vykonana, kdyz nejake zarizeni publikuje zpravu na tema, ke
kteremu je ESP8266 prihlaseno
void callback(String topic, byte* message, unsigned int length) {
  Serial.print("Message arrived on topic: ");
  Serial.print(topic);
  Serial.print(". Message: ");
  String messageTemp;

  for (int i = 0; i < length; i++) {
    Serial.print((char)message[i]);
    messageTemp += (char)message[i];
  }
  Serial.println();
// Pokud je zprava prijata na tema Room / LED……
  if(topic=="Room/LED"){
      Serial.println("Changing Room LED to ");
      if(messageTemp == "true"){
        digitalWrite(LED, HIGH);
        Serial.println("On");
      }
      else if(messageTemp == "false"){
        digitalWrite(LED, LOW);
        Serial.println("Off");
      }
  }
  Serial.println();
}
// Funkce pro opetovne pripoji ESP8266 k MQTT brokeru
void reconnect() {
  // Loop dokud se nepripoji
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    // Pokus o pripojeni
    if (client.connect("ESP8266_Client_Temp", MQTT_username, MQTT_password)) {
      Serial.println("connected");
      // Odebirat nebo odeslat topic
      client.subscribe("Room/LED"); //prijem topicu se zprávou z broukereu
    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      // Wait 5 seconds before retrying
      delay(5000);
    }
  }
}
//inicializace při prvním spusteni
void setup() {
  pinMode(LED, OUTPUT);

  Serial.begin(9600);
  setup_wifi();
  client.setServer(mqtt_server, 1883);
  client.setCallback(callback);

  sensors.begin();
}

void loop() {
//Pokud klient není pripojen, znovu se pripoj
  if (!client.connected()) {
```

```
    reconnect();
  }
 //Pokud nebezi klientova smycka, pripoj klienta
  if(!client.loop())
    client.connect("ESP8266Client", MQTT_username, MQTT_password);

  now = millis();
  //tempInfo();

  if (now - lastMeasure > 5000) {
    //Citac pro hlidani komunikace
    lastMeasure = now;
    counter = counter + 1;
    if (counter > 150){
      counter = 0;
    }
    //odesilani citace
    char counterString[8];
    dtostrf(counter, 1, 0, counterString);
    Serial.print("Counter: ");
    Serial.println(counter);
    Serial.println(" out of 150");
    client.publish("room/counter", counterString);

    Serial.print("Requesting temperatures...");
    sensors.requestTemperatures(); // Odeslat povel pro ziskani teploty
    Serial.println("DONE");

    // Teploty ve stupnich Celsia
    temp1 = sensors.getTempC(sensor1);
    temp2 = sensors.getTempC(sensor2);
    temp3 = sensors.getTempC(sensor3);

    // Prevod hodnoty teploty na retezec stringu pro sensor 1
    char tempString1[8];
    dtostrf(temp1, 1, 2, tempString1);
    Serial.print("Sensor 1(*C): ");
    Serial.println(tempString1);
    //Publikovani teploty
    client.publish("room/temp1", tempString1);

    // Prevod hodnoty teploty na retezec stringu pro sensor 2
    char tempString2[8];
    dtostrf(temp1, 1, 2, tempString2);
    Serial.print("Sensor 2(*C): ");
    Serial.println(tempString2);
    client.publish("room/temp2", tempString2);

    // Prevod hodnoty teploty na retezec stringu pro sensor 3
    char tempString3[8];
    dtostrf(temp1, 1, 2, tempString3);
    Serial.print("Sensor 3(*C): ");
    Serial.println(tempString3);
    client.publish("room/temp3", tempString3);

  }
}
```

Příloha 2: Skript na modulu ESP8266 pro počítaní S0 impulsu a komunikaci s Node-RED

```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>


// Prihlasovaci udaje na WiFi
const char* ssid = "Turris";
const char* password = "********";

// Prihlasovaci udaje na MQTT (NULL jestli nejsou potreba)
const char* MQTT_username = "asuchopar";
const char* MQTT_password = "********";

const char* mqtt_server = "192.168.***.***";

// Initializeace espClient (zmen nazev pro vicero ESP)
WiFiClient espClient2;
PubSubClient client(espClient2);

// Pomocna promena pro casovac
long now = millis();
long lastMeasure = 0;

//promenne pro pocitani S0 pulsu
volatile unsigned long pulseCount = 0;
unsigned long lastPulseTime = 0;
const int pulsePin = D5;

//Promena pro kontrolu komunikace
int counter = 0;

// Tohle by melo pripojit ESP k wifi
void setup_wifi() {
  delay(10);
  // Pokus o pripojeni
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.println("."); //ceka se na pripojeni
  }

  Serial.println("");
  Serial.println("WiFi connected - ESP IP address: ");
  Serial.println(WiFi.localIP());
}
// Zpracovani pulsu
void IRAM_ATTR handlePulse() {
  unsigned long currentPulseTime = millis();
  // Debounce - ignoruje pulzy, které prichazi rychle za sebou
  if (currentPulseTime - lastPulseTime > 10) { // 10 ms debounce
    pulseCount++;
    lastPulseTime = currentPulseTime;
  }
}
//=======Zalozny rizeni PWM======
// Spusti se ve chvili kdy je shodny topic zpravy z MQTT s ESP
void callback(String topic, byte* message, unsigned int length) {
  Serial.print("Message arrived on topic: ");
  Serial.print(topic);
  Serial.print(". Message: ");
```

```
  String messagePWM;

  for (int i = 0; i < length; i++) {
    Serial.print((char)message[i]);
    messagePWM += (char)message[i];
  }
  Serial.println();

// Prevede prijatou zpravu na int, interpol 100 -> 255, zapis na pinout
  int pwmValue = messagePWM.toInt();
  int espPwmValue = map(pwmValue, 0, 100, 0, 255);
  analogWrite(D1, espPwmValue);

  Serial.println();
}

// Pokus o pripojeni k MQTT serveru
void reconnect() {
  // Loop dokud se nepripoji
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    // pokus o pripojeni
     if (client.connect("ESP8266_Client_Mesure", MQTT_username, MQTT_password)) {
      Serial.println("connected");
      // odebirat nebo odeslat topic
      client.subscribe("outside/PWM");
    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      delay(5000);
    }
  }
}
//Inicializace pri prvnim spusteni
void setup() {
  Serial.begin(9600);
  setup_wifi();
  client.setServer(mqtt_server, 1883);
  client.setCallback(callback);
  pinMode(pulsePin, INPUT_PULLUP); // Nastaví pin jako vstup s pull-up rezistorem
// Nastaví preruseni na sestupnou hranu signalu
  attachInterrupt(digitalPinToInterrupt(pulsePin), handlePulse, FALLING);
}
void loop() {
  //Pokud klient není pripojen, znovu se pripoj
  if (!client.connected()) {
    reconnect();
  }
  if(!client.loop())
    client.connect("ESP8266Client2", MQTT_username, MQTT_password);

  now = millis();
  //tempInfo();

  if (now - lastMeasure > 5000) {
  //Cidtac pro seldovani komunikace
    lastMeasure = now;
    counter = counter + 1;
    if (counter > 150){
      counter = 0;
    }
  //Odesilani citace
    char counterString[8];
    dtostrf(counter, 1, 0, counterString);
```

```
    Serial.print("Counter: ");
    Serial.print(counter);
    Serial.println(" out of 150");
    client.publish("outside/counter", counterString);
  }
    static unsigned long lastPrintTime = 0;
  if (millis() - lastPrintTime > 10000) { // Kazdych 10 sekund vypise pocet pulsu
    noInterrupts(); // Zastavi preruseni pro pristup ke sdilene promenne
    Serial.print("Number of pulses: ");
    Serial.println(pulseCount);
    interrupts(); // Povoli preruseni
    char pulseString[8];
    dtostrf(pulseCount, 1, 0, pulseString);
    client.publish("outside/pulses", pulseString);
    lastPrintTime = millis();
  }
}
```

Příloha 3: Funkční uzel Node-RED ukládající hodnoty z ESP do globální proměnný

counterSave

```
// Přijetí a uložení hodnoty do tokových proměnných
let currentValue = msg.payload;
let previousValue = flow.get("previousValue1") || null; // Získání předchozí
hodnoty

// Uložení aktuální hodnoty a času přijetí
flow.set("previousValue1", currentValue);
flow.set("lastUpdateTime1", Date.now());

return null
```

Příloha 4: Funkční uzel Node-RED hlídající funkčnost komunikace se zařízeními

counterCompare_ESP_temp

```
// Získání uložených hodnot
let previousValue = flow.get("previousValue1");
// Pokud lastValue neexistuje, použijte previousValue
let lastValue = flow.get("lastValue1") || previousValue;

// Kontrola, zda se hodnota změnila
if (lastValue === previousValue) {
    // Hodnoty se neshodují, zařízení neodeslalo novou aktualizaci
    msg = {};
    msg.topic = "Node-RED communication timeout";
    msg.payload = "Timeout: Lost communication with ESP8266 - temperature
sensor";
    // Uložení aktuální hodnoty pro porovnání při dalším spuštění
    flow.set("lastValue1", previousValue);
    return msg;
} else {
    // Uložení aktuální hodnoty pro porovnání při dalším spuštění
    flow.set("lastValue1", previousValue);
    msg.payload = true;
    return msg;
}
```

Příloha 5: Funkční uzel Node-RED limitující falešné poplachy

counterLimiter_ESP_temp

```
// Inicializace kontextové proměnné pro počítadlo, pokud ještě neexistuje
var count = context.get('countPuls') || 0;

if (msg.payload === true) {
    // Resetování počítadla, pokud je zpráva true
    count = 0;
} else {
    // Inkrementace počítadla pro každou zprávu s obsahem
    count += 1;
}

// Uložení aktualizované hodnoty počítadla zpět do kontextu
context.set('countPuls', count);

// Kontrola, zda počítadlo dosáhlo hodnoty 10
if (count >= 10) {
    // Resetování počítadla
    context.set('countPuls', 0);
    //Odeslání chybového hlášení
    node.error("Timeout: Lost communication with ESP8266 - temperature
sensor",msg);
    // Odeslání zprávy na výstup
    return msg;
}

return null;
```

Příloha 6: Funkční uzel Node-RED generující medián ze tří hodnot měření teploty

insideTempESP_median

```
// Získání uložených hodnot
let temps = flow.get("temps") || {};
let values = Object.values(temps);

// Seřazení hodnot
values.sort((a, b) => a - b);

let midIndex = Math.floor(values.length / 2);
let median = values[midIndex];
// Odeslání mediánu jako výsledek
msg.payload = median
return msg
```

Příloha 7: Funkční uzel Node-RED převádějící hodnoty z dvou registrů na jednu hodnotu

dataTypeConvertert

```
// Převod registru MSW a registru LSW
let msw = msg.payload[0];
let lsw  = msg.payload[1];

// Spojení hodnot do jednoho 32bitového čísla
let combined = (msw << 16) | lsw;

// Převod spojeného čísla na float
let buffer = Buffer.alloc(4);
buffer.writeUInt32BE(combined, 0);
let float = buffer.readFloatBE(0);

msg.payload = float;
return msg;
```

Příloha 8: Funkční uzel Node-RED pro zaokrouhlení hodnoty

roundNumber

```
let float = msg.payload
// Zaokrouhlení na 2 desetinná místa
let rounded = parseFloat(float.toFixed(2));
msg.payload = rounded

return msg;
```

Příloha 9: Funkční uzel Node-RED pro uložení dat do globální proměnné

pushData_insideTempPLC

```
let data = flow.get("data") || {}; // Získání stávajícího objektu nebo
inicializace nového
data.insideTempPLC = msg.payload;
flow.set("data", data);

return null
```

Příloha 10: Funkční uzel Node-RED pro vyčítání uložených dat z globální proměnné

pullData

```
// Přijetí a uložení hodnoty do tokové proměnné
let data = flow.get("data") || {}; // Získání stávajícího objektu nebo
inicializace nového
msg.payload = data;
return msg
```

Příloha 11: Funkční uzel Node-RED pro převedení formátu dat z databáze na formát vhodný pro zápis do souboru csv

convertToCSV

```
// Vstupní data
const inputData = msg.payload;

// Mapování pro ukládání dat podle _time
let dataMap = {};

// Inicializace hlavičky CSV
let csvHeader = "_time;Control action;Inside temperature (ESP);Inside temperature
(PLC);Outside Temperature (Forecast);Outside Temperature (PLC);Required
temperature\n";
let csvBody = "";

// Iterace přes vstupní data a vytváření mapy
inputData.forEach(item => {
    let key = item._time; // Klíč pro mapu je časové razítko
    if (!dataMap[key]) {
        // Inicializace nového záznamu s časovým razítkem, pokud ještě neexistuje
        dataMap[key] = {
            "Control action": "",
            "Inside temperature (ESP)": "",
            "Inside temperature (PLC)": "",
            "Ouside Temperature (Forecast)": "",
            "Ouside Temperature (PLC)": "",
            "Required temperature": ""
        };
    }
    // Přidání hodnoty do příslušného pole
    dataMap[key][item._field] = item._value;
});

// Generování těla CSV ze shromážděných dat
for (let time in dataMap) {
    csvBody += time + ";" +
        dataMap[time]["Control action"] + ";" +
        dataMap[time]["Inside temperature (ESP)"] + ";" +
        dataMap[time]["Inside temperature (PLC)"] + ";" +
        dataMap[time]["Ouside Temperature (Forecast)"] + ";" +
        dataMap[time]["Ouside Temperature (PLC)"] + ";" +
        dataMap[time]["Required temperature"] + "\n";
}

// Kombinace hlavičky a těla do jednoho CSV řetězce
let csvOutput = csvHeader + csvBody;

msg.payload = csvOutput;
return msg;
```

Příloha 12: Funkční uzly Node-RED pro vyčítání hodnoty z objektu generovaného openweathermap uzlem

outsideTempForecast_now

```
msg.payload = msg.payload.tempc
return msg
```

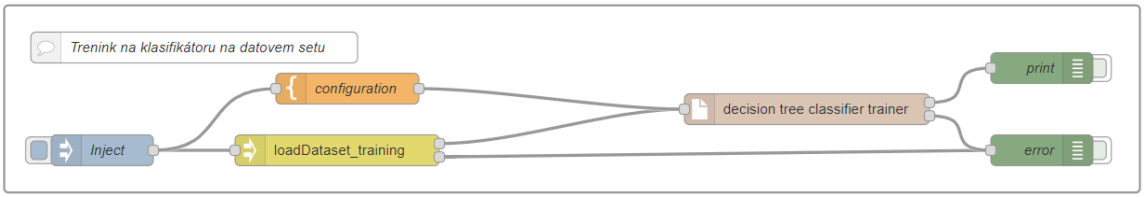
outsideTempForecast_3h

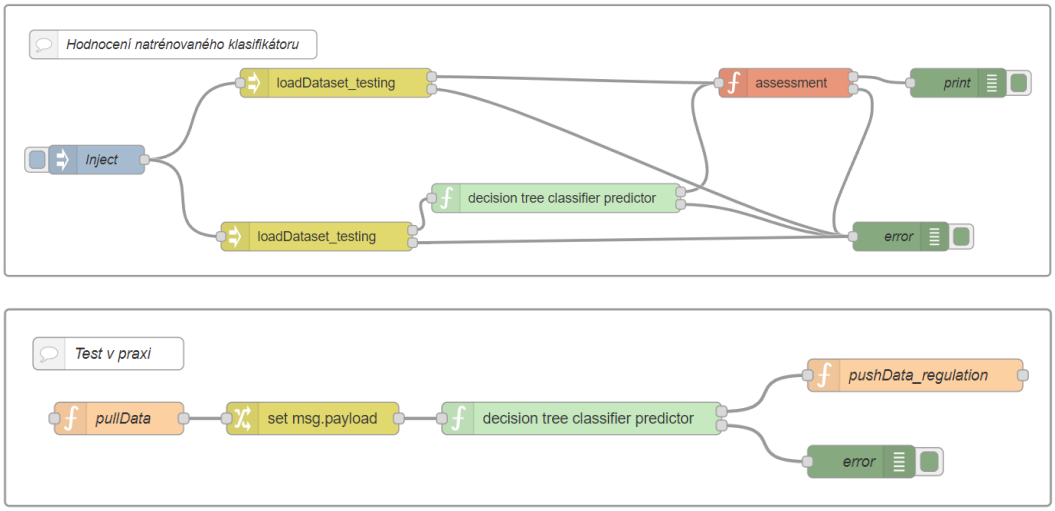
```
msg.payload = msg.payload[1].main.temp;
return msg
```

outsideTempForecast_24h_chart

```
// Inicializace pole pro sérii dat pro chart
let dataSeries = [];

// Určení počtu bodů (8 bodů pro 3h intervaly = 24 hodin)
let points = msg.payload.slice(0, 8);

// Procházení omezeného pole objektů a extrakce teploty a času
points.forEach(item => {
    // Převod UNIX časového razítka na JavaScriptové datum (v milisekundách)
    let date = new Date(item.dt * 1000);

    // Přidání teploty a časového razítka do série dat
    dataSeries.push({
        "x": date.getTime(), // Graf očekává čas v milisekundách
        "y": item.main.temp
    });
});

// Příprava výstupního payloadu pro graf
msg.payload = [{
    "series": ["Teplota"],
    "data": [dataSeries],
    // labels není potřeba, pokud x už obsahuje časová razítka
}];

return msg;
```

Příloha 13: Uzly Node-RED pro trénování ML modelu a jeho hodnocení





Příloha 14: Funkční uzly Node-RED zajišťující logiku za řídícími prvky vizualizace

start/endWard

```
// Získání uložených hodnot
let startTime = flow.get("startTime") || 0;
let endTime = flow.get("endTime") || 1;

if (msg.topic === "startTime") {
    startTime = msg.payload;
    if (startTime >= endTime) {

        endTime = startTime;
        flow.set("endTime", endTime);
        // Odesílá aktualizovanou hodnotu endTime pouze pokud je to nutné
        return { topic: "updateEndTime", payload: endTime };
    }
} else if (msg.topic === "endTime") {
    endTime = msg.payload;
    if (endTime <= startTime) {

        startTime = endTime;
        flow.set("startTime", startTime);
        // Odesílá aktualizovanou hodnotu startTime pouze pokud je to nutné
        return { topic: "updateStartTime", payload: startTime };
    }
}

// Aktualizuje hodnoty ve flow pro budoucí referenci
flow.set("startTime", startTime);
flow.set("endTime", endTime);

return null;
```

tempPlaner

```
// Předpokládáme, že hodnoty ze sliderů jsou v hodinách
let startTime = flow.get("startTime"); // Získání startTime z flow
let endTime = flow.get("endTime"); // Získání endTime z flow
let timestamp = msg.payload; // Předpokládáme, že payload obsahuje timestamp

// Kontrola, zda msg.topic určuje, co zpráva obsahuje
```

```
if (msg.topic === "startTime") {
    flow.set("startTime", msg.payload);
} else if (msg.topic === "endTime") {
    flow.set("endTime", msg.payload);
} else if (msg.topic === "timestamp") {
    // Porovnání timestamp s časem startu a ukončení
    if (startTime !== undefined && endTime !== undefined && timestamp >=
startTime && timestamp <= endTime) {
        // Čas je mezi startem a koncem
        msg.payload = 3;
        msg.topic = "tempPlaner";
    } else {
        // Čas je mimo rozsah
        msg.payload = 0;
        msg.topic = "tempPlaner";
    }
    return msg;
}

return null;
```

timeConvert

```
if ( !msg.timestamp ) msg.timestamp = Math.round(+new Date());

var dt = new Date(msg.timestamp);

msg.payload =dt.getHours();

msg.topic = "timestamp";

return msg;
```

PLC_Control

```
if (msg.topic === "switchPLC") {
    flow.set("switchState1", msg.payload); // Uložení stavu přepínače
} else if (msg.topic === "tempPlaner") {
    flow.set("inputPlaner", msg.payload); // Uložení vstupního čísla
}

// Získání uložených hodnot
var switchState = flow.get("switchState1");
var inputPlaner = flow.get("inputPlaner");

// Logika pro určení výstupní hodnoty
if (switchState === false) {
    if (inputPlaner === 3) {
        msg.payload = 2;
        msg.topic = "tempPlaner";
    }else{
        msg.payload = 0;
        msg.topic = "tempPlaner";
    }
} else {
    msg.payload = inputPlaner; // Pokud je přepínač v pozici 0, použije vstupní
číslo
    msg.topic = "tempPlaner";
}

return msg;
```

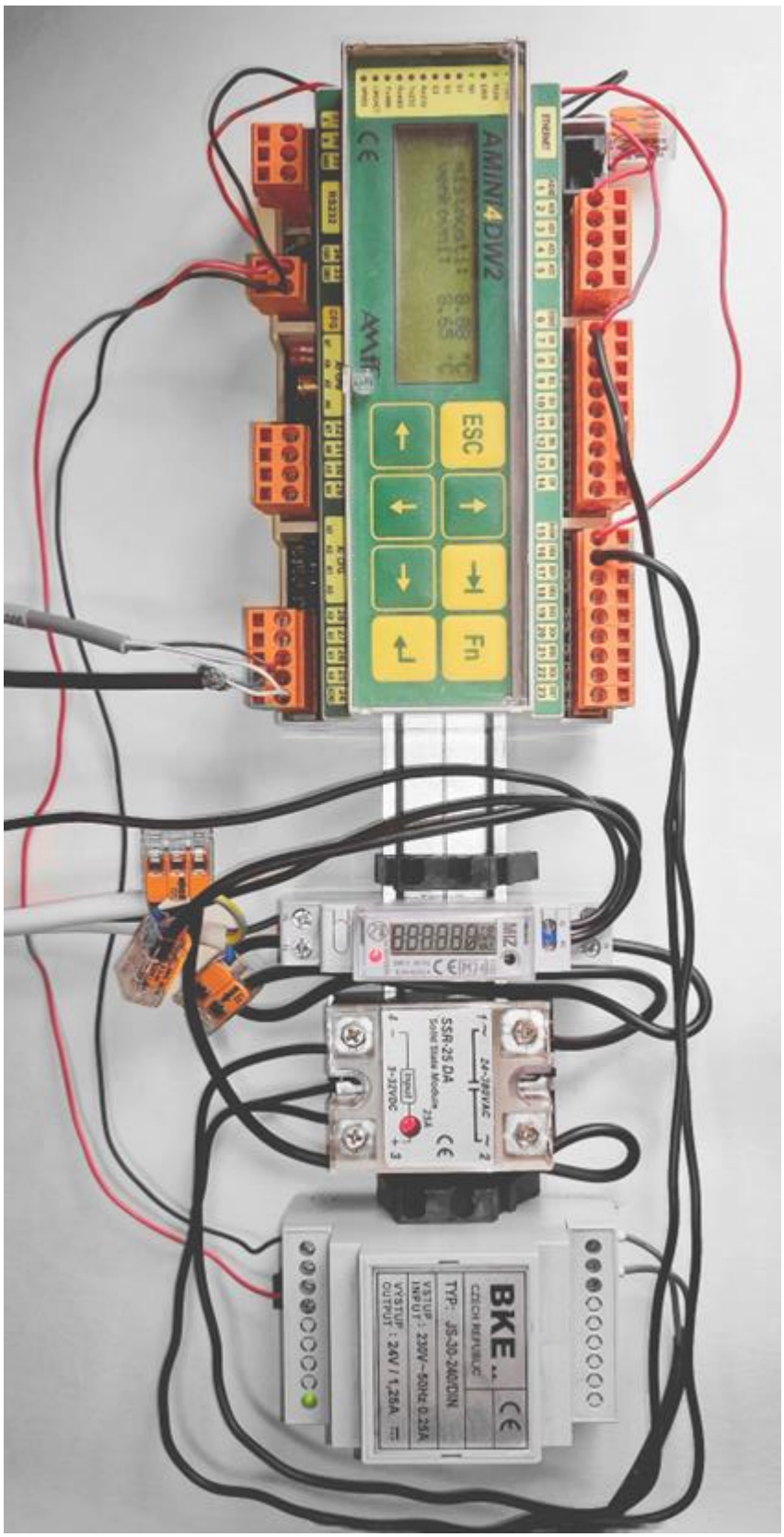
manControl

```
if (msg.topic === "switchMAN") {
    flow.set("switchState2", msg.payload); // Uložení stavu přepínače
} else if (msg.topic === "tempPlaner") {
    flow.set("switchPLC", msg.payload); // Uložení vstupního čísla
}

// Získání uložených hodnot
var switchState = flow.get("switchState2");
var switchPLC = flow.get("switchPLC");

// Logika pro určení výstupní hodnoty
if (switchState === true) {
    msg.payload = 1; // Pokud je přepínač v pozici 1, nastaví výstup na 1
    msg.topic = "tempPlaner";
} else {
    msg.payload = switchPLC; // Pokud je přepínač v pozici 0, použije vstupní
číslo
    msg.topic = "tempPlaner";
}

// Vrácení upravené zprávy
return msg;
```

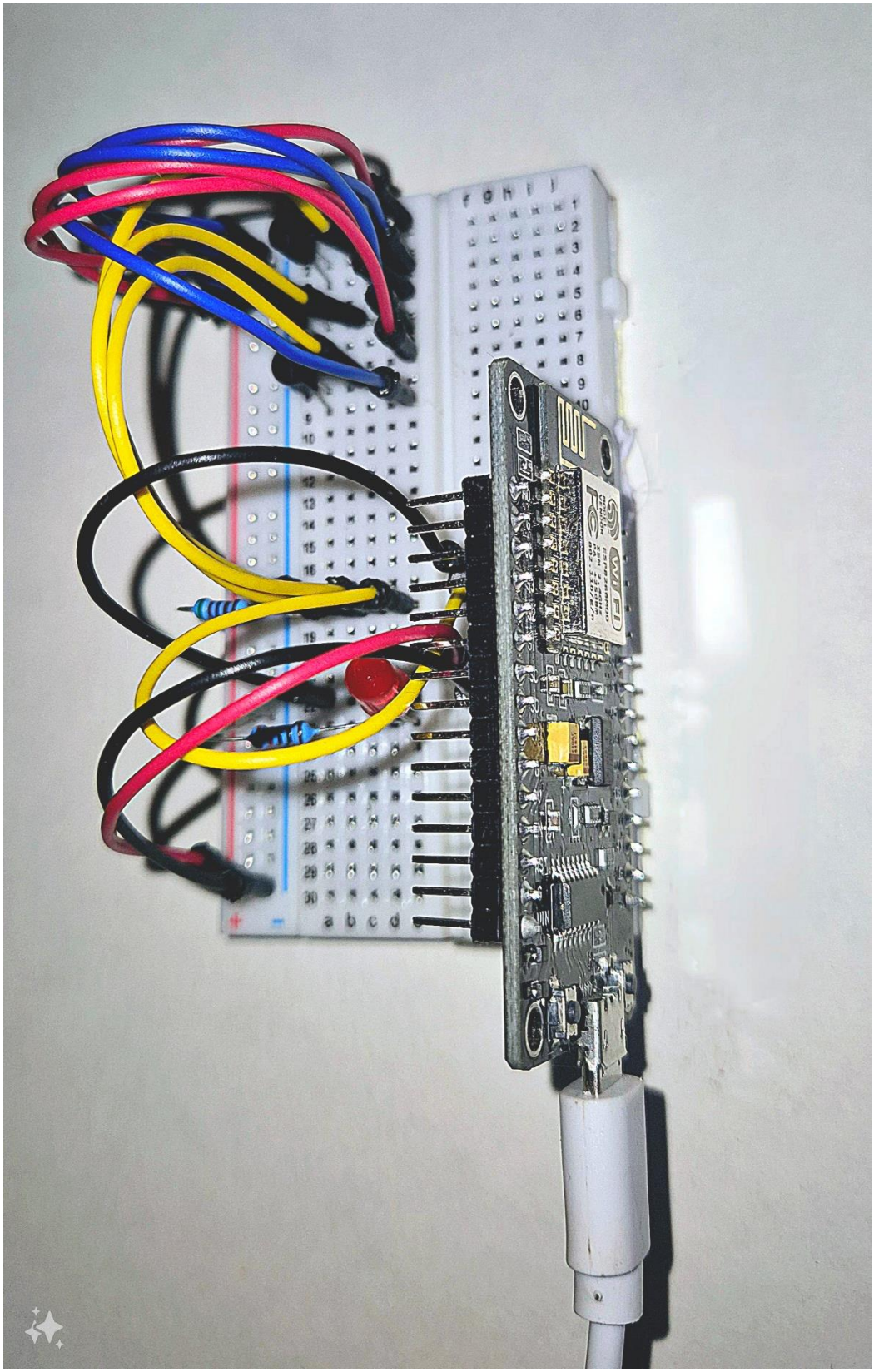
deleteTopic

```
// Odstranění topic z msg objektu
delete msg.topic;
return msg;
```

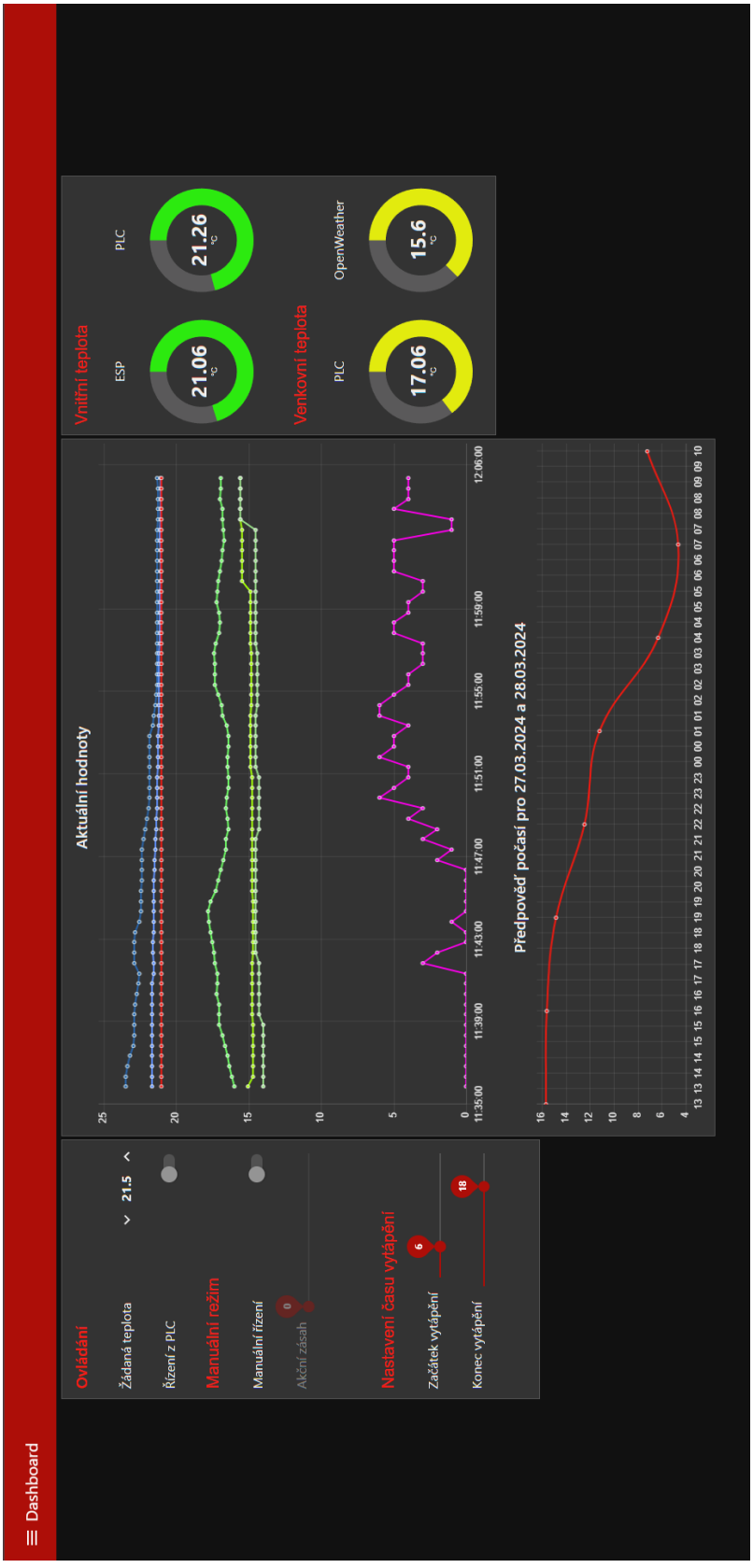
Příloha 15: Fotografie zařízení Amit AMiNi4DW2

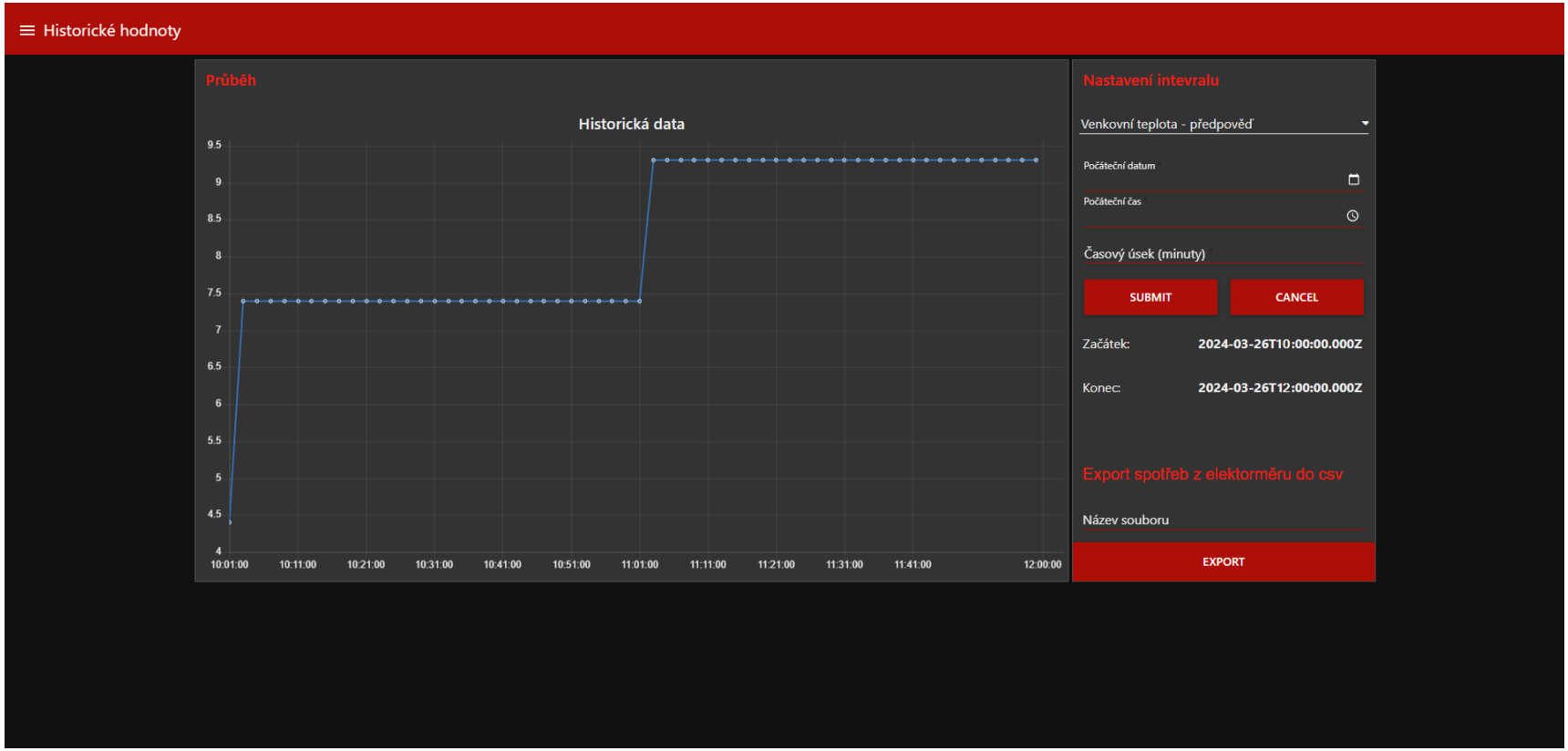


Příloha 16 Fotografie ESP8266 pro měření teploty v místnosti



Příloha 17: Uživatelské rozhraní




Historické hodnoty
Průběh
Historická data
9.5
9
8.5
8
7.5
7
6.5
6
5.5
5
4.5
4
10:01:00
10:11:00
10:21:00
10:31:00
10:41:00
10:51:00
11:01:00
11:11:00
11:21:00
11:31:00
11:41:00
12:00:00
Nastavení intevralu
Venkovní teplota - předpověď
Počáteční datum
Počáteční čas
Časový úsek (minuty)
SUBMIT
CANCEL
Začátek:
2024-03-26T10:00:00.000Z
Konec:
2024-03-26T12:00:00.000Z
Export spotřeb z elektoměru do csv
Název souboru
EXPORT

InfluxDB
Name this Cell
window period: 240000ms
100
80
60
40
20
0
-20
-40
-60
-80
-100
-120
-140
-160
-180
-200
-220
-240
-260
2024-03-26 13:00:00
2024-03-26 16:00:00
2024-03-26 19:00:00
2024-03-26 22:00:00
2024-03-27 01:00:00
2024-03-27 04:00:00
2024-03-27 07:00:00
2024-03-27 10:00:00
_time _value _field _measurement
2024-03-27 10:28:00 94.5 Control action FlowMeter
2024-03-27 10:28:00 25.05 Inside temperature (PLC) FlowMeter
2024-03-27 10:28:00 21 Required temperature FlowMeter
2024-03-27 10:28:00 20.56 Inside temperature (ESP) FlowMeter
2024-03-27 10:28:00 13.05 Ouside Temperature (PLC) FlowMeter
2024-03-27 10:28:00 12.3 Ouside Temperature (Forecast) FlowMeter