Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačních technologií

Diplomová práce

Využití internetu věcí pro solární systémy

David Miller

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ DIPLOMOVÉ PRÁCE

Bc. David Miller

Informatika

Název práce

Využiti internetu věcí pro solární systémy

Název anglicky

IOT for solar systems

Cíle práce

Diplomová práce je tematicky zaměřená na problematiku využiti internetu věcí pro solární systémy.Hlavním cílem práce je implementace internetu věcí pro solární systém.
Dílčí cíle jsou:
- Charakterizovat problematiku internetu věcí a solárních systémů.
- Analyzovat vhodné software a hardware nástroje
- Navrhnout model

Metodika

V teoretické části práce budou studiem literatury získány potřebné znalosti pro zhotovení samotné práce. Analýza současného stavu a návrh nového řešení skládajícího se z jednotlivých kroků. Vytvoření analýzy softwarových a hardwarových nástrojů na základě, které budou vybrány vhodné komponenty k vytvoření systému pro monitorování a ovládání solárního systému. V dalším kroku bude vytvořen model pro solární systém skládající se z centrální jednotky, komponent a softwaru. Na základě syntézy teoretických poznatků a výsledků praktické časti budou formulovány závěry práce

Doporučený rozsah práce

60-80 stran

Klíčová slova

IOT, automatizace, solární systémy, webová aplikace, monitoring

Doporučené zdroje informací

DAYLEY, Brad, Brendan DAYLEY a Caleb DAYLEY. Node.js, MongoDB and Angular web development. Second edition. Boston, MA: Addison-Wesley, [2017]. ISBN 9780134655536.

Dimitrios Serpanos, Marilyn Wolf. Internet-of-Things (IoT). Švýcarsko : Springer International Publishing, 2018. ISBN 978-3-319-69714-7.

Martin Malý. HRADLA, VOLTY, JEDNOČIPY Úvod do bastlení. CZ.NIC, z. s. p. o. (2017). ISBN 978-80-88168-23-2

Pinker, J. Mikroprocesory a mikropočítače. BEN – technická literatura (2011). ISBN 80-7300-110-1

Předběžný termín obhajoby

2023/24 LS – PEF

Vedoucí práce

Ing. Michal Stočes, Ph.D.

Garantující pracoviště

Katedra informačních technologií

Elektronicky schváleno dne 4. 7. 2023

doc. Ing. Jiří Vaněk, Ph.D.
Vedoucí katedry

Elektronicky schváleno dne 3. 11. 2023

doc. Ing. Tomáš Šubrt, Ph.D.
Děkan

V Praze dne 19. 02. 2024

Čestné prohlášení

Prohlašuji, že svou diplomovou práci "Využití internetu věcí pro solární systémy" jsem vypracoval(a) 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(ka) uvedené diplomové práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 30.3.2024 _____

Poděkování

Rád bych touto cestou poděkoval vedoucímu práce panu Ing. Michalu Stočesovi, Ph. D. za vedení mé práce a firmě Intronix s.r.o. za možnost uskutečnit tuto diplomovou práci a poskytnutí cenných rad a informací z praxe.

Využití internetu věcí pro solární systémy

Abstrakt

Diplomová práce se věnuje implementaci IoT pro fotovoltaické systémy. Toho je docíleno za použití databáze MongoDB, REST API serveru a mikrokontroleru ESP32.

V teoretické části jsou popsány veškeré použité technologie potřebné pro využití internetu věcí pro solární systémy.

Vlastní práce se věnuje vytvoření softwaru pro implementaci IoT pro solární systéme. Zprvu byly nutné vytvořit IoT model pro lepší pochopení problému. Dále je popsáno nastavení databáze MongoDB, které zahrnuje zabezpečení databáze a komunikace s ní. Pro komunikace mikrokontroleru, databáze a webu slouží REST API server, jehož kód je zhotoven v programovacím jazyce Python za pomocí frameworku Flask. Samotný kód mikrokontroleru ESP32 je naprogramován v jazyce C++, který má na starosti sbírání dat ze senzorů a řízení výstupních zařízení. Pro zobrazení hodnot senzorů a ovládání systému je zhotovena webová aplikace. Průběh vytváření systému je podrobně popsán a následně testován.

Závěrem práce je popsán budoucí vývoj systému, kde je rozebírán přesun REST API a webové aplikace na soukromý server. Také je zde zahrnut vývoj desky plošného spoje zahrnující modul ESP32 a další potřebné komponenty.

Klíčová slova: IoT, fotovoltaické elektrárny, ESP32, automatizace, API, dashboard, programování

IOT for solar systems

Abstract

The thesis focuses on the implementation of IoT for photovoltaic systems. This is achieved using MongoDB database, REST API server and ESP32 microcontroller.

The theoretical part describes all the technologies used and required for the use of IoT for solar systems.

The actual work is devoted to the development of software for the implementation of IoT for solar systems. Initially, it was necessary to create an IoT model to better understand the problem. Next, the setup of the MongoDB database is described, which includes the security of the database and communication with it. A REST API server is used to communicate between the microcontroller, database and the web, whose code is made in Python programming language using Flask framework. The ESP32 microcontroller code itself is programmed in C++, which is responsible for collecting sensor data and controlling output devices. A web application is made to display the sensor values and control the system. The process of creating the system is described in detail and then tested.

Finally, the future development of the system is described, where the migration of the REST API and the web application to a private server is discussed. Also included is the development of a circuit board incorporating the ESP32 module and other necessary components.

Keywords: IoT, photovoltaic power plants, ESP32, automation, API, dashboard, programming

Obsah

1 Úvod **7**

2 Cíl práce a metodika **8**

2.1 Cíle práce 8

2.2 Metodika 8

3 Teoretická východiska **9**

3.1 Internet věcí 9

3.2 Fotovoltaické elektrárny 10

3.3 Databáze 11

3.3.1 Databáze časových řad 11

3.3.2 JSON 12

3.4 API 13

3.4.1 REST API 13

3.5 Mikrokontrolery 14

3.5.1 ESP32 15

3.5.2 Arduino 16

3.6 Mikroprocesory 17

3.6.1 Raspberry Pi Zero W 18

3.7 Programovací jazyky 19

3.7.1 Python 19

3.7.2 Node.js 20

3.7.3 HTML 21

3.7.4 CSS 21

3.7.5 Javascript 22

3.7.6 Frameworky 22

3.7.7 Knihovny 23

3.8 Vývojová prostředí 24

3.8.1 Visual studio Code 25

3.8.2 PyCharm 25

3.9 Wireframe 26

3.10 Saatyho metoda 26

3.11 cURL 26

4 Vlastní práce **28**

4.1 Analýza trhu 28

4.1.1 Wattrouter Mx 28

4.1.2 GreenBonO 28

4.2 Analýza požadavků na systém 29

4.3 Analýza hardwarových nástrojů 29

4.3.1 Stanovení požadavků na hardware ... 30
4.3.2 Prvotní výzkum a sběr kandidátů ... 30
4.3.3 Vyhodnocení analýzy ... 31
4.3.4 Saatyho metoda ... 31
4.3.5 Potvrzení analýzy ... 34
4.4 Analýza softwarových nástrojů ... 34
4.4.1 Programovací jazyky ... 34
4.4.2 Vývojová prostředí (IDE) ... 34
4.4.3 Databáze ... 35
4.4.4 Vyhodnocení analýzy ... 35
4.5 Návrh systému ... 36
4.5.1 Popis zapojení ... 36
4.5.2 IoT model ... 36
4.5.3 REST API server ... 37
4.5.4 ESP32 ... 38
4.5.5 Databáze ... 38
4.6 Program REST API serveru ... 40
4.6.1 Popis kódu ... 40
4.7 Program ESP32 ... 44
4.7.1 Popis kódu ... 44
4.8 Testování REST API ... 52
4.9 Webová aplikace ... 53
4.9.1 Návrh webové aplikace ... 53
4.9.2 Program webové aplikace ... 55

5 Výsledky a diskuse ... 59
5.1 Shrnutí ... 59
5.2 Budoucí vývoj ... 59
5.2.1 Migrace na server ... 59
5.2.2 Vývoj softwaru ... 59
5.2.3 Vývoj hardwaru ... 60

6 Závěr ... 61

7 Bibliografie ... 62

8 Seznam obrázků, tabulek, grafů a zkratek ... 65
8.1 Seznam použitých zkratek ... 65

9 Přílohy ... 67
9.1 Příloha 1 – program ... 67

1 Úvod

V roce 2020 bylo v České republice instalováno 6293 nových fotovoltaických elektráren, v roce 2021 to bylo o 3028 více, tedy 9321 a i v letošním roce poptávka stále stoupá. Při tomto trendu lze očekávat enormní nárust poptávky po technologiích v tomto odvětví.

Nedílnou součástí fotovoltaických elektráren jsou také systémy pro řízení a zefektivnění činnosti těchto elektráren. Jedním z těchto zařízení je i wattrouter, který má za úkol dokonale vytěžovat přebytky elektrické energie a tím výrazně snížit dobu návratnosti investice do fotovoltaické elektrárny. Některé wattroutery dokonce umožňují uživateli ovládat fotovoltaickou elektrárnu skrze aplikaci s dashboardem.

Tato práce se zabývá vývojem softwaru pro wattrouter, který je vyvíjen společností Intronix s.r.o. Tento software se skládá ze čtyř částí: mikrokontroleru pro sběr hodnot ze senzorů, databáze časových řad pro ukládání dat ze senzorů, REST API serveru pro komunikaci mezi mikrokontrolerem a databází a webovou aplikací s dashboardem.

V diplomové práci bude využito dostupných technologií a open source nástrojů k vytvoření vlastního systému pro implementaci IoT pro solární systémy. Teoretické část práce popisuje a definuje všechny použité technologie, které budou použity k realizaci. Vlastní práce se zabývá analýzou softwarových a hardwarových komponent vhodných pro implementaci internetu věcí pro solární systémy. Také zahrnuje naprogramování REST API serveru, webové aplikace, kódu mikrokontroleru a nastavení databáze časových řad MongoDB. REST API server je naprogramován v jazyce Python za pomocí frameworku Flask a dočasně umístěn na Google cloud pro možné testování. Kód mikrokontroleru je naprogramován v jazyce C++ a má za úkol sbírat data ze senzorů a ovládat výstupy systému. Webová aplikace obsahuje dashboard s třemi grafy, předpovědí počasí a posuvník určující interval nočního proudu.

Paralelně s vývojem softwaru vzniká i hardwarová část wattrouteru, kterou má na starosti firma Intronix s.r.o. To zahrnuje zejména návrh desky plošných spojů obsahující mikrokontroler ESP32 a ostatní komponenty zajišťující správnou funkci. Tak vznikne kompletní systém, který lze nazývat wattrouterem.

Motivací pro výběr tohoto tématu byla absence řídící jednotky pro fotovoltaické systémy ze strany společnosti Intronix. Společnost se chystá koncem roku 2024 proniknout na český trh s vlastním řešením fotovoltaických elektráren, proto bylo nezbytné tuto jednotku sestrojit.

2 Cíl práce a metodika

2.1 Cíle práce

Diplomová práce je tematicky zaměřená na problematiku využiti internetu věcí pro solární systémy.Hlavním cílem práce je implementace internetu věcí pro solární systém.
Dílčí cíle jsou:
- Charakterizovat problematiku internetu věcí a solárních systémů.
- Analyzovat vhodné software a hardware nástroje
- Navrhnout model

2.2 Metodika

V teoretické části práce budou studiem literatury získány potřebné znalosti pro zhotovení samotné práce. Analýza současného stavu a návrh nového řešení skládajícího se z jednotlivých kroků. Vytvoření analýzy softwarových a hardwarových nástrojů na základě, které budou vybrány vhodné komponenty k vytvoření systému pro monitorování a ovládání solárního systému. V dalším kroku bude vytvořen model pro solární systém skládající se z centrální jednotky, komponent a softwaru. Na základě syntézy teoretických poznatkůa výsledků praktické časti budou formulovány závěry práce

3 Teoretická východiska

V této části práce budou popsána teoretická východiska potřebná ke zpracování vlastní práce.

3.1 Internet věcí

Internet věcí (IoT) představuje rozsáhlou síť propojených fyzických přístrojů schopných vzájemné komunikace, což zahrnuje výměnu dat a informací. Tato připojená zařízení jsou vybavena různými technologiemi jako čipy a senzory, což jim umožňuje odesílat a přijímat data. Rozsah IoT zařízení je široký a zahrnuje například inteligentní domácí systémy, čipy, senzory a medicínské přístroje. [1] [2]

Klíčovým přínosem IoT je jeho schopnost propojit rozličná zařízení a systémy, což umožňuje jejich automatizaci a efektivnější řízení. Příkladem může být inteligentní termostat, který automaticky upravuje domácí teplotu podle aktuálního počasí nebo přítomnosti osob v domácnosti, nebo inteligentní systémy zavlažování, které se řídí meteorologickými daty a podmínkami půdy, aby efektivně zavlažovaly zahrady. [2]

IoT nachází uplatnění také v průmyslovém sektoru, kde napomáhá optimalizaci výrobních procesů, zvyšuje bezpečnost a efektivitu v dopravě a logistice a najde využití v dalších oblastech. Dále umožňuje průmyslovým podnikům monitorovat stav jejich zařízení, což může předcházet poruchám a výpadkům ve výrobě. [3]

Pro fungování IoT je nezbytná řada technologií a standardů, včetně bezdrátového připojení, cloudových služeb, analýzy velkých dat. Rozvoj a implementace IoT vyžaduje spolupráci napříč různými odvětvími, včetně výrobců hardwaru, poskytovatelů cloudových platforem a vývojářů softwaru. [3]

V poslední době se Internet věcí (IoT) vyvíjí nebývalou rychlostí a předpokládá se, že tento vzestupný trend bude pokračovat i v budoucnu. Předpokládá se masivní nárůst počtu zařízení připojených k internetu, což otevře nové příležitosti jak pro společnosti, tak pro individuální uživatele. IoT má potenciál nalézt uplatnění v široké škále sektorů, včetně inteligentních domovů a měst, automobilového průmyslu, zdravotní péče a vzdělávání, a očekává se, že bude mít klíčovou roli při řešení světových výzev, jako je boj proti změně klimatu a efektivnější správa zdrojů. [2] [3]

S rozvojem technologií, jako je například 5G síť, která nabídne rychlejší a spolehlivější připojení pro IoT zařízení, lze očekávat rozšíření využití IoT v zákaznických službách, včetně automatizace domácnosti nebo chytrého měření spotřeby energie. Také se očekává nástup nových aplikací IoT, které zjednoduší pracovní procesy a zlepší kvalitu života. [1]

Zajištění bezpečnosti a ochrany soukromí v rámci IoT bude klíčové. Anticipuje se vývoj nových bezpečnostních technologií a metod, které ochrání před kybernetickými útoky a zamezí neoprávněnému přístupu k osobním datům. [1] [3]

3.2 Fotovoltaické elektrárny

Fotovoltaické elektrárny jsou zařízení, která přeměňují sluneční energii přímo na elektrickou energii pomocí fotovoltaických článků, které jsou základním stavebním prvkem fotovoltaických panelů. Tyto elektrárny využívají polovodičové materiály (nejčastěji křemík), které při pohlcení slunečního světla uvolňují elektrony a tím vytvářejí elektrický proud. Fotovoltaické elektrárny lze instalovat v malém měřítku na střechách domů nebo budov, jako jsou kanceláře a školy, nebo v rozsáhlém měřítku ve formě solárních farem či parků na volném prostranství. [4]

Princip Funkce
Když foton slunečního světla narazí na fotovoltaický článek, může uvolnit elektron z jeho atomu. Elektrony se pak pohybují přes materiál a vytvářejí elektrický proud. Tento jev se nazývá fotovoltaický efekt. Fotovoltaické panely jsou obvykle spojeny v sérii nebo paralelně, aby se dosáhlo požadovaného napětí a proudu, a jsou připojeny k invertoru, který přeměňuje stejnosměrný proud (DC) vyrobený panely na střídavý proud (AC), který lze použít v domácnostech nebo předat do elektrické sítě. [4]

Wattroutery
Watt Router představuje inovativní řešení pro správu a optimalizaci energetické spotřeby v domácnostech, komerčních objektech i v průmyslu. Jeho hlavním úkolem je efektivně řídit distribuci elektrické energie mezi různými zdroji a spotřebiči s cílem maximalizovat využití energie, zvýšit energetickou efektivitu a snížit náklady na energie. Zařízení je zvláště užitečné v prostředích, kde je k dispozici energie z obnovitelných zdrojů, jako jsou solární panely nebo větrné turbíny. Umožňuje uživatelům prioritně využívat energii vyrobenou z těchto zdrojů před spotřebou energie z tradiční elektrické sítě.

Jednou z klíčových funkcí Watt Routeru je jeho schopnost automaticky přepínat mezi energií ukládanou v bateriích a energií získanou přímo z obnovitelných zdrojů v závislosti na aktuálním výkonu a potřebách spotřeby. Díky tomu může efektivně řídit spotřebu energie a předejít špičkám ve spotřebě, které by jinak vedly k vyšším nákladům. Tento systém je vybaven i prediktivní analýzou, která umožňuje anticipovat budoucí spotřebu energie a přizpůsobit distribuci energie tak, aby byla co nejefektivnější.

Moderní Watt Routery jsou navrženy s ohledem na uživatelský komfort a nabízejí pokročilá uživatelská rozhraní a aplikace pro monitorování a dálkové ovládání energetické spotřeby. Umožňují uživatelům nejen sledovat, kolik energie jejich domácnost nebo firma spotřebuje, ale také jim dávají kontrolu nad tím, jak a kdy se energie využívá. Navíc se tyto systémy často integrují s dalšími prvky inteligentních domácností, což umožňuje komplexní správu a automatizaci energetické spotřeby v rámci celého objektu.

Celkově Watt Router představuje klíčový prvek pro dosažení energetické soběstačnosti a optimalizaci spotřeby, čímž se stává neocenitelným nástrojem pro každého, kdo hledá způsoby, jak snížit své energetické náklady a zároveň přispět k ochraně životního prostředí. [5]

3.3 Databáze

Databáze je uspořádaný soubor strukturovaných informací nebo dat, obvykle uložených elektronicky v počítačovém systému. Databáze je obvykle řízena systémem pro správu databází (DBMS). Data a DBMS spolu s aplikacemi, které jsou s nimi spojeny, se společně označují jako databázový systém, často zkráceně jen databáze. [6] [7]

Data v nejběžnějších typech dnes provozovaných databází jsou obvykle modelována v řádcích a sloupcích v řadě tabulek, aby bylo možné efektivně zpracovávat data a provádět dotazy. K datům lze pak snadno přistupovat, spravovat je, upravovat, aktualizovat, kontrolovat a organizovat. Většina databází používá pro zápis a dotazování dat strukturovaný dotazovací jazyk (SQL). [6] [7]

3.3.1 Databáze časových řad

Databáze časových řad je databáze optimalizovaná pro data s časovými značkami nebo časové řady. Data časových řad mohou být měření nebo události, které jsou sledovány, měněny a agregovány v čase. Může jít o metriky serverů, sledování výkonu aplikací, síťová data, data ze senzorů, události a další analytická data. [8]

Databáze časových řad je vytvořena speciálně pro zpracování metrik a událostí nebo měření, které jsou opatřeny časovou značkou. Databáze časových řad je optimalizována pro měření změn v čase. [9] [7]

InfluxDB
InfluxDB je databáze časových řad specificky navržená pro rychlé čtení a zápis časově řazených dat, což ji činí ideální pro aplikace monitoringu, telemetrie, real-time analytiky a jiné časově kritické aplikace. Tato databáze je optimalizována pro vysokou dostupnost, zpracování velkých objemů dat a udržení vysokého výkonu. Klíčem k efektivnímu zpracování velkého množství datových bodů generovaných v pravidelných časových intervalech je její jedinečný datový model a indexace založená na čase. InfluxDB podporuje ukládání dat v sériích, kde každá série reprezentuje množinu datových bodů spojených s určitým měřením, sadou tagů a časovými razítky. [8]

Všechna data v InfluxDB jsou implicitně indexována podle času, což umožňuje velmi rychlé dotazování na základě časových rozmezí. Díky flexibilnímu schématu databáze, které nepožaduje předem definované schéma pro data, mohou uživatelé dynamicky přidávat tagy a políčka k datovým bodům. Toto je zvláště výhodné v situacích, kdy se struktura dat může rychle měnit. [8]

InfluxDB nabízí výkonný dotazovací jazyk, InfluxQL, který umožňuje provádět složité dotazy na data, včetně agregací, filtrů podle tagů a časových rozmezí. Databáze je navržena pro snadnou integraci s dalšími nástroji a platformami, včetně populárních nástrojů pro monitorování a vizualizaci jako je Grafana, a podporuje klienty pro mnoho programovacích jazyků. [8]

Pro zajištění vysoké dostupnosti a škálovatelnosti může být InfluxDB nasazena v klastru, což umožňuje spolehlivé zpracování dat i v případě velkých objemů dat nebo vysoké

poptávky na čtení a zápis. Díky pokročilým technikám komprese databáze efektivně ukládá velké objemy časových dat s minimálním využitím diskového prostoru. [8]

MongoDB
MongoDB je non-SQL databáze, což znamená, že ukládá data ve flexibilních dokumentech typu JSON, z toho plyne, že pole se mohou v jednotlivých dokumentech lišit a struktura dat se může v průběhu času měnit. Model dokumentu se mapuje na objekty v kódu aplikace, a to usnadňuje práci s daty. [9] [7]

MongoDB Atlas
MongoDB Atlas je multicloudová databázová služba, která zjednodušuje nasazení a správu databází. [9] [7]

MongoDB Compass
MongoDB Compass je výkonné grafické uživatelské rozhraní (GUI) pro dotazování, agregaci a analýzu dat MongoDB ve vizuálním prostředí. Compass je k dispozici zdarma a se zdrojovými kódy a lze jej spustit v systémech MacOS, Windows a Linux. [9] [7]

3.3.2 JSON

JSON (JavaScript Object Notation) je lehký formát dat, který slouží k výměně dat mezi serverem a klientem nebo mezi různými částmi aplikací. JSON je jednoduchý a srozumitelný, což jej činí oblíbeným výběrem pro přenos dat ve webovém prostředí, ale může být používán i v jiných kontextech. [7] [10]

Struktura dat: JSON data jsou strukturována ve formě párových klíč-hodnota, podobně jako objekty v JavaScriptu. Data mohou být ve formě objektů (sady klíč-hodnota), pole (seznam hodnot), čísel, řetězců, booleovských hodnot nebo hodnoty [7] [10]

Základní pravidla pro formát JSON

Datové typy: JSON podporuje řetězce (řetězce znaků), čísla, logické hodnoty (true/false), pole, objekty a hodnotu null.

Struktura: Data v JSON jsou organizována do dvou základních struktur: objekty a pole. Objekty jsou kolekce párových klíč-hodnota, zatímco pole jsou seznamy hodnot. Obě struktury lze kombinovat k vytváření složitějších datových struktur.

Souběžnost s jazyky: JSON je nezávislý na programovacím jazyce, což znamená, že může být snadno použit s různými programovacími jazyky. Většina moderních programovacích jazyků poskytuje nástroje pro práci s JSON.

Čitelnost: JSON je snadno čitelný pro lidi a lze ho také editovat ručně. Data jsou strukturována pomocí závorek {}, hranatých závorek [], dvojteček :, a čár. [10]

3.4 API

API (rozhraní pro programování aplikací) je sada definicí a nástrojů, které umožňují různým softwarovým aplikacím komunikovat a interagovat mezi sebou. Je to jakýsi most, který umožňuje jedné části softwaru využívat funkce nebo služby jiné části bez nutnosti znalosti interní implementace.

API může být ve formě sad funkcí, procedur, tříd nebo protokolů, které jsou k dispozici pro programátory k integraci do svých vlastních aplikací. Existují různé typy API, včetně webových API, které umožňují komunikaci mezi webovými službami, a knihoven API, které poskytují sadu funkcí nebo metod pro interakci s určitým programem či systémem.

API umožňuje aplikacím získávat data, provádět operace a integrovat se s dalšími aplikacemi a službami. API je klíčový prvek vývoje moderních softwarových aplikací, protože umožňuje efektivní spolupráci mezi různými komponentami a systémy. [11]

3.4.1 REST API

REST API (Representational State Transfer Application Programming Interface) je architektonický styl, který klade důraz na jednoduchost, srozumitelnost, a efektivní komunikaci mezi komponentami distribuovaných systémů, často v kontextu webových služeb. [11]

Zdroje (Resources)
Data nebo služby jsou reprezentovány jako zdroje, které mají jednoznačný identifikátor (URI – Uniform Resource Identifier). Například, pokud máme databázi knih, každá kniha by mohla být zastoupena jako zdroj s unikátním URI, jako např. /books/123. [11]

Reprezentace dat (Representation)
Data jsou reprezentována ve vhodném formátu, často pomocí JSON nebo XML. Klient může získávat nebo modifikovat data pomocí HTTP metod jako GET, POST, PUT, nebo DELETE. [11]

Statelness (Bezstavovost)
Každý požadavek klienta na server obsahuje veškeré informace potřebné k pochopení a zpracování požadavku. Server neudržuje žádný stav o klientovi. [11]

HTTP Metody

GET: Získání informací o zdroji.
POST: Vytvoření nového zdroje.
PUT: Aktualizace existujícího zdroje.
DELETE: Smazání zdroje. [12] [7]

Rozhraní (Interface)
Rozhraní mezi klientem a serverem je jasně definován. Klienti komunikují se serverem pomocí standardních HTTP metod a obdržených odpovědí. REST API je běžně používáno ve webovém vývoji pro vytváření flexibilních systémů, kde různé části aplikace nebo různé aplikace mohou efektivně komunikovat pomocí standardních protokolů HTTP. [12] [7]

Běžné stavové kódy odpovědí HTTP

- 200 Úspěch
- 307 Dočasné přesměrování. Prohlížeč se pokusí o přesměrování pomocí hlavičky Location
- 404 Nenalezeno
- 503 Služba je nedostupná
- 500 Vnitřní chyba serveru
- 501 Neimplementováno
- 400 Špatný požadavek. Nastane v okamžiku, kdy webová aplikace odešle falešný požadavek
- 401 Neoprávněné použití. Nastává, pokud uživatel není přihlášen.
- 403 Zakázán, pokud by nepomohlo ani přihlášení
- 550 Oprávnění odepřeno Není povoleno provést požadavek. [12] [7]

3.5 Mikrokontrolery

Mikrokontroler je integrovaný obvod, který obsahuje procesor (CPU), paměť a periferní zařízení, vše v jednom čipu. Tyto součástky jsou navrženy tak, aby poskytovaly výpočetní a řídící funkce pro různé elektronické systémy. [13] [14]

Procesor (CPU): Mikrokontrolery mají vestavěný procesor, který provádí instrukce uložené v jejich paměti. Typickými architekturami jsou například RISC nebo CISC.

Programová paměť (Flash): Ukládá programový kód, který mikrokontroler vykonává.
Datová paměť (RAM): Slouží pro ukládání dočasných dat a proměnných během provádění programu.
Periferní zařízení:

Komunikační porty: UART, SPI, I2C pro komunikaci s dalšími zařízeními.

GPIO (General-Purpose Input/Output): Slouží k připojení a ovládání různých vnějších zařízení.

Časovače a čítače: Pro generování časových intervalů nebo měření frekvence.

Analogové a digitální převodníky: Pro měření analogových signálů.

PWM (Pulse-Width Modulation): Pro generování pulzně šířkově modulovaných signálů.
Vývojové nástroje:

Vývojové desky a moduly: K vývoji a testování kódu před jeho implementací na konkrétním zařízení.

Kompilátory a vývojová prostředí: K vytváření a ladění kódu pro mikrokontrolery.
Programování:

Mikrokontrolery se používají v široké škále aplikací, včetně spotřební elektroniky, průmyslové automatizace, lékařských zařízení, automobilové elektroniky a mnoha dalších oblastí.
Popis výše je obecný a existuje mnoho různých typů mikrokontrolerů s různými schopnostmi, závislými na konkrétním použití a požadavcích projektu. [14] [13]

3.5.1 ESP32

ESP32 je výkonný mikrokontroler vyvinutý firmou Espressif Systems, který poskytuje širokou škálu funkcí pro různé aplikace v oblasti internetu věcí (IoT), průmyslové automatizace a dalších oblastí. Tento článek se zaměří na klíčové vlastnosti, výhody a použití ESP32 v různých odvětvích. [13] [14]

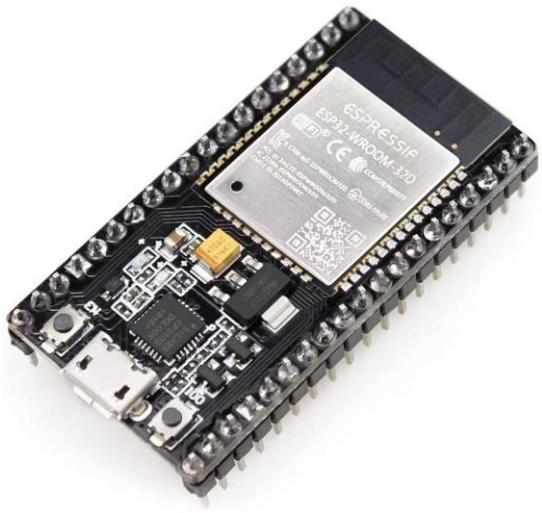
Technické specifikace
ESP32 kombinuje vysoký výkon, nízkou spotřebu energie a bohatou sadu periférií, což ho činí ideálním pro mnoho různých aplikací.

ESP32 je vybavený výkonným dvoujádrovým procesorem s frekvencí až 240 MHz, což umožňuje zpracování složitých úloh a aplikací s vysokými nároky na výkon. Vestavěná podpora Wi-Fi a Bluetooth umožňuje snadné připojení k bezdrátovým sítím a komunikaci s jinými zařízeními, což je klíčové pro aplikace IoT. ESP32 je navržen s ohledem na nízkou spotřebu energie, což umožňuje dlouhou výdrž baterií a využití v energeticky úsporných aplikacích. Wi-Fi a Bluetooth obsahuje ESP32 také řadu dalších periférií, jako jsou například rozhraní UART, SPI, I2C, GPIO piny, ADC a DAC, což umožňuje snadnou integraci s různými senzory a zařízeními. [14] [13]

Použití ESP32
IoT aplikace: Díky podpoře Wi-Fi a Bluetooth je ESP32 ideální pro různé aplikace v oblasti IoT, jako jsou chytrá domácnost, sledování a monitorování, a mnoho dalších.

Průmyslová automatizace: Díky vysokému výkonu a bohaté sadě periférií může být ESP32 použit pro řízení a monitorování průmyslových zařízení a procesů.

Spotřebiče a elektronika: ESP32 může být integrován do spotřebičů a elektroniky pro přidání bezdrátové konektivity a inteligentních funkcí. [14] [13]



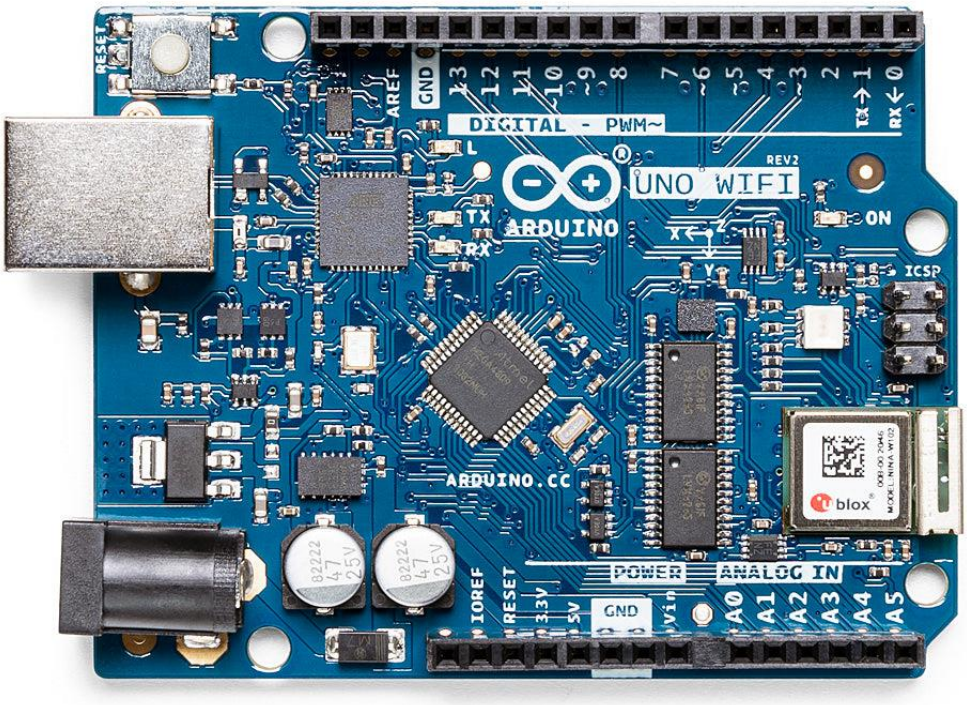
Obrázek 1 mikrokontroler ESP32 (zdroj: https://allegro.cz/nabidka/esp-32-wroom-wifi-bluetooth-esp32-nodemcu-arduino-12134563733)

3.5.2 Arduino

Arduino je otevřená vývojová platforma, která je založena na snadno použitelné hardwarové i softwarové stránce. Tato platforma je navržena s cílem umožnit přístup k vývoji aplikací s mikrokontrolery. V srdci platformy je mikrokontrolerová deska, typicky z rodiny Atmel AVR jako ATmega328, která nabízí digitální a analogové vstupní/výstupní piny, USB rozhraní pro programování a sériovou komunikaci, a někdy i další komunikační rozhraní jako SPI a I2C. Desky lze napájet přes USB, externí zdroj napájení, nebo baterie, což umožňuje jejich použití v různých projektech a aplikacích. [15] [16]

Softwarová stránka Arduina zahrnuje integrované vývojové prostředí (IDE), které je dostupné pro různé operační systémy a slouží k psaní, kompilaci a nahrávání programů do hardwaru. Programovací jazyk je založen na Wiring, což je rozšíření C/C++ a zahrnuje knihovny usnadňující ovládání hardwaru. Toto prostředí spolu s jednoduchým programovacím jazykem umožňuje rychlé a efektivní prototypování nápadů a projektů. [15] [16]

Technické Specifikace
Základní Arduino desky jsou založeny na 8-bitových AVR mikrokontrolerech od Atmelu, jako je ATmega328, ATmega2560, mezi jinými, ačkoliv existují varianty využívající 32-bitové ARM procesory. Tyto mikrokontrolery poskytují dostatečný výkon pro řízení různých senzorů a komunikačních modulů. Arduino desky typicky obsahují digitální I/O piny (včetně několika, které podporují PWM výstup), analogové vstupní piny, USB připojení pro programování a sériovou komunikaci, a externí napájecí konektor. [15] [16]



Obrázek 2 Arduino UNO WiFi (zdroj: https://store.arduino.cc/products/arduino-uno-wifi-rev2)

Arduino UNO WIFI
Arduino Uno WiFi je mikrokontrolerová deska, která rozšiřuje možnosti tradičního Arduino UNO o bezdrátovou konektivitu. Toto zařízení je ideální pro projekty a aplikace, kde je potřeba připojení k internetu nebo lokální síti bez použití externích Wifi modulů. [15] [16]

3.6 Mikroprocesory

Mikroprocesory, často označované jako "mozky" počítačů a elektronických zařízení, jsou integrované obvody (IC), které slouží jako centrální procesorové jednotky (CPU). Tyto komponenty představují základ moderních výpočetních systémů, od osobních počítačů po mobilní telefony a rozsáhlé serverové farmy. První mikroprocesor, Intel 4004, byl uveden na trh v roce 1971 a od té doby technologie mikroprocesorů prošla obrovským vývojem. [17] [18] [19]

Jádro mikroprocesoru spočívá v jeho schopnosti provádět aritmetické a logické operace, čímž řídí tok dat mezi různými komponentami počítače. Mikroprocesory zpracovávají instrukce z paměti, komunikují s periferními zařízeními a spravují datové toky, což umožňuje komplexní interakce, na kterých je založen software. [18] [19]

S vývojem výrobních technologií se zvýšil výkon mikroprocesorů, což umožnilo integraci více funkcí a vyšší rychlosti zpracování při současném snižování spotřeby energie. Současné mikroprocesory mohou obsahovat více jader, což umožňuje paralelní zpracování instrukcí a výrazně zvyšuje výpočetní výkon bez nutnosti zvyšovat frekvenci jednotlivých jader.

Architektury mikroprocesorů se obecně dělí na dvě hlavní kategorie: CISC (Complex Instruction Set Computers) a RISC (Reduced Instruction Set Computers). CISC architektury jsou známé svou bohatou sadou instrukcí, což umožňuje efektivní využití paměti, zatímco RISC se zaměřuje na jednodušší a rychlejší sadu instrukcí, která může vést k efektivnějšímu zpracování ve specifických aplikacích. [18] [18]

V posledních letech došlo k nárůstu významu specializovaných mikroprocesorů, jako jsou DSP (Digital Signal Processors) pro zpracování signálů, a ASIC (Application-Specific Integrated Circuits) navržených pro specifické účely. Také se rozšířily SoC (System on a Chip) řešení, které integrují mikroprocesor s dalšími komponentami, jako jsou grafické procesory a síťové moduly, na jediný čip, což umožňuje vytvářet kompaktnější a energeticky efektivnější zařízení. [18] [19] [18]

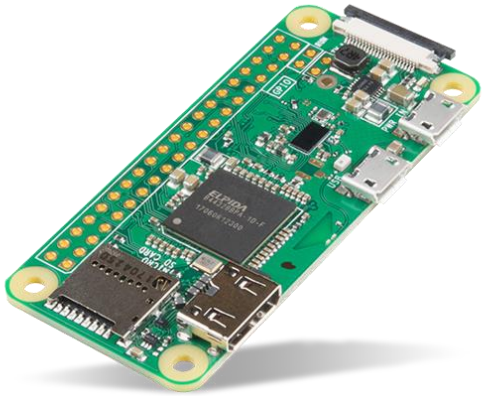
Výroba mikroprocesorů je náročná na technologie a zahrnuje pokročilé techniky litografie, aby bylo možné umístit miliardy tranzistorů na křemíkový wafer velikosti několika centimetrů. Výzkum a vývoj v této oblasti neustále posouvají hranice možného, což se odráží v neustálém zmenšování velikosti tranzistorů a zvyšování výkonu mikroprocesorů. [18] [19]

Jelikož technologie postupuje, mikroprocesory nadále hrají klíčovou roli v digitálním věku, umožňují vývoj nových aplikací a technologií a stávají se stále více integrovanými do každodenního života. [18] [19] [18]

3.6.1 Raspberry Pi Zero W

Raspberry Pi Zero W je jednodeskový počítač z rodiny Raspberry Pi, který je specifický svou kompaktní velikostí a integrovanou bezdrátovou komunikací. Byl uveden na trh v roce 2017, tento model kombinuje základní výpočetní funkce s vysokou mírou konektivity, čímž se otevírají nové možnosti pro vývoj aplikací v oblastech IoT, domácí automatizace, nositelné technologie a vzdělávání. [20] [21]

Technické specifikace
Základní konfigurace Raspberry Pi Zero W zahrnuje jedno jádrový procesor Broadcom BCM2835 s frekvencí 1 GHz a 512 MB LPDDR2 SDRAM. Tento hardware poskytuje dostatečný výkon pro řadu aplikací, od základních výukových nástrojů po specifické projekty v oblasti inteligentních domácností. Dále je vybaven integrovaným Wi-Fi 802.11n a Bluetooth 4.1, což umožňuje bezproblémovou bezdrátovou konektivitu. Pro výstup videa a zvuku je k dispozici mini HDMI port podporující 1080p60 video a kompozitní video výstup je realizovatelný přes GPIO piny. Úložiště je zajištěno prostřednictvím slotu pro microSD kartu. GPIO header s 40 piny (28 z nichž je GPIO) umožňuje připojení široké škály periférií a senzorů. Rozměry desky jsou 65 mm × 30 mm × 5 mm, což zdůrazňuje její kompaktní povahu. [20] [21]



Obrázek 3 Raspberry Pi Zero W (zdroj: https://cz.mouser.com/new/sparkfun/sparkfun-pi-zero-w-basic-kit/)

3.7 Programovací jazyky

Programovací jazyk je formalizovaný jazyk navržený k vytváření instrukcí, které mohou být interpretovány nebo kompilovány pro vykonávání operací na počítači. Skládá se ze syntaktických a sémantických pravidel, které definují, jak mohou být programy pro daný jazyk psány a jaký význam mají. Programovací jazyky umožňují vývojářům specifikovat přesně, jaké výpočty má počítač provést.

Syntaktická pravidla určují, jaké kombinace symbolů jsou v jazyce považovány za správně strukturované programy nebo výrazy. Tato pravidla obvykle zahrnují definice různých typů tokenů (např. identifikátory, laterály, operátory), pravidla pro jejich kombinování do větších struktur (např. výrazy, příkazy) a formální gramatiku, která popisuje, jak jsou tyto struktury organizovány do programů.

Sémantická pravidla určují význam těchto syntakticky správných programů a výrazů, tj. jaký typ výpočtu nebo akce by měl být proveden, když je program spuštěn. Sémantika může být definována formálně pomocí matematických modelů nebo popisem efektů, které mají jednotlivé konstrukce v programovacím jazyce na stav počítače nebo na vstupy a výstupy.

Existují různé paradigmaty programovacích jazyků, včetně imperativního (kde jsou programy sestaveny z příkazů, které mění stav systému), deklarativního (kde jsou programy specifikace požadovaného výsledku bez explicitních instrukcí, jak jej dosáhnout), funkčního (kde jsou programy sestaveny z funkcí a důraz je kladen na aplikaci a kompozici funkcí), objektově orientovaného (kde je důraz na definování datových struktur a operací, které s nimi mohou být prováděny) a mnoho dalších. [22]

3.7.1 Python

Python je vysokoúrovňový interpretovaný programovací jazyk, jehož filozofie klade důraz na čitelnost kódu; používá se k vývoji aplikací všeho druhu, například Instagram, Netflix, Spotify, Panda3D a další.2 Je to multiparadigmatický programovací jazyk, protože částečně podporuje objektové, imperativní a v menší míře i funkcionální programování. Je to interpretovaný, dynamický, multiplatformní jazyk.

Spravuje ho nadace Python Software Foundation a má licenci otevřeného kódu, která se nazývá Python Software Foundation License.3 Python je trvale řazen mezi nejoblíbenější programovací jazyky.4

Jazyk Python vytvořil koncem 80. let 20. století Guido van Rossum ve Stichting Mathematisch Centrum (CWI) v Nizozemsku jako nástupce programovacího jazyka ABC

Python je multiparadigmový programovací jazyk. To znamená, že místo aby nutil programátory k určitému stylu programování, umožňuje několik stylů: objektově orientované programování, imperativní programování a funkcionální programování. Další paradigmata jsou podporována pomocí rozšíření. Používá dynamické typování a počítání referencí pro správu paměti.

Důležitou vlastností jazyka Python je dynamické rozlišování jmen, tj. to, co spojuje jméno metody a proměnné během provádění programu (nazývá se také dynamické linkování metod).

Dalším cílem návrhu jazyka je snadné rozšiřování. Nové moduly lze snadno napsat v jazyce C nebo C++. Python lze začlenit do aplikací, které potřebují programovatelné rozhraní.26

Přestože by programování v jazyce Python mohlo být v některých situacích považováno za nepřátelské vůči tradičnímu funkcionálnímu programování, které vyložil jazyk Lisp, existuje mezi jazykem Python a minimalistickými jazyky rodiny Lisp (například Scheme) poměrně dost analogií. [23] [24]

3.7.2 Node.js

Node.js je open-source, křížově platformní běhové prostředí, které umožňuje vykonávání kódu napsaného v JavaScriptu mimo webový prohlížeč. Původně vytvořený Ryanem Dahlem v roce 2009, Node.js přinesl JavaScript na server, což rozšířilo možnosti tohoto jazyka z klientské strany webových aplikací na serverovou stranu a další výpočetní prostředí. Node.js je založen na V8 JavaScript engine od Google, který je známý svou rychlostí a výkonností díky překladu JavaScriptu do nativního strojového kódu.

Node.js využívá neblokující, událostmi řízený I/O model, který jej činí lehkým a efektivním pro vývoj škálovatelných síťových aplikací, jako jsou webové servery. Tento model umožňuje Node.js zpracovávat více požadavků současně bez nutnosti více vláknového programování. Přestože Node.js běží v jednom vlákně, díky své asynchronní povaze může obsluhovat velké množství souběžných požadavků. Vlákno event loopu se stará o všechny asynchronní operace, což minimalizuje potřebu pro náročné kontextové přepínání mezi vlákny. Node.js přichází s NPM, což je rozsáhlý ekosystém open-source knihoven (modulů), který umožňuje vývojářům snadno sdílet a používat kód. NPM usnadňuje správu závislostí a balíčků v projektech Node.js. Podporuje modulární strukturu programu prostřednictvím svého module systému, který umožňuje izolovat funkcionality do samostatných souborů a balíčků, které lze snadno importovat a využívat v jiných částech aplikace.

Node.js se používá pro širokou škálu aplikací, od webových a síťových serverů přes nástroje pro vývojové prostředí až po IoT (internet věcí). Jeho schopnost rychle zpracovat I/O

operace a snadno integrovat s databázemi a jinými systémy dělá z Node.js populární volbu pro vývoj RESTful API, real-time aplikací (např. chatů) a mikro služeb. [25] [26]

3.7.3 HTML

HTML, zkratka pro HyperText Markup Language, je standardní značkovací jazyk používaný pro tvorbu webových stránek a aplikací. Umožňuje tvůrcům obsahu strukturovat dokumenty pomocí hierarchické organizace prvků, včetně textu, obrázků, videí a odkazů. HTML dokumenty jsou interpretovány webovými prohlížeči, které je pak zobrazují uživatelům.

HTML byl vytvořen Timem Berners-Lee v roce 1990, když pracoval v CERNu ve Švýcarsku. Od svého vzniku prošel HTML několika revizemi, přičemž nejnovější verze je HTML5, která byla oficiálně publikována v říjnu 2014. HTML5 přineslo mnoho nových funkcí a vylepšení, včetně lepší podpory pro multimédia, nové API pro lepší interakce a dynamiku stránek, a nové značky, které umožňují lepší strukturování obsahu. [27] [28]

Základní koncepce

- **Značky (Tags):** HTML dokumenty jsou sestaveny z HTML značek, které určují různé typy obsahu a struktury dokumentu. Každá značka je obvykle obklopena špičatými závorkami, například *<p>* pro odstavec.
- **Elementy:** Elementy se skládají z otevírací značky, volitelného obsahu a uzavírací značky. Některé značky, jako je *<img>*, nemají uzavírací značku a jsou samozavírací.
- Atributy: Elementy mohou obsahovat atributy, které poskytují další informace o elementu. Například *<img src="obrazek.jpg" alt="Popisek">* obsahuje atributy src a alt.
- **Struktura dokumentu:** HTML dokumenty mají obvykle základní strukturu, včetně deklarace typu dokumentu (doctype), *<html>* elementu, hlavičky dokumentu *(<head>)* a těla *(<body>).*

HTML je základem webu a společně s CSS (pro stylování) a JavaScriptem (pro interaktivitu) tvoří trio technologií, které jsou nezbytné pro vývoj webových stránek a aplikací. [27] [28]

3.7.4 CSS

CSS je zkratka pro Cascading Style Sheets (kaskádové styly). Je to styl jazyk používaný pro popis prezentace dokumentu napsaného v značkovacím jazyce, jako je HTML. CSS definuje, jak mají být HTML prvky zobrazeny na obrazovce, papíře, v řeči nebo na jiném médiu. CSS je klíčovým prvkem webu a společně s HTML a JavaScriptem tvoří základní stavební bloky webových stránek. [27]

CSS byl prvně navržen Håkonem Wium Liem a Bertem Bosem v roce 1994. První verze CSS (CSS1) byla dokončena a publikována v roce 1996. Od té doby bylo vyvinuto několik verzí CSS, přičemž CSS3 je nejnovější verze, která se skládá z několika modulů publikovaných v různých časech. [27] [29]

CSS nabízí širokou škálu vlastností, které umožňují přizpůsobení téměř každého aspektu vzhledu webové stránky, včetně:

- Barvy a pozadí: určení barvy textu, pozadí a další.
- Fonty a text: stylizace textu, včetně velikosti fontu, typu písma, řádkování a zarovnání textu.
- Rozložení: kontrola layoutu stránky pomocí flexboxu, gridu a dalších layout modelů.
- Box model: řízení odsazení (margin), okrajů (border), výplní (padding) a rozměrů elementů.
- Pseudo-třídy a pseudo-elementy: definování speciálních stavů elementů, jako jsou hover efekty nebo stylování konkrétních částí elementu. [27] [29]

3.7.5 Javascript

JavaScript je vysokoúrovňový, interpretovaný programovací jazyk známý pro svou roli ve vývoji webových stránek, kde umožňuje interaktivitu s uživateli. Jeho využití však sahá daleko za hranice webových prohlížečů. JavaScript lze použít také na serverech (Node.js), v aplikacích pro mobilní zařízení, desktopové aplikace, a dokonce i v hardwarových projektech. [27] [30]

JavaScript byl původně vyvinut firmou Netscape pod názvem LiveScript jako prostředek pro přidávání programových skriptů do webových stránek. Brzy po svém vzniku byl přejmenován na JavaScript, což odráželo populární technologii té doby, Java, i když mezi oběma jazyky neexistuje přímá souvislost. ECMAScript, standardizovaná specifikace jazyka, byla vytvořena, aby zajistila jeho konzistentnost a interoperabilitu. [27] [30]

JavaScript umožňuje práci s různými typy dat, včetně čísel, řetězců a objektů. Funkce jsou základními stavebními bloky kódu, které lze opakovaně používat. JavaScript podporuje jak deklarativní (funkční), tak i výrazové definice funkcí. JavaScript je založen na prototypovém dědění, což umožňuje objektům dědit vlastnosti a metody od jiných objektů. Promises a async/await umožňují řešit asynchronní operace, například dotazy na server nebo čtení souborů. [27] [30]

3.7.6 Frameworky

Frameword je jako struktura, která poskytuje základ pro proces vývoje aplikace. Pomocí frameworku se můžete vyhnout psaní kódu od začátku. Poskytují sadu nástrojů a prvků, které pomáhají při rychlém procesu vývoje. Funguje jako šablona, kterou lze použít, a dokonce upravit podle požadavků projektu. [27] [31]

Frameworky jsou založeny na programovacích jazycích. Mezi oblíbené a nejpoužívanější frameworky patří Django, Flutter, Angular, Vue, PyTorch, Spring Boot, React Native, Apache Spark, Ionic atd. Tyto frameworky umožňují vývojářům vytvářet robustní a funkčně bohatý software. [27] [31]

Psaní kódu od nuly je zdlouhavý úkol plný možných rizik a chyb. Je třeba, aby byl kód čistý, dobře otestovaný, bez chyb a omylů. Pro ostatní vývojáře bude obtížné kódu porozumět a pracovat na něm. Proto je lepší pracovat s frameworky, které splňují vaše požadavky.

Usnadňují proces vývoje s menším počtem chyb. Pro ostatní bude snadné porozumět vašemu kódu, protože jsou také obeznámeni s frameworky. [32] [27]

Flask

Flask je mikrowebový framework napsaný v jazyce Python. Je klasifikován jako mikroframework, protože nevyžaduje konkrétní nástroje nebo knihovny. Nemá abstraktní vrstvu databáze, validaci formulářů ani žádné další komponenty, kde by běžné funkce poskytovaly již existující knihovny třetích stran. Flask však podporuje rozšíření, která mohou přidávat funkce aplikace, jako by byly implementovány v samotném Flasku. Existují rozšíření pro objektově-relační mapovače, validaci formulářů, zpracování odesílaných dat, různé otevřené autentizační technologie a několik běžných nástrojů souvisejících s frameworkem. [27] [31] [33]

Flask vytvořil Armin Ronacher z mezinárodní skupiny nadšenců Pythonu Pocoo, která vznikla v roce 2004.Podle Ronachera šlo původně o aprílový žert, který byl natolik populární, že se z něj stala seriózní aplikace. [32] [33]

Django

Django je vysokoúrovňový webový framework pro rychlý vývoj bezpečných a udržitelných webových aplikací, napsaný v Pythonu. Od svého vzniku v roce 2005 se stal jedním z nejpopulárnějších webových frameworků. Používá návrhový vzor Model-View-Template (MVT), což je varianta tradičního Model-View-Controller (MVC) vzoru, s tím rozdílem, že kontroler je nahrazen systémem šablon. Django ORM umožňuje interakci s databází pomocí Python tříd, což zjednodušuje proces vytváření a manipulace s databázovými tabulkami. Dále nabízí zabudované ochranné mechanismy proti běžným bezpečnostním hrozbám, jako je SQL injection, cross-site scripting (XSS), cross-site request forgery (CSRF) a clickjacking, a jeho uživatelský systém podporuje bezpečné ukládání hesel.

Django také obsahuje flexibilní systémem šablon, který usnadňuje generování dynamického HTML obsahu. Jeho schopnost generovat administrační rozhraní pro správu databáze výrazně urychluje vývoj. URL Routing umožňuje návrh přívětivých URL adres, které jsou snadno zapamatovatelné a optimalizované pro vyhledávače. Přestože je Django navržen pro rychlý vývoj, je schopen zvládnout velký objem provozu a lze ho škálovat pro velké aplikace.

Nabízí rozsáhlou standardní knihovnu pro různé aspekty webového vývoje, včetně formulářů, autentizace uživatelů, a mezinárodní podpory. Jeho modularita, rozšiřitelnost a velká komunita, která poskytuje mnoho doplňkových balíčků pro rozšíření jeho funkcionalit. [34]

3.7.7 Knihovny

V informatice se knihovnou rozumí soubor funkčních implementací kódovaných v programovacím jazyce, který poskytuje dobře definované rozhraní k vyvolávané funkci.

Na rozdíl od spustitelného programu se u chování implementovaného knihovnou neočekává, že bude používáno samostatně, ale je určeno k tomu, aby byla používána jinými programy,

a to nezávisle a současně. Na druhou stranu se chování knihovny nemusí příliš lišit od chování, které může být specifikováno v programu. Některé knihovny navíc mohou ke svému fungování vyžadovat jiné knihovny, protože chování, které definují, zpřesňuje nebo mění chování původní knihovny; nebo ji zpřístupňuje pro jinou technologii či programovací jazyk.

Knihovny mohou být propojeny s programem (nebo jinou knihovnou) v různých fázích vývoje nebo provádění v závislosti na typu požadovaného propojení. [35]

PyMongo

PyMongo je knihovna jazyka Python obsahující nástroje pro práci s MongoDB. Pymongo umožňuje programům v jazyce Python interagovat s MongoDB, provádět operace čtení a zápisu, a spravovat databázi. [36]

ArduinoJson

ArduinoJson je knihovna pro platformu Arduino, která usnadňuje práci s formátem JSON. JSON (JavaScript Object Notation) je lehký formát dat, který se používá k výměně informací mezi aplikacemi. ArduinoJson umožňuje Arduino mikrokontrolerům generovat a zpracovávat JSON data.

Tato knihovna poskytuje nástroje pro vytváření JSON objektů a pole, jejich serializaci do textové podoby (pro odesílání nebo ukládání) a deserializaci z textu zpět do interní reprezentace dat. To může být užitečné, pokud pracujete s komunikací mezi různými zařízeními nebo serverem, který používá formát JSON. [37]

Adafruit DHT sensor library

Tato knihovna poskytuje jednoduché rozhraní pro komunikaci s DHT senzory, které umožňuje čtení naměřených hodnot teploty a vlhkosti. Pomocí této knihovny můžete snadno integrovat DHT senzory do svých projektů a získávat aktuální informace o teplotě a vlhkosti. [38]

Adafruit Unified Sensor

Adafruit Unified Sensor je knihovna pro platformu Arduino, která slouží k unifikaci rozhraní pro různé senzory od společnosti Adafruit. Tato knihovna poskytuje jednotné rozhraní pro různé typy senzorů, což usnadňuje integraci a programování. [38]

3.8 Vývojová prostředí

Vývojová prostředí, často označovaná jako integrovaná vývojová prostředí (IDE - Integrated Development Environments), jsou softwarové aplikace poskytující komplexní nástroje pro vývojáře softwaru. IDE typicky kombinuje několik nástrojů potřebných pro vývoj softwaru do jediné aplikace, což vývojářům umožňuje psát, testovat a ladit kód efektivněji. [27]

3.8.1 Visual studio Code

Visual Studio Code je bezplatné, open-source, křížově platformní vývojové prostředí (IDE) vyvinuté společností Microsoft. Od svého uvedení v roce 2015 se rychle stalo jedním z nejpopulárnějších nástrojů pro vývojáře díky své všestrannosti, výkonu a rozšiřitelnosti. VS Code podporuje vývoj v mnoha programovacích jazyků a frameworků, nabízí širokou škálu funkcí a nástrojů pro produktivní vývoj softwaru.

VS Code podporuje mnoho programovacích jazyků. Tato podpora je zčásti nativní a zčásti dostupná prostřednictvím rozsáhlého ekosystému rozšíření. Jednou z nejvýraznějších vlastností VS Code je jeho vysoce rozšiřitelná architektura. Trh s rozšířeními nabízí tisíce pluginů a témat, které přidávají nové funkce a nástroje nebo mění vzhled a chování prostředí. VS Code poskytuje integrovanou podporu pro Git, což umožňuje vývojářům provádět běžné operace s verzemi přímo z IDE, včetně commitování změn, pushování a pull requestů. Díky funkcím jako IntelliSense, VS Code poskytuje kontextově relevantní návrhy kódu, automatické dokončování, informace o parametrech funkcí a rychlé informace o typu proměnných. VS Code obsahuje vestavěný debugger, který umožňuje ladit kód přímo v IDE. Podporuje krokování kódu, body zastavení a prohlížení proměnných, což usnadňuje diagnostiku a opravu chyb v kódu. VS Code obsahuje integrovaný terminál a podporuje vývoj v Docker kontejnerech, což vývojářům umožňuje spouštět a testovat aplikace v izolovaném prostředí přímo z IDE.

VS Code je vhodné pro širokou škálu vývojových úkolů, od jednoduchých skriptů a webových stránek až po složité podnikové aplikace. Díky podpoře mnoha jazyků a nástrojů je oblíbenou volbou pro front-end i back-end vývojáře, včetně těch, kteří pracují na full-stack aplikacích, mobilních aplikacích, cloudových službách a aplikacích pro IoT. [27] [39]

3.8.2 **PyCharm**

PyCharm je výkonné integrované vývojové prostředí (IDE) určené primárně pro jazyk Python, vyvinuté společností JetBrains. PyCharm je navržen tak, aby maximalizoval produktivitu vývojářů poskytnutím komplexních nástrojů pro vývoj v Pythonu, včetně inteligentního editování kódu, sofistikovaného debuggování, rychlé navigace, automatického refaktoringu kódu a podpory pro webové technologie jako jsou Django, Flask a další.

PyCharm nabízí pokročilé funkce pro editaci kódu, včetně syntax highlighting, code completion, on-the-fly error checking a quick fixes, což usnadňuje psaní kódu a snižuje pravděpodobnost chyb. IDE podporuje mnoho Python frameworků, včetně Django, Flask a Pyramid, což umožňuje rychlý vývoj webových aplikací. PyCharm rovněž integruje nástroje specifické pro jednotlivé frameworky, jako jsou šablony, správa databází a debuggování. Umožňuje snadnou integraci s virtualenv, Dockerem a Vagrantem pro správu závislostí a izolaci vývojového prostředí, což zvyšuje konzistenci mezi vývojovými a produkčními prostředími. Obsahuje pokročilé nástroje pro debuggování a testování, včetně grafického debuggeru, test runneru a podpory pro testovací frameworky, jako je pytest nebo unittest, což vývojářům usnadňuje identifikaci a opravu chyb.
[40]

3.9 Wireframe

Wireframy jsou základní vizuální návrhy, které demonstrují rozložení, strukturu a funkčnost webových stránek, aplikací nebo jiných digitálních produktů před jejich finálním designem a vývojem. Představují kostru nebo "drátěný model" produktu, který zobrazuje základní uspořádání prvků a interakce uživatele, ale obvykle nepředstavují grafický design, barvy nebo styl.

Wireframy umožňují návrhářům a týmům efektivně plánovat uspořádání a hierarchii informací na stránce nebo v aplikaci. Slouží jako komunikační nástroj mezi designéry, vývojáři, stakeholdery a klienty pro diskusi o funkcích, layoutu a cílech produktu. Umožňují rychlé a levné iterace návrhu, což je zásadní pro testování a zlepšování nápadů před finálním vývojem. [27]

3.10 Saatyho metoda

Saatyho metoda, známá také jako Analytický hierarchický proces (AHP), je rozhodovací technika, kterou vytvořil Thomas L. Saaty v sedmdesátých letech 20. století. AHP je používána k řešení komplexních rozhodovacích problémů, kde je potřeba posoudit více kritérií, a to i v situacích, kdy jsou tato kritéria nekonzistentní nebo kvalitativní.

Metoda kombinuje matematické a psychologické aspekty rozhodování a funguje na principu rozkladu rozhodovacího procesu do hierarchie. Tato hierarchie může zahrnovat cíle, sub-cíle, kritéria a alternativy. AHP pomáhá rozhodovatelům určit priority a nejlepší možnost na základě pečlivě strukturované matematické analýzy. [41]

3.11 cURL

cURL (Client URL) je softwarový projekt a nástroj příkazové řádky dostupný pro různé operační systémy, včetně Linuxu, Windows a macOS. Je určen pro přenos dat s URL syntaxí a podporuje širokou škálu protokolů, včetně HTTP, HTTPS, FTP, FTPS, SCP, SFTP, TFTP, DICT, TELNET, LDAP nebo FILE. cURL je velmi oblíbený mezi vývojáři pro testování API, stahování souborů, odesílání dat formulářů a mnoho dalších úkolů spojených s komunikací mezi klientem a serverem přes různé protokoly.

Hlavní Funkce a Vlastnosti

- Podpora Protokolů: cURL podporuje širokou škálu síťových protokolů, což umožňuje interakci s téměř jakýmkoliv typem serveru nebo zdroje dat.
- Odesílání a Příjem Dat: Umožňuje odesílat data na server (např. při použití HTTP POST nebo PUT požadavků) a stahovat data zpět do klienta.
- Autentizace: Podporuje různé metody autentizace, včetně základní, digest, NTLM a další.
- Manipulace s Cookies: Umožňuje odesílat a přijímat cookies, což je užitečné pro interakci se sezeními na webových stránkách.
- SSL Podpora: cURL podporuje šifrovanou komunikaci přes SSL/TLS, což umožňuje bezpečný přenos citlivých dat.

- Řízení Přenosu: Umožňuje řízení různých aspektů přenosu dat, včetně limitu rychlosti, timeoutů a pokusů o připojení.

[42]

4 Vlastní práce

Tato část práce se bude věnovat samotné implementaci IoT pro solární systémy. Počátkem práce bude provedena analýza trhu ke zjištění klíčových vlastností solárních systémů. Dále proběhne analýza vhodných softwarových a hardwarových komponent pro implementaci IoT pro solární systémy. Následně bude zhotovena analýza požadavků na tento systém po konzultaci s firmou Intronix s.r.o. V návaznosti na získaných informacích bude vytvořen IoT model k hlubšímu porozumění problematiky a stanovení potřebných technologií.

Další část práce se bude zabývat samotnou implementací internetu věcí pro toto řešení. Implementací internetu věcí pro solární systémy vznikne software pro wattrouter, který bude mít na starosti efektivní chod fotovoltaické elektrárny.

4.1 Analýza trhu

Před tím, než bude možné definovat vlastnosti našeho řešení, je nutné udělat průzkum trhu a zjistit jaké specifikace nabízí konkurenční zařízení. Na českém trhu se vyskytují dva hlavní konkurenti v oblasti wattrouterů fotovoltaických elektráren. Jedním z nich je wattrouter od české firmy Solar controls a druhým je GreenBonO od české firmy Yorix s.r.o Jelikož se tato práce zabývá zejména softwarovou částí, nebude brán ohled na hardwarové specifikace konkurenčních řešení.

4.1.1 Wattrouter Mx

Jedná se o sofistikovaný programovatelný regulátor od společnosti Solar controls s.r., který je určen k optimalizaci vlastní spotřeby pro objekty s instalovanou fotovoltaickou nebo větrnou elektrárnou.

Vlastnosti softwaru
Webová Aplikace s dashboardem pro sledování a ovládání chodu fotovoltaické elektrárny. Aplikace Obsahuje informace o výkonu na jednotlivých fázích, které jsou vykreslovány v grafech, nastavení a sledování výkonu jednotlivých zátěží připojených na polovodičových SSR relé, informace o chybových hláškách systému, stavy vstupů, stavy výstupů, informaci o stavu nočního proudu.

4.1.2 GreenBonO

Regulátor GreenBonO je určen k dosažení maximálního možného využití vlastní vyrobené elektřiny v místech s libovolnou lokální elektrárnou, jejíž výroba je přednostně určena k místní spotřebě.
Uplatní se nejen jako wattrouter u elektráren připojených k síti, ale také jako vytěžovač u elektráren hybridních, ostrovních a mikrosítí.
Pestrá škála vyhotovených firmwarů zajišťuje tomuto výrobku univerzální použitelnost od nejtriviálnější jednofázové aplikace s jedním bojlerem až po rozsáhlé třífázové regulace. A právě ve schopnosti regulovat velké střešní fotovoltaické elektrárny (FVE) s velkým počtem spotřebičů přebytků těžko najde rovnocennou konkurenci. Zkuste zajistit např. regulaci tohoto rozsahu, nebo tohoto libovolným konkurenčním výrobkem! [43]

Vlastnosti softwaru
GreenBonO se nastavuje skrze aplikaci za použití rozhraní RS485. Aplikace umožňuje uživateli sledovat výkony na jednotlivých fázích a konfiguraci SSR relé. Dále obsahuje konfiguraci intervalu spínacích hodin pro dvě relé.

Aplikace GreenBonO není určena běžným uživatelům, spíše slouží k prvotnímu nastavení celého systému. Obsahuje však nastavení spínacích hodin, které je nezbytné

4.2 Analýza požadavků na systém

K úspěšnému zhotovení systému je nutné zjistit požadavky na tento systém. Analýza byla zhotovena ve spolupráci se společností Intronix za pomocí brainstormingu a znalostí získaných při tvoření teoretické práce. Dle těchto požadavků se bude dále řídit celý vývoj tohoto systému.

- Systém by měl být modulární (jednoduše by se měli dát přidat nová vstupní a výstupní zařízení)
    - Systém bude obsahovat čtveřici polovodičových relé pro připojení výstupních zařízení
    - Systém bude obsahovat aspoň 4 volné vstupy pro možnost rozšíření o další vstupní zařízení
- Systém musí obsahovat webovou aplikaci pro jednoduché ovládání a sledování hodnot.
    - Pro sledování hodnot by systém měl obsahovat spojnicový graf s historií naměřených dat
        - Spojnicový graf bude sloužit pro sledování výkonu fotovoltaické elektrárny a výkonu boileru
    - Webová aplikace by měla obsahovat dlouhodobou předpověď počasí
- Systém by měl být schopný komunikovat i mimo lokální síť
- Systém bude schopný sám vyhodnotit kdy sepnout zátěž (například boiler) v návaznosti na naměřených hodnotách přetoků do sítě.

Při návrhu systému budou zohledněny tyto požadavky. Možné odchylky od analýzy budou dále se společností Intronix konzultovány a rozebrány v závěru práce.

4.3 Analýza hardwarových nástrojů

Tato část práce se zabývá analýzou vhodných hardwarových komponent pro úspěšnou implementaci internetu věcí pro solární systémy. Analýza má za úkol podle stanovených kritérií zvolit nejvhodnější řídící jednotku pro tento systém. Pro vybrání nejvhodnější komponenty byla zvolena Saatyho metoda párového porovnání. Pro porovnání byla zvolena tato kritéria: výkon, velikost, průměrná spotřeba elektrické energie, uživatelská podpora a cena.

4.3.1 Stanovení požadavků na hardware

Účel projektu: Sběr dat o stavu baterie, přetocích do sítě a teploty boileru v reálném čase.
Funkční požadavky: Možnost bezdrátového připojení k internetu pro odesílání dat do cloudové služby.
Technické požadavky: Wi-Fi konektivita, podpora pro připojení senzorů, nízká spotřeba energie, kompaktní velikost.
Rozpočet a náklady: Maximálně 500Kč za zařízení.

4.3.2 Prvotní výzkum a sběr kandidátů

Po provedené rešerše v teoretické části práce byli vybráni potencionální zařízení pro chod systému. V této části bude proveden prvotní výzkum k předběžnému určení nejvhodnějšího zařízení pro implementaci internetu věcí pro solární panely. Tento výběr bude následně ověřen Saatyho metodou.

ESP32
Výhody:
- Vestavěná Wi-Fi a Bluetooth konektivita.
- Vysoký výpočetní výkon a nízká spotřeba energie.
- Dostupnost mnoha vývojových nástrojů a knihoven pro snadnou práci s HTTP a REST API.
- Relativně nízká cena a široká dostupnost.
- Podpora pro různé vývojové prostředí včetně Arduino IDE a ESP-IDF.

Nevýhody:
- Vyžaduje dodatečné znalosti pro správu energetické efektivity a bezpečnosti připojení.

Raspberry Pi Zero W
Výhody:
- Flexibilní jednodeskový počítač s plnou podporou operačního systému Linux, což usnadňuje vývoj a integraci s REST API.
- Podpora Wi-Fi, což umožňuje snadné připojení k internetu.
- Velké množství dostupných knihoven a nástrojů pro práci s REST API v různých programovacích jazycích (Python, Node.js, Java, atd.).

Nevýhody:
- Vyšší spotřeba energie ve srovnání s mikrokontrolery jako je ESP32, což může být problém pro baterií poháněné aplikace.
- Vyšší cena v porovnání s jednoduššími mikrokontrolery.

Arduino UNO Wi-Fi
Výhody:
- Snadné zapojení a programování pro začátečníky.
- Podpora pro Wi-Fi umožňuje jednoduchou síťovou konektivitu.
- Široká uživatelská a vývojová komunita s množstvím dostupných tutoriálů a knihoven.

Nevýhody:
- Nižší výpočetní výkon

- Potřeba dodatečného hardwaru (Ethernetový/Wi-Fi shield) pro připojení k internetu, což zvyšuje celkové náklady.

Seznam vybraných zařízení pro analýzu:
- Arduino s Wi-Fi shieldem
- Raspberry Pi Zero W
- ESP32

4.3.3 Vyhodnocení analýzy

Pro většinu IoT projektů, kde je důležitá komunikace s REST API a není vyžadována vysoká výpočetní síla přímo na zařízení je ESP32 obvykle nejlepší volbou kvůli své ceně, výkonu a integrované podpoře pro Wi-Fi. Toto zařízení nabízí skvělý poměr cena/výkon, je dostatečně výkonné pro komunikaci s REST API a je energeticky efektivní, což je ideální pro tento účel.

4.3.4 Saatyho metoda

Pro ověření správného výběru řídící jednotky bude provedena Saatyho metoda párového porovnávání.

Porovnávána budou tyto zařízení:
- Arduino UNO Wi-Fi
- ESP32
- Raspberry Pi Zero W

Pro porovnání byla zvolena tato kritéria:
- **Výkon**, kde je zohledněn kmitočet procesoru a kapacita paměti.
- **Velikost** zařízení v milimetrech
- Průměrná elektrická **spotřeba** v miliampérech.
- Uživatelská **podpora**, která byla ohodnocena body 1 až 9 (1 - slabá uživatelská podpora, 9 – silná uživatelská podpora). Body byly stanoveny na základě předešlé rešerše.
- **Cena** je uvedena v korunách Českých.

Tabulka 1 Zařízení a jejich parametry

	zařízení	Výkon	Velikost	Spotřeba	Podpora	Cena
L1	Arduino Uno Wifi	16MHz, 2KB	68,6x53,4 mm	50 mA	9	768 Kč
L2	ESP32	2x240MHz, 520KB	54,4x27,9 mm	100 mA	8	207 Kč
L3	Raspberry Pi ZERO W	1x1GHZ, 512 MB	65x30 mm	300 mA	7	420 Kč

Škálovatelnost kritérií:
1 – rovnocenné části i a j
3 – slabě preferovaná část i před j
5 – silně preferovaná část i před j
7 – velmi silně preferovaná část i před j

9 – absolutně preferovaná část i před j

Písmeno i bylo zvoleno pro označení řádku a písmeno j pro označení sloupce tabulky.

Pro lepší přehlednost budou jednotlivá kritéria označena jako k1-k5.
k1 – výkon
k2 – velikost
k3 – spotřeba
k4 – podpora
k5 – cena

Tabulka 2 Stanovení vah

	k1	k2	k3	k4	k5	Geometrický průměr	Váhy kritérií
k1	1	3	1/7	3	5	1,450851272	19,62 %
k2	1/3	1	1/5	3	3	0,901067424	12,18 %
k3	7	5	1	5	7	4,145980143	56,06 %
k4	1/3	1/3	1/5	1	3	0,581810759	7,87 %
k5	1/5	1/3	1/7	1/3	1	0,316473893	4,28 %
					celkem	**7,396183491**	**100,00 %**

Prvotním krokem Saatyho metody je stanovení vah. Nejdříve je nutné porovnat jednotlivá kritéria vůči sobě a tím stanovit váhy jednotlivých kritérií. Dále je nutné vypočítat geometrický průměr každého řádku. Váhy kritérií pak lze získat podílem geometrického průměru daného řádku a celkovým geometrickým průměrem. Z výpočtu nám plyne, že nejsilnější váhu má kritérium k3 čili spotřeba elektrické energie.

Tabulka 3 Výpočet váhy pro výkon

		16MHz, 2KB	2x240MHz, 520KB	1x1GHZ, 512MB		
	k1	**L1**	**L2**	**L3**	**Geo.P.**	**Váha**
16MHz, 2KB	**L1**	1	1/5	1/7	0,305711	7,19 %
2x240MHz, 520KB	**L2**	5	1	1/3	1,185631	27,90 %
1x1GHZ, 512MB	**L3**	7	3	1	2,758924	64,91 %
				celkem	**4,250266**	**100,00 %**

Tabulka 4 Výpočet váhy pro velikost

		68,6x53,4	54,4x27,9	65x30		
	k2	**L1**	**L2**	**L3**	**Geo.P.**	**Váha**
68,6x53,4	**L1**	1	1/3	1/3	0,48075	12,68 %
54,4x27,9	**L2**	3	1	5	2,466212	65,06 %
65x30	**L3**	1/5	3	1	0,843433	22,25 %
				celkem	**3,790395**	**100,00 %**

Tabulka 5 Výpočet váhy pro spotřebu

		100	100	300		
	k3	**L1**	**L2**	**L3**	**Geo.P.**	**Váha**
100	**L1**	1	1	5	1,709976	45,45 %
100	**L2**	1	1	5	1,709976	45,45 %
300	**L3**	1/5	1/5	1	0,341995	9,09 %
				celkem	**3,761947**	**100,00 %**

Tabulka 6 Výpočet váhy pro podporu

		9	8	7		
	k4	**L1**	**L2**	**L3**	**Geo.P.**	**Váha**
9	**L1**	1	1	2	1,259921	41,26%
8	**L2**	1	1	1	1	32,75%
7	**L3**	1/2	1	1	0,793701	25,99%
				celkem	**3,053622**	**100,00%**

Tabulka 7 Výpočet váhy pro cenu

		768	207	420		
	k5	**L1**	**L2**	**L3**	**Geo.P.**	**Váha**
768	**L1**	1	1/5	1/3	0,40548	10,47%
207	**L2**	5	1	3	2,466212	63,70%
420	**L3**	3	1/3	1	1	25,83%
				celkem	**3,871692**	**100,00%**

Dalším krokem Saatyho metody je párové hodnocení jednotlivých kritérií. Nejdříve porovnáme kritéria vůči sobě a dosadíme hodnocení. Opět je nutné spočítat geometrický průměr jednotlivých řádku a následně vypočítat podíl geometrického průměru řádku a celkového geometrického průměru pro každý řádek. Tím získáme výsledné váhy daných kritérií.

Tabulka 8 Výsledné hodnocení

Váha	**19,62 %**	**12,18 %**	**56,06 %**	**7,87 %**	**4,28 %**	**Výsledné hodnocení**
výsledek	**k1**	**k2**	**k3**	**k4**	**k5**	
L1	0,0719	0,1268	0,4545	0,4126	0,1047	32 %
L2	0,279	0,6506	0,4545	0,3275	0,637	44 %
L3	0,6491	0,2225	0,0909	0,2599	0,2583	24 %

V poslední části dosadíme do tabulky hodnoty jednotlivých kritérií vypočtených v předešlých tabulkách a vynásobíme je s váhami kritérií. Tyto hodnoty sečteme a dostaneme výsledné hodnocení, které nám určí nejvhodnější zařízení na implementaci IoT pro solární systémy. V tomto případě nám vyšla jako nejpreferovanější varianta L2 čili mikrokontroler ESP32. Výsledek se shoduje i s prvotní analýzou, proto lze konstatovat, že ESP32 je nejlepší volbou pro toto řešení.

4.3.5 Potvrzení analýzy

Pro IoT projekt, který vyžaduje komunikaci s REST API serverem a databází, je klíčové zvolit zařízení s dostatečným výpočetním výkonem, stabilním internetovým připojením a podporou pro softwarové knihovny nezbytné pro HTTP komunikaci. Je také nutné vzít v potaz spotřebu elektrické energie, a to z důvodu ještě větší efektivity celého systému. Dalším kritériem je i velikost zařízení, z důvodu budoucího osazení tohoto zařízení na desku plošného spoje.

Po provedení Saatyho metody tyto kritéria nejlépe splňoval mikrokontroler ESP32. Tím se potvrdil výběr v prvotní analýze.

4.4 Analýza softwarových nástrojů

Analýza softwarových nástrojů se zaměří na výběr vhodných programovacích jazyků, vývojových prostředí (IDE) a databázových řešení. Při volbě vhodných softwarových nástrojů bude kladen důraz na to, aby bylo možné realizovat komunikaci skrze REST API server.

4.4.1 Programovací jazyky

Python
Výhody:
- Jednoduchost, čitelnost a mnoho knihoven podpory.
- Vynikající knihovny pro práci s HTTP požadavky (jako je request)
- Rychlý vývoj.

Nevýhody:
- Může být méně efektivní z hlediska výkonu ve srovnání s kompilovanými jazyky, jako je C++.

JavaScript/Node.js
Výhody:
- Umožňuje vývoj serverové i klientské části v jediném programovacím jazyce.
- Node.js je široce používán pro vývoj back-endu a je ideální pro asynchronní operace, což je časté v IoT aplikacích.

Nevýhody:
- Asynchronní programování a callbacky mohou ztížit čitelnost kódu

4.4.2 Vývojová prostředí (IDE)

Visual Studio Code
Výhody:
- Bezplatné, lehké a vysoce konfigurovatelné IDE s podporou pro Python, JavaScript a mnoho dalších jazyků. Obsahuje rozsáhlé rozšíření pro IoT vývoj, včetně podpory pro Docker a integraci s verzovacími systémy.

Nevýhody:
- Může vyžadovat dodatečnou konfiguraci a instalaci rozšíření pro optimalizaci vašeho vývojového workflow.

PyCharm (pro Python)
Výhody:

- Nabízí pokročilé funkce pro vývoj Python aplikací, včetně debuggeru, testovacího prostředí a integrace s různými frameworky.

Nevýhody:

- Základní verze je zdarma, ale profesionální verze s dalšími funkcemi je placená.

4.4.3 Databáze

MongoDB
Výhody:

- Dokumentově orientovaná NoSQL databáze, která je vhodná pro ukládání velkého množství strukturovaných i nestrukturovaných dat.
- Je dobře škálovatelná a flexibilní, což je užitečné pro IoT aplikace s různorodými datovými modely.

Nevýhody:

- Může být méně vhodná pro aplikace vyžadující komplexní transakce a spojení, jako jsou tradiční relační databáze.

InfluxDB
Výhody:

- Časově řazená databáze optimalizovaná pro rychlé zápisy a dotazy na časové řady dat, což je běžné v IoT aplikacích pro monitorování a telemetrii.

Nevýhody:

- Zaměření na časové řady může omezit její použití v projektech, které vyžadují více obecné databázové schéma.

4.4.4 Vyhodnocení analýzy

Pro vývoj softwaru bylo vybráno vývojové prostředí Visual Studio Code a to z důvodu rozsáhlé podpory mnoha programovacích jazyků a frameworků, což umožňuje práci na rozmanitých projektech internetu věcí, od vývoje embedded softwaru až po serverovou část a front-end. Další velká výhoda je podpora pro Docker a možnost remote developmentu.

Jako programovací jazyk byl vybrán Python, a to z důvodů, že je ideální pro serverovou část projektu díky své snadné čitelnosti, široké podpoře knihoven a frameworků pro vývoj REST API (např. Flask nebo Django). Tento jazyk také umožňuje snadné zpracování dat a je dobře podporován v IoT komunitě.

Přestože databáze MongoDB postrádat některé specializované funkce pro časové řady, jako jsou pokročilé možnosti komprese dat časových řad nebo optimalizované dotazovací plány, které jsou běžné v databázích specificky navržených pro časové řady jako je InfluxDB, pro tento projekt je její funkcionalita dostačující. Databáze MongoDB byla vybrána hlavně z důvodu předešlých zkušeností s touto databází.

4.5 Návrh systému

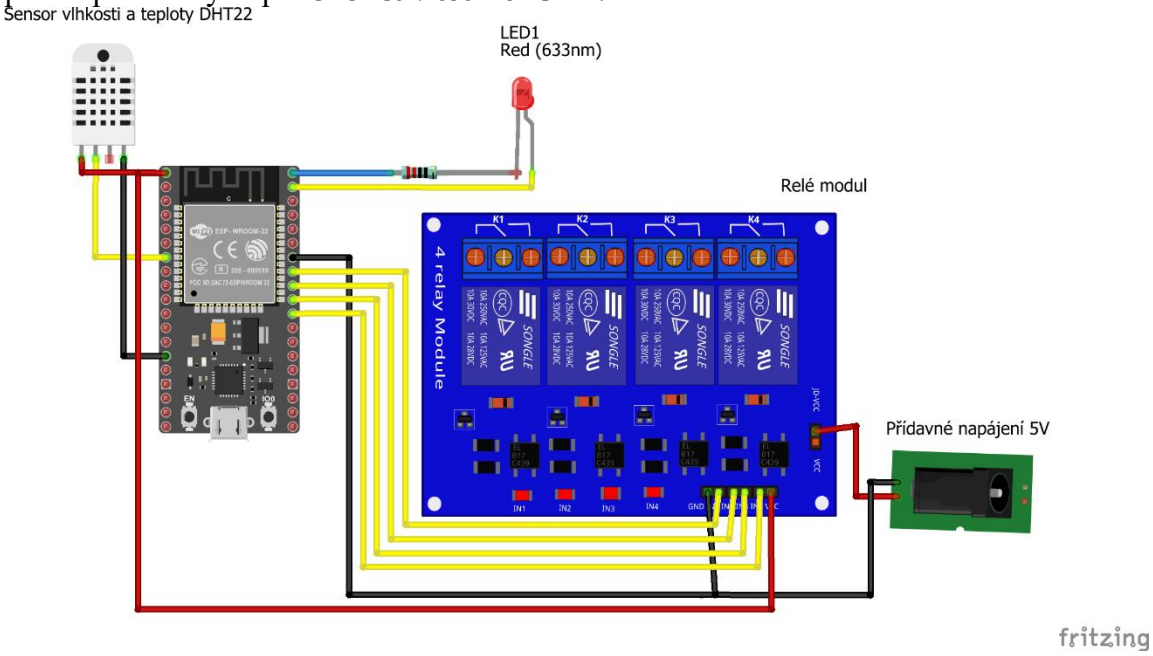
Pro návrh hardwarového prototypu bylo zapotřebí se seznámit s teoretickými informacemi uvedenými v prvotní časti práce. K ověření funkčnosti programu bude nejdříve zhotoveno testovací zapojení. Toto zapojení bude obsahovat červenou diodu a čtyř kanálový relé modul pro otestování výstupu a senzor teploty a vlhkosti DHT22 pro otestování vstupů.

Testovací zapojení bude obsahovat tento hardware:

- 1x červená LED
- DHT22 (senzor pro měření vlhkosti a teploty)
- Nepájivé pole
- ESP32
- Zdroj stejnosměrného napětí 5V
- Čtyř kanálový relé modul

4.5.1 Popis zapojení

GND (zem) je propojen s GND na relé modulu, GND na LED diodě a GND na senzoru DHT22. VCC 3,3V (napájecí vstup) je propojen s VCC na senzoru DHT22. D2 pin je propojen s datovým vstupem DAT na senzoru DHT22. D1, D5, D6, D7 piny jsou propojeny s IN1, IN2, IN3, IN4 na relé modulu, což umožňuje mikrokontroleru ovládat relé. Anoda led diody (delší nožička) je propojena s rezistorem, který je pak propojen s GND na ESP32. Katoda led diody (kratší nožička) je přímo propojena s GND. Pro úpravu napětí na diodě je použit předřadný odpor s rezistivitou 10kOhm.

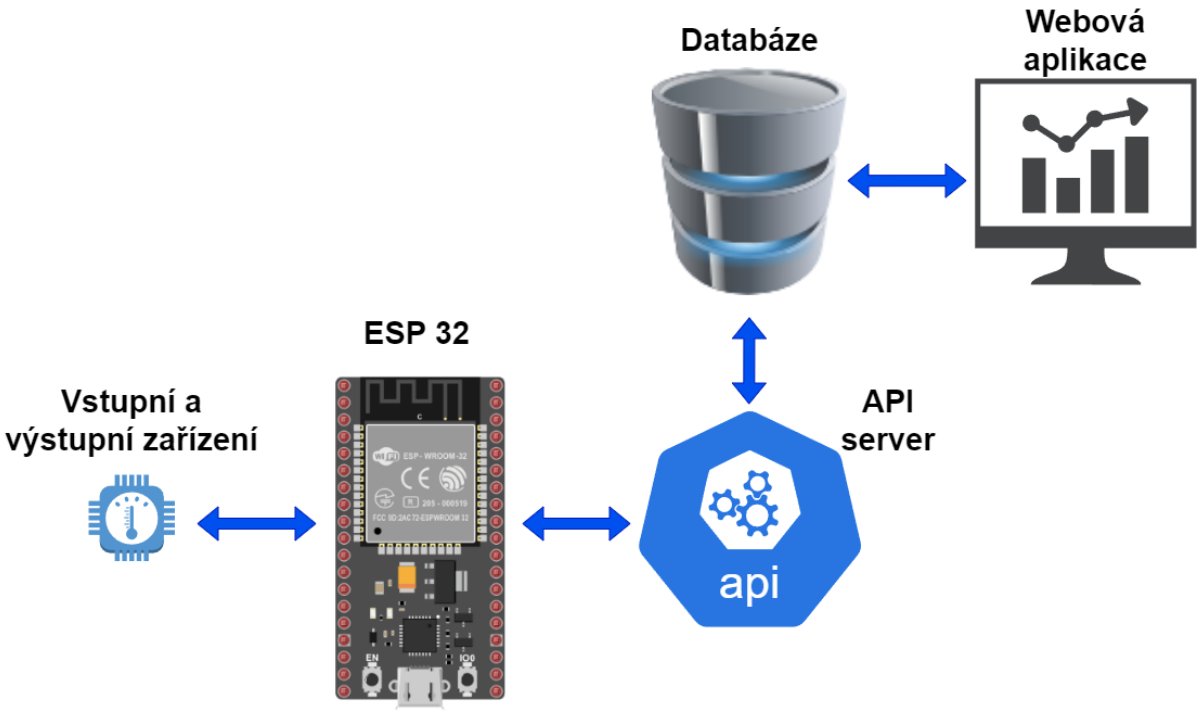


Obrázek 4 Schéma testovacího zapojení (vlastní zpracování)

4.5.2 IoT model

Pro lepší pochopení celého systému byl zhotoven IoT model obsahující vstupní a výstupní zařízení, která jsou přímo připojeny do mikrokontroleru ESP32. REST API server komunikuje skrze HTTP protokol s databází a mikrokontrolerem ESP32. Databáze zapisuje

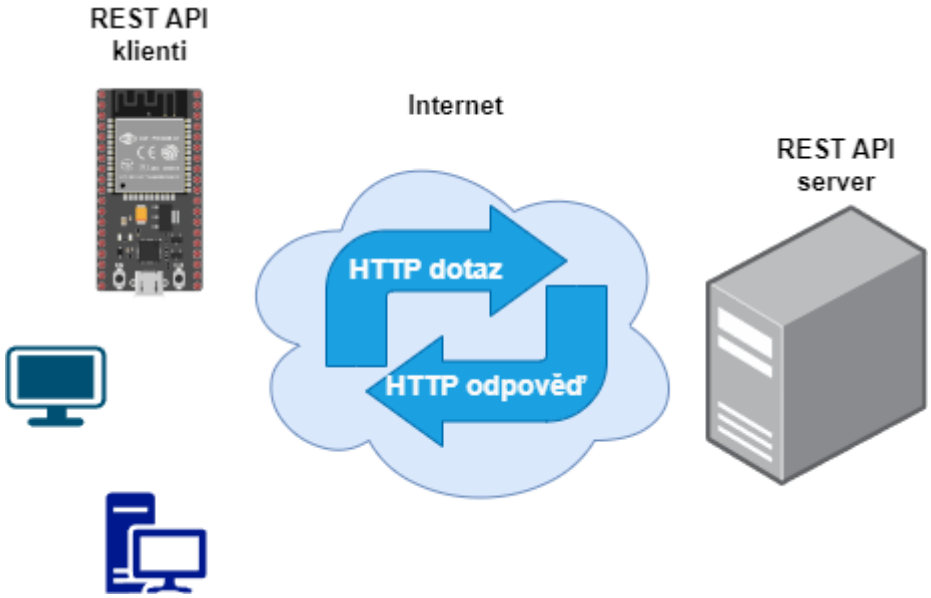
naměřená data senzorů, která zpracoval mikrokontroler ESP32 a dále odeslal skrze REST API server. Tato komunikace se děje i v opačném směru čili ESP32 provádí úkony (sepnutí výstupu) v závislosti na získaných datech z databáze. Webová aplikace vykresluje data v grafech a obsahuje přepínače pro ovládání výstupů systému (dashboard).



Obrázek 5 IoT model (vlastní zpracování)

4.5.3 REST API server

Z důvodu zajištění komunikace mezi mikrokontrolerem a databází bylo nutné využít REST API server, a to kvůli nemožnosti připojit ESP32 napřímo do databáze. REST API server tedy bude fungovat jako prostředník při komunikaci mezi ESP32 a databází. Dokud systém nebude finálně hotový, bude využito cloudových služeb pro REST API server. K tomuto účelu byla vybrána cloudová platforma Google Cloud, která umožňuje hosting Python kódu. Po úspěšném odlazení programu bude REST API umístěno na serveru společnosti Intronix.



Obrázek 6 Komunikace REST API serveru a klienta (vlastní zpracování)

4.5.4 ESP32

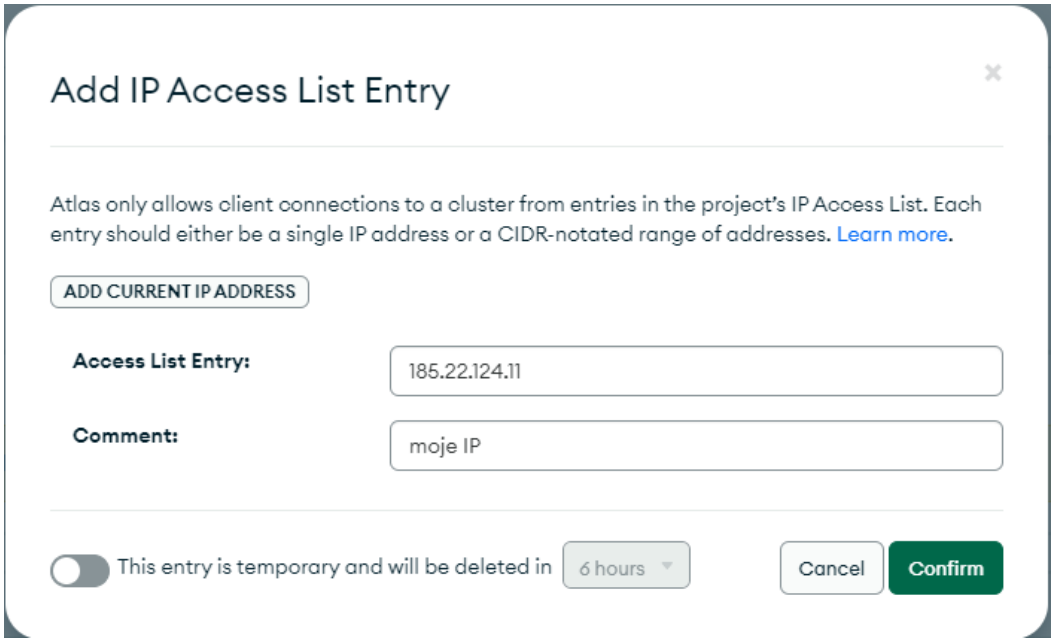
Úkolem mikrokontroleru ESP32 bude sběr hodnot a ovládání výstupních zařízení. Dále bude tyto hodnoty odesílat skrze REST API server do databáze časových řad MongoDB. Vstupy a výstupy budou ovládány v návaznosti na přijatých datech od REST API serveru.

4.5.5 Databáze

Pro tento systém byla vybrána cloudová databáze časových řad MongoDB Atlas, a to z důvodu nutnosti ukládat data v návaznosti na čas. Naměřená data se následně mohou zobrazovat v lineárních grafech, a tak sledovat historii naměřených dat. Data budou ukládána ve formátu JSON.
MongoDB Atlas lze nasadit u různých poskytovatelů cloudových služeb, jako je Amazon, Azure nebo Google. Pro účely tohoto systému byl vybrán cloudový server od společnosti Azure s lokalizací v Nizozemsku, kde byl vytvořen náš cluster. Tento cloudový server byl vybrán z důvodu nejmenší vzdálenosti od nás. Tím se může snížit riziko vyšší latence. Pro práci s databází bude použit GUI od společnosti MongoDB.

Zabezpečení
Je důležité zabezpečit cluster tak, aby k němu neměli přístup jiní uživatelé, a to omezením IP adres, které mohou k clusteru přistupovat. Do tohoto seznamu lze přidat libovolný počet IP adres. V našem případě byla použita naše aktuální IP adresa, která byla automaticky zjištěna MongoDB Atlas viz obrázek níže.

Déle je třeba vytvořit uživatele pro přístup do databáze. Jednotlivým uživatelům lze přidávat role, které určují pravomoci daného uživatele. Uživateli lze nastavit, zdali je administrátorem databáze nebo budeme mít pouze přístup ke čtení záznamů uložených v databázi. V našem případě byl vytvořen uživatel „iotuser“ s pravomocemi číst a psát to jakékoliv databáze. Přidávání a změna rolí je možná kdykoliv během vývoje systému.



Obrázek 7 Přidání IP adresy pro přístup do databáze

Po úvodním nastavení databáze lze přejít k samotnému vytvoření naší databáze časových řad obsahující data z našich senzorů. Naměřené hodnoty senzorů se budou nacházet v databázi pojmenované „sensors_db“. Pro možnost testování naplníme databázi JSON objekty (záznamy) o senzorech viz obrázek níže.

```
_id: ObjectId('65ccc7980f0d362c9f7746d5')
sensor_id : "led_2"
description : "LED"
location : "outside"
enabled : true
type : "toggle"
value : "on"
```

```
_id: ObjectId('65ccc7980f0d362c9f7746d4')
sensor_id : "led_1"
description : "LED"
location : "bedroom"
enabled : true
type : "toggle"
value : "on"
```

Obrázek 8 Vložené JSON objekty pro testování spojení s databází

4.6 Program REST API serveru

V této části práce bude popsán kód REST API serveru. který bude uložen v souboru app.py a citlivá data jako je token a link pro připojení k databázi budou uloženy v souboru .env z důvodu bezpečnosti. Tím, že token a připojovací link k databázi nebude obsažen přímo v hlavním kódu se zvýší bezpečnost. REST API server byl naprogramován v jazyce Python za pomocí frameworku Flask. Pro zlepšení přehlednosti a následného testování bylo využito vývojové prostředí Visual Studio Code obsahující terminál pro sledování HTTP dotazů. Pro komunikaci s databází MongoDB bude využívána knihovna PyMongo.

REST API se řídí topologií klient-server, kdy existuje server a klient. Oba jsou od sebe odděleni a komunikují spolu prostřednictvím internetu. Klienti a server mohou být naprogramováni v různých programovacích jazycích, protože jsou na sobě nezávislí. Vzájemně si rozumí, protože se řídí protokolem REST API. Klient obvykle odešle požadavek HTTP a tento požadavek zpracuje server a odpoví mu příslušnou odpovědí. Kompletní zdrojový kód programu je uveden v příloze 1.

4.6.1 Popis kódu

Program nejprve naimportuje knihovny potřebné pro správný chod programu. Voláním funkce load_dotenv() načteme proměnné prostředí, které obsahuje link pro připojení k databázi a vytvoříme aplikaci Flask. Nakonec načteme náš řetězec připojení k adrese URL MongoDB a přiřadíme jej do proměnné mongo_db_url.

```
import os
from flask import Flask, Response, request, jsonify, make_response, render_template
from dotenv import load_dotenv
from pymongo import MongoClient
from bson.json_util import dumps
from bson.objectid import ObjectId

load_dotenv()

app = Flask(__name__)
mongo_db_url = os.environ.get("MONGO_DB_CONN_STRING")
```

Obrázek 9 popis kódu REST API – import knihoven, funkce dotenv a připojení k databázi

Dále je zapotřebí vytvořit objekt MongoClient, který je použit ke komunikaci s naší databází MongoDB. Poté byl vytvořen odkaz na naši databázi sensors_db a přidán do proměnné db.

```
client = MongoClient(mongo_db_url)
db = client["sensors_db"]
```

Obrázek 10 popis kódu REST API – komunikace s databází

GET

V dalším kroku byla vytvořena funkce, která po přivolání koncového bodu (endpoint) poskytne odpověď. Funkce get_sensors() je rozšířena o funkci @app.get, což znaména, že

bude odpovídat na jakýkoli požadavek HTTP GET s cestou „/api/sensors“. Dále se extrahuje parametr dotazu sensor_id, a zkontroluje se, zdali byl opravdu zadán. Pokud ano, vytvoříme slovník Pythonu {"sensor_id": sensor_id}, který použijeme k filtrování naší kolekce senzorů. V opačném případě jsme předali prázdný objekt slovníku {}, takže se žádný filtr nepoužije.

Poté je zavolána funkce PyMongo find() na naši kolekci senzorů pomocí objektu filter a přiřadíme ji do seznamu Pythonu. Nakonec jsme vytvořili objekt Response, ve kterém převedeme výsledný dokument MongoDB BSON na objekt JSON a před jeho vrácením volajícímu nastavíme stav na 200 a typ MimeType na „application/json“.

```python
@app.get("/api/sensors")
def get_sensors():
    sensor_id = request.args.get("sensor_id")
    filter = {} if sensor_id is None else {"sensor_id": sensor_id}
    sensors = list(db.sensors.find(filter))
    print(sensors)

    response = Response(
        response=dumps(sensors), status=200, mimetype="application/json"
    )
    return response
```

Obrázek 11 popis kódu REST API – funkce GET

POST

Pro přidání dalších dat o sensorech do naší databáze byla vytvořena funkce add_sensor(). Tato funkce byla rozšířena o funkci @app.post, což znamená, že odpoví na HTTP požadavek POST.

Pomocí kódu request.json extrahujeme data JSON a zavoláme metodu PyMongo insert_one(), abychom přidali záznam do naší kolekce senzorů. Poté voláním funkce jsonify() vrátíme zprávu, že informace o senzorech byly úspěšně přidány.

```python
@app.post("/api/sensors")
def add_sensor():
    _json = request.json
    db.sensors.insert_one(_json)

    resp = jsonify({"message": "Sensor added successfully"})
    resp.status_code = 200
    return resp
```

Obrázek 12 popis kódu REST API – funkce POST

DELETE

K vymazání senzorů z naší kolekce bude použita funkce delete_sensor() s rozšířením o funkci @app.delete. Tato funkce se provede, když bude zavolána metodu HTTP Delete. Ta

zavolá funkci PyMongo delete_one(), kde dodáme třídu s názvem ObjectId. Pokud vše funguje podle očekávání, vrátím se odpověď JSON s úspěšnou zprávou.

```
@app.delete("/api/sensors/<id>")
def delete_sensor(id):
    db.sensors.delete_one({"_id": ObjectId(id)})

    resp = jsonify({"message": "Sensor deleted successfully"})
    resp.status_code = 200
    return resp
```

Obrázek 13 popis kódu REST API – funkce DELETE

PUT

Tato funkce slouží pro aktualizování dat o našich senzorech**.** Vytvořili jsme funkci update_sensor(), která používá cestu "/api/sensors/<id>", přičemž <id> je mapováno na ObjectId našeho dokumentu senzorů. Tato funkce zobrazení se zavolá, pokud je z klienta zahájen HTTP dotaz PUT.

Nejprve se extrahuje dokument JSON, který chcete aktualizovat, a zavolá funkci PyMongo update_one(), která vyžaduje ObjectId a používá funkci $set k nastavení konkrétních hodnot v dokumentu. Pokud vše proběhne v pořádku, odešle se zpět odpověď JSON, že aktualizace proběhla úspěšně.

```
@app.put("/api/sensors/<id>")
def update_sensor(id):
    _json = request.json
    db.sensors.update_one({"_id": ObjectId(id)}, {"$set": _json})

    resp = jsonify({"message": "Sensor updated successfully"})
    resp.status_code = 200
    return resp
```

Obrázek 14 popis kódu REST API – funkce PUT

Chybové hlášky

Pokud je náš server nedostupný nebo jsou špatně zadána vstupní data od klienta musí REST API nějak zareagovat. Z toho důvodu byly do kódu přidány odpovědi na chybové hlášky http 400, 404 a 500.

```python
@app.errorhandler(400)
def handle_400_error(error):
    return make_response(
        jsonify(
            {
                "errorCode": error.code,
                "errorDescription": "Bad request!",
                "errorDetailedDescription": error.description,
                "errorName": error.name,
            }
        ),
        400,
    )


@app.errorhandler(404)
def handle_404_error(error):
    return make_response(
        jsonify(
            {
                "errorCode": error.code,
                "errorDescription": "Resource not found!",
                "errorDetailedDescription": error.description,
                "errorName": error.name,
            }
        ),
        404,
    )


@app.errorhandler(500)
def handle_500_error(error):
    return make_response(
        jsonify(
            {
                "errorCode": error.code,
                "errorDescription": "Internal Server Error",
                "errorDetailedDescription": error.description,
                "errorName": error.name,
            }
        ),
        500,
    )
```

Obrázek 15 popis kódu REST API – chybové hlášky

4.7 Program ESP32

V této kapitole bude detailně popsán kód nahraný v mikrokontroleru ESP32, jehož úkolem je naměřené hodnoty senzorů odesílat do databáze a přijímat zprávy o stavu polovodičových relé. Data jsou odesílána do databáze skrze REST API server ve formátu JSON. Kompletní zdrojový kód programu je uveden v příloze 1.

4.7.1 Popis kódu

Jako první je třeba importovat potřebné knihovny ke správnému chodu kódu. Knihovna Arduino.h má nastarosti správné rozpoznání programu operačním systémem. Pro implementaci http klienta je zapotřebí importovat knihovnu HTTPClient.h, která zajišťuje připojení k Wi-Fi a pracuje s http metodami (GET, POST, PUT a další). Knihovna ArduinoJson.h umožňuje Arduino mikrokontrolérům generovat a zpracovávat JSON data. DHT.h knihovna poskytuje jednoduché rozhraní pro komunikaci s DHT senzory, které umožňuje čtení naměřených hodnot teploty a vlhkosti. Kód pro mikrokontroler ESP32 bude uložen v souboru app.py.

```
#include <Arduino.h>
#include <HTTPClient.h>
#include <ArduinoJson.h>
#include "DHT.h"
```

Obrázek 16 popis kódu ESP32 – knihovny

Připojení k síti Wi-Fi, které obsahuje jméno a heslo sítě. Proměnná base_rest_url obsahuje IP adresu a port REST API serveru.

```
const char *ssid = "SSID";
const char *password = "PASSWORD";

const char *base_rest_url = "http://10.0.1.30:5000/";
```

Obrázek 17 popis kódu ESP32 – připojení k síti Wi-Fi a REST API serveru

Abychom mohli volat náš koncový bod REST API z mikrokontroleru ESP32, budeme používat HTTPClient. V tomto bodě je definován WiFiClient a HTTPClient.

```
WiFiClient client;
HTTPClient http;
```

Obrázek 18 popis kódu ESP32 – definování

Dalším krokem je definování vstupů a výstupů mikrokontroleru ESP32, kde se nachází naše komponenty (DHT22, LED a relé modul). Zpoždění intervalu čtení je 5 sekund. To znamená, že budeme volat náš koncový bod rozhraní REST API a načítat záznam z databáze MongoDB každých 5 sekund.

```
unsigned long previousMillis = 0;
const long readInterval = 5000;

const int LED_PIN = 23;

const int RELAY_PIN_1 = 19;
const int RELAY_PIN_2 = 18;
const int RELAY_PIN_3 = 5;
const int RELAY_PIN_4 = 17;

char dhtObjectId[30];
#define DHTPIN 32

#define DHTTYPE DHT22

DHT dht(DHTPIN, DHTTYPE);
```

Obrázek 19 popis kódu ESP32 – definování I/O a nastavení časového intervalu čtení

Definování datové struktury pro záznamy
Jakmile mikrokontroler ESP32 získá odpověď z REST API serveru, můžeme vrácené záznamy namapovat pomocí struktury v jazyce C/C++.

HTTP metoda GET(LED)
Nejprve je potřeba vypočítat velikost dokumentu JSON pomocí nástroje ArduinoJson JSON Assistant. Tato hodnota byla přiřazena do proměnné JSON_DOC_SIZE.

Dále se zavolá náš koncový bod REST API a předá se přiřazené sensor_id.

Poté se odpověď deserializuje a přiřadí se do lokální proměnné StaticJsonDocument<JSON_DOC_SIZE> doc; a vrátí se volajícímu.

Funkce extractLEDConfiguration zavolá předchozí funkci callHTTPGet a zkontroluje výsledek. Z databáze očekáváme pouze jeden záznam nebo dokument.

Pomocí dříve definované struktury LED vytvoříme lokální proměnnou tohoto typu a naplníme ji zkopírováním hodnot do pole pomocí příkazu strcpy funkce C++. Tuto dočasnou lokální proměnnou pak vrátíme volajícímu.

```
struct DHT22Readings
{
  float temperature;
  float humidity;
};

struct LED
{
  char sensor_id[10];
  char description[20];
  char location[20];
  bool enabled;
  char type[20];
  char status[10];
};

struct RELAY
{
  char sensor_id[10];
  char description[20];
  char location[20];
  bool enabled;
  char type[20];
  char in1[10];
  char in2[10];
  char in3[10];
  char in4[10];
};
```

Obrázek 20 popis kódu ESP32 – definování datové struktury

```
const int JSON_DOC_SIZE = 384;

StaticJsonDocument<JSON_DOC_SIZE> callHTTPGet(const char *sensor_id)
{
  char rest_api_url[200];

  sprintf(rest_api_url, "%sapi/sensors?sensor_id=%s", base_rest_url, sensor_id);
  Serial.println(rest_api_url);

  http.useHTTP10(true);
  http.begin(client, rest_api_url);
  http.addHeader("Content-Type", "application/json");
  http.GET();

  StaticJsonDocument<JSON_DOC_SIZE> doc;

  DeserializationError error = deserializeJson(doc, http.getStream());

  if (error)
  {
    Serial.print("deserializeJson() failed: ");
    Serial.println(error.c_str());
    return doc;
  }

  http.end();
  return doc;
}

LED extractLEDConfiguration(const char *sensor_id)
{
  StaticJsonDocument<JSON_DOC_SIZE> doc = callHTTPGet(sensor_id);
  if (doc.isNull() || doc.size() > 1)
    return {};
  for (JsonObject item : doc.as<JsonArray>())
  {

    const char *sensorId = item["sensor_id"];
    const char *description = item["description"];
    const char *location = item["location"];
    bool enabled = item["enabled"];
    const char *type = item["type"];
    const char *status = item["status"];

    LED ledTemp = {};
    strcpy(ledTemp.sensor_id, sensorId);
    strcpy(ledTemp.description, description);
    strcpy(ledTemp.location, location);
    ledTemp.enabled = enabled;
    strcpy(ledTemp.type, type);
    strcpy(ledTemp.status, status);

    return ledTemp;
  }
  return {};
}
```

Obrázek 21 popis kódu ESP32 – HTTP GET metoda (LED)

HTTP metoda GET(Relay)

Funkce extractRelayConfiguration() je téměř podobná funkci extractLEDConfiguration() a jediným rozdílem je návratový typ, kterým je struct RELAY. Kromě toho je volán stejný koncový bod rozhraní REST API, ale předává se jiné sensor_id, jak je uvedeno níže.

```
RELAY extractRelayConfiguration(const char *sensor_id)
{
  StaticJsonDocument<JSON_DOC_SIZE> doc = callHTTPGet(sensor_id);
  if (doc.isNull() || doc.size() > 1)
    return {}; // or RELAY{}
  for (JsonObject item : doc.as<JsonArray>())
  {
    const char *sensorId = item["sensor_id"];
    const char *description = item["description"];
    const char *location = item["location"];
    bool enabled = item["enabled"];
    const char *type = item["type"];

    JsonObject values = item["values"];
    const char *valuesIn1 = values["in1"];
    const char *valuesIn2 = values["in2"];
    const char *valuesIn3 = values["in3"];
    const char *valuesIn4 = values["in4"];

    RELAY relayTemp = {};
    strcpy(relayTemp.sensor_id, sensorId);
    strcpy(relayTemp.description, description);
    strcpy(relayTemp.location, location);
    relayTemp.enabled = enabled;
    strcpy(relayTemp.type, type);
    strcpy(relayTemp.in1, valuesIn1);
    strcpy(relayTemp.in2, valuesIn2);
    strcpy(relayTemp.in3, valuesIn3);
    strcpy(relayTemp.in4, valuesIn4);

    return relayTemp;
  }
  return {};
}
```

Obrázek 22 popis kódu ESP32 – HTTP GET metoda (Relay)

HTTP PUT (DHT22)

K aktualizování záznamů ze senzoru DHT22 je nutné zavolat koncový bod rozhraní REST API, který bude aktualizovat záznamy v databázi MongoDB. K aktualizaci konkrétního záznamu je potřeba ObjectId záznamu k určení, který záznam aktualizovat.

Dále se musí připravit objekt JSON, který se bude předávat při volání rozhraní API pomocí HTTPClient. Pomocí knihovny ArduinoJSON se zakóduje objekt JSON, který bude obsahovat údaje senzoru DHT22. Pomocí funkce HTTPClient.PUT() se odešle požadavek HTTP PUT spolu s serializovaným JSON dokumentem.

Následující funkce slouží k ovládání komponent (relé, LED a DHT22) a převádí HIGH a LOW na hodnoty kompatibilní s Arduinem.

```
DHT22Readings readDHT22()
{
  float humidity = dht.readHumidity();
  float temperatureInC = dht.readTemperature();
  return {temperatureInC, humidity};
}
int convertStatus(const char *value)
{
  if (strcmp(value, "HIGH") == 0)
  {
    Serial.println("Setting LED to HIGH");
    return HIGH;
  }
  else
  {
    Serial.println("Setting LED to LOW");
    return LOW;
  }
}

void setLEDStatus(int status)
{
  Serial.printf("Setting LED status to : %d", status);
  Serial.println("");
  digitalWrite(LED_PIN, status);
}

void setRelayStatus(int pin, int status)
{
  Serial.printf("Setting Relay %d status to : %d", pin, status);
  Serial.println("");
  digitalWrite(pin, status);
}

void turnOffAllRelay()
{
  digitalWrite(RELAY_PIN_1, HIGH);
  digitalWrite(RELAY_PIN_2, HIGH);
  digitalWrite(RELAY_PIN_3, HIGH);
  digitalWrite(RELAY_PIN_4, HIGH);
}
```

Obrázek 23 popis kódu ESP32 – HTTP PUT metoda funkce

```
void sendDHT22Readings(const char *objectId, DHT22Readings dhtReadings)
{
  char rest_api_url[200];

  sprintf(rest_api_url, "%sapi/sensors/%s", base_rest_url, objectId);
  Serial.println(rest_api_url);

  String jsondata = "";
  StaticJsonDocument<JSON_DOC_SIZE> doc;
  JsonObject readings = doc.createNestedObject("readings");
  readings["temperature"] = dhtReadings.temperature;
  readings["humidity"] = dhtReadings.humidity;

  serializeJson(doc, jsondata);
  Serial.println("JSON Data...");
  Serial.println(jsondata);

  http.begin(client, rest_api_url);
  http.addHeader("Content-Type", "application/json");

  int httpResponseCode = http.PUT(jsondata);
  if (httpResponseCode > 0)
  {
    String response = http.getString();
    Serial.println(httpResponseCode);
    Serial.println(response);
  }
  else
  {
    Serial.print("Error on sending POST: ");
    Serial.println(httpResponseCode);
    http.end();
  }
}

void getDHT22ObjectId(const char *sensor_id)
{
  StaticJsonDocument<JSON_DOC_SIZE> doc = callHTTPGet(sensor_id);
  if (doc.isNull() || doc.size() > 1)
    return;
  for (JsonObject item : doc.as<JsonArray>())
  {
    Serial.println(item);
    const char *objectId = item["_id"]["$oid"];
    strcpy(dhtObjectId, objectId);

    return;
  }
  return;
}

DHT22Readings readDHT22()
{
  float humidity = dht.readHumidity();
  float temperatureInC = dht.readTemperature();
  return {temperatureInC, humidity};
}
```

Obrázek 24 popis kódu ESP32 – HTTP PUT metoda (DHT22)

Funkce setup() během spouštění provádí následující úkony:

- Připojí se k WiFi
- Začne komunikovat se senzorem DHT22
- Nastaví piny GPIO pro LED a relé a na začátku vypne všechna relé.
- Získá z databáze DHT22 ObjectId, aby se mohly ukládat naměřené hodnoty, a to voláním funkce *getDHT22ObjectId("dht22_1");*

```
void setup()
{
  Serial.begin(9600);
  for (uint8_t t = 2; t > 0; t--)
  {
    Serial.printf("[SETUP] WAIT %d...\n", t);
    Serial.flush();
    delay(1000);
  }

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED)
  {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());

  dht.begin();
  getDHT22ObjectId("dht22_1");
  pinMode(LED_PIN, OUTPUT);
  pinMode(RELAY_PIN_1, OUTPUT);
  pinMode(RELAY_PIN_2, OUTPUT);
  pinMode(RELAY_PIN_3, OUTPUT);
  pinMode(RELAY_PIN_4, OUTPUT);
  turnOffAllRelay();
}
```

Obrázek 25 popis kódu ESP32 – funkce setup

Funkce loop() je standardní funkcí Arduina, ve které v pětisekundových intervalech komunikuje s REST API serverem.

Nejprve je získána konfigurace pro LED voláním funkce *extractLEDConfiguration()*. Dále se zkontroluje návratový stav a zjistí, zda je *HIGH* nebo *LOW*, a podle toho se nastaví hodnota LED. Pak je získána konfigurace pro relé voláním funkce *extractRelayConfiguration("relay_1");*. Poté se zkontroluje návratová hodnota proměnných *in1*, *in2*, *in3* a *in4* a podle toho se nastaví hodnota relé.

```
void loop()
{
  unsigned long currentMillis = millis();

  if (currentMillis - previousMillis >= readInterval)
  {
    previousMillis = currentMillis;

    Serial.println("---------------");
    LED ledSetup = extractLEDConfiguration("led_1");
    Serial.println(ledSetup.sensor_id);
    Serial.println(ledSetup.description);
    Serial.println(ledSetup.location);
    Serial.println(ledSetup.enabled);
    Serial.println(ledSetup.type);
    Serial.println(ledSetup.status);
    setLEDStatus(convertStatus(ledSetup.status));
    Serial.println("---------------");
    RELAY relaySetup = extractRelayConfiguration("relay_1");
    Serial.println(relaySetup.sensor_id);
    Serial.println(relaySetup.description);
    Serial.println(relaySetup.location);
    Serial.println(relaySetup.enabled);
    Serial.println(relaySetup.type);
    Serial.println(relaySetup.in1);
    Serial.println(relaySetup.in2);
    Serial.println(relaySetup.in3);
    Serial.println(relaySetup.in4);
    setRelayStatus(RELAY_PIN_1, convertStatus(relaySetup.in1));
    setRelayStatus(RELAY_PIN_2, convertStatus(relaySetup.in2));
    setRelayStatus(RELAY_PIN_3, convertStatus(relaySetup.in3));
    setRelayStatus(RELAY_PIN_4, convertStatus(relaySetup.in4));
    Serial.println("---------------");
    Serial.println("Sending latest DHT22 readings");
    DHT22Readings readings = readDHT22();
    if (isnan(readings.humidity) || isnan(readings.temperature))
    {
      Serial.println(F("Failed to read from DHT sensor!"));
      return;
    }
    Serial.print("Temperature :: ");
    Serial.println(readings.temperature);
    Serial.print("Humidity :: ");
    Serial.println(readings.humidity);
    sendDHT22Readings(dhtObjectId, readings);
  }
}
```

Obrázek 26 popis kódu ESP32 – funkce loop

4.8 Testování REST API

Funkčnost REST API serveru bude testována použitím platformy cURL

Zkusme získat všechny senzory v naší kolekci provedením níže uvedeného příkazu. Ten vrátí všechny dokumenty, které jsme vložili do naší kolekce senzorů.

```
$ curl http://localhost:5000/api/sensors

[{"_id": {"$oid": "6426689cb39a9135e7b7e63a"}, "sensor_id": "led_1", "description": "This is o
```

Obrázek 27 Testování REST API – Získání všech záznamů v kolekci

Pokud chceme získat konkrétní jeden záznam, který odpovídá konkrétnímu *sensor_id*, použijeme příkaz uvedený níže.

```
$ curl http://localhost:5000/api/sensors?sensor_id=led_1

[{"_id": {"$oid": "6426689cb39a9135e7b7e63a"}, "sensor_id": "led_1", "description": "This is
```

Obrázek 29 Testování REST API – získání záznamu led_1

Pro otestování metody POST, tedy vložení nového záznamu do databáze můžeme otestovat pomocí příkazu uvedeného níže. Příkaz nám vloží do kolekce *sensors* nový záznam s určenými daty.

```
$ curl -X POST http://localhost:5000/api/sensors -H 'Content-Type: application/json' -d '{"sens

{
  "message": "Sensor added successfully"
}
```

Obrázek 28 Testování REST API – vložení nového záznamu

Pokud chceme otestovat koncový bod API pro odstranění senzoru pomocí příkazu curl, spustíme níže uvedený příkaz. Zadaný ObjectId by měl být přítomen v naší kolekci senzorů MongoDB. Pro otestování, zdali byl záznam opravdu smazán můžeme použít dotaz GET.

```
curl -X "DELETE" 'http://localhost:5000/api/sensors/6426d2a6fcd3368dd29f7f56'
```

Obrázek 30 Testování REST API – smazání záznamu

Provedením níže uvedeného curl příkazu otestujeme metodu PUT, tedy aktualizování stávajícího záznamu.

```
curl -X PUT 'http://localhost:5000/api/sensors/6426689cb39a9135e7b7e63a' -H 'Content-Type: app
```

Obrázek 31 Testování REST API – metoda PUT

Z testování bylo zjištěno, že veškeré potřebné metody REST API serveru jsou funkční. Z toho plyne, že REST API bude možné nasadit na komunikaci přímo s mikrokontrolerem ESP 32.

4.9 Webová aplikace

Webová aplikace bude primárně vyvíjena pro mobilní zařízení tedy pro velikost 360×640 a to z důvodu, že bude využívána zejména těmito zařízeními. Webová aplikace bude sloužit k získání informací o stavu fotovoltaické elektrárny (stav baterie, předpověď počasí, nastavení nočního proudu, sledování výkonu zátěže a fotovoltaické elektrárny). Všechny tyto informace se budou dát vyčíst z dashboardu obsahujícího grafy. Kód webové aplikace bude uložen ve 3 souborech. Kód HTML bude uložen v soubor Index.html. Kód kaskádových stylů bude uložen v Style.css a kód s Javascriptem bude uložen v souboru Script.js.

4.9.1 Návrh webové aplikace

V této kapitole bude vytvořen návrh webové aplikace, který je vyobrazen na obrázku 28.

Seznam prvků:

- Přihlašovací obrazovka
- Dlouhodobá předpověď počasí
- Měrný graf znázorňující stav baterie
- Spojnicový graf pro sledování výkonu fotovoltaické elektrárny
- Spojnicový graf pro sledování výkonu zátěže
- Posuvný pás s dvěma body pro nastavení intervalu nočního proudu
- On/off tlačítka pro manuální spínání relé

Přihlášení
Po zadání adresy webové stránky je uživateli zobrazeno dialogové okno pro přihlášení k dashboardu. Po správném vyplnění uživatelského jména a hesla uživatel klikne na tlačítko přihlásit a je přesměrován na úvodní stránku s dashboardem. Při špatném zadání hesla nebo uživatelského jména se zobrazí chybová hláška „chybně zadané heslo nebo uživatelské jméno“. Uživatel může v případě zapomenutého hesla využít odkaz „zapomenuté heslo“.

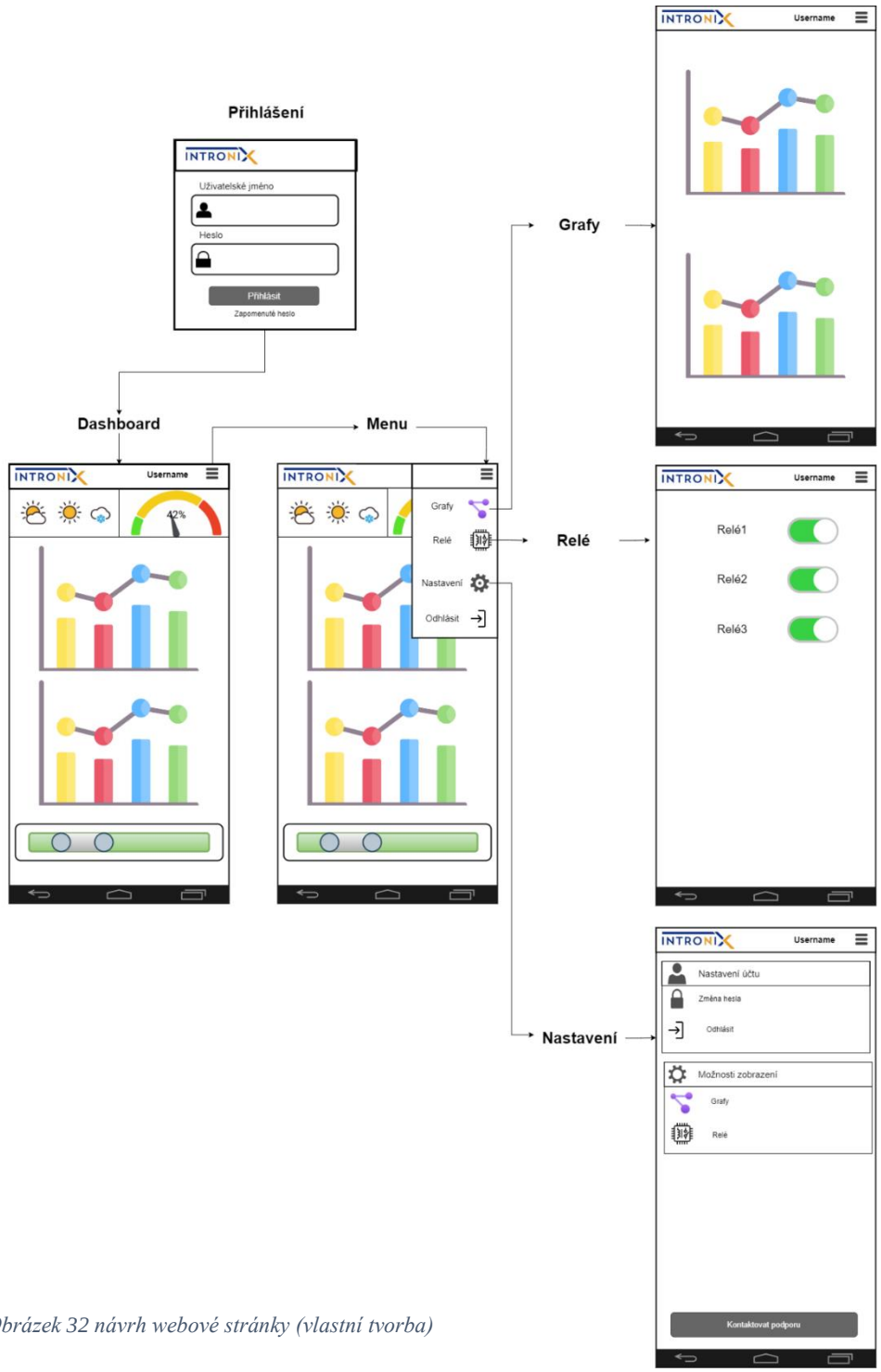
Dashboard
Po úspěšném přihlášení je uživatel přesměrován na hlavní stránku s dashboardem. Stránka obsahuje informace o dlouhodobé předpovědi počasí, měrný graf pro stav baterie, lineární graf informující o výkonu fotovoltaické elektrárny, lineární graf informující o výkonu zátěže a posuvník pro nastavení intervalu nočního proudu.

Záhlaví stránky obsahuje logo firmy Intronix, které odkazuje na webové stránky společnosti. V pravé části stránky se nachází rozklikávací menu obsahující odkazy. Menu bude obsahovat odkazy na grafy, relé, nastavení a odhlášení.

Grafy
Tato stránka obsahuje detailnější pohled na grafy s mnohými prvky zlepšující orientaci v grafech. Pro zlepšení orientace je možné data sledovat v různém časovém rozmezí. Toho lze docílit přepínáním časového rozmezí (minuty, hodiny, dny, měsíce) nebo scrollováním v grafu.

Relé
Tato stránka zobrazuje stav výstupů, zdali jsou zapnuté či nikoli. Jednotlivé výstupe také lze zapnout či vypnout manuálně.

Nastavení
Tato stránka umožňuje uživateli změnu hesla, odhlášení a možnost změnit zobrazení stránky grafů a relé.



Obrázek 32 návrh webové stránky (vlastní tvorba)

4.9.2 Program webové aplikace

V této kapitole bude podrobně popsán kód webové aplikace. Webová stránka byla naprogramována pomocí jazyka HTML, CSS a Javascriptu. Jedná se pouze o front end webové stránky, back end webové stránky bude dodělám v pozdějším vývoji projektu.

Webová aplikace je tvořena celkem třemi soubory (HTML, CSS, Javascript). Primárním souborem je soubor index.php, který definuje základní strukturu webové aplikace. V tomto souboru jsou linkovány soubory kaskádových stylů a javascriptu. Kaskádové styly jsou využity pro popis způsobu zobrazení elementů. Javascript přidává nezbytné funkce pro obrazovku přihlášení a vykreslování grafů. Kompletní zdrojový kód programu je uveden v příloze 1.

HTML Struktura

- **Přihlašovací formulář:** Na začátku je skrytý formulář pro přihlášení s textovými poli pro jméno a heslo a tlačítkem pro přihlášení. Pokud uživatel zadá správné přihlašovací údaje (v tomto případě "admin" pro obě pole), JavaScriptová funkce zobrazí dashboard.
- **Dashboard:** Obsahuje hlavičku s logem a menu, předpověď počasí na tři dny s ikonami, měrný graf pro zobrazení stavu baterie, dva lineární grafy pro výkon fotovoltaické elektrárny a boileru, a posuvník pro nastavení intervalu nočního proudu.

Body HTML souboru je vyobrazeno na obrázku 33.

```
<title>Dashboard</title>
</head>
<body>
    <div id="loginPage" class="login-page">
        <div class="login-form">
            <form id="loginForm">
                <input type="text" id="username" placeholder="Jméno" required>
                <input type="password" id="password" placeholder="Heslo" required>
                <input type="submit" value="Přihlásit se">
            </form>
            <div id="errorMessage" style="color: red;"></div>
        </div>
    </div>
    <div id="dashboardPage" class="dashboard-page hidden">
        <div class="header">
            <div class="logo">
                <a href="https://www.intronix.cz/"><img src="Intronix_logo.png" alt="INTRONIX Logo"></a>
            </div>
            <div class="menu">
                <img src="menu.png" alt="Menu" />
                <div class="menu-content">
                    <a href="#">Profil</a>
                    <a href="#">Nastavení</a>
                    <a href="#" onclick="logout()">Odhlásit</a>
                </div>
            </div>
        </div>
        <div class="weather-forecast">
            <div class="weather-day">
                <img src="sunny.png" alt="Sunny" class="weather-icon">
                <p><strong>Pondělí:</strong> 20°C, Jasno</p>
            </div>
            <div class="weather-day">
                <img src="cloudy.png" alt="Cloudy" class="weather-icon">
                <p><strong>Úterý:</strong> 18°C, Oblačno</p>
            </div>
            <div class="weather-day">
                <img src="rainy.png" alt="Rainy" class="weather-icon">
                <p><strong>Středa:</strong> 16°C, Déšť</p>
            </div>
        </div>
        <div class="battery-gauge">
            <canvas id="batteryCapacityGauge"></canvas>
        </div>
        <h2 class="battery-gauge-title">Stav baterie</h2>
        <div class="charts">
            <canvas id="solarPowerChart"></canvas>
            <canvas id="boilerPowerChart"></canvas>
        </div>
        <div class="Slider">
            <h2 class="slider-title">Nastavení intervalu nočního proudu</h2>
            <div id="timeSlider"></div>
        </div>
    </div>
<script src="script.js"></script>
</body>
</html>
```

Obrázek 33 Obsah souboru index.html

CSS

- Přidává základní styly pro celou stránku, jako je nastavení fontu, margin a padding.
- Definuje styly pro přihlašovací formulář, včetně rozměrů a stínů.
- Styly pro hlavičku určují, jak bude logo a menu zobrazeno.

- Předpověď počasí, gauge graf, lineární grafy a slider jsou stylovány tak, aby byly vizuálně příjemné a responzivní.
- Speciální stylizace pro canvas elementy zajišťuje, že grafy jsou řádně vycentrované a mají maximální šířku.

```css
body, html {
    margin: 0;
    padding: 0;
    font-family: Arial, sans-serif;
}
.hidden {
    display: none;
}
.login-form {
    width: 360px;
    padding: 20px;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
    background-color: white;
    margin: 100px auto;
}
input[type="text"], input[type="password"], input[type="submit"] {
    width: calc(100% - 20px);
    padding: 10px;
    margin: 10px 0;
    display: block;
}
input[type="submit"] {
    background-color: #007bff;
    color: white;
    border: none;
    cursor: pointer;
}
.header {
    position: relative;
    background-color: #ffffff;
    color: #000;
    padding: 10px 20px;
}
.menu {
    position: absolute;
    top: 10px;
    right: 10px;
    z-index: 100;
}
.menu img {
    width: 30px;
    height: auto;
    cursor: pointer;
}
.menu-content {
    display: none;
    position: absolute;
    right: 0;
    background-color: #f9f9f9;
    min-width: 160px;
    box-shadow: 0px 8px 16px 0px rgba(0,0,0,0.2);
    z-index: 1;
}
.menu-content a {
    color: black;
    padding: 12px 16px;
    text-decoration: none;
    display: block;
}
.menu:hover .menu-content {
    display: block;
}
canvas {
    max-width: 400px;
    margin: auto;
    display: block;
}
#timeSlider {
  margin: 5px auto 0 auto;
```

Obrázek 34 Obsah souboru style.css 1

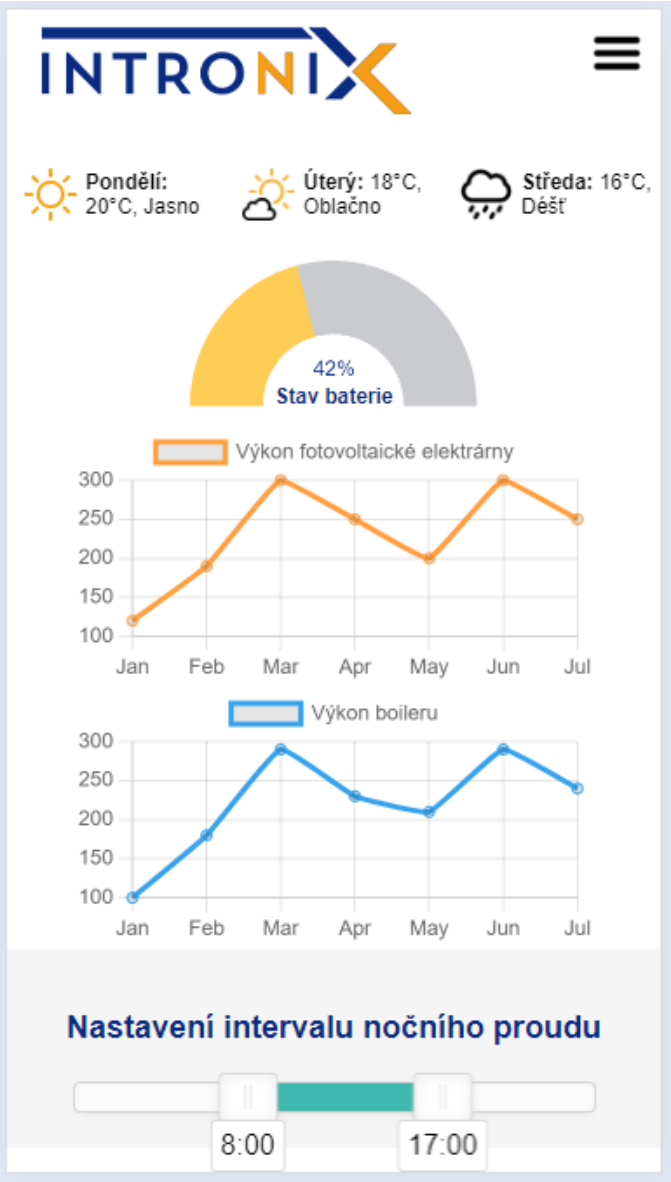
```css
.charts {
margin-top: 5px;
}
.weather-forecast {
display: flex;
justify-content: center;
flex-wrap: wrap;
position: absolute;
top: 70px;
left: 0;
width: 100%;
padding: 5px;
background-color: rgba(255, 255, 255, 0.9);
z-index: 2;
}
.weather-day {
display: flex;
align-items: center;
flex: 1 0 30%;
margin: 5px;
font-size: 12px;
}
.weather-icon {
width: 30px;
height: 30px;
margin-right: 5px;
}
.Slider {
text-align: center;
padding: 20px 0;
background-color: #f5f5f5;
margin-top: 1px;
}
.slider-title {
margin-bottom: 20px;
color: #0b2b81;
font-size: 18px;
}
.noUi-tooltip {
bottom: -30px !important;
}
top: auto !important;
.battery-gauge {
position: relative;
text-align: center;
margin-top: 60px;
width: 100%;
height: auto;
}
.battery-gauge canvas {
width: 50% !important;
height: auto !important;
}
.battery-gauge-title {
color: #0b2b81;
font-size: 12px;
text-align: center;
margin-top: -13px
```

Obrázek 35 Obsah souboru style.css 2

JavaScript

- Skript ovládá funkčnost přihlášení a zobrazuje chybovou zprávu v případě nesprávných přihlašovacích údajů.
- Funkce showDashboard skryje přihlašovací stránku a zobrazí dashboard.
- Funkce logout opět zobrazí přihlašovací stránku a skryje dashboard.
- renderCharts inicializuje grafy využívající Chart.js a noUiSlider pro vytvoření lineárních grafů a sliderů.
- Vlastní inicializace pro noUiSlider definuje jeho vzhled a chování, včetně rozsahu hodnot a kroků.
- Měrný graf je nakonfigurován s použitím Chart.js, kde onComplete funkce přidá textovou etiketu s procentuální hodnotou přímo do grafu, což uživateli umožňuje ihned vidět stav baterie.

Z důvodu obsáhlosti souboru script.js bude uveden kód programu pouze v příloze.



Obrázek 36 Dashboard webové aplikace

5 Výsledky a diskuse

Tato část diplomové práce shrnuje dosažené poznatky při jejím tvoření.

5.1 Shrnutí

Úvodem práce byly analýzami softwarových a hardwarových nástrojů zvoleny vhodné komponenty potřebné k implementaci internetu věcí pro solární systémy. Na základě těchto poznatků a podmínek stanovených firmou Intronix vznikl IoT model a zkušební zapojení pro následovné testování systému. Následně byla popsána požadována funkcionalita na vybrané komponenty. V další části práce byly detailně popsány programy jednotlivých komponent (ESP32 a REST API). Poslední část práce se zabývala návrhem webové aplikace. Vznikly wireframy potřebné pro vývoj aplikace, díky čemuž mohl být zhotoven frontend webu.

5.2 Budoucí vývoj

V rámci diplomové práce byl zhotoven základ softwaru systému pro efektivní řízení fotovoltaických elektráren. Tento software bude muset být dále vyvíjen a pozměněn z testovacího režimu na stav, kdy bude možné tento software reálně nasadit jako komerční řešení. To zahrnuje několik dílčích bodů uvedených níže.

5.2.1 Migrace na server

Aktuálně pro testovací účely je umístěný REST API na cloudové platformě Google cloud. Dá se předpokládat, že REST API bude migrován na server vlastněný společnostní Intronix s.r.o. lokalizovaný přímo v serverovně firmy. To může zajistit lepší správu nad serverem a snížit tak možná rizika spojená s cloudovými službami.

5.2.2 Vývoj softwaru

Odladění kódu
V následujících měsících se bude upravovat kód, tak aby vyhovoval požadavkům společnosti Intronix a mohl být nasazen pro komerční účely. Programy se aktuálně nachází ve stavu, kdy je možné testovat komunikaci a funkcionalitu jednotlivých vstupů a výstupů. Bude však nutné upravit program tak aby odpovídaly reálným požadavkům při nasazení u zákazníků.

Řídící algoritmus
Do budoucna se předpokládá vývoj řídícího algoritmu, který bude schopen v návaznosti na naměřených hodnotách sám vyhodnotit kdy sepnout zátěž systému, a tak zefektivnit výtěžnost fotovoltaické elektrárny. V potaz se budu brát předpověď počasí, interval nočního proudu, stav zátěže (v případě boileru to může být jeho teplota).

Webová aplikace
Webová aplikace je aktuálně ve stavu funkčního front endu. Do budoucna bude nutné propojit webovou aplikaci s databází. Webová aplikace bude z databáze čerpat data o uživatelích, která využije v přihlašovacím formuláři. Dále také budu data využity ve třech

grafech pro zobrazení stavu baterie, výkon fotovoltaické elektrárny a výkon boileru. V poslední řadě budou data využity pro určení intervalu nočního proudu.

Dále bude nutné získávat data o předpovědi počasí. Toho bude docíleno za použití API od společnosti OpenWeather, která nabízí toto API zdarma.

5.2.3 Vývoj hardwaru

Tato část se zabývá budoucím vývojem hardwarové části wattrouteru.

Deska plošného spoje
Tato práce se zabývá pouze softwarovou částí dané problematiky, proto bude nutné navrhnout desku plošného spoje, která bude obsahovat mikrokontroler ESP32 a další komponenty potřebné pro správný chod. Deska plošného spoje bude vyvíjena a osazena společností Intronix a také dále testována.

Krabička wattrouteru
Pro umístění desky plošného spoje by mohla být vybrána krabička vyobrazena na obrázku 34. Tato krabička je vhodná jak svou velikostí, tak i možnosti montáže na lištu DIN. To by zajišťovalo pohodlnou montáž přímo do útrob rozvaděče fotovoltaické elektrárny, bez nutnosti montáže krabičky šrouby. Vršek krabičky by mohl také obsahovat displej pro zobrazení informací o stavu wattrouteru.

Obrázek 37 krabička wattrouteru

6 Závěr

Cílem práce bylo implementovat IoT pro solární systémy. Pro splnění tohoto cíle bylo nutné v teoretické části analyzovat vhodné a softwarové komponenty a charakterizovat problematiku internetu věcí a solárních systémů. V úvodu práce bylo nezbytné udělat průzkum trhu a zjistit jaké vlastnosti přináší konkurenční řešení. Po vyhodnocení průzkumu trhu byly zjištěny klíčové vlastnosti, které by měl řídící systém pro solární systémy splňovat. Následně byly tyto informace analyzovány společně s firmou Intronix s.r.o. a byly stanoveny požadavky na daný systém. Dalším krokem byla analýzy softwarových a hardwarových komponent pro úspěšnou implementaci internetu věcí pro solární systémy, kde bylo nutné správně zvolit řídící jednotku a softwarovou část systému. Výsledky této analýzy byly potvrzeny Saatyho metodou párového porovnání. Analýzami byl zjištěn nejvhodnější hardware a software pro toto řešení.

Jako řídící jednotka byl vybrán mikrokontroler ESP32, a to hlavně z důvodu nízké spotřeby elektrické energie. Pro naprogramování mikrokontroleru byl zvolen programovací jazyk C++ a vývojové prostředí Visual Studio Code. Komunikaci systému zajišťuje REST API server naprogramovaný v programovacím jazyce Python. K ukládání dat byla zvolena databáze časových řad MongoDB.

Díky předešlým krokům mohlo dojít k samotné implementaci internetu věcí pro solární systémy, která zahrnuje vytvoření programu mikrokontroleru ESP32, REST API, webové aplikace a vytvoření databáze časových řad pro ukládání záznamů ze senzorů.

Úvodem samotné implementace vzniklo testovací zapojení pro snadné ověření funkce softwarové části. Toto zapojení obsahuje vstupní a výstupní zařízení pro ověření komunikace. Prvotním úkolem softwarové části práce bylo vytvoření a zabezpečení databáze časových řad MongoDB. Databáze byla využita pro ukládání naměřených hodnot vstupních zařízení. Pro komunikaci mezi databází a mikrokontroler vznikl REST API server. Ke komunikaci dochází v obou směrech, kdy mikrokontroler odesílá data senzorů a přijímá informace o změnách na výstupních zařízeních. Pro lepší interakci uživatele se systémem vznikl dashboard ve webové aplikaci, který informuje uživatele o stavu fotovoltaické elektrárny a umožňuje natavení proměnných ovlivňujících efektivitu systému. Webová aplikace je dostupná i mimo lokální síť a tím umožňuje uživateli provádět tyto úkony na dálku. Dashboard webové aplikace obsahuje dlouhodobou předpověď počasí, dva lineárních grafy s přehledy o výkonech zátěže a elektrárny, měrný graf pro sledování kapacity baterie a posuvník pro nastavení intervalu nočního proudu. V záložce relé uživatel nalezne manuální ovládání pro sepnutí výstupních zařízení.

Výsledkem je systém postavený na jednotce ESP32, který je schopen komunikovat skrze REST API server s databází a webová aplikace s dashboardem. Vzniklá softwarová část bude základem pro wattrouter, které bude mít na starosti efektivní využití fotovoltaické elektrárny.

Do budoucna bude nutné tento prototyp upravit tak aby vyhovoval reálným požadavkům solárních systémů. Tyto úpravy zahrnují zejména změnu vstupních a výstupních zařízen, kdy bude nutné program přizpůsobit jejich požadavkům. Také bude nutné dodělat backendovou část webové aplikace, k získávání dat. Systém bude dále vyvíjen a do budoucna se dá počítat s komerčním využitím.

7 Bibliografie

[1] AVAST. Blog.avast. online. In: ELDER, Jeff. AVAST. *How Kevin Ashton named The Internet of Things*. 2019. Dostupné z: https://blog.avast.com. [cit. 2024-03-31].

[2] TECHTARGET. Internet of things (IoT). online. In: TECHTARGET. *What is the internet of things (IoT)?*. 2023. Dostupné z: https://www.techtarget.com. [cit. 2024-03-31].

[3] IOT ANALYTICS. IoT Analytics. online. In: IOT ANALYTICS. *Why the Internet of Things is called Internet of Things: Definition, history, disambiguation*. 2014. Dostupné z: https://iot-analytics.com. [cit. 2024-03-31].

[4] R.M, SWANSON. *Photovoltaics Power Up*. 1. Science, 2009. ISBN není uvedeno.

[5] *Co je Wattrouter? A proč ho použít?*. online. In: Co je Wattrouter? A proč ho použít?. 2021. Dostupné z: https://www.nemakej.cz. [cit. 2024-03-31].

[6] GARCIA-MOLINA, Hector; D. ULLMAN, Jeffrey a WIDOM, Jennifer. *Database Systems*. 2. Department of Computer Science Stanford University: Pearson Education Inc., 2009. ISBN ISBN D-13-bQb?Dl-fl 178-0-13-b0b?01-b.

[7] DAYLEY, Brad a DALEY, Caleb. *MongoDB and Angular web development*. 2. Boston: Addison-Wesley, 2017. ISBN 9780134655536.

[8] *Wearecommunity*. online. In: NAIR, Anurag. Introduction to InfluxDB: A time-series database. 2021. Dostupné z: https://wearecommunity.io. [cit. 2024-03-31].

[9] MONGODB. *What Is MongoDB?*. online. In: MONGODB. Mongodb. není uvedeno. Dostupné z: https://www.mongodb.com/what-is-mongodb. [cit. 2024-02-25].

[10] MDN CONTRIBUTORS. *JSON*. online. In: MDN CONTRIBUTORS. JSON. 2024. Dostupné z: https://developer.mozilla.org. [cit. 2024-03-31].

[11] COOKSEY, Brian. *An introduction to APIs*. 1. Zapier, Inc., 2014. ISBN není uvedeno.

[12] CAIN, Jerry. *Introduction to HTTP*. PDF. 1. 2023.

[13] CAMERON, Neil. *Electronics Projects with the ESP8266 and ESP32: Building web pages, applications, and WiFi enabled devices*. 1. Edinburgh: Apress, 2021. ISBN 978-1-4842-63358.

[14] MALÝ, Martin. *Hradla, volty, jednočipy: úvod do bastlení*. 1. CZ.NIC. Praha: CZ.NIC, z.s.p.o., 2017. ISBN ISBN978-80-88168-23-2.

[15] ARTHUR, James. *Arduino: The complete guide to Arduino for beginners, including projects, tips, tricks, and programming!*. 2. Ingram Publishing, 2019. ISBN 1925989704.

[16] BANZI, Massimo a SHILOH, Michael. *Getting started with Arduino*. Third edition. Sebastopol, CA: Maker Media, 2015. ISBN 978-144-9363-338.

[17] SKALICKÝ, Petr. *Procesory řady 8051*. 2. rozš. vyd. Praha: BEN - technická literatura, 1998. ISBN ISBN 80-7300-110-1.

[18] *How Microprocessors Work*. online. In: BRAIN, Marshall a POLLETTE, Chris. How Microprocessors Work. 2021, 22.8.2021. Dostupné z: https://computer.howstuffworks.com. [cit. 2024-03-31].

[19] PINKER, Jiří. *Mikroprocesory a mikropočítače*. 2. Praha: BEN - technická literatura, 2004. ISBN 80-730-0110-1.

[20] *Raspberry Pi Projects Book*. 1. Raspberry Pi Trading Ltd., 2016. ISBN 1908256699.

[21] *Spiceworks*. online. In: BASUMALLICK, Chiradeep. What Is Raspberry Pi? Models, Features, and Uses. 2022. Dostupné z: https://www.spiceworks.com. [cit. 2024-03-31].

[22] *Introduction To Computers And C Programming*. 1. New Age International, 2007. ISBN 81-224-1379-X.

[23] COURSERA. *Coursera*. online. In: COURSERA. What Is Python Used For? A Beginner's Guide. 2023. Dostupné z: https://www.coursera.org. [cit. 2024-03-31].

[24] RAMALHO, Luciano. *Fluent Python: clear, concise, and effective programming*. Second edition. Beijing: O'Reilly, 2022. ISBN ISBN:9781492056355.

[25] FREECODECAMP. *What Exactly is Node.js? Explained for Beginners*. online. In: FREECODECAMP. What Exactly is Node.js? Explained for Beginners. 2022. Dostupné z: https://www.freecodecamp.org. [cit. 2024-03-31].

[26] SIMPLELEARN. *What is Node.js: A Comprehensive Guide*. online. In: SUFIYAN, Taha. SIMPLELEARN. What is Node.js: A Comprehensive Guide. 2023. Dostupné z: https://www.simplilearn.com. [cit. 2024-03-31].

[27] *Full Stack Web Development*. 1. Rheinwerk Verlag GmbH, 2024. ISBN 1493224379.

[28] *Investipedia*. online. In: HAYES, Adam. HyperText Markup Language (HTML): What It Is, How It Works. 2022. Dostupné z: https://www.investopedia.com. [cit. 2024-03-31].

[29] *Skillcursh*. online. In: MORRIS, Scott. The Ultimate Guide To CSS. není uvedeno. Dostupné z: https://skillcrush.com. [cit. 2024-03-31].

[30] *Code institude*. online. In: O'GRADY, Brian. JavaScript Demystified: Why You Should Learn This Essential Language. není uvedeno. Dostupné z: https://codeinstitute.net. [cit. 2024-03-31].

[31] CODECADEMY. *What Is a Framework?*. online. In: TEAM, Codecamedy. CODECADEMY. What Is a Framework?. 2021. Dostupné z: https://www.codecademy.com. [cit. 2024-03-31].

[32] SAINI, Ashray. *Understanding Flask Framework: Installation, features & Expert Insights*. online. In: Flask-python. 2024, 7.2.2024. Dostupné z: https://www.analyticsvidhya.com/blog/2021/10/flask-python/. [cit. 2024-02-29].

[33] *An introduction to the Flask Python web app framework*. online. In: HUNT-WALKER, Nicholas. An introduction to the Flask Python web app framework. 2018. Dostupné z: https://opensource.com. [cit. 2024-03-31].

[34] *Understanding Django Framework: Its Purpose and Strengths*. online. In: KORSUN, Julia. Understanding Django Framework: Its Purpose and Strengths. 2024. Dostupné z: https://djangostars.com/blog/why-we-use-django-framework/. [cit. 2024-03-31].

[35] TECHOPEDIA. *Software Library*. online. In: ROUSE, Margaret. Software Library. 2016. Dostupné z: https://www.techopedia.com. [cit. 2024-03-31].

[36] MONGODB. *What is PyMongo? Getting Started with Python and MongoDB*. online. není uvedeno. Dostupné z: https://www.mongodb.com/languages/python/pymongo-tutorial. [cit. 2024-02-29].

[37] LUIS LLAMAS. *How to use Json files in Arduino with Arduino Json*. online. 2019, 15.9.2019. Dostupné z: https://www.luisllamas.es/en/arduino-json/. [cit. 2024-02-29].

[38] *Arduino DHT22 Library Code Examples & Tutorial*. online. In: MAGDY, Khaled. Arduino DHT22 Library Code Examples & Tutorial. není uvedeno. Dostupné z: https://deepbluembedded.com. [cit. 2024-03-31].

[39] INFOWORLD. *What is Visual Studio Code? Microsoft's extensible code editor*. online. In: HELLER, Martin. INFOWORLD. What is Visual Studio Code? Microsoft's extensible code editor. 2022. Dostupné z: https://www.infoworld.com. [cit. 2024-03-31].

[40] *What is PyCharm*. online. In: What is PyCharm. 2023. Dostupné z: https://www.educba.com/what-is-pycharm/. [cit. 2024-03-31].

[41] THOMAS L., Saaty. *Drawing out and Reconciling Differences*. 1. Pittsburgh, Pennsylvania: RWS Publications, 2008. ISBN 978-1-888603-08-8.

[42] *CURL, 17 YEARS OLD TODAY*. online. In: STENBERG, Daniel. CURL, 17 YEARS OLD TODAY. 2015. Dostupné z: https://daniel.haxx.se. [cit. 2024-03-31].

[43] GreenBonO – reference. online. In: TROCHTA, Ing.Richard. *GreenBonO*. 2012. Dostupné z: yorix.cz. [cit. 2024-03-29].

8 Seznam obrázků, tabulek, grafů a zkratek

Obrázek 1 mikrokontroler ESP32 (zdroj: https://allegro.cz/nabidka/esp-32-wroom-wifi-bluetooth-esp32-nodemcu-arduino-12134563733) 16
Obrázek 2 Arduino UNO WiFi (zdroj: https://store.arduino.cc/products/arduino-uno-wifi-rev2) 17
Obrázek 3 Raspberry Pi Zero W (zdroj: https://cz.mouser.com/new/sparkfun/sparkfun-pi-zero-w-basic-kit/) 19
Obrázek 4 Schéma testovacího zapojení (vlastní zpracování) 36
Obrázek 5 IoT model (vlastní zpracování) 37
Obrázek 6 Komunikace REST API serveru a klienta (vlastní zpracování) 38
Obrázek 7 Přidání IP adresy pro přístup do databáze 39
Obrázek 8 Vložené JSON objekty pro testování spojení s databází 39
Obrázek 9 popis kódu REST API – import knihoven, funkce dotenv a připojení k databázi 40
Obrázek 10 popis kódu REST API – komunikace s databází 40
Obrázek 11 popis kódu REST API – funkce GET 41
Obrázek 12 popis kódu REST API – funkce POST 41
Obrázek 13 popis kódu REST API – funkce DELETE 42
Obrázek 14 popis kódu REST API – funkce PUT 42
Obrázek 15 popis kódu REST API – chybové hlášky 43
Obrázek 16 popis kódu ESP32 – knihovny 44
Obrázek 17 popis kódu ESP32 – připojení k síti Wi-Fi a REST API serveru 44
Obrázek 18 popis kódu ESP32 – definování 44
Obrázek 19 popis kódu ESP32 – definování I/O a nastavení časového intervalu čtení 45
Obrázek 20 popis kódu ESP32 – definování datové struktury 45
Obrázek 21 popis kódu ESP32 – HTTP GET metoda (LED) 46
Obrázek 22 popis kódu ESP32 $-$ $HTTP\ GET\ metoda\ (Relay)$ 47
Obrázek 23 popis kódu ESP32 – HTTP PUT metoda funkce 48
Obrázek 24 popis kódu ESP32 – HTTP PUT metoda (DHT22) 49
Obrázek 25 popis kódu ESP32 – funkce setup 50
Obrázek 26 popis kódu ESP32 – funkce loop 51
Obrázek 27 Testování REST API – Získání všech záznamů v kolekci 52
Obrázek 28 Testování REST API – vložení nového záznamu 52
Obrázek 29 Testování REST API – získání záznamu led_1 52
Obrázek 30 Testování REST API – smazání záznamu 52
Obrázek 31 Testování REST API – metoda PUT 52
Obrázek 32 návrh webové stránky (vlastní tvorba) 54
Obrázek 33 Obsah souboru index.html 56
Obrázek 34 Obsah souboru style.css 1 57
Obrázek 35 Obsah souboru style.css 2 57
Obrázek 36 Dashboard webové aplikace 58
Obrázek 37 krabička wattrouteru 60

8.1 Seznam použitých zkratek

IoT – Internet of Things
HTTP – Hypertext Transfer Protocol

REST – Representational state transfer
API – Application programming interface
I/O – input/output

9 Přílohy

V této kapitole jsou uvedeny veškeré přílohy spojené s touto prací.

9.1 Příloha 1 – program

V této příloze se nachází celý program, který zahrnuje zdrojové kódy webové aplikace, mikrokontroleru ESP32 a REST API serveru. Program se nachází na CD a v příloze *projekt.rar*.